THE UNIVERSITY OF CHICAGO

ESSAYS IN BAYESIAN INFERENCE AND DEEP LEARNING

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE UNIVERSITY OF CHICAGO
BOOTH SCHOOL OF BUSINESS
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

BY
JIANENG XU

CHICAGO, ILLINOIS
AUGUST 2021

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my sincerest gratitude to my advisor Prof. Nicholas Polson. Without his support and guidance, my wonderful experience in Booth School of Business would not have been possible. Prof. Polson teaches me a lot about Bayesian statistics, financial markets and wisdom of life with his humor. I always feel lucky and grateful for being his student and having him as my role model. I would also like to thank the rest of my committee members, Prof. Veronika Rockova, Prof. Ruey Tsay and Prof. Dacheng Xiu for their valuable discussion.

During the last six years, my life in Chicago has been full of fun. I must thank all my fellow students, to name but a few, Bumeng Zhuo, Jianfei Shen, Siao Lu, Yuexi Wang, Danna Zhang, Zhipeng Lou, Yuefei Han, Jingwen Jiang, Shihao Gu, Yang Su and Zuguang Gao. I also want to thank Prof. Gavin Feng and Prof. Jingyu He in City University of Hong Kong. I wrote research papers with them and learned a lot from them.

I would like to express my deepest thanks to my mother Yuefen Gao and my father Jianzhong Xu. My parents support me with their love and care. They didn't receive much schooling when they were young. But they make every effort to support my education so that I have enough freedom and time to explore anything that interests me.

Last but not least, I would like to especially thank Chizhi Xia. Your love has changed my life.

CHAPTER 1

BAYESIAN INFERENCE FOR GAMMA MODELS

We use the theory of normal variance-mean mixtures to derive a data augmentation scheme for models that include gamma functions. Our methodology applies to many situations in statistics and machine learning, including Multinomial-Dirichlet distributions, Negative binomial regression, Poisson-Gamma hierarchical models, Extreme value models, to name but a few. All of those models include a gamma function which does not admit a natural conjugate prior distribution providing a significant challenge to inference and prediction. To provide a data augmentation strategy, we construct and develop the theory of the class of Exponential Reciprocal Gamma distributions. This allows scalable EM and MCMC algorithms to be developed. We illustrate our methodology on a number of examples, including gamma shape inference, negative binomial regression and Dirichlet allocation. Finally, we conclude with directions for future research.

Key Words: Data Augmentation, Exponential Reciprocal Gamma, Pólya Gamma, Latent Dirichlet Allocation, Gamma Shape, Markov Chain Monte Carlo, Expectation-Maximization;

1.1 Introduction

Statistical models involving gamma functions are prevalent in statistics and machine learning. For example, topic models, negative binomial regression, time series count models, Poisson-Gamma hierarchical models, non-parametric Bayes, to name but a few (Rossell, 2009; Aktekin, Polson, and Soyer, 2018; Lijoi, Muliere, Prünster, and Taddei, 2016). Gamma distribution also serves as the conjugate prior for many model parameters, such as a normal precision and Poisson intensity. The normalizing constant depends on gamma function whose argument is the shape parameter. Bayesian Inference in gamma models is a long standing problem that presents significant technical and computational difficulties (Damsleth, 1975; Rossell, 2009; Miller, 2018). Similar issue also occurs to the learning of other widely-used gamma models. Table 1.1 gives a list of distributions, where the shape/concentration/dispersion parameters are nested in gamma functions.

By exploiting normal variance-mean mixture identities related to gamma function, we derive a general data augmentation strategy. Our main result is given in Proposition 1, according to which a MCMC algorithm is built in Proposition 2 and an expectation-maximization algorithm is developed in Section 1.3.1.

The main difficulty is to address the reciprocal gamma function $1/\Gamma(\alpha)$. Following Barndorff-Nielsen, Kent, and Sørensen (1982), Hartman (1976) and Roynette and Yor (2005), we represent $1/\Gamma(\alpha)$ as a mean-variance mixture of normals, then we define our new class of distributions named the exponential reciprocal gamma (ERG) class. This novel distribution places gamma models in the same footing as other commonly used Bayesian models, such as sparsity (lasso, horseshoe) and logit (Pólya-Gamma). As a by-product, we show how to nest the latter within our framework. Thus we unify the inference procedure for many models.

Our data augmentation strategy with ERG auxiliary variables may be utilized to design efficient Markov chain Monte Carlo (MCMC) algorithms in latent Dirichlet allocation (Blei, Ng, and Jordan, 2003), Beta-negative binomial models (Zhou, Hannah, Dunson, and Carin,

Table 1.1: List of Gamma Models

Distribution	Likelihood	Applications
Gamma	$\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$	Gamma process, Poisson regression
Inverse Gamma	$\frac{\beta^\alpha}{\Gamma(\alpha)} x^{-\alpha-1} e^{-\beta/x}$	survival analysis, conjugate prior
Beta	$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$	order statistics, wavelet analysis
Dirichlet	$\frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K x_k^{\alpha_k-1}$	topic model, Bayesian networks
Negative Binomial	$\frac{\Gamma(x+\alpha)}{\Gamma(\alpha)\Gamma(\alpha+1)} p^r (1-p)^x$	stock control problems, negative binomial regression

2012), and Gamma-Gamma (GaGa) hierarchical models (Rossell, 2009). It adds to the literature on Bayesian computation with auxiliary variables, which have proven useful in computing posterior distributions in logistic regression (Polson, Scott, and Windle, 2013), negative binomial regression (Pillow and Scott, 2012), multinomial factor models (Holmes and Held, 2006), support vector machines (Mallick, Ghosh, and Ghosh, 2005; Polson and Scott, 2011), and dependent multinomial models (Linderman, Johnson, and Adams, 2015).

To illustrate our methodology, we show examples including gamma shape inference, negative binomial regression and multinomial-Dirichlet model. The first example has a posterior density which exactly matches with the above form. Hence our algorithms can be straightforwardly applied. The posterior samples are efficiently drawn and the posterior mode is easily found by EM algorithm. The second example extends our results to incorporate Pólya-Gamma mixture representation described in (Polson, Scott, and Windle, 2013). The last example demonstrates how our methodology generalizes to high-dimensional case. Conditioned on the auxiliary variables, all elements of the multivariate concentration parameter become mutually independent, which reduces the sampling difficulty significantly. We also suggest the use of normal approximation to speed up the sampling procedure in this example.

The ERG family of distributions is defined as an infinite convolution of Generalized

inverse Gaussian (GIG) distributions and is related to the class of Pólya-Gamma (PG) distributions (Polson, Scott, and Windle, 2013) for logistic regression. Other mixture representations related to GIG are introduced in Zhang, Wang, Liu, Jordan, and Lawrence (2012) and Barndorff-Nielsen and Shephard (2012), with applications in sparse regression and stochastic volatility modelling. The ERG(0) distribution is also a special case of H_a^{Γ} distribution family studied in Roynette and Yor (2005), who provide a representation of the ratio gamma functions as a scale mixture of normals. This adds to scale mixtures results in Bayesian inference, see Andrews and Mallows (1974), Barndorff-Nielsen, Kent, and Sørensen (1982), West (1987), and Polson, Scott, and Windle (2013). Scale mixtures of normals are increasingly used in modeling complex high-dimensional distributions, and Bhattacharya, Chakraborty, and Mallick (2016) provide fast sampling strategies, adding to the practical use of scale mixture distributions in scalable stochastic simulations. Equivalently constructed scalable PG sampling schemes are provided in Windle, Polson, and Scott (2014) and Glynn, Tokdar, Howard, and Banks (2019).

1.1.1 Connections with Existing Work

Obtaining random draws or finding the mode from a posterior distribution involving gamma functions is computationally challenging it requires accurate gamma function evaluations. Approximation methods have been proposed to handle the computational burden. Minka (2000) describes an efficient iterative schemes for maximum likelihood estimate of Dirichlet distribution. The likelihood of Dirichlet precision is approximated by a simpler function (gamma density) by matching the first two derivatives, while the multivariate Dirichlet mean is estimated separately by fixed-point iteration. Miller (2018) applies the same idea of derivative-matching and approximate the full conditional distribution of gamma shape parameter by a gamma density function. Rossell (2009) defines a gamma shape distribution in differential expression analysis. By approximating gamma function with Stirling's formula

and evaluating the limit of the expression, the proposed distribution is roughly proportional to a gamma density. However, These ad-hoc methods are all essentially derived in univariate case. It's not straightforward to generalize them in multivariate cases as we need to deal with correlations, and the computation of approximating parameters itself will get cumbersome. Our framework instead provides an easy and unified way to derive both MCMC and EM algorithms for multivariate models, without the need to approximate density functions.

The rest of our paper proceeds as follows: Section 1.2 defines the class of ERG distributions; Section 1.3 illustrates our data augmentation strategy, developing a parameter expanded Gibbs sampler as well as EM algorithm; Section 1.4 presents examples of gamma shape inference, negative binomial regression and multinomial-Dirichlet model; and Section 1.5 concludes with directions for future research.

1.2 Exponential Reciprocal Gamma (ERG) Distribution

In this section, we present the theoretical development of the ERG distribution class, defining the ERG distribution by the form of its integral transform. In Section 1.2.1, we define a baseline case of the ERG distribution and prove that it is an infinite convolution of independent GIG distributions; in Section 1.2.2, the general class of exponential reciprocal distributions is constructed with an exponential tilting of the baseline case defined in Section 1.2.1. Other convolutions of GIG distribution are discussed in Section 1.2.3.

1.2.1 ERG(0) Distribution

Let ERG(0) denote the exponential reciprocal gamma distribution. The parameter, which is a tilting parameter fixed at zero in this case, will be discussed in greater detail in Section 1.2.2.

Definition 1.2.1. Random variable X_0 has an exponential reciprocal gamma distribution,

ERG(0), if and only if its density function $p_0(x)$ satisfies the following identity

$$E\left(e^{-s^2X_0}\right) = \int_0^\infty e^{-s^2x} p_0(x)dx = \frac{e^{-\gamma s}}{\Gamma(1+s)}, \qquad s > 0. \tag{1.1}$$

where $\gamma = -\psi(1) \approx 0.57721$ is the Euler-Mascheroni constant and $\psi(s) = \frac{d}{ds}\log\Gamma(s)$ is the digamma function.

Remark 1. *The product representation for the reciprocal gamma function due to Weierstrass is,*

$$\frac{e^{-\gamma s}}{\Gamma(1+s)} = \prod_{k=1}^{\infty}\left(1+\frac{s}{k}\right)e^{-\frac{s}{k}},\ s \in \mathbb{C}/\{0,-1,-2,...\}$$

Remark 2. *Roynette and Yor (2005) prove the existence of an infinitely divisible distribution H_a^Γ, with density function $p_a^\Gamma(x)$, such that for $a > 0$*

$$E\left(e^{-\frac{1}{2}s^2H_a^\Gamma}\right) = \int_0^\infty e^{-\frac{1}{2}s^2x} p_a^\Gamma(x)dx = \frac{\Gamma(a)}{\Gamma(a+s)}e^{\psi(a)s}$$

which follows from the general product representation for the reciprocal gamma function,

$$\frac{\Gamma(a)}{\Gamma\left(a+s\right)}e^{\psi(a)s} = \prod_{k=0}^{\infty}\left(1+\frac{s}{a+k}\right)e^{-\frac{s}{a+k}}. \tag{1.2}$$

Note that the representation in Remark 1 is a special case with $a = 1$. Hence

$$ERG(0) \overset{D}{=} \frac{1}{2}H_1^\Gamma.$$

Remark 3. *The ERG$(n,0)$ is defines as follows. If $X_{n,0} \sim ERG(n,0)$,*

$$E\left(e^{-s^2X_{n,0}}\right) = \int_0^\infty e^{-s^2x} p_{n,0}(x)dx = \frac{e^{-\gamma ns}}{\Gamma(1+s)^n}, \qquad n > 0, s > 0.$$

Note that ERG$(n,0)$ is the equivalent in distribution to the sum of n independent ERG$(1,0)$

when n is a positive integer.

1.2.2 General ERG(c) Distribution

We now construct the general class of ERG distributions, ERG(c), by exponentially tilting the ERG(0) distribution. The exponential tilting strategy – similar to the one used by Polson, Scott, and Windle (2013) – allows a second parameter $c \in \mathbb{R}^+$ to inform a priori the precision of the ERG random variable.

Definition 1.2.2. The ERG(c) distribution is constructed as an exponential tilting of the ERG(0) density. Its density function is

$$p_c(x) = Z_c \cdot \exp\left(-c^2 x\right) p_0(x), \quad x, c > 0.$$

The normalizing constant, namely $Z_c = 1/E\left(\exp\left(-c^2 X_0\right)\right)$ where X_0 is an ERG(0) random variable, can be calculated using the identity in Remark 1. Similarly, the integral identity of ERG(c) is given by

$$E\left(e^{-s^2 X_c}\right) = \prod_{k=1}^{\infty} \left(\frac{k + \sqrt{s^2 + c^2}}{k + c}\right) e^{-\frac{\sqrt{s^2+c^2}-c}{k}} \tag{1.3}$$

$$= \frac{\Gamma(1+c)}{\Gamma(1+\sqrt{s^2+c^2})} e^{-\gamma(\sqrt{s^2+c^2}-c)}. \tag{1.4}$$

Our main result, presented in Theorem 1, is that a random variable $X_c \sim \text{ERG}(c)$ may be constructed from an infinite sum of independent generalized inverse Gaussian (GIG) random variables. The power of the result lies in the ability to identify previously unknown conditional posterior distributions.

Theorem 1. *The ERG(c) class of distributions can be constructed as an infinite sum of*

independent generalized inverse Gaussian (GIG) distributions as follows

$$ERG(c) \stackrel{D}{=} \sum_{k=1}^{\infty} GIG\left(-\frac{3}{2}, 2c^2, \frac{1}{2k^2}\right).$$

In particular, when $c = 0$, the GIG distribution reduces to inverse gamma distribution. Hence,

$$ERG(0) \stackrel{D}{=} \sum_{k=1}^{\infty} \frac{1}{4k^2}\Gamma_k^{-1}.$$

where Γ_k^{-1} are i.i.d. inverse gamma random variables with shape $\frac{3}{2}$ and scale 1.

Proof. See Appendix 1.6.1. □

The following theorem concerns the first two moments of ERG(c) which will be used to construct our EM algorithm in Section 1.3.1 and the approximate Gibbs sampler in Appendix 1.6.4.

Theorem 2. *If G_k is a GIG random variable with $p = -\frac{3}{2}, a = 2c^2, b = \frac{1}{2k^2}$, then the mean and variance of the tail infinite sum $\sum_{k=N}^{\infty} G_k$ are*

$$E\left(\sum_{k=N}^{\infty} G_k\right) = \frac{1}{2c}\left(\psi(N+c) - \psi(N)\right)$$
$$Var\left(\sum_{k=N}^{\infty} G_k\right) = \frac{1}{4c^3}\left(\psi(N+c) - \psi(N) - c\psi'(N+c)\right),$$

where $\psi(s) = \frac{d}{ds}\log\Gamma(s)$ is the digamma function. Setting $N = 1$ gives us the first two moments of ERG(c).

Proof. If g_k is a GIG random variable with $p = -\frac{3}{2}, a = 2c^2, b = \frac{1}{2k^2}$, then its mean and

variance are given by

$$E(g_k) = \frac{1}{2}\left(\frac{1}{k^2+ck}\right), \quad \mathrm{Var}(g_k) = \frac{1}{4c}\left(\frac{1}{k^3+2ck^2+c^2k}\right).$$

Theorem 2 is thus a direct application of Theorem 1. □

Remark 4. *The ERG(c) distribution class belongs to the family of generalized gamma convolutions (GGC), Bondesson (1992). Its Laplace transform in Equation (1.4) also satisfies*

$$E(e^{-sX_c}) = \exp\left\{\int_0^\infty (e^{-sx}-1)\nu(x)dx\right\}, \quad s \geq 0.$$

Its Lévy density $\nu(x)$ and Thorin density $\mu(t)$ are

$$\begin{aligned}\nu(x) &= \frac{1}{x}\int_0^\infty e^{-tx}\mu(t)dt,\\ \mu(t) &= \mathbb{1}_{t\geq c^2}\frac{\psi(1-\sqrt{c^2-t})+\psi(1+\sqrt{c^2-t})+2\gamma}{4\pi\sqrt{t-c^2}}.\end{aligned}$$

This result can be used to generate ERG random variables as it shows that it falls into the class of Generalized Gamma Convolution, see Bondesson (1982) and Rosiński (2001).

1.2.3 GIG Mixtures

The ERG distribution allows us to represent the unnormalized density $\frac{e^{ax}}{\Gamma(1+x)}$ as a normal variance-mean mixture. That is,

$$\frac{e^{ax}}{\Gamma(1+x)} = \int_0^\infty \phi(x \mid \mu(\omega), \sigma^2(\omega)) \cdot \tau(\omega) p_0(\omega) d\omega$$

where $\phi(\cdot \mid \mu, \sigma^2)$ is the normal density with mean μ and variance σ^2 and $P_0(\cdot)$ is the distribution function of ERG(0). $\mu(\omega) = \frac{a+\gamma}{2\omega}, \sigma^2(\omega) = \frac{1}{2\omega}$ and $\tau(\omega) = \sqrt{\frac{\pi}{\omega}}\exp\left\{\frac{a+\gamma}{4\omega}\right\}$. In statistics and machine learning, probability distributions whose density function $p(x)$ is of

the following form are used explicitly and implicitly:

$$p(x) = \int_0^\infty f(x \mid \theta(\omega))p(\omega)d\omega. \tag{1.5}$$

Here $f(x \mid \theta(\omega))$ is some well-known density function, e.g. normal, and the mixing $p(\omega)$ is the distribution of a single GIG or an infinite convolution of GIG's. Combined with a data-augmentation scheme, the above mixture representation provides a powerful framework to solve many non-Gaussian models.

1. When $f = \phi$ and the mixing distribution is GIG, Polson and Scott (2013) give the variance-mean mixture representations for many common loss functions in regression and binary classification problems, which corresponds to different choices of the function $\theta(\omega) = (\mu(\omega), \sigma^2(\omega))$ and parameters of GIG. For example, absolute loss $L(y) = |y|$, hinge loss for support vector machine $L(y) = \max(1 - y, 0)$ and check loss for quantile regression $L(y) = |y| + (2q - 1)y$. The representations then help reduce those non-Gaussian models to Gaussian linear models with heteroscedastic errors. Note that GIG is a very general family with many common distributions as its special cases, such as gamma, inverse gamma and inverse Gaussian distribution.

2. For the logistic loss in binary classification $L(y) = \log(1 + e^y)$, Polson and Scott (2013) show that it is also a normal variance-mean mixture. The mixing distribution is Pólya distribution, which is constructed as an infinite sum of exponentials. Note that exponential distribution is again a special case of GIG.

3. By choosing f to be the exponential power density, $f(x \mid \eta(\omega), q) \propto \exp\left\{-\frac{1}{2\eta(\omega)}|x|^q\right\}$, and the mixing distribution $p(\omega)$ to be GIG, Zhang, Wang, Liu, Jordan, and Lawrence (2012) introduce a sparsity-inducing prior called EP-GIG and develop EM algorithms for sparse learning. The density function of EP-GIG is given explicitly and special cases (generalized t distribution and exponential power-gamma distribution) are discussed

when the mixing GIG reduces to inverse gamma and gamma respectively.

4. The Pólya-Gamma distribution class is proposed by Polson, Scott, and Windle (2013) to solve the inference problem in models with binomial likelihoods, including logistic regression and negative binomial regression. Pólya-Gamma distribution $\mathrm{PG}(b, c)$ can be written as an infinite convolution of gamma distributions whose shapes are all equal to b and scales depend on c, or equivalently $\mathrm{GIG}(b, 2, 0)$.

$$\mathrm{PG}(b, c) \stackrel{D}{=} \frac{1}{2\pi^2} \sum_{k=1}^{\infty} \frac{\mathrm{GIG}(b, 2, 0)}{(k - 1/2)^2 + c^2/(4\pi^2)}.$$

Choose $f(z \mid \omega) = 2^{-b} \exp\left\{-\omega z^2/2 + \kappa z\right\}$ with $\kappa = a - b/2$ and $z = x'\beta$, the likelihood of logistic regression is a mixture

$$\frac{(e^z)^a}{(1 + e^z)^b} = \int_0^{\infty} f(z \mid \omega) p_{\mathrm{PG}}(\omega \mid b, 0) d\omega. \tag{1.6}$$

Here $p_{\mathrm{PG}}(\omega \mid b, 0)$ is the density of $\mathrm{PG}(b, 0)$ and $f(z \mid \omega)$ is proportional to the normal density where the mean and variance are functions of ω.

5. Barndorff-Nielsen and Shephard (2012) use a normal variance-mean mixture as a general approach of building densities on the real line. Here $f(x \mid \theta(\omega)) = \phi(x \mid \mu + \beta\omega, \omega)$. When $p(\omega)$ is GIG density, the resulted mixture is generalized hyperbolic distribution, which includes many special cases such as normal inverse Gaussian, normal gamma, Laplace, skewed Student's t distribution. Furthermore, normal distribution can also be written as a limiting case of generalized hyperbolic distribution.

1.3 MCMC and Data Augmentation

This section illustrates the data augmentation strategy and sampling scheme for Gamma inference using the ERG class distributions. First, notice that many Bayesian gamma models

Table 1.2: Integral Representation for Gamma Functions

	Integral Representation	Auxiliary Variables
$\Gamma(\alpha)$	$\int_0^\infty x^{\alpha-1}e^{-x}dx$	Gamma
$\frac{1}{\Gamma(\alpha)}$	$\int_0^\infty \alpha e^{-\alpha^2 x+\gamma\alpha}p_0(x)dx$	ERG
$\frac{\Gamma(\alpha)}{\Gamma(\alpha+\beta)}$	$\int_0^1 \frac{1}{\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1}dx$	Beta

involve a posterior density of the form

$$p(x) = C\cdot\left(\prod_{\ell=1}^{L}\Gamma(g_\ell(x))\right)\left(\prod_{m=1}^{M}\frac{1}{\Gamma(h_m(x))}\right)\left(\prod_{n=1}^{N}\frac{\Gamma(j_n(x))}{\Gamma(j_n(x)+\beta_n)}\right)\cdot x^{p-1}e^{-ax^2+bx},\ x>0, \tag{1.7}$$

where $x\in\mathbb{R}^+$ is on the positive real line, and C is the normalizing constant. The arguments in gamma functions, $\{g_\ell(\cdot)\}_{\ell=1}^L, \{h_m(\cdot)\}_{m=1}^M$ and $\{j_n(\cdot)\}_{n=1}^N$ are nonnegative increasing linear functions of x on $(0,\infty)$. We assume $M\geq 1$, otherwise the form might not be integrable. Parameters $p, a, b, \beta_1, ..., \beta_N$ are scalars. β_n's are positive. p and a are nonnegative. In order to perform full posterior inference on the variable x, a sampling procedure for $p(x)$ is needed, while a Maximum A Posteriori (MAP) estimate requires finding the maximizer of it.

The idea of the data augmentation strategy is to introduce a group of auxiliary random variables $\boldsymbol{\omega}=(\omega_1,...,\omega_n)^T$ such that for $\boldsymbol{\omega}\in\Omega$,

$$p(x)=\int_\Omega p(x,\boldsymbol{\omega})d\boldsymbol{\omega}$$

and the joint density $p(x,\boldsymbol{\omega})$ after augmentation is easier to deal with, as it doesn't consist of gamma functions any longer.

Returning to Equation (1.7), the posterior density involves gamma functions $\Gamma(\cdot)$, reciprocal gamma functions $1/\Gamma(\cdot)$ and gamma ratios $\Gamma(\cdot)/\Gamma(\cdot+\beta)$. We will then express each of them using the corresponding integral representation in Table 1.2. This is equivalent to

introducing an auxiliary random variable for each of them. The total number of the auxiliary random variables is then $L + M + N$.

Proposition 1. The posterior density admits an integral representation as follows

$$p(x) = C \cdot \int_{(0,\infty)^{L+M}\times(0,1)^N} G(x,\boldsymbol{\tau}) \cdot H(x,\boldsymbol{\omega}) \cdot J(x,\boldsymbol{\eta}) \cdot Q(x) \cdot e^{-ax^2+bx} d\boldsymbol{\tau} d\boldsymbol{\omega} d\boldsymbol{\eta} \quad (1.8)$$

where

$$\begin{aligned}
G(x,\boldsymbol{\tau}) &= \prod_{\ell=1}^{L} \tau_\ell^{g_\ell(x)-1} e^{-\tau_\ell} \\
H(x,\boldsymbol{\omega}) &= \exp\left\{ -\sum_{m=1}^{M} h_m(x)^2 \omega_m + \gamma \sum_{m=1}^{M} h_m(x) \right\} \prod_{m=1}^{M} p_0(\omega_m) \\
J(x,\boldsymbol{\eta}) &= \prod_{n=1}^{N} \eta_n^{j_n(x)-1} (1-\eta_m)^{\beta_n - 1} \\
Q(x) &= x^{p-1} \prod_{m=1}^{M} h_m(x).
\end{aligned}$$

$p_0(\cdot)$ is the probability density of ERG(0).

Remark 5. *When $N \geq 1$ in the form (1.7), the third term $\left(\prod_{n=1}^{N} \frac{\Gamma(j_n(x))}{\Gamma(j_n(x)+\beta_n)}\right)$ can be absorbed into the first two terms. However, we still recommend using the Beta representation if possible. Otherwise the total number of auxiliary variables needed is increased by N.*

Remark 6. *We may generalize the form (1.7) in a few ways:*

1. *For multivariate x of dimension d, if g, h, j are all linear functions mapping from $\mathbb{R}^{+,d}$ to $\mathbb{R}^+$, then the data augmentation is still valid.*

2. *The x^p term can be replaced with a polynomial function of x, as long as it's always positive for $x > 0$.*

3. *If the posterior density $p(x)$ has extra factors which are not included in the form (1.7), the strategy still works as long as we can find integral representations for those extra factors.*

Although the posterior joint distribution in Equation (1.8) looks forbidding at the first glance, in many applications the linear functions g, h, j are simple enough, e.g. $h_m(x) = x$ for all m, and $N = 0$, which simplifies the expression a lot. More importantly, the conditional posteriors can be derived easily. To derive that of τ_ℓ, for example,

$$p(\tau_\ell \mid x, \boldsymbol{\tau}_{-\ell}, \boldsymbol{\omega}, \boldsymbol{\eta}) = \tau_\ell^{g_\ell(x)-1} e^{-\tau_\ell} \cdot \frac{\left(\prod_{\ell' \neq \ell} \tau_{\ell'}^{g_{\ell'}(x)-1} e^{-\tau'_\ell} \cdot H(x, \boldsymbol{\omega}) \cdot J(x, \boldsymbol{\eta}) \cdot Q(x)\right)}{\int G(x, \boldsymbol{\tau}) \cdot H(x, \boldsymbol{\omega}) \cdot J(x, \boldsymbol{\eta}) \cdot Q(x) d\tau_\ell},$$

notice that it's proportional to $\tau_\ell^{g_\ell(x)-1} e^{-\tau_\ell}$. Therefore the conditional posterior distribution of τ_ℓ is $\Gamma(g_\ell(x), 1)$. Similarly for ω_m's and η_n's. For the conditional posterior of x, since g, h, j are all linear, it's proportional to $Q_1(x)e^{-Q_2(x)}$ where $Q_1(x)$ is a polynomial of degree $p + M - 1$ and $Q_2(x)$ is a quadratic function.

Before we summarize the conditional posteriors with respect to the augmented vector $(x, \boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\eta})$, the following definition of power truncated normal (PTN) distribution is useful.

Definition 1.3.1. The power truncated normal distribution, PTN(p, a, b), has density function

$$p(x) = C \cdot x^{p-1} e^{-ax^2+bx}, x > 0 \tag{1.9}$$

where $p, a > 0$ and $b \neq 0$.

Finally, the following proposition is helpful in developing the Gibbs sampler.

Proposition 2 (Gibbs Sampler)**.** If the joint probability density of the augmented vector $(x, \boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\eta})$ is proportional to $G(x, \boldsymbol{\tau}) \cdot H(x, \boldsymbol{\omega}) \cdot J(x, \boldsymbol{\eta}) \cdot Q(x)$ given in Proposition 1, then

the conditional distributions are

$$\tau_\ell \mid x, \boldsymbol{\tau}_{-\ell}, \boldsymbol{\omega}, \boldsymbol{\eta} \sim \Gamma(g_\ell(x), 1), \quad \ell = 1, 2, ..., L$$

$$\omega_m \mid x, \boldsymbol{\tau}, \boldsymbol{\omega}_{-m}, \boldsymbol{\eta} \sim \mathrm{ERG}(h_m(x)), \quad m = 1, 2, ..., M$$

$$\eta_n \mid x, \boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\eta}_{-n} \sim \mathrm{Beta}(j_n(x), \beta_n), \quad n = 1, 2, ..., N$$

$$x \mid \boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\eta} \sim \sum_{m=0}^{M} \pi_k \cdot \mathrm{PTN}(p + m, \tilde{a}, \tilde{b})$$

Here the conditional distribution of x is a finite discrete mixture of PTN distributions. $\{\pi_m\}_{m=0}^M$ are proportional to the coefficients in the polynomial $Q(x)$.

$$\tilde{a} = a + \sum_{m=1}^{M} \omega_m \cdot (h_m'(0))^2$$

$$\tilde{b} = b + \left(\sum_{\ell=1}^{L} g_\ell'(0) \cdot \log \tau_\ell \right) + \left(\sum_{m=1}^{M} h_m'(0)(\gamma - 2\omega_m h_m(0)) \right) + \left(\sum_{n=1}^{N} j_n'(0) \cdot \log \eta_n \right).$$

Furthermore, given x, all auxiliary random variables are conditionally independent. The sampling methods for ERG and PTN are given in Appendix 1.6.2 and 1.6.3.

In many statistical applications, the number of auxiliary random variables grows linearly with the sample size and problem dimension, but the forms of g, h, j are relatively simple. Observing that $\tilde{a}$ and $\tilde{b}$ are both the sum of numerous terms, we may use normal variables to approximate them, by matching the moments, so that the sampling procedure for $(\boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\eta})$ can be skipped in Gibbs sampling. Appendix 1.6.4 gives the approximate Gibbs sampler when $g_\ell(x) = g(x)$ for all ℓ (similarly for h and j), and L, M, N are all large enough.

1.3.1 Expectation-Maximization Algorithm

By exploiting Proposition 1, MAP of the model can be found using an expectation-maximization algorithm. The complete-data log posterior is

$$\begin{aligned}
&\log p(x, \boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\eta}) \\
&= c_0 + \log G(x, \boldsymbol{\tau}) + \log H(x, \boldsymbol{\omega}) + \log J(x, \boldsymbol{\eta}) + \log Q(x) - ax^2 + bx \\
&= c_1 + \left(\sum_{\ell=1}^{L} g_\ell(x) \log \tau_\ell \right) + \left(\sum_{m=1}^{M} \gamma h_m(x) - h_m(x)^2 \omega_m \right) \\
&\quad + \left(\sum_{n=1}^{N} j_n(x) \log \eta_n \right) + \log Q(x) - ax^2 + bx
\end{aligned} \tag{1.10}$$

for some constants c_0, c_1 (with respect to x). In the t-th expectation step, we compute the expected value of $\log p(x, \boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\eta})$ under the current conditional posterior $p(\boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\eta} \mid x_{(t)})$, denoted as $C(x \mid x_{(t)})$. Then in the maximization step, $C(x \mid x^{(t)})$ is maximized as a function of x. We now derive the expectation and maximization steps.

- **The Expectation Step**

  From equation (1.10), notice that $\log p(x, \boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\eta})$ is linear in terms of $\log \tau_\ell, \omega_m$ and $\log \eta_n$. Therefore, we replace them with their conditional expectations in the expectation step. Applying Proposition 2 yields

$$\begin{aligned}
E\left(\log \tau_\ell \mid x_{(t)}\right) &= \psi(g_\ell(x_{(t)})), \quad \ell = 1, 2, ..., L \\
E\left(\omega_m \mid x_{(t)}\right) &= \frac{\psi(1 + h_m(x_{(t)})) + \gamma}{2h_m(x_{(t)})}, \quad m = 1, 2, ..., M \\
E\left(\log \eta_n \mid x_{(t)}\right) &= \psi(j_n(x_{(t)})) - \psi(j_n(x_{(t)}) + \beta_n), \quad n = 1, 2, ..., N.
\end{aligned}$$

  The derivation above uses the properties of gamma and beta distribution, as well as Theorem 2 which calculates the expectation of $\mathrm{ERG}(c)$ distribution. Finally, one can

represent the function $C(x \mid x_{(t)})$ (up to a constant) as

$$C(x \mid x_{(t)}) = \left(\sum_{\ell=1}^{L} g_\ell(x)\psi(g_\ell(x_{(t)})) \right) + \left(\sum_{m=1}^{M} \gamma h_m(x) - h_m(x)^2 \frac{\psi(1+h_m(x_{(t)})) + \gamma}{2h_m(x_{(t)})} \right) + \left(\sum_{n=1}^{N} j_n(x)\psi(j_n(x_{(t)})) - \psi(j_n(x_{(t)}) + \beta_n) \right) + \log Q(x) - ax^2 + bx$$

Given the linearity of g, h, j functions, it can be further simplified as

$$C(x \mid x_{(t)}) = \log Q(x) - \kappa_1 x^2 + \kappa_2 x \tag{1.11}$$

for some constant $\kappa_1 > 0$ and $\kappa_2 \in \mathbb{R}$, depending on $x_{(t)}$.

- **The Maximization Step**

  Since $C(x \mid x_{(t)}) \to -\infty$ as $x \to \infty$, we conclude that $C(x \mid x_{(t)})$ as a function of x has a maximizer on $(0, \infty)$, which can be found numerically. Furthermore, when $h_m(0) = 0$ for all m, the unique maximizer has a closed form

$$x^* := \arg\max_{x>0} C(x \mid x_{(t)}) = \frac{\kappa_2 + \sqrt{\kappa_2^2 + 8\kappa_1(p + M - 1)}}{4\kappa_1}. \tag{1.12}$$

1.4 Examples

1.4.1 Inference for Gamma Shape

The gamma distribution, parameterized by shape α and rate β, is a component of many probability models. For instance, a gamma prior distribution for the precision parameter in Gaussian linear models is quite common. In fact, Normal-gamma distributions are workhorse models for shrinkage estimation in regression problems (Griffin and Brown, 2010). Gamma distribution is also widely used in modelling of extreme values, where it serves as the prior for

the shape parameter of Pareto distribution (Arnold and Press (1989)) and helps to construct a quasi-conjugate prior for generalized Pareto distribution (Diebolt, El-Aroui, Garrido, and Girard (2005)). While a gamma prior distribution for a parameter is common, it is less common to model hyperparameters of the gamma distribution itself as random variables – particularly the shape parameter, α. Although posterior inference of the rate parameter is straightforward – since the gamma distribution itself is a conjugate prior for the rate parameter – posterior inference of the gamma shape parameter is a long-standing problem (Damsleth, 1975; Damien, Laud, and Smith, 1995; Rossell, 2009; Miller, 2018) and efficient posterior computation remains an open problem.

Damsleth (1975) discussed two conjugate priors for α, the gamma shape parameter. They are called GamCon distributions of type I and type II. Type I assumes the rate β is known, while the type II doesn't. In this section, we focus on the latter one and replicate Damsleth's example, showing how to utilize the data augmentation scheme and build algorithms for posterior inference.

Let's consider the following hierarchical model

$$\begin{aligned} x \mid \alpha, \beta &\sim \Gamma(\alpha, \beta), \\ \beta \mid \alpha &\sim \Gamma(\delta\alpha + 1, \delta\eta), \\ \alpha &\sim \xi_2(\eta/\mu, \delta). \end{aligned}$$

Here ξ_2 is GamCon distribution of type II. $\eta > \mu > 0, \delta > 0$. The probability density of $\xi_2(\mu, \delta)$ is

$$p(\alpha \mid \mu, \delta) = C_{\mu,\delta} \cdot \frac{\Gamma(\delta\alpha + 1)}{\Gamma(x)^\delta} (\delta\mu)^{-\delta\alpha},\ \alpha > 0, \mu > 1, \delta > 0.$$

Suppose observations $\boldsymbol{x} = (x_1, x_2, ..., x_n)'$ are independently and identically distributed

gamma random variables drawn from the above model. The likelihood is

$$f(\boldsymbol{x} \mid \alpha, \beta) = \prod_{i=1}^{n} \frac{\beta^{\alpha}}{\Gamma(\alpha)} x_i^{\alpha-1} e^{-\beta x_i} \propto \frac{(x_g \beta)^{n\alpha}}{\Gamma(\alpha)^n} e^{-n x_g \beta}$$

where x_g is the geometrical mean, $x_g = (\prod_{i=1}^{n} x_i)^{1/n}$.

The marginal posterior distribution of α is then calculated as

$$p(\alpha \mid \boldsymbol{x}) = \int_0^{\infty} f(\boldsymbol{x} \mid \alpha, \beta) p(\beta \mid \alpha) p(\alpha) d\beta$$

$$\text{where} \qquad p(\beta \mid \alpha) = \frac{(\delta\eta)^{\delta\alpha+1}}{\Gamma(\alpha\delta+1)} \beta^{\alpha\delta} e^{-\delta\eta\beta}, \text{ and} \qquad p(\alpha) = C \cdot \frac{\Gamma(\delta\alpha+1)}{\Gamma(\alpha)^{\delta}} (\delta\eta/\mu)^{-\delta\alpha}.$$

By construction, the marginal posterior of α given $\boldsymbol{x}$ also follows ξ_2, with updated parameters (η', μ', δ'). That is, $\alpha \mid \boldsymbol{x} \sim \xi_2(\eta'/\mu', \delta')$ and

$$\begin{aligned} \delta' &= \delta + n, \\ \eta' &= \frac{\delta}{\delta+n}\eta + \frac{n}{\delta+n} x_a, \\ \mu' &= \mu^{\frac{\delta}{\delta+n}} \cdot x_g^{\frac{n}{\delta+n}}. \end{aligned}$$

where x_a is the arithmetical mean, $x_a = \frac{1}{n}\sum_{i=1}^{n} x_i$. One can observe that η' is a weighted arithmetical mean of η and x_a with weights δ and n respectively and μ' is a weighted geometrical mean of μ and x_g, also with weights δ and n. Here x_a and x_g are two sufficient statistics. δ is viewed as the prior sample size while η and μ are the prior means. The problem of gamma shape inference is thus translated to that of ξ_2 distribution.

MCMC and EM for ξ_2 when $\delta \in \mathbb{N}^+$

We first develop the Gibbs sampler for a general GamCon distribution of type II, using data augmentation strategy. Suppose $x \sim \xi_2(x; \mu, \delta)$ and δ is a positive integer. Then the

probability density can be rewritten in the form (1.7)

$$p(x \mid \mu, \delta) = C_{\mu,\delta} \cdot \Gamma(\delta x + 1) \left(\prod_{m=1}^{M} \frac{1}{\Gamma(x)} \right) e^{-\delta \log(\delta\mu) x} \tag{1.13}$$

Hence $L = 1$ with $g_1(x) = \delta x + 1$, $M = \delta$ with $h_m(x) = x$ and $N = 0$. $p = 1, a = 0$ and $b = -\delta \log(\delta\mu)$.

After introducing the auxiliary variables τ and $\omega_1, \omega_2, ..., \omega_\delta$, Proposition 2 then immediately gives the conditional posteriors:

$$\begin{aligned} \tau \mid x, \boldsymbol{\omega} &\sim \Gamma(\delta x + 1, 1) \\ \omega_1, \omega_2, ..., \omega_\delta \mid x, \tau &\overset{i.i.d.}{\sim} \mathrm{ERG}(x) \\ x \mid \tau, \boldsymbol{\omega} &\sim \mathrm{PTN}\left(\delta + 1, \sum_{i=1}^{\delta} \omega_i, \delta\left(\gamma - \log(\delta\mu/\tau)\right) \right) \end{aligned}$$

Damsleth Examples with Non-Informative Prior

Here we replicate Damsleth's example of no prior information with sample size $n = 5, 10, 30$. Putting $\delta = 0$, the posterior parameters are

$$\delta' = n, \quad \eta' = x_a, \quad \mu' = x_g.$$

The resulted posterior of gamma shape is thus $\alpha \mid \boldsymbol{x} \sim \xi_2(x_a/x_g, n)$. Instead of actually generating Gamma random variables, we directly use the sufficient statistics (x_a, x_g), given in Table 2 of Damsleth (1975). The true value of α is 5. The histogram of 5000 posterior samples for each case are shown in Figure 1.1. The colored solid lines denote the true posterior density and the black dashed line denotes the true α. We see that the posterior samples are indeed sampled from the target distributions. Figure 1.2 shows the sampling trace plots. In Table 1.3, the sample moments are compared with their theoretical counterparts calculated

by numerical integration. When $n = 30$, the deviations from the theoretical values are $-1.0\%, -4.6\%, -8.2\%$ and 0.1% for mean, variance, skewness and kurtosis respectively.

Figure 1.1: Histogram of Posterior Samples

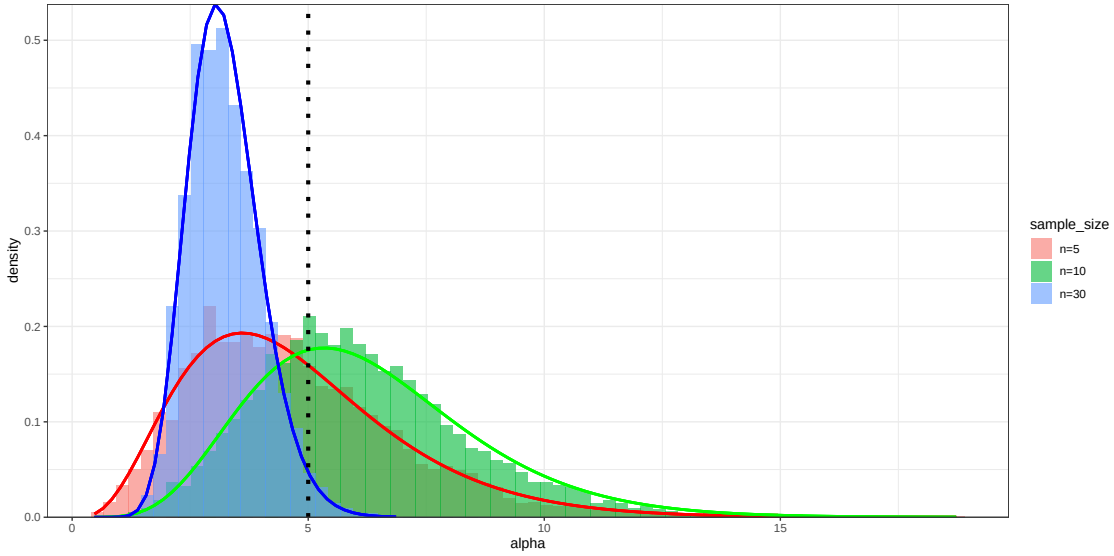
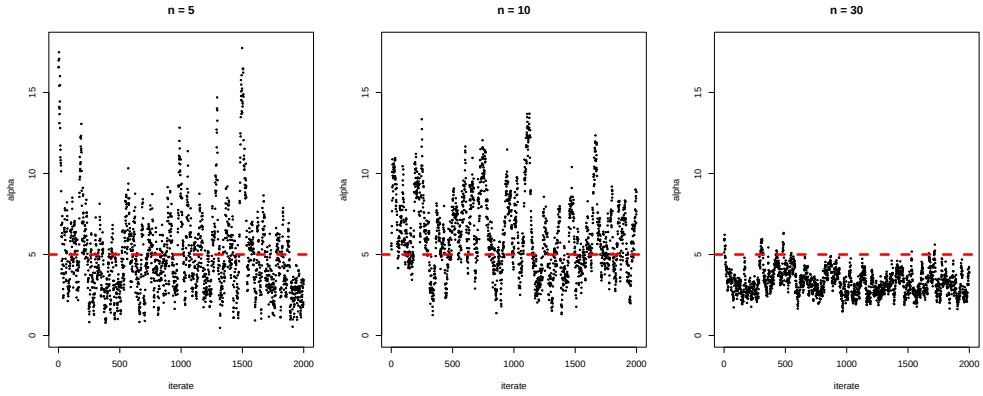


Figure 1.2: Trace Plot of Posterior Samples



To find the posterior mode of α, we exploit the EM algorithm developed in Section 1.3.1. Specially, in the expectation step, the conditional expected value of complete-data log

Table 1.3: Posterior Moments

n	x_a	x_g	Method	Mean	Variance	Skewness	Kurtosis
5	7.19	6.05	Numerical Integration	4.768	5.399	0.997	4.494
			Posterior Sampling	4.779	5.630	1.305	6.148
10	5.57	5.01	Numerical Integration	6.271	5.780	0.783	3.921
			Posterior Sampling	6.101	4.857	0.634	3.427
30	5.09	4.26	Numerical Integration	3.252	0.585	0.490	3.361
			Posterior Sampling	3.250	0.605	0.537	3.155

posterior (at t-th iteration) given by Equation (1.11) now has the parameters

$$\begin{aligned}\log Q(x) &:= \delta' \log \alpha \\ \kappa_1 &:= \frac{\delta' \left(\gamma + \psi(\alpha^{(t)} + 1)\right)}{2\alpha^{(t)}} > 0 \\ \kappa_2 &:= \delta' \left(\gamma - \log \delta' \eta'/\mu' + \psi(\delta' \alpha^{(t)} + 1)\right)\end{aligned}$$

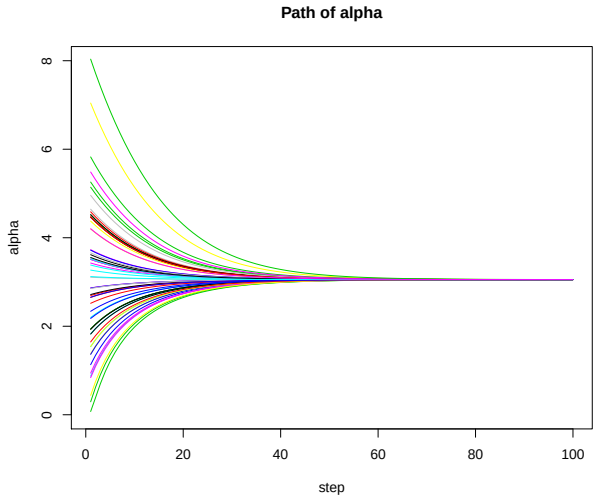
Then in the maximization step, α is updated by Equation (1.12), with $p - M + 1 = \delta'$. In Figure 1.3, we start with 30 different initial values in EM algorithm and show the optimizing paths for the case $n = 30$. In this particular case, the algorithm converges after around 50 steps. The 30 numerical solutions given by EM has mean 3.054, which matches with the maximizer found by Mathematica. And the standard deviation is as small as 0.0003.

1.4.2 Negative Binomial Regression

Next, we proceed to how our data augmentation strategy can be used to fit negative binomial regression models. The count data $\{y_i\}_{i=1}^n$ are assumed to follow the negative binomial distribution

$$y_i \mid r, p_i \sim \mathrm{NB}(r, p_i), \quad p_i = \frac{1}{1 + e^{-\boldsymbol{x_i'}\boldsymbol{\beta}}}$$

Figure 1.3: Optimizing Paths of EM Algorithm



where $\{\boldsymbol{x}_i\}_{i=1}^n$ are observed covariates and $\boldsymbol{\beta}$ is the regression coefficients. r is interpreted as the number of failures until the experiment is stopped, while the success probability p_i is related to $\boldsymbol{x_i'\beta}$ via the logistic transformation. This model specification is equivalent to the following Gamma-Poisson mixture,

$$y_i \mid \lambda_i \sim \text{Poisson}(\lambda_i),$$
$$\lambda_i \mid r, \boldsymbol{x_i'\beta} \sim \Gamma\left(r, e^{-\boldsymbol{x_i'\beta}}\right).$$

The likelihood is

$$f(y_i \mid \boldsymbol{x_i}, \boldsymbol{\beta}, r) = \frac{\Gamma(y_i + r)}{y_i! \cdot \Gamma(r)} \left(\frac{e^{\boldsymbol{x_i'\beta}}}{e^{\boldsymbol{x_i'\beta}} + 1}\right)^{y_i} \left(\frac{1}{e^{\boldsymbol{x_i'\beta}} + 1}\right)^{r}.$$

Pillow and Scott (2012) adopt exactly the same model with known r and use a data augmentation strategy for the inference on β. Our example here extends their method and allows for inference on both r and β. The parameter r controls the dispersion of observations, as the expectation of y_i is $re^{\boldsymbol{x_i'\beta}}$ and variance is $re^{\boldsymbol{x_i'\beta}}\left(1 + e^{\boldsymbol{x_i'\beta}}\right)$.

Let the prior for $\boldsymbol{\beta}$ and r be $N(\mathbf{0}, \Sigma)$ and $p(r)$. We then calculate the joint posterior of

$(\boldsymbol{\beta}, r)$ as

$$p(\boldsymbol{\beta}, r \mid \boldsymbol{X}, y) = Cp(r) \exp\left(-\frac{1}{2}\boldsymbol{\beta}'\Sigma^{-1}\boldsymbol{\beta}\right) \prod_{i=1}^{n} \frac{\Gamma(y_i + r)}{\Gamma(r)} \left(\frac{e^{\boldsymbol{x_i'}\boldsymbol{\beta}}}{e^{\boldsymbol{x_i'}\boldsymbol{\beta}} + 1}\right)^{y_i} \left(\frac{1}{e^{\boldsymbol{x_i'}\boldsymbol{\beta}} + 1}\right)^{r}$$

$$= Cp(r) \exp\left(-\frac{1}{2}\boldsymbol{\beta}'\Sigma^{-1}\boldsymbol{\beta}\right) \left(\frac{1}{\Gamma(r)}\right)^{n} \left(\prod_{i=1}^{n} \Gamma(y_i + r)\right) \left(\prod_{i=1}^{n} \frac{\left(e^{\boldsymbol{x_i'}\boldsymbol{\beta}}\right)^{y_i}}{\left(e^{\boldsymbol{x_i'}\boldsymbol{\beta}} + 1\right)^{r+y_i}}\right)$$

Notice that if we assign gamma prior for r, then its conditional posterior density is close to the form (1.7), except for the extra factor $\prod_{i=1}^{n} \left(e^{\boldsymbol{x_i'}\boldsymbol{\beta}}\right)^{y_i} / \left(e^{\boldsymbol{x_i'}\boldsymbol{\beta}} + 1\right)^{r+y_i}$. Similarly for β, the posterior is close to normal density, except for the same factor. This is however not an issue as we can write this factor as a scale mixture of normals, where the mixing distribution is Pólya-Gamma from Polson, Scott, and Windle (2013). Setting $a = y_i$ and $b = r + y_i$, the key mixture representation in Equation (1.6) related to it now becomes

$$\frac{\left(e^{\boldsymbol{x_i'}\boldsymbol{\beta}}\right)^{y_i}}{\left(e^{\boldsymbol{x_i'}\boldsymbol{\beta}} + 1\right)^{r+y_i}} \propto \int_0^{\infty} f(r, \boldsymbol{\beta} \mid \xi) \cdot p_{\mathrm{PG}}(\xi_i \mid r + y_i, 0) d\xi_i$$

$$f(r, \boldsymbol{\beta} \mid \xi) = \exp\left[\frac{1}{2}\left(-r(2\log 2 + \boldsymbol{x_i'}\boldsymbol{\beta}) + y_i(\boldsymbol{x_i'}\boldsymbol{\beta}) - \xi_i(\boldsymbol{x_i'}\boldsymbol{\beta})^2\right)\right].$$

The integrand $f(r, \boldsymbol{\beta} \mid \xi)$ is an exponential density when viewed as a function of r, and a normal density when viewed as a function of $\boldsymbol{\beta}$.

Finally, we introduce the auxiliary variables $\boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\xi}$ which follow gamma, ERG and PG distribution respectively. Let $\Xi := \mathrm{diag}(\xi_1, ..., \xi_n)$ and $\boldsymbol{z} := \left(\frac{y_1 - r}{2\xi_1}, ..., \frac{y_n - r}{2\xi_n}\right)'$ and $p(r) \sim$

$\Gamma(a_0, b_0)$. The conditional posteriors are derived as follows:

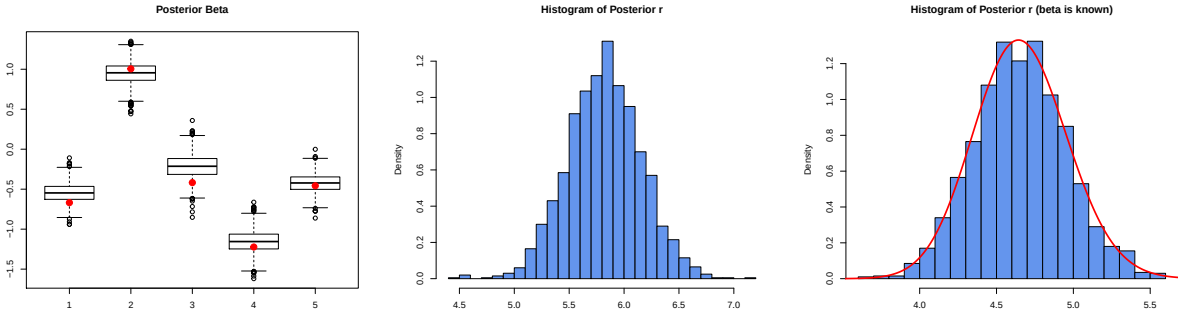
$$
\begin{aligned}
\tau_i \mid r, \boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\omega}, \boldsymbol{X}, \boldsymbol{y} &\sim \Gamma(y_i + r, 1) \\
\omega_i \mid r, \boldsymbol{\beta}, \boldsymbol{\tau}, \boldsymbol{\xi}, \boldsymbol{X}, \boldsymbol{y} &\overset{i.i.d.}{\sim} \mathrm{ERG}(r) \\
\xi_i \mid r, \boldsymbol{\beta}, \boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{X}, \boldsymbol{y} &\sim \mathrm{PG}(r + y_i, \boldsymbol{x_i'}\boldsymbol{\beta}) \\
\boldsymbol{\beta} \mid r, , \boldsymbol{\tau}, \boldsymbol{\xi}, \boldsymbol{\omega}, \boldsymbol{X}, \boldsymbol{y} &\sim N(\boldsymbol{m}, V) \\
r \mid \boldsymbol{\beta}, \boldsymbol{\tau}, \boldsymbol{\xi}, \boldsymbol{\omega}, \boldsymbol{X}, \boldsymbol{y} &\sim \mathrm{PTN}(a_0 + n, a, b + b_0)
\end{aligned}
$$

where

$$
\begin{aligned}
a &= \sum_{i=1}^{n} \omega_i \\
b &= n(\gamma - \log 2) + \sum_{i=1}^{n} \left(\log \tau_i - \boldsymbol{x_i'}\boldsymbol{\beta}/2 \right) \\
V &= \left(\Sigma^{-1} + X' \Xi X \right)^{-1} \\
m &= V X' \Omega z
\end{aligned}
$$

Figure 1.4 shows an illustrating simulation example where we set true $r = 5$ and generate $n = 100$ count observations. We draw the coefficient vector $\boldsymbol{\beta} \in \mathbb{R}^5$ and covariates $x_{ij} \overset{i.i.d.}{\sim} N(0, 0.5^2)$ for $1 \leq i \leq n$ and $1 \leq j \leq 5$. The prior for r is $p(r) \sim 1/r$ which is the limit case of gamma prior with $a_0 = 0, b_0 = 0$. For β, we choose $\Sigma = 10^6 I_5$ so that the prior information is relatively weak. The left panel shows the boxplots of posterior β samples, with those red dots denote the true β's. As the figure shows, all 5 β's fall in the 95% credible interval. The middle panel is the histogram of the posterior samples for r. Since the actual marginal posterior of r is hard to compute given the complicated form of the joint $p(\boldsymbol{\beta}, r \mid \boldsymbol{X}, y)$, in the right panel we use the same dataset, plug in the true β's, and run the Gibbs sampler again, so that we are able to compare the resulted histogram with the true density (red line).

Figure 1.4: Negative Binomial Regression Results



They are quite close to each other, indicating that the sampling procedure works well for this example.

1.4.3 Multinomial-Dirichlet Model

In this section, we develop the Markov chain Monte Carlo (MCMC) algorithm for fully posterior inference of the concentration parameter vector in the Dirichlet distribution. Such inference problems commonly arise in applied analyses of categorical data. Section 1.4.3 presents the general hierarchical multinomial-Dirichlet model class for which the ERG data augmentation scheme may be utilized. Section 1.4.3 develops a Gibbs sampler for inferring the concentration parameter in the Dirichlet distribution and conducts a simulation study comparing the performance of our data augmentation strategy with Metropolis-Hasting algorithm.

A Hierarchical Multinomial-Dirichlet Model

The multinomial-Dirichlet framework presented herein is closely related to the latent Dirichlet allocation model of Blei, Ng, and Jordan (2003) for topic modeling of text data, and we use text analysis as a motivating context. Suppose that for document $s \in \{1, ..., S\}$, each of N_s words in the document is independently allocated to K topics conditional on

probability vector $\boldsymbol{p}_s = (p_{s1}, p_{s2}, \ldots, p_{sK})$. For each document s, the number of words allocated to each topic is denoted by $\boldsymbol{n}_s = (n_{s1}, \cdots, n_{sK})$, which follows a multinomial distribution. The sampling model for the count vector $\boldsymbol{n}_s$ is then a multinomial distribution given probability vector $\boldsymbol{p}_s$,

$$\boldsymbol{n}_s \mid N_s, \boldsymbol{p}_s \sim \text{Multinomial}\left(N_s, \boldsymbol{p}_s\right).$$

The probability vector $\boldsymbol{p}_s$ is the proportional allocation of each document to the K topics. In a Bayesian analysis, the probability vector for each document $\boldsymbol{p}_s$ is typically assigned a Dirichlet distribution with concentration parameter vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$,

$$\boldsymbol{p}_s \mid \boldsymbol{\alpha} \sim \text{Dirichlet}\left(\boldsymbol{\alpha}\right).$$

Rather than fixing $\boldsymbol{\alpha} = \left(\frac{1}{K}, \ldots, \frac{1}{K}\right)$, as is common, we complete the model with a prior distribution $p(\boldsymbol{\alpha})$. This hierarchical prior distribution for $\boldsymbol{\alpha}$ facilitates more efficient information sharing across documents (observational units), and it yields practical advantages for out-of-sample prediction, which we discuss below. The model framework and ERG augmentation admit independent uniform, truncated normal, and exponential prior distributions for the elements α_k. Section 1.4.3 presents analyses based on independent gamma priors $p(\boldsymbol{\alpha}) = \prod_{k=1}^{K} p(\alpha_k)$.

In application, model inferences are often summarized by the posterior predictive distribution for the topic proportion vector $\boldsymbol{p}^*$ in a new document. Computing the posterior predictive distribution $p\left(\boldsymbol{p}^* \mid \boldsymbol{n}_1, ..., \boldsymbol{n}_S\right) = \int_{\boldsymbol{\alpha}} p\left(\boldsymbol{p}^* \mid \boldsymbol{\alpha}\right) p(\boldsymbol{\alpha} \mid \boldsymbol{n}_1, ..., \boldsymbol{n}_S) d\boldsymbol{\alpha}$ requires posterior computation of $p\left(\boldsymbol{\alpha} \mid \boldsymbol{n}_1, \ldots, \boldsymbol{n}_S\right) \propto p\left(\boldsymbol{\alpha}\right) \prod_{s=1}^{S} p\left(\boldsymbol{n}_s \mid \boldsymbol{\alpha}\right)$; however, when the probability vectors $\boldsymbol{p}_s$ are integrated out of the multinomial likelihood, the marginal likelihood $p\left(\boldsymbol{n}_s \mid \boldsymbol{\alpha}\right)$

includes elements of $\boldsymbol{\alpha}$ inside the gamma function,

$$\prod_{s=1}^{S} p\left(\boldsymbol{n}_s \mid \boldsymbol{\alpha}\right) = \prod_{s=1}^{S} \left(\frac{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}{\Gamma\left(\sum_{k=1}^{K} \left(n_{sk} + \alpha_k\right)\right)} \prod_{k=1}^{K} \frac{\Gamma\left(n_{sk} + \alpha_k\right)}{\Gamma\left(\alpha_k\right)} \right).$$

Because $\boldsymbol{\alpha}$ is nested inside the gamma function, computing $p(\boldsymbol{\alpha} \mid \boldsymbol{n}_1, \ldots, \boldsymbol{n}_S)$ is a challenge. Previous inference strategies relied on approximations, but in Section 1.4.3 we introduce a new data augmentation scheme for computing the full posterior $p\left(\boldsymbol{\alpha} \mid \boldsymbol{n}_1, \ldots, \boldsymbol{n}_S\right)$.

Data Augmentation and Simulation Study

Assume independent gamma prior distributions for each element of vector $\boldsymbol{\alpha}$ so that $p\left(\boldsymbol{\alpha}\right) \propto \prod_{k=1}^{K} \alpha_k^{a_0 - 1} e^{-b_0 \alpha_k}$, with hyperparamter a_0 and b_0. Note that gamma priors on each α_k give closed-form full conditional distributions in a Gibbs sampler, which is shown below. When $\alpha_k \sim \Gamma(a_0, b_0)$, where a_0 denotes the shape parameter and b_0 the rate, the expectation $E[\alpha_k] = a_0/b_0$. We can set $E[\alpha_k] = 1/K$, a standard choice for the Dirichlet concentration parameter, by choosing $a_0 = b_0/K$. The prior variance then depends on both the dimension of the Dirichlet distribution, K, and the rate parameter, b_0.

We now reorganize the multivariate posterior density of $\boldsymbol{\alpha}$ as

$$p(\boldsymbol{\alpha} \mid \boldsymbol{n}_1, \ldots, \boldsymbol{n}_S) = \left(\prod_{k=1}^{K} f(\alpha_k) \right) \left(\prod_{s=1}^{S} \frac{\Gamma(\sum_k \alpha_k)}{\Gamma(\sum_k \alpha_k + N_s)} \right) \tag{1.14}$$

where

$$f(\alpha_k) = \left(\prod_{s=1}^{S} \Gamma(\alpha_k + n_{sk}) \right) \left(\prod_{s=1}^{S} \frac{1}{\Gamma(\alpha_k)} \right) \alpha_k^{a_0 - 1} e^{-b_0 \alpha_k}. \tag{1.15}$$

Note that for each α_k, Equation (1.15) is exactly of the form (1.7). And the extra factor, $\prod_{s=1}^{S} \frac{\Gamma(\sum_k \alpha_k)}{\Gamma(\sum_k \alpha_k + N_s)}$, can be replaced with a beta integral representation. A multivariate

version of Proposition 2 produces the conditional posteriors as follows

$$\begin{aligned}
\tau_{sk} \mid \boldsymbol{\alpha}, \boldsymbol{\omega}, \boldsymbol{\eta} &\sim \Gamma(\alpha_k + n_{sk}), \\
\omega_{sk} \mid \boldsymbol{\alpha}, \boldsymbol{\tau}, \boldsymbol{\eta} &\sim \mathrm{ERG}(\alpha_k), \\
\eta_s \mid \boldsymbol{\alpha}, \boldsymbol{\tau}, \boldsymbol{\omega} &\sim \mathrm{Beta}\left(\sum_{k=1}^{K} \alpha_k, N_s\right), \\
\alpha_k \mid \boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\eta} &\sim \mathrm{PTN}(S + a_0, a_k, b_k)
\end{aligned}$$

where $a_k = \sum_{s=1}^{S} \omega_{sk}, b_k = S\gamma - b_0 + \sum_{s=1}^{S} \log(\tau_{sk} \cdot \eta_s)$. Conditioned on the introduced auxiliary variables $(\boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\eta})$, all elements of the vector $\boldsymbol{\alpha}$ are now mutually independent, which significantly reduces the difficulty of sampling procedure as we can now sample separately from K univariate distributions.

The total number of auxiliary variables to be sampled at each iterate is $S(1 + 2K)$, which may greatly slow down the MCMC algorithm for large values of S and K. However, we observe that these variables affect the distribution of α_k only through the parameters (a_k, b_k). The summation form of (a_k, b_k) suggests that we may apply central limit theorem and approximate them by normal variables. Therefore, as mentioned in Section 1.3, the Gibbs sampling algorithm can be approximately simplified to

1. Initialize $a_k^{(0)}, b_k^{(0)}$ for $1 \leq k \leq K$.

2. At step t, sample $a_k^{(t)}$ and $b_k^{(t)}$ from $N(\mu_{a,k}, \sigma_{a,k}^2)$ and $N(\mu_{b,k}, \sigma_{b,k}^2)$ respectively, where

$$\begin{aligned}
\mu_{a,k} &= \frac{S}{2\alpha_k}\left(\psi(1+\alpha_k)+\gamma\right) \\
\sigma_{a,k}^2 &= \frac{\mu_{a,k}}{2\alpha_k^2} - \frac{S}{4\alpha_k^2}\psi'(1+\alpha_k) \\
\mu_{b,k} &= S\gamma - b_0 + S\psi(\alpha_0) + \sum_{s=1}^{S}\psi(\alpha_k+n_{sk}) - \psi(\alpha_0+N_s) \\
\sigma_{b,k}^2 &= S\psi'(\alpha_0) + \sum_{s=1}^{S}\psi'(\alpha_k+n_{sk}) - \psi'(\alpha_0+N_s)
\end{aligned}$$

   and $\alpha_0 = \sum_{k=1}^{K}\alpha_k$.

3. Sample $\alpha_k^{(t)}$ from PTN$(S+a_0, a_k^{(t)}, b_k^{(t)})$ for $1 \le k \le K$. Increase t by 1 and return to (2).

Fixing the dataset dimensions S and K, we consider two settings of the true $\boldsymbol{\alpha}$ for our simulation experiment: (A) heterogeneous $\alpha_k = k/K$ and (B) homogeneous $\alpha_k = 1/K$ for $k = 1, 2, ..., K$. Probability vector $\boldsymbol{p}_s$ are drawn independently from Dirichlet($\boldsymbol{\alpha}$) and vector of counts $\boldsymbol{n}_s$ are drawn from Mutinomial $(N_s, \boldsymbol{p}_s)$ with $N_s = 500$ for $s = 1, 2, ..., S$. We set the hyperparamters $a_0 = b_0/K$ in independent gamma priors of α_k's, so that the prior expectations are all equal to $1/K$. The setting of our simulation experiment is summarized below:

- $S \in \{100, 1000\}, K \in \{10, 50\}$ and $b_0 \in \{0.1, 1, 5\}$.
- 4 MCMC algorithms in comparison:
  - DA: Gibbs sampler which iteratively samples $\boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\eta}$ and $\boldsymbol{\alpha}$.
  - DA-N: replace the parameters in the conditional posterior of $\boldsymbol{\alpha}$ with approximating normal variables and skip the sampling of $\boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\eta}$ in DA.

- DA-E: replace the parameters in the conditional posterior of $\boldsymbol{\alpha}$ with corresponding expectations and skip the sampling of $\boldsymbol{\tau}, \boldsymbol{\omega}, \boldsymbol{\eta}$ in DA.
- MH: random-walk Metropolis-Hasting sampler.

- Metrics:
  - Root Mean Square Error:

$$\text{RMSE} = \sqrt{\frac{1}{TK}\sum_{t=1}^{T}\|\boldsymbol{\alpha}^{(t)} - \boldsymbol{\alpha}\|^2}.$$

    $T = 500$ is the posterior sample size.
  - Effective sample size ratio (ESSR):

$$\text{ESSR} = \frac{1}{K}\sum_{k=1}^{K}\frac{\lambda_k^2}{\sigma_k^2},$$

    which measures the serial correlation between posterior samples. λ^2 is the sample variance and σ^2 is the estimate of the spectral density at frequency zero.
  - Algorithm running time in seconds

In simulation experiments, we initialize each α_k with a random draw from Lognormal with parameters $\mu = 0, \sigma^2 = 1/K$ and multiply by $1/K$, so that the prior median is $1/K$. For each combination of (b_0, S, K), we run the simulation for 50 times (the first 200 samples are dropped each time). Table 1.4 and 1.5 show the results, averaged over 50 runs, for the homogeneous and heterogeneous setting respectively. Despite being slow, our data augmentation strategy with ERG auxiliary variables produces more accurate estimates of $\boldsymbol{\alpha}$ than Metropolis-Hasting does, in heterogeneous setting. While in homogeneous setting, the two methods have similar RMSE. Metropolis-Hasting gets significantly worse when K grows to 50. This is not surprising at all. As we notice that, the effective sample size ratios

for Metropolis-Hasting are as low as 0.01, indicating strong serial correlations in posterior samples drawn by Metropolis-Hasting, of which the acceptance rate is around 0.02. Therefore, the RMSE for Metropolis-Hasting is entirely up to the distance between the initial $\boldsymbol{\alpha}_{(0)}$ and the true $\boldsymbol{\alpha}$. Our Gibbs sampler instead enjoys the advantage of being less sensitive to initialization and not requiring further tuning. The different choices of b_0 don't seem to affect the posteriors too much. Once replacing auxiliary variables with normal approximations or expectations, the Gibbs sampler gets much faster. Meanwhile, root mean squared errors and effective sample sizes get slightly improved. With DA-N and DA-E, we can expect the resulted posterior samples to have slightly less variation as well as less correlation.

Table 1.4: Homogeneous Setting

		$b_0 = 0.1$				$b_0 = 1$				$b_0 = 5$			
		Root Mean Square Error ($\times 10^3$)											
S	K	DA	DA-N	DA-E	MH	DA	DA-N	DA-E	MH	DA	DA-N	DA-E	MH
100	10	20.93	20.95	18.95	20.56	20.25	20.36	18.31	19.99	20.22	20.25	18.38	20.19
100	50	7.78	8.03	6.93	5.88	7.71	7.95	6.86	6.03	7.66	7.95	6.84	5.95
1000	10	6.54	6.53	5.95	7.13	6.58	6.54	6.01	7.26	6.38	6.36	5.80	6.87
1000	50	2.49	2.51	2.19	2.48	2.49	2.51	2.20	2.50	2.48	2.50	2.18	2.46
		Effective Sample Size Ratio											
S	K	DA	DA-N	DA-E	MH	DA	DA-N	DA-E	MH	DA	DA-N	DA-E	MH
100	10	0.32	0.33	0.32	0.04	0.32	0.32	0.32	0.04	0.32	0.32	0.32	0.04
100	50	0.08	0.08	0.07	0.01	0.08	0.08	0.07	0.01	0.08	0.08	0.07	0.01
1000	10	0.32	0.32	0.32	0.01	0.33	0.33	0.32	0.01	0.32	0.32	0.32	0.01
1000	50	0.08	0.08	0.07	0.01	0.08	0.07	0.07	0.01	0.08	0.08	0.07	0.01
		Running Time											
S	K	DA	DA-N	DA-E	MH	DA	DA-N	DA-E	MH	DA	DA-N	DA-E	MH
100	10	32.76	0.63	0.22	0.93	31.52	0.61	0.22	0.89	33.37	0.62	0.23	0.89
100	50	174.29	3.61	1.12	2.51	166.49	3.43	1.08	2.43	165.67	3.43	1.07	2.45
1000	10	236.33	5.13	1.31	9.47	206.65	4.59	1.21	8.70	204.72	4.62	1.22	8.79
1000	50	1099.74	30.67	6.18	26.05	1012.35	28.12	5.77	23.99	1014.13	28.14	5.75	23.90

Table 1.5: Heterogeneous Setting

		$b_0 = 0.1$				$b_0 = 1$				$b_0 = 5$			
		Root Mean Square Error ($\times 10^3$)											
S	K	DA	DA-N	DA-E	MH	DA	DA-N	DA-E	MH	DA	DA-N	DA-E	MH
100	10	82.76	82.25	76.54	82.34	82.56	81.90	77.33	83.49	81.95	82.11	78.46	86.96
100	50	81.50	81.41	75.83	553.05	78.53	78.39	73.47	553.83	81.99	81.98	79.15	553.38
1000	10	26.11	26.00	24.50	41.00	25.76	25.61	24.18	41.08	25.88	25.71	24.25	41.39
1000	50	24.80	24.79	23.19	553.35	25.22	25.11	23.66	553.30	25.26	25.21	23.71	552.86
		Effective Sample Size Ratio											
S	K	DA	DA-N	DA-E	MH	DA	DA-N	DA-E	MH	DA	DA-N	DA-E	MH
100	10	0.47	0.49	0.50	0.03	0.48	0.51	0.49	0.03	0.49	0.51	0.51	0.03
100	50	0.44	0.45	0.44	0.01	0.44	0.45	0.44	0.01	0.45	0.46	0.46	0.01
1000	10	0.48	0.50	0.50	0.02	0.47	0.50	0.50	0.01	0.47	0.50	0.50	0.01
1000	50	0.44	0.44	0.44	0.01	0.44	0.45	0.45	0.01	0.44	0.44	0.45	0.01
		Running Time											
S	K	DA	DA-N	DA-E	MH	DA	DA-N	DA-E	MH	DA	DA-N	DA-E	MH
100	10	42.28	0.58	0.26	0.93	32.97	0.49	0.23	0.85	33.36	0.49	0.24	0.86
100	50	186.00	3.07	1.23	2.48	164.91	2.75	1.11	2.25	164.60	2.75	1.11	2.26
1000	10	230.54	3.69	1.27	8.92	205.64	3.33	1.18	8.41	204.56	3.34	1.18	8.51
1000	50	1104.85	22.43	6.25	24.09	1014.25	21.06	5.85	22.41	1013.74	21.06	5.86	22.46

1.5 Discussion

The class of Exponential Reciprocal Gamma (ERG) distributions are developed as mixing distributions for models with Gamma functions. This adds to the literature on normal variance-mean mixtures by showing that they extend to a wide class of applications. Our ensuing data augmentation strategy facilitates full posterior inference for parameters in models which were hitherto hard to provide inference. The algorithms are scalable and are a fast efficient simulation method for drawing from posterior distributions with applications to many area, such as non-parametric Bayes, latent Dirichlet allocation, Gamma-Gamma hierarchical models, extreme value models, and many other Bayesian mixture models.

The focus of our paper is on theoretical and algorithmic development of ERG auxiliary variables. Our work contributes to the literature on scale mixtures of normals (see, e.g., (Andrews and Mallows, 1974; West, 1987; Polson, Scott, and Windle, 2013)). We believe that the computational strategies developed here will provide the foundation for new and

richly structured hierarchical gamma models. Applied Bayesian analyses of categorical data will benefit from increased model flexibility and information borrowing strategies.

There are a number of avenues for future research. In particular, regularized scale allocation models can be implemented using data augmentation methods of Polson and Scott (2013) with ERG distribution. Barndorff-Nielsen, Blaesild, and Seshadri (1992) provide multivariate GIG distribution theory and relationships with Poisson processes.

1.6 Appendix

1.6.1 Proof of Theorem 1

The generalized inverse Gaussian distribution, GIG(p, a, b), has probability density function

$$p\left(x\right) \propto x^{p-1} \exp\left\{-\frac{1}{2}\left(ax + b/x\right)\right\}, \quad a, b, x > 0, p \in \mathbb{R}.$$

It suffices to show that if $G_k \sim GIG\left(-\frac{3}{2}, 2c^2, \frac{1}{2k^2}\right)$, then the following integral identity holds,

$$E(e^{-s^2 G_k}) = \left(\frac{k + \sqrt{s^2 + c^2}}{k + c}\right) e^{-\frac{\sqrt{s^2+c^2}-c}{k}}.$$

The density of G_k given by

$$p_{k,c}(x) = m\left(k, c\right) x^{-\frac{5}{2}} \exp\left(-\frac{1}{4k^2 x} - c^2 x\right).$$

with normalizing constant,

$$m(k, c) = \frac{1}{\Gamma\left(\frac{3}{2}\right)} \frac{(2k)^{-3}}{ck^{-1} + 1} e^{ck^{-1}}.$$

It follows by the algebraic calculation,

$$\begin{aligned}\int_0^\infty e^{-t^2x} p_{k,c}(x)dx &= m(k,c)\int_0^\infty x^{-\frac{5}{2}} \exp\left(-\frac{1}{4k^2}x^{-1} - (t^2+c^2)x\right) dx \\ &= \frac{m(k,c)}{m\left(k, \sqrt{t^2+c^2}\right)} \\ &= \frac{(\sqrt{t^2+c^2}k^{-1}+1)\exp\left(\sqrt{t^2+c^2}k^{-1}\right)}{(ck^{-1}+1)\exp(ck^{-1})} \\ &= \left(\frac{k+\sqrt{t^2+c^2}}{k+c}\right) e^{-\frac{\sqrt{t^2+c^2}-c}{k}},\end{aligned}$$

as required.

1.6.2 Simulating ERG Random Variables

We consider below 3 different ways to generate independent random variables from ERG distribution.

(a) Since Theorem 1, we can approximate an ERG(c) with the finite sum

$$X_N = \sum_{k=1}^{N-1} \text{GIG}\left(-\frac{3}{2}, 2c^2, \frac{1}{2k^2}\right) + \Gamma(a_N, b_N) \tag{1.16}$$

where the gamma random variable $\Gamma(a_N, b_N)$ are to approximate the tail part by matching the first two moments given in Theorem 2. The shape and rate parameter are

$$\begin{aligned}b_N &= \frac{2c^2\left(\psi(N+c) - \psi(N)\right)}{\psi(N+c) - \psi(N) - c\psi'(N+c)} \\ a_N &= \frac{b_N}{2c}\left(\psi(N+c) - \psi(N)\right)\end{aligned}$$

(b) Since the generation of i.i.d. inverse gamma variables is faster than that of GIG

variables, for small values of c, we may consider rejection sampling with ERG(0) as the proposal density.

1. Generate a sample W from ERG(0) using the method in (a) and U from Unif[0,1].
2. If $U < e^{-c^2 W}$, accept W as a sample drawn from ERG(c). Otherwise, reject W and return to the sampling step.

(c) McLeish (2014) shows that one can simulate random variables using the saddlepoint approximation, given the cumulant generating function $k(t) = \log E(e^{tW})$ is known. The saddlepoint approximation is

$$\begin{aligned} P(W \leq x) &\approx \Phi(w) + \phi(w)\left(\frac{1}{w} - \frac{1}{u}\right) \\ w &= w(t) = \operatorname{sgn}(t)\sqrt{2(tk'(t) - k(t))} \\ u &= u(t) = t\sqrt{k''(t)} \end{aligned}$$

where t solves $k'(t) = x$. $\Phi(\cdot)$ and $\phi(\cdot)$ are the cdf and density of standard normal distribution. We can first generate a random variable T using inverse transform method, such that its cdf $F(t) = \Phi(w(t)) + \phi(w(t))\left(\frac{1}{w(t)} - \frac{1}{u(t)}\right)$, then $W = k'(T)$ has cdf given by the saddlepoint approximation above.

1. Genearte a sample U from Unif[0,1].

2. Solve $U = F(t)$ using Newton-Raphson iteration

$$
\begin{aligned}
t_{n+1} &= t_n - \frac{F(t_n) - U}{\sqrt{k''(t_n)}\phi(w(t_n))} \\
k(t) &= -\gamma\sqrt{c^2 - t} - \ln\Gamma(1 + \sqrt{c^2 - t}) + (\gamma c + \ln\Gamma(1 + c)) \\
k'(t) &= \frac{\gamma + \psi(1 + \sqrt{c^2 - t})}{2\sqrt{c^2 - t}} \\
k''(t) &= \frac{\gamma + \psi(1 + \sqrt{c^2 - t}) - \sqrt{c^2 - t}\psi'(1 + \sqrt{c^2 - t})}{4(c^2 - t)^{3/2}}
\end{aligned}
$$

3. $W = k'(t)$.

1.6.3 Simulating PTN Random Variables

The probability density of a random variable PTN(p, a, b) is given as

$$
p(x \mid p, a, b) = \frac{x^{p-1}e^{-ax^2+bx}}{\int_0^\infty x^{p-1}e^{-ax^2+bx}dx}, \quad (x, p, a > 0, b \neq 0).
$$

Note that we can write it as a multiplication

$$
\begin{aligned}
p(x \mid p, a, b) &= \frac{x^{p-1}e^{-ax^2+bx}}{\int_0^\infty x^{p-1}e^{-ax^2+bx}dx} \\
&= \frac{x^{p-1}e^{-(\tau|b|-b)x}e^{-a(x-\tau|b|/2a)^2}}{\int_0^\infty x^{p-1}e^{-(\tau|b|-b)x}e^{-a(x-\tau|b|/2a)^2}dx} \\
&= ce^{-a(x-\tau|b|/2a)^2}g(x \mid p, \tau|b| - b)
\end{aligned}
$$

where $g(x|p, b)$ is the density of a gamma random variable with shape p and rate $\tau|b| - b > 0$.

$$
0 \leq e^{-a(x-\tau|b|/2a)^2} \leq 1,\ c = \frac{\int_0^\infty x^{p-1}e^{-(\tau|b|-b)x}dx}{\int_0^\infty x^{p-1}e^{-(\tau|b|-b)x}e^{-a(x-\tau|b|/2a)^2}dx} \geq 1
$$

We sample from $p(x \mid p, a, b)$ by rejection method:

1. Generate $X \sim \Gamma(p, \tau|b| - b)$ and $U \sim Unif[0,1]$

2. Return X until $U \leq e^{-a(X-\tau|b|/2a)^2}$.

If $b > 0$, we set $\tau = \sqrt{1/4 + 2ap/b^2} + 1/2$; if $b < 0$, we set $\tau = \sqrt{1/4 + 2ap/b^2} - 1/2$. Then $e^{-a(E(X)-\tau|b|/2a)^2} = 1$.

1.6.4 Approximating Gibbs Sampler

If $g_\ell(x) = g(x)$ for all ℓ (similarly for h and j), and L, M, N are all very large, then an approximate Gibbs sampler based on the conditional posterior in Proposition 2 is

1. Initialize $\tilde{a}^{(0)}, \tilde{b}^{(0)}$.

2. At step t, sample $(\tilde{a}^{(t)}, \tilde{b}^{(t)})$ from $N(\mu_a(t), \sigma_a^2(t))$ and $N(\mu_b(t), \sigma_b^2(t))$ respectively, where

$$\begin{aligned}
\mu_a(t) &= a + \frac{M(h'(0))^2}{2h(x^{(t-1)})}\left(\psi(1 + h(x^{(t-1)})) + \gamma\right) \\
\mu_b(t) &= b + Lg'(0)\psi(g(x^{(t-1)})) + M\gamma h'(0) - 2(\mu_a(t) - a) + \\
&\quad Nj'(0)\left(\psi(j(x^{(t-1)})) - \psi(j(x^{(t-1)}) + \beta_n)\right) \\
\sigma_a^2(t) &= \frac{M(h'(0))^4}{4(h(x^{(t-1)}))^3}\left(\psi(1 + h(x^{(t-1)})) + \gamma - h(x^{(t-1)})\psi'(1 + h(x^{(t-1)}))\right) \\
\sigma_b^2(t) &= L(g'(0))^2\psi'(g(x^{(t-1)})) + 4(h'(0))^2\sigma_a^2(t) \\
&\quad + N(j'(0))^2\left(\psi'(j(x^{(t-1)})) - \psi'(j(x^{(t-1)}) + \beta_n)\right)
\end{aligned}$$

3. Sample $x^{(t)}$ from $\sum_{m=0}^{M} \pi_k \cdot \text{PTN}(p + m, \tilde{a}^{(t)}, \tilde{b}^{(t)})$.

CHAPTER 2

WEIGHTED BAYESIAN BOOTSTRAP FOR SCALABLE POSTERIOR DISTRIBUTIONS

We introduce and develop a weighted Bayesian bootstrap (WBB) for machine learning and statistics. WBB provides uncertainty quantification by sampling from a high dimensional posterior distribution. WBB is computationally fast and scalable using only off-the-shelf optimization software. First-order asymptotic analysis provides a theoretical justification under suitable regularity conditions on the statistical model. We illustrate the proposed methodology in regularized regression, trend filtering and deep learning and conclude with directions for future research.

Key Words: Deep learning, Markov chain Monte Carlo, Regularization, Trend Filtering, Weighted Bootstrap.

2.1 Introduction

Weighted Bayesian Bootstrap (WBB) is a simulation-based algorithm for assessing uncertainty in Machine Learning and Statistics. Uncertainty quantification (UQ) is an active area of research, particularly in high-dimensional inference problems (Wang & Swiler, 2018). Whilst there are computationally fast and scalable algorithms for training models in a wide variety of contexts, uncertainty assessments are still required, as are methods to compute these assessments. Bayesian analysis offers a general solution, but developing computationally fast scalable algorithms for sampling a posterior distribution is a notoriously hard problem. WBB makes a contribution to this literature by showing how off-the-shelf optimization algorithms, such as convex optimization or stochastic gradient descent (SGD), can be adapted to provide uncertainty assessments. Our goal here is to marry Bayesian uncertainty techniques with state-of-the-art optimization methods and software systems.

For relatively simple statistical models, the weighted likelihood bootstrap (WLB) method provides approximate posterior sampling through repeated optimization of a randoml weight likelihood function (Newton & Raftery, 1994). The proposed WBB extends the WLB to a broad class of contemporary statistical models by leveraging advances in optimization methodology. Essentially, the WLB used optimization of certain randomized objective functions to enable approximate marginalization (i.e., integration) required in Bayesian analysis. The same idea – optimize a randomized objective function to achieve posterior sampling – is at the heart of the proposed WBB method, though some changes to the WLB procedure are required to carry out this program for the models considered. Theoretical support for the WLB approximation is based on connections between posterior variation and curvature of the log-likelihood revealed through repeated optimization of randomly weighted likelihoods. By contrast, the proposed WBB calculates a series of randomized posterior modes rather than randomized likelihood maximizers. A key rationale for this proposal is that high dimensional posterior modes are now readily computable, thanks to systems such as

`TensorFlow` (Abadi et al., 2015) and `Keras` (Chollet, 2015) that deploy stochastic gradient descent (SGD) and convex optimization methods for large-scale problems, such as on neural network architectures used in deep learning (LeCun, Bengio, & Hinton, 2015). By linking random weighting with advanced optimization, we expose a simple scheme for approximate uncertainty quantification in a wide class of statistical models.

Quantifying uncertainty is typically unavailable in a purely regularization optimization method. We contend that UQ is available directly by repeated optimization of randomized objective functions, using the same computational tools that produce the primary estimate, rather than through Markov chain Monte Carlo, variational methods, approximate Bayesian computation, or other techniques. See Green et al. (2015) for a good summary of Bayesian computation history. Thus, uncertainty assessments are provided at little extra effort over the original training computations. A further benefit is that with extra computational cost, it is straightforward to add a regularization path across hyper-parameters (e.g., simply repeat WBB on different λ), which is usually difficult to compute in traditional Bayesian sensitivity analysis. We use predictive cross-validation techniques in this regard.

The rest of the paper is outlined as follows. Section 2 develops our weighted Bayesian Bootstrap (WBB) algorithm. Section 3 provides applications to high dimensional sparse regression, trend filtering and deep learning. WBB can also be applied to Bayesian tree models (Taddy et al., 2015). Section 4 indicates several directions for future research, including bootstrap filters in state-space models (Gordon, Salmond, & Smith, 1993) and connections to the resampling-sampling perspective in sequential Bayesian inference (Lopes, Polson, & Carvalho, 2012).

2.2 Weighted Bayesian Bootstrap

2.2.1 Setting

We work with a broad class of statistical models for data structures involving outcomes and covariates. Examples considered in Section 3 include regression, trend-filtering, and deep learning. Let $y = (y_1, y_2, \cdots, y_n)$ be an n-vector of outcomes and let $\theta = (\theta_1, \theta_2, ..., \theta_p)$ be a p-dimensional parameter of interest. Covariate data may be organized in an $n \times p$ matrix A whose rows are the design points (or "features") a_i^T where we index observations by i and parameters by j. A large number of estimation/training problems can be expressed in the form

$$\underset{\theta \in \mathcal{R}^d}{\text{minimize}} \quad \mathcal{L}(\theta) := l(y|\theta) + \lambda\phi(\theta) \tag{2.1}$$

where $l(y|\theta) = \sum_{i=1}^n l_i(y_i|\theta)$ is a measure of fit (or "empirical risk function") depending on θ and y and implicitly on A. The penalty function, or regularization term, $\lambda\phi(\theta)$, may encode soft or hard constraints, and is controlled by a hyper-parameter, $\lambda > 0$, whose values index an entire path of solutions. The penalty function $\phi(\theta)$ effects a favorable bias-variance estimation tradeoff and provides extensive modeling flexibility (Wellner & Zhang, 2012). To accommodate contemporary applications we allow $\phi(\theta)$ to have points in its domain where it fails to be differentiable (e.g., L^1 norm). If we treat data, y, as arising from a probabilistic model parameterized by θ, then the likelihood function $p(y|\theta)$ yields the model-associated measure of fit $l(y|\theta) = -\log p(y|\theta)$. The maximum likelihood estimator (MLE) is $\hat{\theta} := \text{argmax}_\theta\ p(y|\theta)$, though of course this usually differs from the solution to Equation (2.1): $\theta^* := \arg\min\{l(y|\theta) + \lambda\phi(\theta)\}]$. We recall a key duality between regularization and Bayesian analysis.

2.2.2 Bayesian Regularization Duality

From the Bayesian perspective, the measure of fit, $l(y|\theta) = -\log p(y|\theta)$, and the penalty function, $\lambda\phi(\theta)$, correspond to the negative logarithms of the likelihood and prior distribution in the model

$$\begin{aligned} p(y|\theta) &\propto \exp\{-l(y|\theta)\}\,, \\ p(\theta) &\propto \exp\{-\lambda\phi(\theta)\}, \\ p(\theta|y) &\propto \exp\{-(l(y|\theta) + \lambda\phi(\theta))\}. \end{aligned} \tag{2.2}$$

This posterior $p(\theta|y)$ is often a proper distribution over $\mathcal{R}^d$, even if the prior $p(\theta)$ is not proper. The well-known equivalence between regularization and Bayesian methods is seen, for example, in regression with a Gaussian regression model subject to a penalty such as an L^2-norm (ridge) Gaussian prior or L^1-norm (LASSO) double exponential prior. By this duality, the posterior mode, or maximum a posteriori (MAP) estimate, is θ^*, a solution to Equation (2.1). See Gribonval & Machart (2013) for a nuanced view of the connection between Equation (2.1) and Equation (2.2) in Gaussian regression models.

2.2.3 Optimization

Advances in optimization methodology provide efficient algorithms to compute $\theta^* = \arg\min \mathcal{L}(\theta)$ for a wide range of loss and penalty functions. Theory is well developed in the case of convex objective functions (e.g., Bertsekas, Nedi, & Ozdaglar, 2003; Boyd & Vandenberghe, 2004). For example if loss l is convex and differentiable in θ and penalty ϕ is convex, then a necessary and sufficient condition for θ^* to minimize $l(y|\theta) + \lambda\phi(\theta)$ is

$$0 \in \partial\left\{l(y|\theta^*) + \lambda\phi(\theta^*)\right\} = \nabla l(y|\theta^*) + \lambda\partial\phi(\theta^*) \tag{2.3}$$

where ∂ is the subdifferential operator (the set of subgradients of the objective), in this case the sum of a point and a set. Though not a formula for θ^*, such as given by the normal equations in linear regression, Equation (2.3) usefully guides algorithms that aim to solve θ^*. For example, under separability conditions on the penalty function, coordinate descent algorithms effectively solve for θ^*; see Wright (2015), or Hastie, Tibshirani, & Wainwright (2015, chap. 5) for a statistical perspective. The optimization literature also characterizes θ^* as the fixed point of a proximal operator $\text{prox}_{\gamma\mathcal{L}}(\theta) = \arg\min_z\{\mathcal{L}(z) - \frac{1}{2\gamma}||z-\theta||_2^2\}$, which opens the door to powerful MM algorithms and related schemes; see Lange (2016, chap. 5), Polson & Scott (2015), and Polson, Scott, & Willard (2015). Beyond convexity, the guarantees are weaker (e.g., local not global minima) and the algorithms are many (e.g., Nocedal & Wright, 2006). Gradient descent or stochastic gradient descent (SGD) are effective in many cases, owing to parameter dimensionality and structure of the gradients. The appendix develops SGD for one example.

Advances in applied optimization provide effective software tools for data analysis. For example, the R package `glmnet` deploys coordinate descent for loss functions arising from generalized linear models and LASSO or elastic net penalties (Friedman, Hastie, & Tibshirani, 2010). To solve the generalized LASSO problem, the R package `genlasso` deploys a dual path algorithm (Arnold & Tibshirani, 2014). A variety of general purpose optimization tools for statistics are compiled at the optimization view at CRAN (`https://cran.r-project.org`). For machine learning, the `TensorFlow` system has greatly simplified gradient descent, SGD, and related algorithms for many applications (Abadi et al., 2015).

2.2.4 WBB Algorithm

We now define the weighted Bayesian bootstrap (WBB). Recalling the original objective function Equation (2.1), we form the randomized objective

$$\mathcal{L}_{\mathbf{w}}(\theta) = \left\{ \sum_{i=1}^{n} w_i l_i(y_i|\theta) \right\} + \lambda w_0 \phi(\theta) \tag{2.4}$$

where entries of $\mathbf{w} = (w_0, w_1, \cdots, w_n)$ are independent and identically distributed (i.i.d.) standard exponentially distributed random weights, generated by the analyst and independently from the data y. Equivalently, $w_i = \log(1/u_i)$ where u_i's are i.i.d. Uniform$(0, 1)$. When p is relatively large and $\phi(\theta)$ is separable as $\phi(\theta) = \sum_{j=1}^{p} \phi_j(\theta_j)$, we recommend an extension in which $w_0 = (w_{0,1}, w_{0,2}, \cdots, w_{0,p})$ allows separate i.i.d. random weights on the prior regularization terms $\phi_j(\theta_j)$, namely $\lambda \sum_{j=1}^{p} w_{0,j}\phi_j(\theta_j)$, to help with sparsity and to reduce the effect of occasionally large scalar weight. In either case, associated with any vector $\mathbf{w}$ is the solution, $\theta^*_{\mathbf{w}} = \arg\min \mathcal{L}_{\mathbf{w}}(\theta)$. Our basic conjecture is that the conditional distribution of $\theta^*_{\mathbf{w}}$ – the distribution induced by $\mathbf{w}$ with the data fixed – approximates the Bayesian posterior Equation (2.2). For any measurable set $\mathcal{B}$ in the parameter space,

$$\Pr\left(\theta^*_{\mathbf{w}} \in \mathcal{B}|y\right) \approx \int_{\mathcal{B}} p(\theta|y)\, d\theta. \tag{2.5}$$

Section 2.5 provides an asymptotic argument in support of Equation (2.5), and we investigate the approximation numerically in a few examples in Section 3. Assuming this conjecture is true, we have a straightforward optimization-based algorithm for approximate posterior sampling:

Algorithm 1 Weighted Bayesian Bootstrap

Input:

data: $\mathcal{D} = (y, A)$

model structure: $\mathcal{M} = (\{l_i\}, \lambda, \phi)$

number of draws: T

Output: T parameter samples $\{\theta^{*,t}\}$

Function **WBB**$(\mathcal{D}, \mathcal{M}, T)$:

for all $t = 1$ to T **do**

Realize: $(u_1, \cdots, u_n) \sim_{i.i.d.} \text{Uniform}(0, 1)$

Construct: $w_i \leftarrow \log(1/u_i), \forall i.$

Independently construct w_0 as either a univariate Exponential (common weight case) or as a vector of i.i.d. Exponentials (separate weights case)

Set $\mathbf{w} = (w_0, w_1, \cdots, w_n)$

Compute: $\theta^{*,t} \leftarrow \arg\min \mathcal{L}_{\mathbf{w}}(\theta)$

end for

When optimization on the original problem Equation (2.1) is fast and scalable, so too is the WBB. Weights $\mathbf{w}$ and the corresponding $\theta^*_{\mathbf{w}}$ are independent across t, making it possible to speed up the algorithm via parallel computing. To choose the amount of regularization λ which is assumed to be fixed for all sets of $\mathbf{w}$, we can use the marginal likelihood $m_\lambda(y)$, estimated by bridge sampling (Gelman & Meng, 1998) or simply using predictive cross-validation. Next we consider the approximation Equation (2.5) from an asymptotic perspective.

2.2.5 WBB Properties

Conditions under which the target posterior distribution Equation (2.2) is approximately Gaussian are well established (Kleijn & van der Vaart, 2012). For example, when data form a random sample from fixed distribution $p(y_i|\theta_0)$ that resides in a sufficiently regular model, and when the prior is smooth and positive around $\theta_0 \in \mathcal{R}^p$, we have

$$\theta|\, y \sim_{\text{approx}} N_p\left(\theta_n^*, J_n^{-1}(\theta_n^*)\right), \tag{2.6}$$

where, including sample size n as an explicit subscript, we have posterior mode $\theta_n^* = \arg\min \mathcal{L}_n(\theta)$, and where $J_n(\theta_n^*) = nj(\theta_n^*)$ is the Fisher information matrix evaluated at θ_n^*. Here $j(\theta)$ is the information per sample, and $y = (y_1, \cdots, y_n)$ denotes data. Johnstone (2010) studies Bernstein-von Mises theorem in high dimensional settings where p grows with n and the situation is very different. Centering on the posterior mode, rather than the MLE, improves accuracy in many cases (Bertail & Lo, 1991).

As to the WBB distribution, consider a one-term Taylor expansion of $\nabla\mathcal{L}_{\mathbf{w},n}(\theta)$ about the posterior mode θ_n^*, which is allowable for sufficiently smooth loss and penalty terms:

$$\nabla\mathcal{L}_{\mathbf{w},n}(\theta) = \nabla\mathcal{L}_{\mathbf{w},n}(\theta_n^*) + \nabla^2\mathcal{L}_{\mathbf{w},n}(\theta_n^*)(\theta - \theta_n^*) + R_n \tag{2.7}$$

where R_n is an error term and ∇ and ∇^2 record the gradient vector and matrix of second partial derivatives, respectively, of the weighted objective function. Evaluating this expansion at $\theta_{\mathbf{w},n}^* = \arg\min \mathcal{L}_{\mathbf{w},n}(\theta)$ zeros out the left hand side of Equation (2.7) and leads to:

$$\sqrt{n}\left(\theta_{\mathbf{w},n}^* - \theta_n^*\right) = -\left(\frac{1}{n}\nabla^2\mathcal{L}_{\mathbf{w},n}(\theta_n^*)\right)^{-1}\left(\frac{1}{\sqrt{n}}\nabla\mathcal{L}_{\mathbf{w},n}(\theta_n^*)\right) + \tilde{R}_n \tag{2.8}$$

where $\tilde{R}_n$ is another error term. Following Newton & Raftery (1994), we recognize that the

$\mathbf{w}$-induced variation in Equation (2.8), conditional upon the data, causes the matrix factor to be approximately the inverse information $[J_n(\theta_n^*)]^{-1}$, the score-like second factor to be approximately mean-zero Gaussian with covariance equal to $J_n(\theta_n^*)$, and the error $\tilde{R}_n$ to be negligible. Thus, compared to the target posterior variation Equation (2.6), we have WBB variation:

$$\theta^*_{\mathbf{w},n} \big| \, y \sim_{\text{approx}} N_p \left(\theta_n^*, J_n^{-1}(\theta_n^*) \right).$$

In a relatively narrow asymptotic sense, therefore, the WBB procedure is approximating the target posterior distribution, as both are approximately Gaussian with the same mean and covariance. Details of the asymptotic analysis follow the WLB case presented in Newton & Raftery (1994), and differ only slightly in our use of the posterior mode θ_n^* in place of the maximum likelihood estimator $\hat{\theta}_n$, and also in our incorporation of weight w_0 on the penalty term of the objective function. At this level of first-order asymptotic analysis, neither of these features affects the limiting conditional Gaussian distribution of $\theta^*_{\mathbf{w},n}$.

We note that rescaling in Equation (2.4) has no effect on solutions $\theta^*_{\mathbf{w},n}$, and so it is equivalent in the construction to use normalized weights $\tilde{\mathbf{w}}$ that sum to unity. Such $\tilde{\mathbf{w}}$ are uniformly distributed over the unit simplex, and thus correspond to a specific Dirichlet distribution. Exponential/Dirichlet weights are motivated from the perspective of both inference, related to the original Bayesian bootstrap (Rubin, 1981; Muliere & Secchi, 1996), and computation, owing to benefits of smoothly varying weights. Other weight distributions may also be effective (Barbe & Bertail, 2012).

We aim to use WBB samples as approximate posterior samples for uncertainty quantification. It remains unknown in general what is the relationship between this WBB distribution and the posterior distribution associated with any specific prior. The first-order asymptotic approximation above is constructed so that the prior structure is asymptotically negligible; one could say, then, that WBB approximates any one of a number of different posterior

distributions. In particular, the WBB samples may not provide a close approximation to the posterior formed from the same prior as used in the penalty term in the objective function. We deploy numerical experiments to understand the approximation better in finite samples.

2.3 Numerical experiments

We illustrate the proposed methodology with a number of scenarios to assess the quality of the WBB approximation.

2.3.1 LASSO Experiment

First, consider a simple univariate normal means problem with a LASSO prior where

$$y|\theta \sim N(\theta, 1^2), \quad \theta \sim \mathit{Laplace}(0, 1/\lambda).$$

Given the i.i.d. exponential weights w_1 and w_0, the weighted posterior mode $\theta^*_{\mathbf{w}}$ is given by

$$\theta^*_{\mathbf{w}} = \underset{\theta \in \Theta}{\arg\min} \left\{ \frac{w_1}{2}(y-\theta)^2 + \lambda w_0 |\theta| \right\}.$$

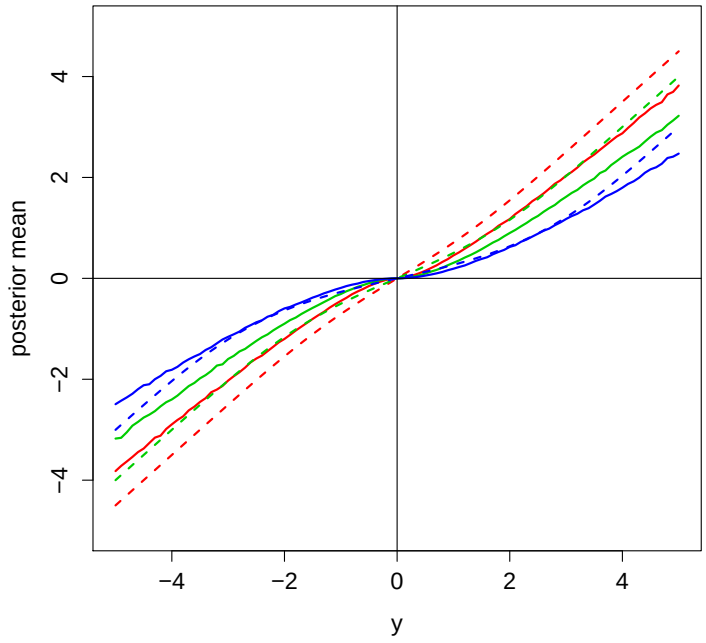
This is sufficiently simple for an exact WBB solution in terms of the soft thresholding proximal operator:

$$\theta^*_{\mathbf{w}} = \begin{cases} y - \lambda w_0/w_1 & \text{if } y > \lambda w_0/w_1, \\ y + \lambda w_0/w_1 & \text{if } y < -\lambda w_0/w_1, \\ 0 & \text{if } |y| \leq \lambda w_0/w_1. \end{cases}$$

The WBB mean $E_{\mathbf{w}}(\theta^*_{\mathbf{w}}|y)$ is approximated by the sample mean of $\{\theta^{*,t}_{\mathbf{w}}\}_{t=1}^T$. On the other

Figure 2.1: Normal Means Model with LASSO Prior

WBB mean $E_{\mathbf{w}}(\theta^*_{\mathbf{w}}|y)$ (in solid lines) versus exact posterior mean $E(\theta|y)$ (in dashed lines).



hand, Mitchell (1994) gives the expression for the posterior mean,

$$\begin{aligned} E(\theta|y) &= \frac{\int_{-\infty}^{\infty} \theta \exp\left\{-(y-\theta)^2/2 - \lambda|\theta|\right\} d\theta}{\int_{-\infty}^{\infty} \exp\left\{-(y-\theta)^2/2 - \lambda|\theta|\right\} d\theta} \\ &= \frac{F(y)}{F(y)+F(-y)}(y+\lambda) + \frac{F(-y)}{F(y)+F(-y)}(y-\lambda) \\ &= y + \frac{F(y)-F(-y)}{F(y)+F(-y)}\lambda \end{aligned}$$

where $F(y) = \exp(y)\Phi(-y-\lambda)$ and $\Phi(\cdot)$ is the c.d.f. of standard normal distribution. We plot the WBB mean versus the exact posterior mean in Figure 2.1. Interestingly, the WBB algorithm shrinks the posterior means towards zero. WBB samples are approximate posterior draws, though the algorithm structure does not permit a clear description of the prior that is in force for this sampled distribution. Whatever is this effective prior, it may be different from the prior encoded in the penalty used in repeated optimization, as the result

in Figure 1 suggests.

A simulation study is conducted in order to further investigate WBB in the context of regression. We simulate data from the linear model,

$$y = X\beta + \epsilon,\ \epsilon \sim N(0, \sigma_\epsilon^2 I)$$

where X is $n \times p$ and y is $n \times 1$. The design matrix X is generated by drawing its n rows independently from a p-dimensional normal distribution, $N(\mathbf{0}, \Sigma)$. Within each row, the covariance between entries in columns i and j is set to be $\Sigma_{i,j} = 0.1 \times 0.8^{|i-j|}$. Further we set the noise variance $\sigma_\epsilon^2 = \|X\beta\|_2^2/(2n)$. In this setting, the signal-to-noise ratio is 2. Figure 2.2 displays WBB samples (kernel density estimates) and posteriors via MCMC in one of the simulation settings; $n = 50, p = 2, \sigma_\epsilon^2 = 1, \Sigma = [[1, 0.3]', [0.3, 1]']$ and $\beta_1 = 0, \beta_2 = 2$. The WBB (on the L^1, LASSO prior penalty) is compared to posteriors computed via MCMC under several different priors: the same L^1 prior encoded in the WBB penalty, and also two L^0 penalties, $p(\beta) \propto \exp\{-\lambda\|\beta\|_0\}$ and $p(\beta) \propto \exp\{-5\lambda\|\beta\|_0\}$ (Polson & Sun, 2019). The WBB samples entail a spike at $\beta_1 = 0$, indicating a positive probability mass in the effective prior. No such posterior mass is evident in the L^0 priors nor possible in the L^1 prior.

Next we use the simulation to compare WBB distributions to MCMC-computed posteriors in a range of regression settings. We consider both sparse and dense cases for the coefficient vector $\beta = [\beta_1, \beta_2, ..., \beta_p]'$:

A(i). $\beta_j = 1$ for $1 \leq j \leq 10$ and $\beta_j = 0$ for $j > 10$.

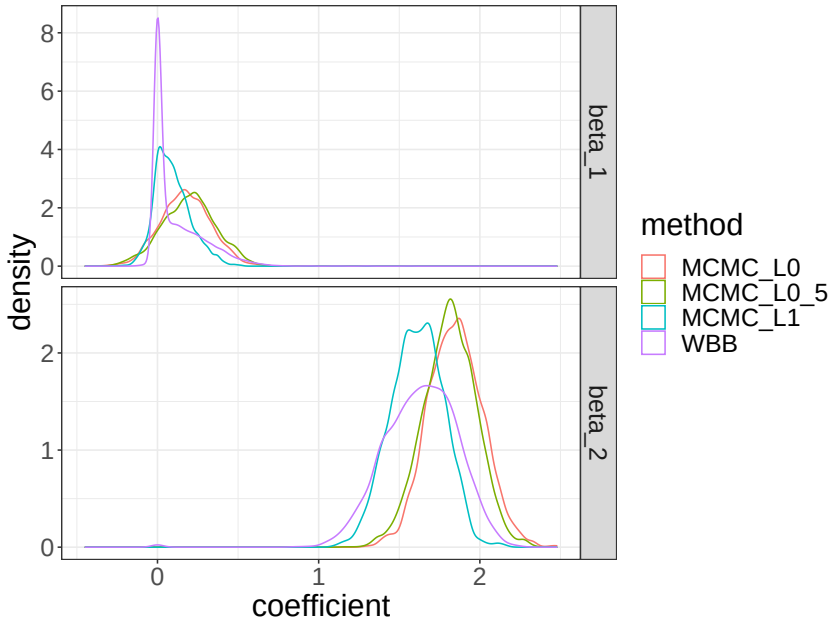
A(ii). $\beta_j = 1$ for $1 \leq j \leq 5$, $\beta_j = 10$ for $6 \leq j \leq 10$ and $\beta_j = 0$ for $j > 10$.

B. $\beta_j = 1$ for $1 \leq j \leq p$.

For each setting of β, we consider $p = 40, 60, 80, 100, 120$. To investigate the performance of WBB under different sample sizes, we also choose two settings of n: $n = 100$ for all p, or $n = p/2$ for all p.

Figure 2.2: WBB Samples Under LASSO Penalty

Compared with posteriors under LASSO prior and L^0 prior computed by MCMC.



We compare the WBB with LASSO prior and separate weights to Markov chain Monte Carlo under the Bayesian LASSO (Park & Casella, 2008; Carlin & Polson, 1991). The two methods entail different prior penalties:

$$\text{WBB} : \lambda\phi(\beta|\mathbf{w}) = \lambda \sum_{j=1}^{p} w_{0,j}|\beta_j|,$$

$$\text{BLASSO} : \phi(\beta|\sigma^2) = -\log\left(\frac{\lambda}{2\sigma}\right) + \frac{\lambda}{\sigma}\sum_{j=1}^{p}|\beta_j|.$$

Bayesian LASSO imposes a noninformative marginal prior on σ^2, $\pi(\sigma^2) = 1/\sigma^2$ and the posterior distribution is sampled by a Gibbs sampler, with λ chosen by maximizing the marginal likelihood. In WBB, λ is chosen via cross-validation, using standard unweighted LASSO. Here the original LASSO prior is separable and we study the separate-weights version of WBB. In high-dimensional cases, a large common w_0 multiplied to $\phi(\beta)$ can introduce extra sparsity to all marginal posteriors. We use separate weights in an effort to

overcome this issue. For comparison criteria, we present the estimation MSE of coefficients β (use posterior mean as our estimate), out-of-sample prediction MSE (test sets are of the same size as the corresponding training sets), and 95% credible interval coverage. Fixing (p, n, β), we draw $T = 200$ posterior samples by each method and the estimation procedure is repeated over $B = 500$ independent datasets $\{(X, y)^{(b)}\}_{b=1}^{B}$. Let $\beta^{*,t}(b)$ denote the t-th posterior draw, with respect to the b-th dataset. For each coordinate j, coverage is calculated using $\{\beta_j^{*,t}(b) : 1 \le t \le T, 1 \le b \le B\}$. The coverages are then averaged across $1 \le j \le p$.

Table 2.1: Coefficient Estimation Mean Squared Error (MSE)

		p	40	60	80	100	120
$n = 50$	A(i)	BLASSO	0.13	0.09	0.06	0.05	0.04
		WBB	0.19	0.06	0.05	0.04	0.03
	A(ii)	BLASSO	5.59	3.74	2.90	2.33	1.96
		WBB	7.40	2.89	2.31	1.90	1.62
	B	BLASSO	0.67	0.67	0.68	0.70	0.70
		WBB	1.77	0.49	0.50	0.51	0.52
$n = p/2$	A(i)	BLASSO	0.13	0.08	0.06	-	0.05
		WBB	0.14	0.08	0.05	-	0.03
	A(ii)	BLASSO	6.88	4.09	2.88	-	1.94
		WBB	6.58	3.80	2.53	-	1.47
	B	BLASSO	0.52	0.56	0.64	-	0.74
		WBB	0.68	0.62	0.55	-	0.48

Table 2.2: Out-of-sample Prediction Mean Squared Error (MSE)

		p	40	60	80	100	120
$n = 50$	A(i)	BLASSO	3.35	3.42	3.43	3.61	3.68
		WBB	3.34	3.37	3.40	3.60	3.72
	A(ii)	BLASSO	119.65	124.03	130.49	132.21	135.46
		WBB	121.45	123.40	129.89	134.04	140.44
	B	BLASSO	20.24	33.18	46.75	61.86	80.57
		WBB	20.61	32.47	46.27	60.59	78.71
$n = p/2$	A(i)	BLASSO	4.55	4.15	3.83	-	3.61
		WBB	3.65	3.75	3.66	-	3.65
	A(ii)	BLASSO	180.56	153.56	136.82	-	128.39
		WBB	145.73	141.35	133.66	-	132.23
	B	BLASSO	26.35	39.23	50.77	-	73.63
		WBB	21.80	34.82	47.76	-	75.29

The complete results for estimation error, out-of-sample prediction error, and credible interval coverage are displayed in Tables 1, 2, and 3, respectively. In almost all cases when $p \ge 60$, WBB has lower estimation error than BLASSO. WBB shows comparable or superior

Table 2.3: 95% Credible Interval Coverage

		p	40	60	80	100	120
$n = 50$	A(i)	BLASSO	0.91	0.92	0.93	0.94	0.94
		WBB	0.92	0.92	0.93	0.94	0.95
	A(ii)	BLASSO	0.91	0.93	0.95	0.96	0.96
		WBB	0.91	0.92	0.94	0.94	0.95
	B	BLASSO	1.00	1.00	1.00	1.00	1.00
		WBB	0.95	0.96	0.94	0.93	0.91
$n = p/2$	A(i)	BLASSO	0.93	0.93	0.94	-	0.94
		WBB	0.91	0.92	0.93	-	0.94
	A(ii)	BLASSO	0.92	0.94	0.95	-	0.96
		WBB	0.91	0.93	0.94	-	0.95
	B	BLASSO	1.00	1.00	1.00	-	1.00
		WBB	0.94	0.93	0.93	-	0.92

out-of-sample prediction in most cases. Credible intervals of neither WBB nor BLASSO have exact 95% coverage. WBB performs well when $p \geq 100$ and β is sparse, though it often under covers. When $\beta_j = 1$ for all $1 \leq j \leq p$, BLASSO intervals are too wide while WBB intervals have coverage close to 95% when $40 \leq p \leq 80$. Though limited in scope, this simulation reveals some distinctions and also broad similarities between WBB and Bayesian LASSO in both posterior structure and the sampling performance of approximate posterior summaries. It may provide a useful basis for further methodological development. Serial computations were used in the simulation above, and WBB was slightly slower than BLASSO (by a factor of 1.5 on CPU time); the computational advantage of WBB shows up in parallel computations, since weight vectors induce separate optimizations.

2.3.2 Diabetes Data

To further illustrate the WBB methodology, we apply it to the often-analyzed diabetes dataset (Efron et al., 2004). Data from $n = 442$ diabetes patients are available, with response a measure of disease progression and with 10 baseline variables ($p = 10$), including age, sex, body mass index, average blood pressure, and six blood serum measurements. We apply Algorithm 1 with $T = 1000$. (R code is in Supplementary Material file.) For each weight vector $\mathbf{w}$, the WBB estimate $\beta^*_{\mathbf{w}}$ is calculated using Equation (2.9) via the regularization

method in the package `glmnet`.

$$\beta_{\mathbf{w}}^{*,\text{common}} := \arg\min_{\beta} \sum_{i=1}^{n} w_i (y_i - x_i'\beta)^2 + \lambda w_0 \sum_{j=1}^{p} |\beta_j| \ . \tag{2.9}$$

The regularization factor λ is chosen by cross-validation with unweighted likelihood. The WBB is also performed with separate weights on each $|\beta_j|$,

$$\beta_{\mathbf{w}}^{*,\text{separate}} := \arg\min_{\beta} \sum_{i=1}^{n} w_i (y_i - x_i'\beta)^2 + \lambda \sum_{j=1}^{p} w_{0,j} |\beta_j| \ .$$

As in the simulation study, WBB is compared to the Bayesian LASSO.

Figure 2.3 shows the results of all these three methods (the WBB with common/separate weight on prior terms , and Bayesian LASSO). Marginal posteriors for β_j's are presented. For some coefficients there is very good agreement among the methods (e.g., `bmi` and `map`). One notable feature is that the WBB tends to introduce less sparsity than Bayesian LASSO. For example, the Bayesian LASSO posteriors of `age, tc, ldl, tch` and `glu` have higher spikes near zero compared with the WBB.

2.3.3 Trend Filtering

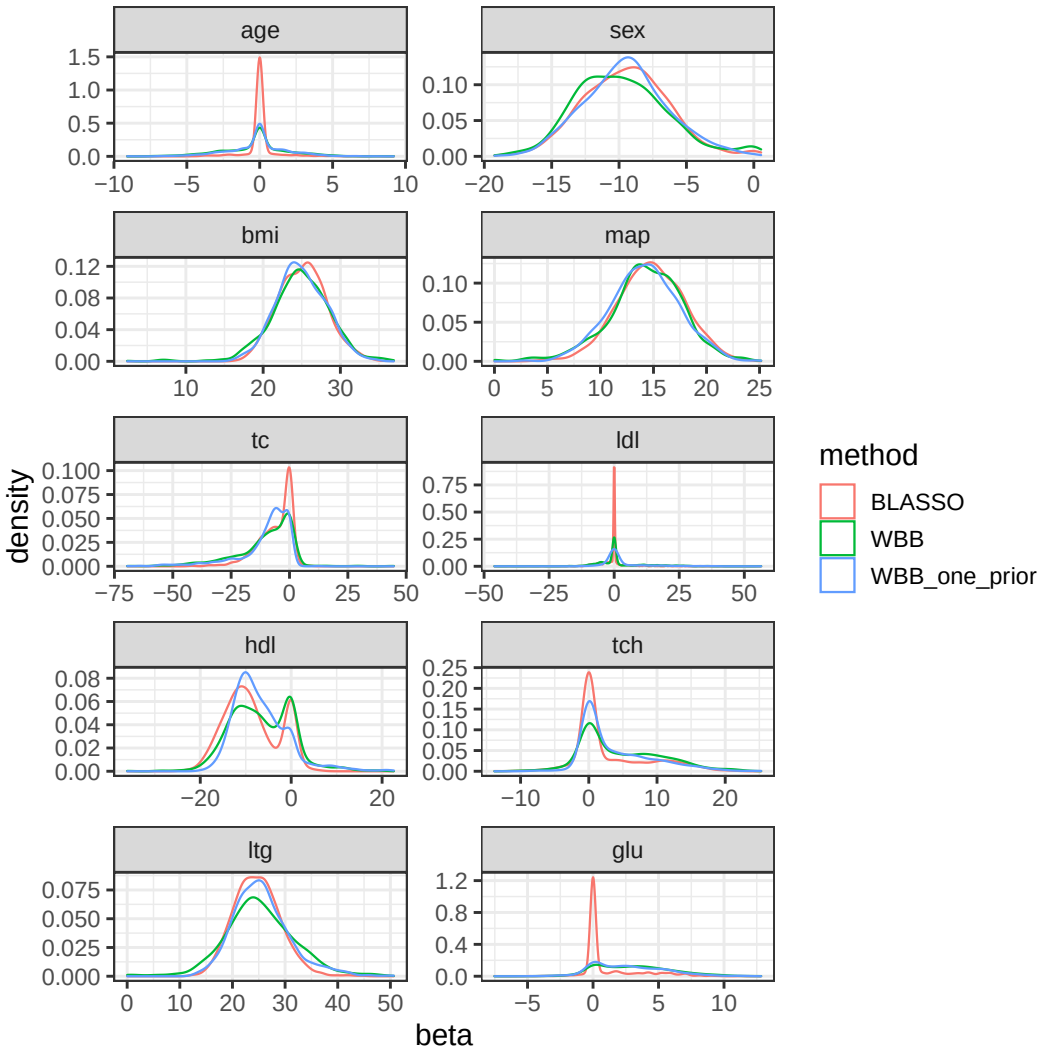
The generalized LASSO solves the optimization problem:

$$\begin{aligned} \beta^* &= \arg\min_{\beta} \{l(y|\beta) + \lambda\phi(\beta)\} \\ &= \arg\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|D\beta\|_1 \end{aligned}$$

where $l(y|\beta) = \frac{1}{2}\|y - X\beta\|_2^2$ is the negative log-likelihood. $D \in \mathcal{R}^{m\times p}$ is a penalty matrix and $\lambda\phi(\beta) = \lambda\|D\beta\|_1$ is the negative log-prior or regularization penalty. There are fast path algorithms for solving this problem (see `genlasso` package).

Figure 2.3: Diabetes Example

The WBB (with common prior weight (blue) and separate prior weights (green)) and Bayesian LASSO (red) are used to draw from the marginal posteriors for β_j's, j = 1,2,...10.

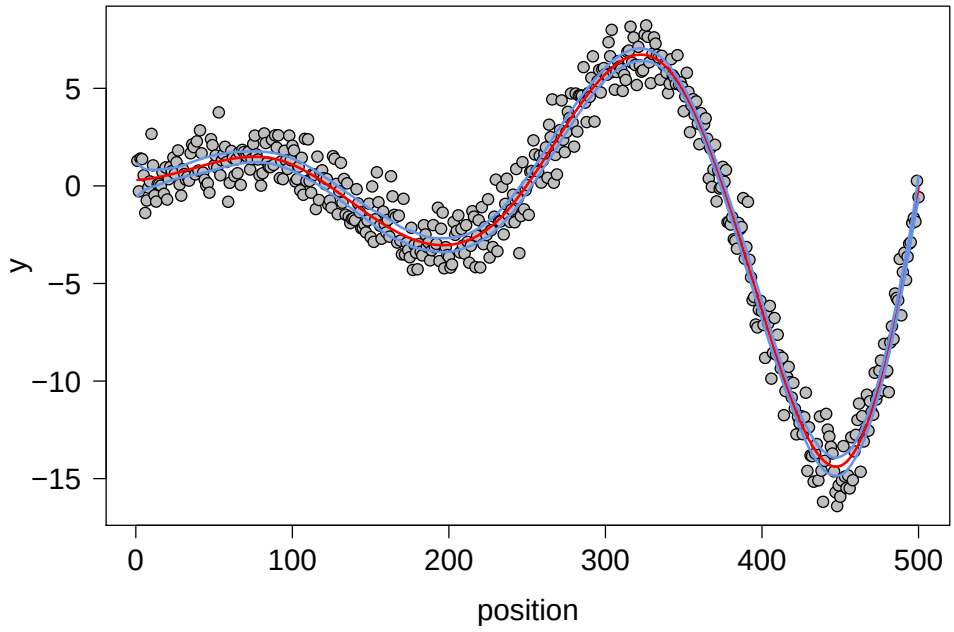


As a subproblem, polynomial trend filtering (Tibshirani, 2014; Polson & Scott, 2015) allows for piece-wise polynomial curve-fitting, where the knots and the parameters are chosen adaptively. Intuitively, the trend-filtering estimator is similar to an adaptive spline model: it penalizes the discrete derivative of order k, resulting in piecewise polynomials of higher degree for larger k.

Specifically, $X = I_p$ in the trend filtering setting and the data $y = (y_1, ..., y_p)$ are assumed

Figure 2.4: Cubic Trend Filtering

The red line is $\hat{\beta}_i$ for $i = 1, 2, ...500$; the blue line is $\hat{\beta}_i \pm 2 \times se$ where the standard errors are computed by WBB; $\lambda = 1000$.



to be meaningfully ordered from 1 to p. The penalty matrix is specially designed by the discrete $(k+1)$-th order derivative,

$$D^{(1)} = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & -1 & 1 \end{bmatrix}_{(p-1)\times p}$$

and $D^{(k+1)} = D^{(1)} D^{(k)}$ for $k = 1, 2, 3....$ For example, the log-prior in linear trend filtering is explicitly written as $\lambda \sum_{i=1}^{p-2} |\beta_{i+2} - 2\beta_{i+1} + \beta_i|$. For a general order $k > 1$,

$$\|D^{(k+1)}\beta\|_1 = \sum_{i=1}^{p-k-1} \Big| \sum_{j=i}^{i+k+1} (-1)^{(j-i)} \binom{k+1}{j-i} \beta_j \Big|.$$

WBB solves the following generalized LASSO problem in each draw:

$$\begin{aligned}\beta^*_{\mathbf{w}} &= \arg\min_{\beta} \frac{1}{2}\sum_{i=1}^{p} w_i(y_i - \beta_i)^2 + \lambda w_0 \|D^{(k)}\beta\|_1 \\ &= \arg\min_{\beta} \frac{1}{2}\|Wy - W\beta\|_2^2 + \lambda\|D^{(k)}\beta\|_1 \\ &= W^{-1}\arg\min_{\tilde{\beta}} \frac{1}{2}\|\tilde{y}_{\mathbf{w}} - \tilde{\beta}_{\mathbf{w}}\|_2^2 + \lambda\|\tilde{D}^{(k)}_{\mathbf{w}}\tilde{\beta}_{\mathbf{w}}\|_1\end{aligned}$$

where $W = \text{diag}\left(\sqrt{w_1}, \cdots, \sqrt{w_p}\right)/\sqrt{w_0}$ and

$$\tilde{y}_{\mathbf{w}} = Wy,\ \tilde{\beta}_{\mathbf{w}} = W\beta,\ \tilde{D}^{(k)}_{\mathbf{w}} = D^{(k)}W^{-1}.$$

To illustrate, we simulate data y_i from a Fourier series regression

$$y_i = \sin\left(\frac{4\pi}{500}i\right)\exp\left(\frac{3}{500}i\right) + \epsilon_i$$

for $i = 1, 2, \cdots, n = 500$, where $\epsilon_i \sim N(0, 2^2)$ are i.i.d. Gaussian deviates. The cubic trend filtering result is given in Figure 2.4. For each i, the WBB gives a group of estimates $\left\{\beta^{*,t}_{\mathbf{w}}(i)\right\}_{t=1}^{T}$ where T is the total number of draws. The standard deviation of these weighted solutions constitutes a posterior standard deviation, or essentially a standard error for the estimator $\hat{\beta}_i$.

2.3.4 Deep Learning: MNIST Example

Deep learning is a form of machine learning that uses hierarchical abstract layers of latent variables to perform pattern matching and prediction. A Bayesian probabilistic perspective provides a number of insights into more efficient algorithms for optimization and hyper-parameter tuning. The general goal is to find a predictor of an output y given a high dimensional input x. For a classification problem, $y \in \{1, 2, ..., K\}$ is a discrete variable and

can be coded as a K-dimensional 0-1 vector. The model is as follows. Let $z^{(l)}$ denote the l-th layer, and so $x = z^{(0)}$. The final output is the response y, which can be numeric or categorical. A deep prediction rule (Polson & Sokolov, 2017) is then

$$\begin{aligned}
z^{(1)} &= f^{(1)}\Big(W^{(0)}x + b^{(0)}\Big),\\
z^{(2)} &= f^{(2)}\Big(W^{(1)}z^{(1)} + b^{(1)}\Big),\\
&\cdots\\
z^{(L)} &= f^{(L)}\Big(W^{(L-1)}z^{(L-1)} + b^{(L-1)}\Big),\\
\hat{y}(x) &= z^{(L)}.
\end{aligned}$$

Here, $W^{(l)}$ are weight matrices, and $b^{(l)}$ are threshold or activation levels. $f^{(l)}$ is the activation function. Probabilistically, the output y in a classification problem is generated by a probability model

$$p(y|x, W, b) \propto \exp\{-l(y|x, W, b)\}$$

where $l(y|x, W, b) = \sum_{i=1}^{n} l_i(y_i|x_i, W, b)$ is the negative cross-entropy,

$$l_i(y_i|x_i, W, b) = l_i(y_i, \hat{y}(x_i)) = \sum_{k=1}^{K} y_{ik} \log \hat{y}_k(x_i)$$

where y_{ik} is 0 or 1. Adding the negative log-prior $\lambda\phi(W, b)$, the objective function (negative log-posterior) to be minimized by stochastic gradient descent is

$$\mathcal{L}_\lambda(y, \hat{y}) = \sum_{i=1}^{n} l_i(y_i, \hat{y}(x_i)) + \lambda\phi(W, b).$$

Accordingly, with each draw of weights $\mathbf{w}$, WBB provides the estimates $(W^*_{\mathbf{w}}, b^*_{\mathbf{w}})$ by solving the following optimization problem:

$$(W^*_{\mathbf{w}}, b^*_{\mathbf{w}}) = \arg\min_{W,b} \sum_{i=1}^{n} w_i l_i(y_i|x_i, W, b) + \lambda w_0 \phi(W, b).$$

We take the classic MNIST example (LeCun & Cortes, 2010) to illustrate the application of WBB in deep learning. The MNIST database of handwritten digits, available from Yann LeCun's website, has 60,000 training examples and 10,000 test examples. Here the high-dimensional x is a normalized and centered fixed-size (28×28) image and the output $\hat{y}$ is a 10-dimensional vector, where i-th coordinate corresponds to the probability of that image being the i-th digit.

For simplicity, we build a 2-layer neural network with layer sizes 128 and 64 respectively. Therefore, the dimensions of parameters are

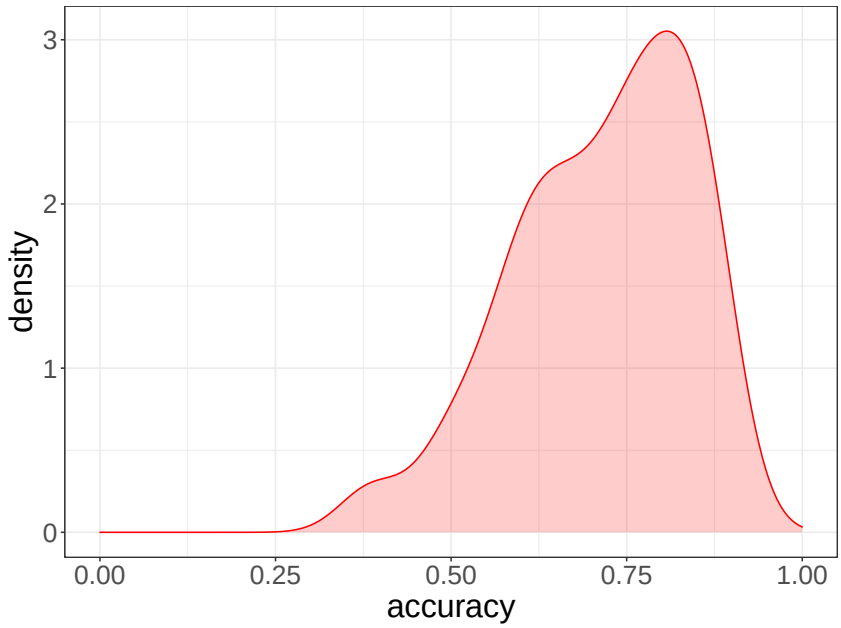
$$W^{(0)} \in \mathcal{R}^{128\times 784},\ b^{(0)} \in \mathcal{R}^{128},$$
$$W^{(1)} \in \mathcal{R}^{64\times 128},\ b^{(1)} \in \mathcal{R}^{64},$$
$$W^{(2)} \in \mathcal{R}^{10\times 64},\ b^{(0)} \in \mathcal{R}^{10}.$$

The activation function $f^{(i)}$ is ReLU, $f(x) = \max\{0, x\}$, and the negative log-prior is specified as

$$\lambda\phi(W, b) = \lambda \sum_{l=0}^{2} \|W^{(l)}\|_2^2$$

where we manually set $\lambda = 10^{-4}$. Figure 2.5 shows the posterior distribution of the classification accuracy in the test dataset. We see that the test accuracies are centered around 0.75 and the posterior distribution is left-skewed. Furthermore, the accuracy is higher than 0.35 in 99% of the cases. The 95% interval is [0.407, 0.893]. Due to the simple 2-layer neural network structure, the classification accuracy is admittedly low compared with the

Figure 2.5: Posterior Distribution of the Classification Accuracy by WBB



state-of-the-art (e.g., Liang & Hu, 2015). This illustrative example shows how WBB can be easily implemented in deep learning. Implementing Variational Bayes in deep learning is discussed in Polson & Sokolov (2017).

2.4 Discussion

Weighted Bayesian Bootstrap (WBB) provides a computationally attractive solution to scalable Bayesian inference (Minsker et al., 2014; Welling & Teh, 2011) whilst accounting for parameter uncertainty by maximizing a weighted posterior distribution. WBB can also be used in conjunction with proximal methods (Parikh & Boyd, 2013; Polson, Scott, & Willard, 2015) to provide sparsity in high dimensional statistical problems. With a similar ease of computation, WBB provides an alternative to approximate Bayesian computation (ABC) methods (Beaumont, 2009) and variational Bayes (VB) methods (Blei, Kucukelbir, & McAuliffe, 2017). A fruitful area for future research is the comparison of WBB to recent extensions of the weighted likelihood bootstrap for generalized Bayesian inference (Lyddon,

Holmes, & Walker, 2019).

For a wide range of non-smooth objective functions/statistical models, recent regularization methods provide fast, scalable algorithms for calculating estimates of the form Equation (2.1), which can also be viewed as posterior modes. Therefore as λ varies we obtain a full regularization path as a form of prior sensitivity analysis. Further, Strawderman, Wells, & Schifano (2013) and Polson & Scott (2015) considered scenarios where posterior modes can be used as posterior means from augmented probability models. There may be useful interpretations of the random weights from the data-augmentation perspective.

Extending WBB asymptotics presents an exciting research opportunity. The argument in Section 2.5 relies on smoothness in both the sampling model and prior, and it retains fixed parameter dimension as n increases. Theoretical guarantees remain unavailable for relatively large parameter dimension or for non-smooth penalty functions. Fortunately, groundwork has been done, for example by van der Pas, Kleijn, & van der Vaart (2014), Narisety & He (2014), and others on the asymptotic behaviour of the posterior distribution, and by Knight & Fu (2000) and others on sampling theory of optimization-based estimators,

2.5 Appendix

Here we consider how stochastic gradient descent (SGD) may be deployed to minimize the penalized loss function, $\sum_{i=1}^{n} w_i l_i(y_i;\theta) + \lambda w_0 \phi(\theta)$. The method minimizes the function by taking a negative step along an estimate g^k of the gradient $\nabla \left[\sum_{i=1}^{n} w_i l_i(y_i;\theta^k) + \lambda w_0 \phi(\theta^k)\right]$ at iteration k. The approximate gradient is estimated by calculating

$$g^k = \frac{n}{b_k} \sum_{i \in E_k} w_i \nabla l_i(y_i;\theta^k) + \lambda w_0 \frac{n}{b_k} \nabla \phi(\theta^k),$$

where $E_k \subset \{1,\ldots,n\}$ and $b_k = |E_k|$ is the number of elements in E_k. When $b_k > 1$ the algorithm is called batch SGD and simply SGD otherwise. A usual strategy to choose subset

E is to go cyclically and pick consecutive elements of $\{1, \ldots, T\}$, $E_{k+1} = [E_k \mod n] + 1$. The approximated direction g^k is calculated using a chain rule (i.e., back-propagation) for deep learning. It is an unbiased estimator. Thus, at each iteration, the SGD updates the solution

$$\theta^{k+1} = \theta^k - t_k g^k.$$

For deep learning applications the step size t_k (i.e., learning rate) is usually kept constant or some simple step size reduction strategy is used, $t_k = a \exp(-kt)$. Appropriate learning rates or the hyperparameters of the reduction schedule are usually found empirically from numerical experiments and observations of the loss function progression.

CHAPTER 3

DEEP LEARNING IN CHARACTERISTICS-SORTED FACTOR MODELS

To study the characteristics-sorted factor model in empirical asset pricing, we design a non-reduced-form feedforward neural network with the non-arbitrage objective to minimize pricing errors. Our model starts from firm characteristics *[inputs]*, generates risk factors *[intermediate features]*, and fits the cross-sectional returns *[outputs]*. A nonlinear activation in deep learning approximates the traditional security sorting on characteristics to create long-short portfolio weights, like a hidden layer, from lag characteristics to realized returns. Our model offers an alternative approach for dimension reduction in empirical asset pricing on characteristics *[inputs]*, rather than factors *[intermediate features]*, and allows for both nonlinearity and interactions directly through *[inputs]*. Our empirical findings are threefold. First, we find substantial and robust asset pricing improvements of multiple performance measures, such as Cross-Sectional R^2, in both in-sample and out-of-sample analysis. Second, the deep learning augmented models produce all positive improvements regarding return prediction over the benchmark factor models. Finally, we show significant increases in factor investing, nonlinear relationships in deep characteristics, and their importance on raw characteristics.

Key Words: Alpha, Characteristics-Sorted Factor Models, Cross-Sectional Returns, Deep Learning, Firm Characteristics, Non-Arbitrage, Pricing Errors.

3.1 Introduction

According to ICAPM of Merton (1973), a combination of common factors captures the cross-section of expected returns, and the regression intercept should be zero. Unlike the regular objective function in statistics and machine learning, the model fitness for asset pricing is not about the explained variation in time series but the magnitude of intercepts, alphas, in the cross-section. This non-arbitrage restriction on alphas implies that simply adding factors leads to statistical overfitting (increase time series R^2) but does not cause economic overfitting (decrease intercepts).

In empirical asset pricing studies, researchers typically sort securities on firm characteristics and create long-short portfolios as proxies for common risk factors to build factor models. For example, the Nobel prize research of Fama and French (1993) add SMB (small-minus-big) and HML (high-minus-low) to CAPM. However, almost all proposed factor models have rejected the zero-alpha hypothesis. Our paper approaches this puzzle with a machine learning perspective as an optimization problem: How does one construct a factor model to minimize pricing errors or alphas?

Our paper aims to investigate the underlying mechanism of the characteristics-sorted factor models, which includes sorting securities, generating factors, and fitting security returns. We define a non-arbitrage objective function, pricing errors, for deep learning optimization. We show the characteristics-sorted factor models can be dissembled as a feed-forward neural network:

(1) *Inputs* are firm characteristics. The neural network starts from sorting securities on firm characteristics, a nonlinear activation to create long-short portfolio weights.

(2) *Intermediate features* are risk factors. The factors are linear activations (long-short portfolio weights) on realized returns from the sorting directions.

(3) *Outputs* are security returns. Reaching the non-arbitrage objective is equivalent to

minimizing pricing errors for fitting the factor model to security returns.

Distinct from the risk factor literature, our focus is constructing a factor model rather than testing a factor or characteristic. We argue the current literature is mostly about intermediate features and outputs (security returns), whereas ours illustrates the complete channel between inputs and outputs. We adopt a non-reduced-form neural network and develop such a bottom-up approach that includes security sorting, factor generation, and fitting the cross-section of security returns. The Fama-French-type characteristics-sorted factor models can be shown as "shallow" learning models.

A particular asset pricing researcher performs a GRS-type statistical test, from Gibbons et al. (1989), on the proposed factor model and stops if the hypothesis is rejected. We approach this procedure as an optimization problem. The deep learning optimization continues to search for optimal solutions because our bottom-up approach provides a non-reduced-form mechanism. We show a "feed-forward" neural network consisting of an "input layer" of firm characteristics, "hidden layers" of factors, and an "output layer" of security returns (check Figure 3.2 for the Fama-French example). The "automatic" factor generation receives training feedback through backward propagation, which addresses how much pricing errors can be reduced by optimizing over model parameters.

On the methodological side, we marry state-of-the-art deep learning optimization with asset pricing factor models. Deep learning is well known for its superior pattern-matching performance, flexible architecture, and yet mysterious "black box." This paper aims to introduce deep learning into asset pricing with a transparent "white box" model. We disassemble the asset pricing mechanism with deep learning concepts: inputs, intermediate features, outputs, and the objective function. We show the routine procedure financial economists have been working on for decades is a transparent "white box" model. The "deep" part of the past asset pricing research is to *manually discover* those useful firm characteristics from all economic information.

On the economic side, long-short factors are useful because they reflect compensation for exposure to underlying characteristics and can be evaluated as tradable portfolios. However, many of these characteristics' formulas are highly similar to accounting, trading, and behavioral perspectives. One unresolved issue is how minor differences in characteristics' formulas affect the corresponding security sorting, long-short factors, and even model fitness. The routine procedure has one long-time overlooked the built-in problem. Most research focuses on factors [intermediate features] and security returns [outputs], whereas the inputs are firm characteristics. Our research attempts to fill in this missing piece. Our non-reduced-form deep learning approach investigates the complete channel from characteristics [inputs] to security returns [outputs].

On the empirical side, we study the universe for the largest 3,000 stocks in the U.S. equity market over the recent 45 years. Our library consists of 61 published firm characteristics. For statistical model fitness (Total R^2 in Table 3.2), we find the deep learning augmented model with robust improvement over the benchmark model, such as CAPM, across multiple sets of test portfolios. We also find substantial and robust improvement of Pricing Error R^2 and cross-sectional R^2 in both in-sample and out-of-sample analysis for the asset pricing improvement. The return prediction performance is illustrated by the Predictive R^2 in Table 3.5, where the deep learning augmented models produce all positive results. Directly investing deep factors, we obtain a significantly higher Sharpe ratio over the Fama-French 3 factors. Finally, to interpret our deep characteristics in a nonlinear dimension reduction perspective, we find Dividend-to-Pricing and Corporate Investment are the two most important to reduce pricing errors by controlling Fama-French three factors.

The rest of the paper is organized as follows. We compare and position our study with the relevant literature in section 3.1.1. Section 3.2 introduces deep learning into empirical asset pricing. Section 3.3 provides details for our deep learning augmented factor model. Section 3.4 illustrates the empirical study design and main empirical findings. Section 3.5

summarizes the paper.

3.1.1 Connections with Previous Literature

Our paper builds on several strands of the asset pricing literature. The most related literature are dimension reduction techniques via principal components and its generalizations. For example, Kim et al. (2018) and the IPCA (Instrumental PCA) of Kelly et al. (2019) use firm characteristics as instruments to model the time-varying coefficients and estimate PCs. Lettau and Pelger (2020) provide a regularized estimation for risk premia, RP-PCA (Risk Premia PCA), that identify factors with small time series variance but useful in the cross-section. Kozak et al. (2018) show that PCA of anomaly portfolios works as well as a reduced-form factor model in explaining anomaly portfolios. Light et al. (2017) use partial least squares (PLS) to aggregate information on firm characteristics.

We compare the results of our method with IPCA and RP-PCA in the empirical analysis. Generally, our deep learning framework differs from traditional PCA in three ways:

(1) Our dimension reduction is applied directly on firm characteristics [inputs] rather than factors [intermediate features] or security returns [outputs].

(2) Our dimension reduction also allows for both nonlinearity and interactions on inputs, whereas PCA only extracts linear components.

(3) PCA relies on a balanced data structure, whereas security sorting allows for an unbalanced data structure, such as individual stock returns and characteristics.

As discussed initially, our approach is closely related to the recent literature on applying machine learning methods to the asset pricing model. Harvey et al. (2016) raise a multiple testing issue to challenge 300 factors discovered in the literature. Feng et al. (2020) develop a regularized two-pass cross-sectional regression to tame the factor zoo, and find only a small

number of factors with the incremental contribution. Kozak et al. (2020) use a shrinkage estimator on the SDF coefficients for characteristic-based factors with economic interpretation. Kelly et al. (2019) and Kelly et al. (2020) evaluate the contribution of individual characteristics under a nested-model comparison for equity and corporate bond returns. A recent article of DeMiguel et al. (2020) shows the economic rationale of why many characteristics are needed in investing portfolios.

Our paper adds to the literature on forecasting asset returns with machine learning. Freyberger et al. (2020) apply adaptive group LASSO for selecting firm characteristics and show evidence of nonlinearity. Gu et al. (2020b) provide a comprehensive empirical investigation of forecasting performance using multiple machine learning algorithms. Han et al. (2018) employ a forecast combination approach, and Bianchi et al. (2020) find that machine learning can forecast treasury bond returns as well. The Bayesian predictive regression of Feng and He (2019) uses lag characteristics for the linear conditional factor coefficients, which is a reduced-form approach for approximation.

Our model provides a pricing kernel (factor models) that can be used to predict cross-sectional returns, and we find all positive Predictive R^2 in Table 3.5. However, the prediction literature studies the time series predictive performance between inputs and outputs, and skips the intermediate channel involved with risk factors. We fill in the missing pieces with our bottom-up approach: lag characteristics, realized factor returns, and realized security returns.

Besides, we provide an out-of-sample evaluation in section 3.4, which can be implemented either on the cross-section or time-series. We use one set of portfolios to train the factor model in the training sample and evaluate its performance with another set of unseen test portfolios in the test sample. This clean research design is a solution to the skepticism of Lewellen et al. (2010), who question the standard protocol of using Fama-French 25 size-B/M portfolios for both factor discovery and model testing. We have presented the training and

test sample design in Figure 3.7.

Finally, we add to the recent development of deep learning in finance and economics. Heaton et al. (2017) introduces deep learning decision models for problems in financial prediction and classification. Gu et al. (2020a) incorporate one deep learning technique, autoencoder, within the Instrumental PCA framework to construct nonlinear principal components. Feng et al. (2020) find it useful when implementing a multivariate-outcome model within the flexible deep learning framework to predict overlapping bond returns. A recent paper of Chen et al. (2019) uses a neural network to estimate the SDF model that explains all asset returns from the conditional moment constraints implied by no-arbitrage and finds promising investment performances. This continued progress in deep learning research is promising for both academic study and practical application in finance.

3.2 Deep Learning and Empirical Asset Pricing

Section 3.2.1 explains why we can treat the asset pricing test via an optimization problem for pricing errors. We demonstrate how a characteristics-sorted factor model can be reformulated within a deep learning architecture in section 3.2.2. Fama-French-type factor models are shown to be deep learning models, and we discuss implementation is discussed in section 3.2.3.

3.2.1 Minimizing pricing errors

From the economic constraint of the beta-pricing model, it follows that the excess asset return can be explained by the risk premia of factors in a linear equation. For tradable factors f_t and g_t,

$$\mathrm{E}(R_{i,t}) = \alpha_i + \beta_i^{\intercal}\mathrm{E}(f_t) + \gamma_i^{\intercal}\mathrm{E}(g_t). \tag{3.1}$$

The GRS test suggests a joint test on the vector of time series model intercepts, α, for all test assets. The kernel for the GRS test statistic is a weighted sum for the quadratic alphas, $\alpha^\intercal \Sigma_\alpha^{-1} \alpha$. If a sufficient factor model exists, this weighted sum for pricing errors should be statistically and economically insignificant, though the academic literature still fails to find such a model.

For this reason, we switch to an optimization problem for an alternative perspective. Machine learning and deep learning methods have been criticized for easily over-fitting the data. However, adding more factors on the regression's right-hand side increases the time series R^2 but unnecessarily reduces the magnitude of the regression intercept. The asset pricing optimization target is different from the time series model fitness. Therefore, we design such a deep learning framework that pushes the model fitting to the lower bound for pricing errors.

Denote $\widehat{R}_{i,t}$ as a linear portfolio constructed with factors to mimic the asset return $R_{i,t}$. Because all regressors need to be tradable portfolios in the time series regression, $\widehat{R}_{i,t}$ is formed as a linear combination of portfolios *without an intercept*. The expected difference, α_i, is the pricing error.

$$\mathrm{E}(R_{i,t} - \widehat{R}_{i,t}) = \alpha_i \tag{3.2}$$

The tradability for alphas determines the unique objective function in our optimization. The core of our objective function design is $\frac{1}{N}\sum_{i=1}^{N} \alpha_i^2$, an equally weighted version for the GRS test statistic kernel, and measures the average pricing errors. To the best of our knowledge, this paper is the first that focuses on minimizing alphas. We define an economic-driven objective function, minimizing pricing errors, which follows the non-arbitrage restriction from asset pricing models.

3.2.2 Characteristics-Sorted Factors in Deep Learning

By following the standard literature, we use excess returns in the study. Our model is to generate additional factors from the deep learning model, f_t, on a benchmark model g_t, which can be CAPM or Fama-French type models. We form the realized return predictor $\widehat{R}_{i,t}$ as a linear combination of f_t and g_t without an intercept. Therefore, the zero mean residual, $\epsilon_{i,t}$, measures the time series variation in forecasting error, and α_i refers to the potential pricing error.

$$\widehat{R}_{i,t} = \beta_i^{\intercal} f_t + \gamma_i^{\intercal} g_t, \tag{3.3}$$

$$R_{i,t} - \widehat{R}_{i,t} = \alpha_i + \epsilon_{i,t}, \tag{3.4}$$

$$f_t = W_{t-1} r_t, \tag{3.5}$$

$$W_{t-1} = H(z_{t-1}). \tag{3.6}$$

The additional deep factors, f_t, are long-short portfolios constructed by sorting individual firms on lag firm characteristics z_{t-1}. We use r_t, thousands of individual firm returns at month t, and W_{t-1}, the long-short portfolio weight determined at month $t-1$. $H(\cdot)$ represents a complex (and hidden) function for z_{t-1} that reflects underlying cross-sectional predictability. The $H(\cdot)$ transformation is the depth for the deep neural network. For example, $H(\cdot)$ can be the sorting function as a shallow network, then it transforms z_{t-1} to the long-short directions $\{1, 0, -1\}$. With the long-short directions, researchers multiply the long-short directions by equal or value weights to form W_{t-1}.

With the notation $\{f_t, r_t, W_{t-1}, z_{t-1}\}$, the characteristics-sorted factor model is clear and transparent in the above deep learning architecture. First, the inputs are lag characteristics z_{t-1}. Second, by sorting securities in the month $t-1$, we obtain the intermediary features W_{t-1}. Third, by adding the second inputs, individual firm realized returns r_t, we generate f_t. Finally, by adding the third inputs, the benchmark model g_t, we fit R_t. The predictive

structure for characteristics-sorted factor investing is due to the lag portfolio construction. The factor f_t is built with long-short portfolio weights at month $t-1$ and individual firm returns at month t.

In our framework, f_t is generated while controlling g_t within the deep learning model fitting. This procedure is consistent with the standard protocol that researchers admit new factors for their significance over a benchmark model. The estimated alphas or pricing errors are constructed as

$$\hat{\alpha}_i = \frac{1}{T}\sum_{t=1}^{T}\Big(R_{i,t} - \widehat{R}_{i,t}\Big) = \frac{1}{T}\sum_{t=1}^{T}\Big(R_{i,t} - \widehat{\beta}_i^{\intercal} f_t - \widehat{\gamma}_i^{\intercal} g_t\Big). \tag{3.7}$$

Our optimization objective is to minimize a weighted sum for the time-series variation and cross-sectional pricing errors:

$$\begin{aligned}
\mathcal{L}_\lambda &= \frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\Big(R_{i,t} - \widehat{R}_{i,t}\Big)^2 + \lambda * \frac{1}{N}\sum_{i=1}^{N}\Big(\frac{1}{T}\sum_{t=1}^{T}(R_{i,t} - \widehat{R}_{i,t})\Big)^2 && (3.8)\\
&= \frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\Big(R_{i,t} - \hat{\beta}_i^{\intercal} f_t - \hat{\gamma}_i^{\intercal} g_t\Big)^2 + \frac{\lambda}{N}\sum_{i=1}^{N}\hat{\alpha}_i^2 && (3.9)\\
&= \frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\Big(\hat{\epsilon}_{i,t} + \hat{\alpha}_i\Big)^2 + \frac{\lambda}{N}\sum_{i=1}^{N}\hat{\alpha}_i^2 && (3.10)\\
& && (3.11)\\
&= \underbrace{\frac{1+\lambda}{N}\sum_{i=1}^{N}\hat{\alpha}_i^2}_{\text{pricing errors}} + \underbrace{\frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\hat{\epsilon}_{i,t}^2}_{\text{time-series variation}}, && (3.12)
\end{aligned}$$

where $\hat{\epsilon}_{i,t} = R_{i,t} - \widehat{R}_{i,t} - \hat{\alpha}_i$, $\sum_{t=1}^{T}$ and $\hat{\epsilon}_{i,t} = 0$.

Here, λ is a tuning parameter that controls the balance between time-series and cross-sectional pricing errors. If λ is too big, we lose the weight for time series variation that supports the factor structure. If λ is too small, the objective function does not help reduce pricing errors. In our empirical study, we perform validation using a sequence of λ's. Our

objective function design follows the RP-PCA of Lettau and Pelger (2020), which also adds a penalty to account for cross-sectional pricing errors in average returns. Their regularized estimation is to identify those factors with small time-series variation but help to price the cross-section. In the empirical study, we further add an extra penalty on the parameters of $H(\cdot)$ (see Section 3.3.4), which helps to stabilize the network model training and introduces sparsity.

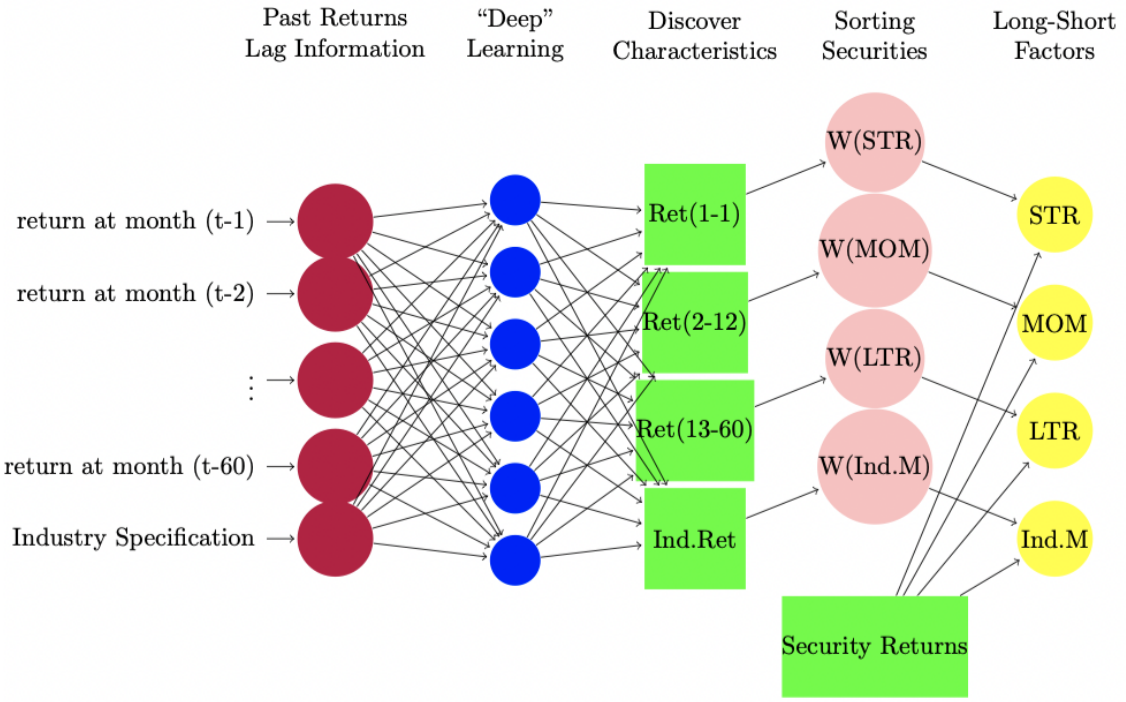
3.2.3 Fama-French Examples in Deep Learning

The current factor zoo contains too many similar firm characteristics to proxy for the same asset pricing anomalies. For example, many measures are proposed for value investing, such as book-to-market ratio, dividend yield, earning-to-price ratio, cash flow to price ratio, and so forth. Sorting securities on these "seemingly" related characteristics might be a trial-and-error experiment, which finds the one proxy with the best in-sample performance for the specific test assets in the test period. A deep learning structure can help pick the best proxy (model selection), combine proxies (dimension reduction), or create the best proxy for the non-arbitrage objective.

For example, multiple momentum factors exist: long-term reversal, short-term reversal, the Carhart Momentum, seasonality, industry momentum, and so forth. All of these similar momentum characteristics are (weighted) sums of past individual security returns. Therefore, the raw inputs z_{t-1} in Figure 3.1 are past returns in purple circles. These momentum characteristics are "calculated" or "combined" from raw inputs and become the new inputs for the actual characteristics sorting. The blue circles in the hidden layers might include many trial-and-error experiments or manual "deep" learning for data mining concerns. The second-to-the-last layer combines the long-short portfolio weights, W, and individual security returns to generate the long-short factors. Figure 3.1 provides the procedure for calculating characteristics and creating factors. The deep learning philosophy has been adopted in

Figure 3.1: Sorting Securities and Generating Factors

This figure provides the procedure for calculating characteristics and creating factors. We start with past returns as raw inputs, and then calculate characteristics as the new inputs for security sorting. The last layer is the factor generation on long-short portfolio weights obtained from the previous layer plus individual security returns.



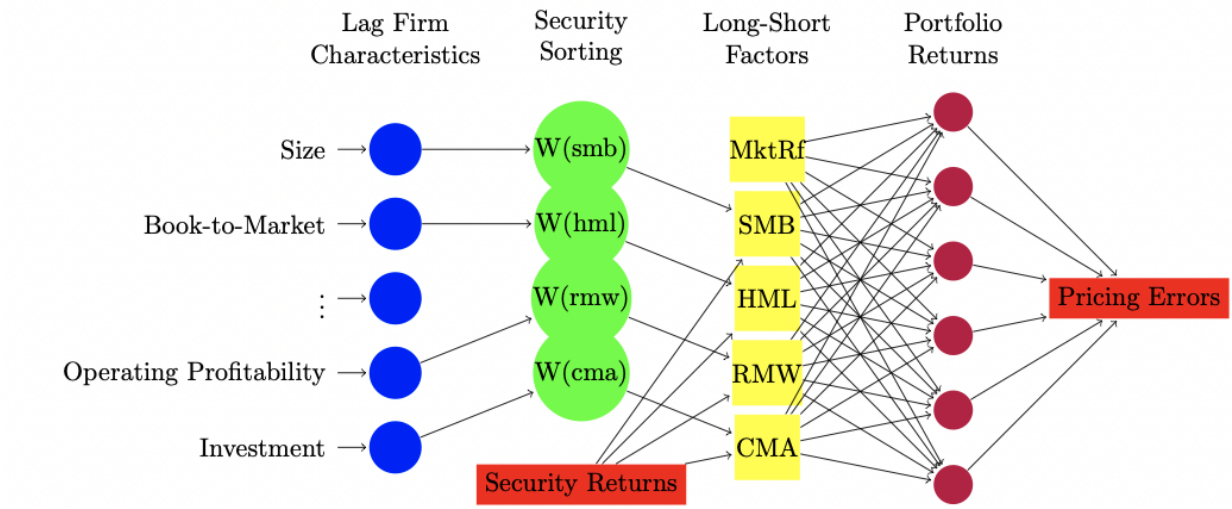
empirical asset pricing for a long time but is manually implemented.

Figure 3.2 shows a complete deep learning architecture for a characteristics-sorted factor model, which is the example for the Fama-French five-factor model. Researchers typically start with a formula to calculate characteristics used for security sorting, as in the blue circles. Then, they sort individual firms on the lag characteristics to determine the long-short portfolio weights as in the green circles.[1] Then, in the yellow rectangles, researchers construct factors as long-short portfolios using the portfolio weights from the last layer along with realized security returns. Adding the market factor produces an augmented factor model

1. If a firm does not exist or has missing characteristics in some periods, it is not included in the security sorting for those periods. Therefore, security sorting works perfectly for the imbalanced panel data structure with missing values, which is the nature of firm dynamics.

Figure 3.2: Fama-French 5-Factor Model in Deep Learning

This figure provides a deep learning representation of building the Fama-French five-factor model using firm characteristics to calculate the objective function, pricing errors, for portfolio returns. The lag characteristics are inputs. The long-short factors are hidden neurons. The portfolio returns are outputs.



to explain realized returns of test assets in the purple circles. The last red rectangle is the objective function for pricing errors.

In our notation, g_t is MktRf if CAPM is the benchmark. The Fama-French five-factor model adds four additional factors, f_t, to the benchmark. Those four characteristics (size, book-to-market, operating profitability, and investment) are z_t. W_t is determined with the bivariate-sorting directions and the lag market equity for value weights. In the standard literature, these four additional factors are tested with significance over CAPM with test assets in purple circles, 25 size-B/M portfolios. In our deep learning diagram, these four additional factors are trained by controlling the benchmark model CAPM to minimize the objective function, pricing errors.

The potential multi-layer transformations and combinations, denoted by $H(\cdot)$, of characteristics are determined before the blue circles. Here, researchers typically choose the formula for anomalies that help pricing in the cross-section. A significant drawback of this

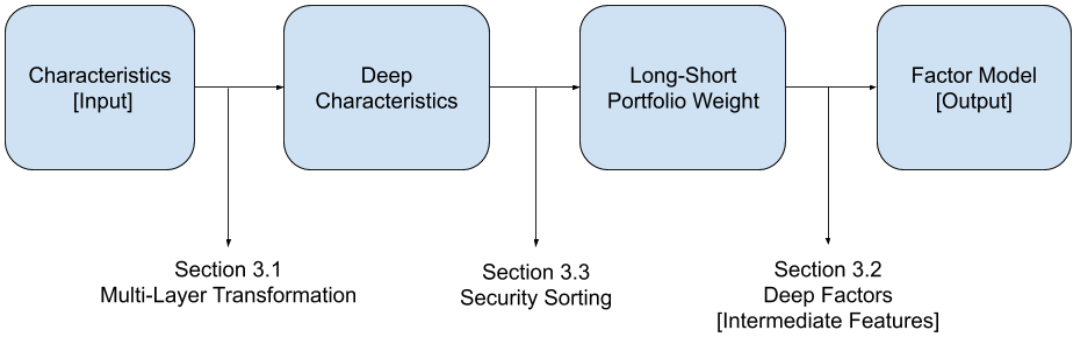
approach is that the characteristics' usefulness is tested ex-post statistically, but the model fitting feedback is never returned to characteristics' construction. With the new technology of backward propagation, our model can be refitted sequentially by the feedback on the change in the objective function.

3.3 Deep Learning in Characteristics-Sorted Factor Models

This section introduces a bottom-up approach to our deep learning model, which provides a non-reduced-form mechanism. Figure 3.3 shows a clear roadmap for how we disassemble the characteristics-sorted factor model within deep learning. Section 3.3.1 illustrates how the dimension reduction on the *[inputs]* firm characteristics performed in the feed-forward neural network via multi-layer transformations. Then, we get the deep characteristics. Section 3.3.2 calculates the *[intermediate features]* deep factors, whose long-short portfolio weights are calculated in section 3.3.3. Section 3.3.4 describes the optimization objective and summarizes the complete deep learning model. In Appendix 3.6.1, we also provide the optimization details.

In section 3.3.3, one can simply adopt equal weights to create the long-short factors. For this reason, we put the weighting scheme of section 3.3.3 after the factor model in section 3.3.2 of the below text. However, we suggest adopting our softmax rank-weighting scheme from an optimization perspective, which is differentiable and provides an economic-driven weighting scheme. We also explain why the neural network optimization requires a differentiable activation function in Appendix 3.6.1.

Figure 3.3: Map for Deep Learning Model Description



A typical training observation indexed by time t includes five types of data:

$$\{R_{i,t}\}_{i=1}^{N}, \text{ excess returns of } N \text{ test portfolios}$$
$$\{r_{j,t}\}_{j=1}^{M}, \text{ excess returns of } M \text{ individual stocks}$$
$$\{z_{k,j,t-1} : 1 \le k \le K\}_{j=1}^{M}, \ K \text{ lagged characteristics of } M \text{ firms}$$
$$\{g_{d,t}\}_{d=1}^{D}, \ D \text{ benchmark factors.}$$

We use a matrix notation for $\{R_t, r_t, z_{t-1}, g_t\}$, where R_t is an $N \times 1$ vector, r_t is an $M \times 1$ vector, z_{t-1} is a $K \times M$ matrix, and g_t is a $D \times 1$ vector. In section 3.4.1, we have $M = 3,000$ stocks, $K = 61$ characteristics, and $D = 1$ or 3 for CAPM or Fama-French 3-factor model. Before introducing each part of the deep learning implementation, we provide a summary of parameter notations and dimensions in Table 3.1.

3.3.1 Deep Characteristics

We introduce how to design an L-layer neural network to generate P deep characteristics. This operation is the "deep" part to induce nonlinearity and interaction within the dimension reduction from K to P characteristics. All transformations performed in this part are within each stock. The data (and intermediate results) of two different stocks are separated and

Table 3.1: Deep Learning Mechanism

This table provides an algorithm summary for the bottom-up approach of our deep learning model. The deep learning network feeds forward from the bottom to the top in the table. The initial input is firm characteristics, and the final outputs are security returns. For each layer, the network takes the output from the immediate lower layer as its new inputs and the additional input if needed. The other inputs include individual security returns r for deep factors and the benchmark factor model g for security returns.

	Dimension	Output	Inputs	Operation	Parameters
Security Returns	$N \times 1$	$\hat{R}$	g	$\beta f + \gamma g$	(β, γ)
Deep Factors	$P \times 1$	f	r	Wr	
Rank Weights	$P \times M$	W		$\text{softmax}(y^+) - \text{softmax}(y^-)$	
Deep Characteristics	$K_L \times M$	Y		$F^{[L]}\left(Z^{[L-1]}\right)$	$(A^{[L]}, b^{[L]})$
	$\vdots$	$\vdots$		$\vdots$	$\vdots$
	$K_l \times M$	$Z^{[l]}$		$F^{[l]}\left(Z^{[l-1]}\right)$	$(A^{[l]}, b^{[l]})$
	$\vdots$	$\vdots$		$\vdots$	$\vdots$
Firm Characteristics	$K_0 \times M$	$Z^{[0]}$	z	$Z^{[0]} := z$	

don't interfere with each other. We drop for now the subscript t, bearing in mind that the inputs z are lagged variables. The architecture is as follows, for $j = 1, 2, ..., M$:

$$
\begin{aligned}
&Z^{[l]}_{\cdot,j} = F\left(A^{[l]} Z^{[l-1]}_{\cdot,j} + b^{[l]}\right), \text{ for } l = 1, 2, ..., L \\
&Z^{[0]}_{\cdot,j} := [z_{1,j}, ..., z_{K,j}]^\intercal,
\end{aligned}
$$

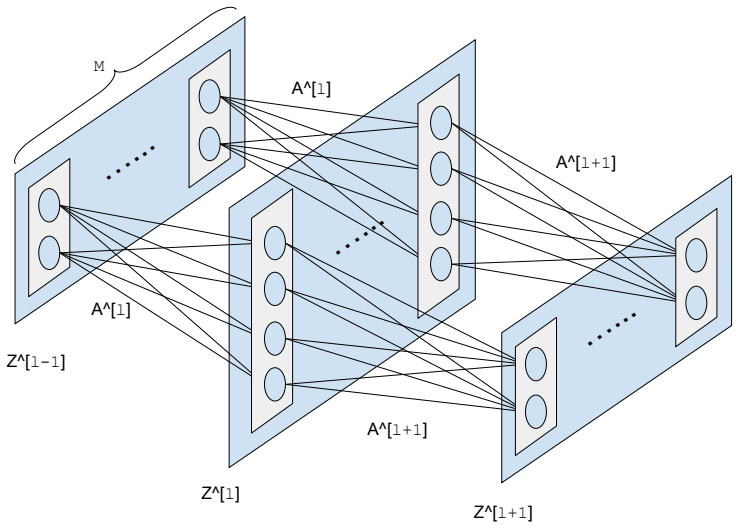
where $Z^{[l]}_{\cdot,j}$ is the j-th column of a $K_l \times M$ matrix $Z^{[l]}$. We set $K_0 = K$ and $K_L = P$. F is the univariate activation function, broadcasting to every element of a matrix. The parameters to be trained in this part are deep learning weights A's and biases b's, namely,

$$
\left\{(A^{[l]}, b^{[l]}) : A^{[l]} \in \mathbb{R}^{K_l \times K_{l-1}}, b^{[l]} \in \mathbb{R}^{K_l}\right\}_{l=1}^{L}.
$$

The transformations are performed column by column with no communication across different firms. This univariate transformation is perfectly built for the security sorting for

Figure 3.4: Deep Network of $Z^{[l-1]} \to Z^{[l]} \to Z^{[l+1]}$.

This figure shows how the deep learning network forwards from $Z^{[l-1]}$ to $Z^{[l+1]}$. $K_{l-1} = K_{l+1} = 2$, $K_l = 4$. The lines connecting two layers represent affine transformation, and the circles represent activation function.



different stock universes. Notice the input layer for deep characteristics is a linear function for firm characteristics. The multi-layer structure helps train the parameters for this linear equation. Our deep characteristics are not built for one particular characteristic but rather for the linear combinations.

With some abuse of notation, we rewrite the architecture for the output Y as our $P \times M$ deep characteristics,

$$\begin{aligned} Y &:= Z^{[L]}, \\ Z^{[l]} &= F\Big(A^{[l]} Z^{[l-1]} + b^{[l]}\Big), \text{ for } l = 1, 2, 3, ..., L \\ Z^{[0]} &:= z. \end{aligned}$$

Unlike a standard feed-forward neural network, the l-th layer in our architecture is a neural matrix $Z^{[l]}$. Each row of $Z^{[l]}$ is a $1 \times M$ vector representing the k_l-th intermediary characteristics for M firms, $k_l = 1, 2, ..., K_l$. We explicitly make all the columns (firms) share

the same parameters $A^{[l]}$ and $b^{[l]}$, whose dimensions are independent of M. Therefore, the formula for deep characteristics is the same for every firm.

Here, K_l denotes the dimension of the l-th layer because the number of columns is fixed as M for all $Z^{[l]}$'s. Figure 3.4 illustrates how our deep-learning network operates by showing a sample architecture from the $(l-1)$-th to the $(l+1)$-th layer, where $K_{l-1} = K_{l+1} = 2$ and $K_l = 4$. The Fama-French approach drops all hidden layers and uses $Y := z$ for sorting in the latter part. By contrast, $Z^{[0]} := z$ in our deep network goes through multiple layers of affine transformations and nonlinear activations and ends up with a low-dimensional deep characteristic Y. Here, the layer sizes $\{K_l\}_{l=1}^{L}$, and the number of layers L are architecture parameters chosen by model designers.

3.3.2 Deep Factors

In this section, we continue with the construction of deep factors based on long-short portfolio weights W (discussed in section 3.3.3), and then an augmented factor model for asset pricing. To create the long-short factors, we need the individual stock returns and the corresponding weights. The architecture after obtaining W is as follows:

$$\hat{R} := Z^{[L+3]} = h^{[2]}\left(Z^{[L+2]}, g\right) \tag{3.13}$$

$$f := Z^{[L+2]} = h^{[1]}\left(Z^{[L+1]}, r\right) \tag{3.14}$$

$$W := Z^{[L+1]} \tag{3.15}$$

Here, $h^{[1]}$ and $h^{[2]}$ are no longer univariate activation functions. Instead, they are operators specially defined to conduct important transformations, which take two arguments: one from the previous layer and another from additional inputs.

We now describe these operators in detail. $h^{[2]} : \mathbb{R}^P \times \mathbb{R}^D \to \mathbb{R}^N$ is a linear transforma-

tion of its two arguments, and the parameters are denoted as $\beta \in \mathbb{R}^{N\times P}$ and $\gamma \in \mathbb{R}^{N\times D}$:

$$h^{[2]}(f, g) = [\beta\ \gamma] \begin{bmatrix} f \\ g \end{bmatrix}. \tag{3.16}$$

Therefore, g represents the benchmark model, such as Fama-French three factors. $h^{[2]}$ is the augmented factor model by adding our deep factors f along with g. $h^{[1]} : \mathbb{R}^{P\times M} \times \mathbb{R}^{M} \to \mathbb{R}^{P}$ defines how we construct deep factors as tradable portfolios. Once given the portfolio weights W and individual stock returns r, it is simply a matrix production:

$$h^{[1]}(W, r) = Wr. \tag{3.17}$$

The tradability for factor and individual stock returns $\{f, g, r\}$ is crucial to determine our economic-driven loss function, which follows the non-arbitrage condition.

3.3.3 Nonlinear Rank Portfolio Weights

The design of long-short portfolio weights is presented in this section. Researchers usually long (and short) top (bottom) 10% or 20% of stocks for equal or value weights to create factors. However, two recent papers adopt the rank weights for creating factors in a general discussion. Frazzini and Pedersen (2014) construct their factor, Betting-against-Beta, with a "rank weighting." They assign each stock to either the "high" portfolio or the "low" portfolio with a weight proportional to the cross-sectional rank of the stock's estimate beta. Novy-Marx and Velikov (2018) add a further discussion to compare different portfolio weighting schemes: rank (linear) weights versus equal weights.

Following these studies, our procedure here generalizes the standard equal weights and introduces the nonlinearity. We define

$$h^{[0]} : \mathbb{R}^{M} \to [-1, 1]^{M}$$

to calculate the portfolio weights based on the rankings of deep characteristics. When the argument is a matrix, it broadcasts to all rows. Let y be a $M \times 1$ vector representing some deep characteristic, that is, a row of Y.

$$h^{[0]}(y) = \underbrace{\begin{bmatrix} \text{softmax}(y_1^+) \\ \text{softmax}(y_2^+) \\ \vdots \\ \text{softmax}(y_M^+) \end{bmatrix}}_{\text{long portfolio}} - \underbrace{\begin{bmatrix} \text{softmax}(y_1^-) \\ \text{softmax}(y_2^-) \\ \vdots \\ \text{softmax}(y_M^-) \end{bmatrix}}_{\text{short portfolio}} \tag{3.18}$$

where $y^+ := y, y^- := -y$ in the simplest case and the nonlinear softmax activation function is an increasing function,

$$\text{softmax}(y_j) = \frac{e^{y_j}}{\sum_{j'=1}^{M} e^{y_{j'}}},$$

and $\sum_{j=1}^{M} \text{softmax}(y_j) = 1$. The first softmax vector in the expression of $h^{[0]}$ represents the weights of stocks in the long portfolio (large y leads to large weight), and the second vector represents the short portfolio (large y leads to small weight). To prevent the exponential operator in softmax from introducing asymmetry and exaggerating the effect of extreme values, we need to first standardize y along the same axis and apply an additional nonlinear transformation before feeding into $h^{[1]}$.

To demonstrate properties of the rank-weight scheme, we use the following example. The left panel of Figure (3.5) shows the final output of $h^{[0]}$ (red line), the portfolio weights W, when

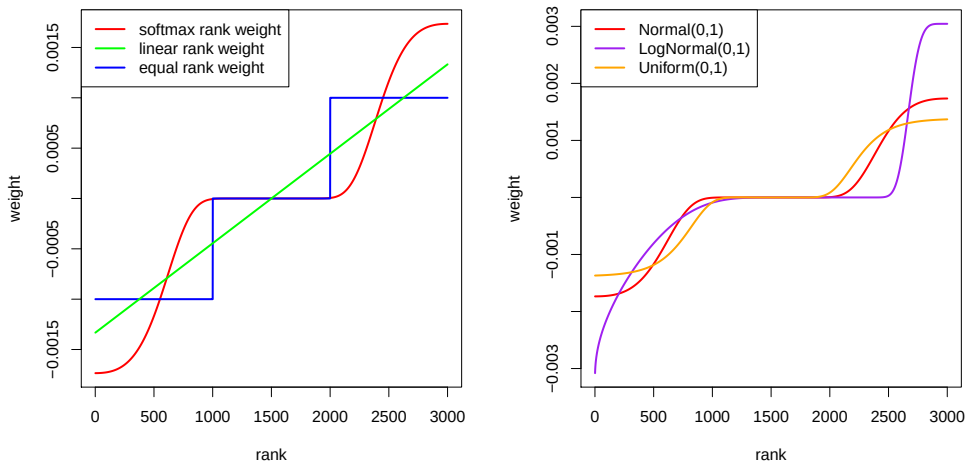
$$y^+ = -50e^{-5y}, y^- = -50e^{5y},$$

and $y = [y_1, y_2, ..., y_{3000}]^\intercal$ is drawn from standard normal distribution $N(0, 1)$. The x-axis shows the cross-sectional ranks of stocks.

For comparison, we also plot the standard equal weights (blue line) with the top and bottom 1/3 of stocks as well as the rank weights introduced by Frazzini and Pedersen (2014)

Figure 3.5: Comparison: Weight vs. Rank

This figure shows the example of softmax rank weights for 3,000 stocks, $h^{[0]}(y) = \text{softmax}(-50e^{-5y}) - \text{softmax}(-50e^{5y})$. In the right panel, y_j's are distributed as standard normal. The red line is the softmax weight; the blue line is the equal weight (with threshold $= 1/3$); the green line is the linear rank weights. In the left panel, the red line remains the same. The purple line is the softmax weights when y_j's are standardized samples from LogNormal$(0, 1)$. The orange line is the softmax weights when y_j's are standardized samples from Uniform$(0, 1)$.



(green line). Whereas their rank weights are linear in firms' cross-sectional rankings, our weighting scheme adds nonlinearity. In terms of actual holdings, Novy-Marx and Velikov (2018) point out that linear rank-weighted portfolios and equal-weighted portfolios are highly overlapped (83%). The departure of our nonlinear rank-weighted portfolio from these latter two portfolios is obvious: it "tilts" even more toward stocks with extreme characteristics. The flexibility of deep learning allows us to tune portfolio weights by changing the functional form of (y^+, y^-).

Unlike equal weights and linear rank weights, our softmax weights depend not only on cross-sectional rank information but also on distributional features (like skewness) for the standardized distribution. For illustration, in the right panel of Figure 3.5, we plot the softmax weights when characteristics are drawn from the skewed distribution LogNormal$(1, 3)$[2]

2. For example, all size-related characteristics follow a lognormal distribution.

(purple line) and the bounded distribution Uniform$(0,1)^3$ (orange line). Notably, the distribution of characteristics affects the symmetry and curvature of the weight curve. We see that compared with the standard normal case, uniform characteristics lead to more holdings of stocks with middle ranks and fewer holdings of stocks in the top and bottom. The log-normal distribution breaks the symmetry of weights in the long and short portfolios. In this case, the long portfolio only holds a small proportion of stocks in the right tail, and the short portfolio holds almost all stocks in the lower half but still favors those with smaller characteristics.

3.3.4 Minimizing Loss Function

The function H maps the lag predictors to the portfolio long-short weights,

$$W_{t-1} = H\Big(z_{t-1}\Big),$$

is essentially a composite given by $H(z) = h^{[0]} \circ F^{[L]} \circ \cdots \circ F^{[1]}(z)$. This multi-layer structure is the key idea of interpreting the security sorting as an activation function within a deep learner. In practice, we choose $F^{[1]} = F^{[2]} = ... = F^{[L]}$ to be the *tanh* function, i.e. $F(z) = (e^x - e^{-x})/(e^x + e^{-x})$.

Fixing L, $\{K_l\}_{l=1}^{L}$, which are architecture parameters, our objective function is the mean squared prediction error regularized by mean squared pricing error and absolute values of off-diagonal weights

$$\mathcal{L}_\lambda(\hat{A}, \hat{b}, \hat{\beta}, \hat{\gamma}) := \frac{1}{NT} \sum_{t=1}^{T} \sum_{i=1}^{N} \Big(R_{i,t} - \widehat{R}_{i,t}\Big)^2 + \frac{\lambda_1}{N} \sum_{i=1}^{N} \hat{\alpha}_i^2 + \lambda_2 \sum_{l=1}^{L-1} \sum_{i \neq j} |A_{i,j}^{[l]}|, \tag{3.19}$$

3. For example, characteristics such as performance scores follow a bounded distribution.

where

$$\widehat{R}_{i,t} = \hat{\beta}_i^{\intercal} f_t + \hat{\gamma}_i^{\intercal} g_t,$$

$$\hat{\alpha}_i = \frac{1}{T}\sum_{t=1}^{T}(R_{i,t} - \widehat{R}_{i,t}),$$

and

$$\hat{\beta} = \left[\hat{\beta}_1, \hat{\beta}_2, ..., \hat{\beta}_N\right]^{\intercal}, \ \hat{\gamma} = [\hat{\gamma}_1, \hat{\gamma}_2, ..., \hat{\gamma}_N]^{\intercal}.$$

Here, $\lambda = (\lambda_1, \lambda_2)$ are the regularization parameters. The penalization of the off-diagonal weights aims to stabilize the model and keep the combination of characteristics sparse. To train the deep network is then equivalent to obtaining a joint estimation of $(A, b) := \left\{A^{[l]}, b^{[l]}\right\}_{l=1}^{L}$ and (β, γ). The corresponding estimates are

$$(\hat{A}, \hat{b}, \hat{\beta}, \hat{\gamma}) = \arg\min \mathcal{L}_{\lambda}.$$

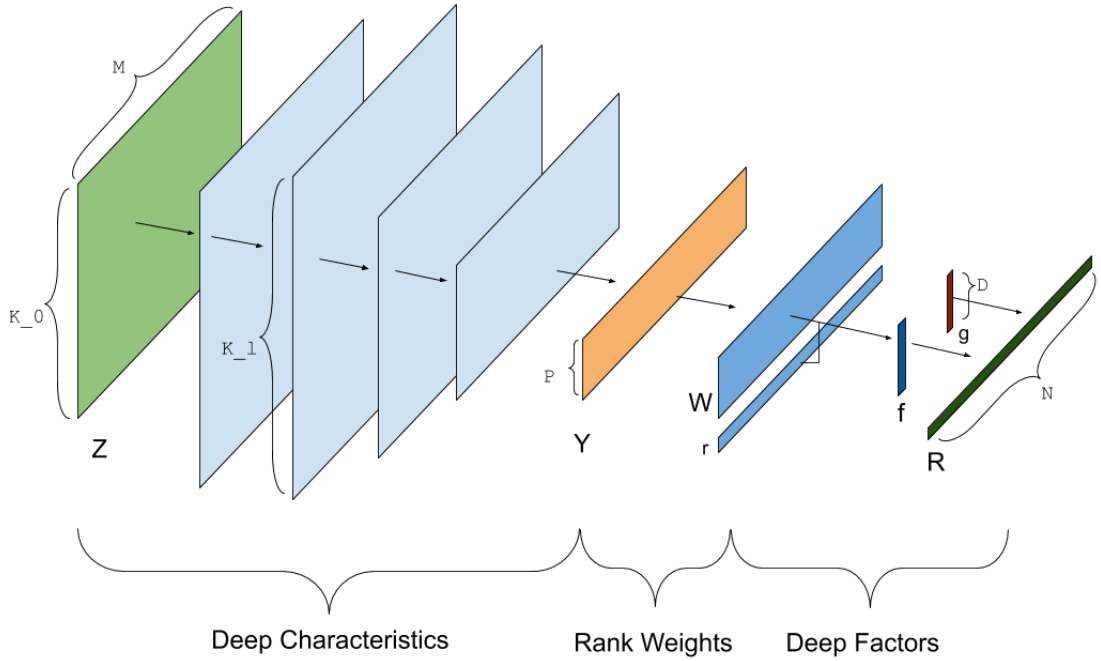
We summarize the above deep learning framework in Table 3.1 and Figure 3.6. Empirically, we set layer size K_l for the l-th layer all equal to K and the number of layers $1 \leq L \leq 4$. For example, a four-layer ($L = 4$) network has layer sizes $K - K - K - K - P$ from $Z^{[0]}$ to Y. We also set $P = 1$ to construct deep factors sequentially. At each time, we train the deep learning model with only one deep factor ($P = 1$), then augment g with this new factor and re-train the model to get the next one.

3.4 Empirical Findings

We report our empirical findings in this section. Section 3.4.1 describes the data and the training-test research design. Multiple asset pricing performance measures used in the model evaluation are introduced in Section 3.4.2. We present the results of statistical model fitness and asset pricing improvement in Section 3.4.3 and 3.4.4. Section 3.4.5 and 3.4.6 illustrate examples of investing deep factors and interpreting deep characteristics.

Figure 3.6: Deep Learning Network Architecture

This figure provides a visualization of deep learning architecture. The firm's characteristics $z$are transformed to deep characteristics Y via the deep network. Then, we "sort" Y to generate factor weight W. The deep factors f and benchmark factors g are used to price the asset return R.



3.4.1 Data and Train-Test Sample

Our monthly data sample is from January 1975 to December 2019. We follow the Fama-French factor in constructing individual stock filtering and using the largest 3,000 firms for lag market equity. We only include stocks for companies listed on the three main exchanges in the United States: NYSE, AMEX, or NASDAQ. We use those observations for firms with a CRSP share code of 10 or 11. We only include observations for firms listed for more than one year. We exclude observations with negative book equity or negative lag market equity.

We take 61 firm characteristics from six major categories: momentum, value, investment, profitability, frictions (or size), and intangibles. We follow Hou et al. (2020) to calculate the characteristic but make some minor differences because we choose to adopt a monthly sorting scheme. First, we modify the characteristics formulas, so they are updated monthly. Follow-

ing Kelly et al. (2019), We standardize the cross-section of firms' monthly characteristics in the range of $[-1, 1]$.[4]

We have the training sample from 1975 to 2009 and the test sample from 2010 to 2019. For the training assets, we fit the deep learning model with the monthly bivariate-sorted 3×2 portfolios between size and other characteristics ($3 \times 2 \times 60 = 360$). These bivariate-sorted portfolios are shown to have stable factor loadings in the literature. For the test assets, we try to show the robustness of our trained deep learning model. We also provide results for the monthly univariate-sorted 5×1 portfolios ($5 \times 1 \times 61 = 306$), Fama-French 25 size-B/M portfolios and the Fama-French 49-industry portfolios.

Testing the model performance on other assets or the test sample from 2010 to 2019 is a clean train-test design. To avoid overfitting, we have implemented a validation scheme to determine the best tuning parameters (number of factors and regularization penalties) for each layer's deep learning models. The deep factors are generated by fitting the cross-section of bivariate-sorted portfolio returns in the training sample from 1975 to 2009. We use the Fama-French 25 size-B/M portfolios in the training sample for the model validation, which is a common use in empirical asset pricing. We show the training and test sample design across the time dimension and asset dimension in Figure 3.7, where bivariate-sorted portfolios are training assets and Fama-French 25 size-B/M portfolios are validation assets.

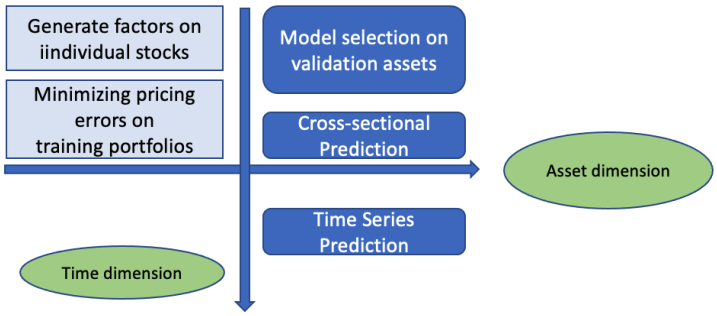
3.4.2 Performance Measures

We follow Kelly et al. (2019) and Kelly et al. (2020) to include multiple performance measures for economical and statistical purposes. Pricing Error R^2 and Cross-Sectional R^2 are designed for evaluating economical asset pricing performance, the non-arbitrage condition

4. For example, the market equity in 2018 December is uniformly standardized to $[-1, 1]$. The firm with the lowest market equity is -1, and the firm with the highest market equity is 1. Therefore, this uniform standardization is a non-standard standardization that transforms the data onto $[-1, 1]$ every month. If a firm has missing values for some characteristics, the imputed values are 0, which implies the firm is not important in security sorting.

Figure 3.7: Training and Test Sample Design

This figure shows the training and test sample design across the time dimension and asset dimension.



for zero alphas. Total R^2 and Predictive R^2 are designed for measuring statistical model fitness.

(1) Total R^2:

$$\text{Total } R^2 = 1 - \frac{\frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\left(R_{i,t} - \widehat{R}_{i,t}\right)^2}{\frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}R_{i,t}^2}, \tag{3.20}$$

where $\widehat{R}_{i,t} = \hat{\beta}_i^{\intercal} f_t + \hat{\gamma}_i^{\intercal} g_t$. The Total R^2 represents the fraction of realized return variation explained by the contemporaneous factor realizations, aggregated over all assets and all periods. It quantifies a model's success in describing the common risks in cross-sectional returns. In the training period, the factor loadings (β, γ) are only estimated once using all the training samples. And $\widehat{R}_{i,t}$ are calculated accordingly. In the test period, we estimate (β, γ) for each month, using the portfolio returns and factor values from the past 60 months.

(2) Predictive R^2:

$$\text{Predictive } R^2 = 1 - \frac{\frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}\left(R_{i,t} - \widehat{R}_{i,t}\right)^2}{\frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N}R_{i,t}^2}, \tag{3.21}$$

where $\widehat{R}_{i,t} = \hat{\beta}_i^{\intercal} \hat{\lambda}_f + \hat{\gamma}_i^{\intercal} \hat{\lambda}_g$. The risk premia estimates $\hat{\lambda}_f$ and $\hat{\lambda}_g$ are the 60-month

moving average for f_t and g_t. The factor loadings $(\hat{\beta}, \hat{\gamma})$ are the same as those in Total R^2 calculation. The Predictive R^2 represents the fraction of realized return variation explained by the model-implied expected returns. For a "predictive" measure, both factor loadings $[\hat{\beta}_i, \hat{\gamma}_i]$ and risk premia $[\hat{\lambda}_f, \hat{\lambda}_g]$ are estimated using the data from immediate past. In contrast to the Total R^2's focus on contemporaneous factors, this focuses on the fraction of panel return variation explained by the model-implied expected returns in the test sample.

(3) Pricing Error R^2:

$$\text{Pricing Error } R^2 = 1 - \frac{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}(R_{i,t} - \widehat{R}_{i,t})\right)^2}{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T} R_{i,t}\right)^2}, \tag{3.22}$$

where $\widehat{R}_{i,t} = \hat{\beta}_i^{\intercal} f_t + \hat{\gamma}_i^{\intercal} g_t$. The Pricing Error R^2 represents the fraction of the squared unconditional mean returns that is described by the common factors. The nominator shares the same objective as our deep learning augmented model. In contrast to the Total R^2's focus on the explanatory power in time series variation of asset returns, this focuses on the explanatory power in the cross-sectional variation of assets' average returns.

(4) Cross-Sectional R^2:

$$\text{Cross-Sectional } R^2 = 1 - \frac{\frac{1}{N}\sum_{i=1}^{N}\left(\bar{R}_i - \widehat{\bar{R}_i}\right)^2}{\frac{1}{N}\sum_{i=1}^{N}\bar{R}_i^2}, \tag{3.23}$$

where $\widehat{\bar{R}_i} = \hat{\beta}_i^{\intercal}\tilde{\lambda}_f + \hat{\gamma}_i^{\intercal}\tilde{\lambda}_g$. The risk premia estimates $\tilde{\lambda}_f$ and $\tilde{\lambda}_g$ are the cross-sectional regression coefficients by regressing $\bar{R}_i$ on $[\hat{\beta}_i, \hat{\gamma}_i]$. The Cross-Sectional R^2 is commonly used to represent the fraction of assets' average returns explained by the model-implied expected returns. In contrast to the Pricing Error R^2 that can be only used for tradable

factors, this can be used for both tradable and nontradable factors.

3.4.3 Statistical Model Fitness and Return Predictability

In table 3.2, we report the Total R^2 for our deep learning augmented models on different sets of test portfolios. We have produced results with two different benchmark models, CAPM and the Fama-French 3-factor model (FF3), to construct our augmented deep factors. The model is trained with the bivariate-sorted portfolios and selected with Fama-French 25 size-B/M portfolios' validation data. Hence, both univariate-sorted and Fama-French 49 industry portfolios are "unseen" and out-of-sample assets in the research design, even in the training sample. We show the validation selected model for each layer, so those four columns are independent of each other. According to the deep learning algorithm, we add factors one by one on the benchmark model. The model validation also determines the number of factors[5].

It is not surprised we find excellent Total R^2 improvement in the in-sample analysis. However, the positive progress in the out-of-sample analysis, where we fix the factor loadings, illustrates adding the deep factors is extremely useful to a weak benchmark model, like CAPM, but insignificantly improves a strong benchmark model, like FF3. We also add two strong benchmark models, Risk-Premium Principal Component Analysis (RP-PCA) of Lettau and Pelger (2020), and Instrumental Principal Component Analysis (IPCA) of Kelly et al. (2019), both include the information of high-dimensional characteristics in estimating principal components. However, both PCA models underperform our deep learning augmented model. These comparison results are pretty consistent across different sets of portfolios for different cross-sections.

Table 3.5 reports the Predictive R^2, which focuses on the cross-sectional variation ex-

5. For the benchmark of CAPM, there are 5, 3, 5, and 5 factors selected for 1-, 2-, 3- , and 4-layer models. For the benchmark of the Fama-French 3-factor model, there are 3, 4, 4, and 3 factors selected for 1-, 2-, 3- , and 4-layer models.

plained by the model-implied expected returns in the test sample, the recent decade. This performance measure is equivalent to those out-of-sample evaluations to the return predictability using zero as the denominator's benchmark forecast. We here to report the predictability for these factor models to test portfolio returns. Notice that it is usually difficult for an augmented large model to underperform a small model in the out-of-sample analysis due to higher estimation errors.

Different test portfolios have different signal-to-noise ratios for return predictability. CAPM and the Fama-French 3-factor model outperform the zero benchmark forecast and produce particular positive results to three sets of characteristics-sorted portfolios[6]. Notably, we still find our model robustly outperforms these two models by adding deep factors regarding all different sets of test portfolios. The two principal component methods' performances are mixed, while RP-PCA delivers similar performances as our deep learning augmented model. Finally, we find weak evidence that a shallow deep learning structure might produce more positive improvements than a deeper one.

3.4.4 Asset Pricing Improvement

In addition to the positive return predictability by the deep learning augmented models, we present the economic evidence in this subsection. Table 3.3 shares the same format and the Pricing Error R^2, whose nominator is the asset pricing models' objective. As discussed, even for the in-sample analysis, adding factors does not necessarily reduce the "alpha" and increase Pricing Error R^2. However, we find our model improves over CAPM and FF3 substantially in both in-sample and out-of-sample analysis. These are strong economic evidence for asset pricing improvement.

For example, in the out-of-sample study, when we fix the factor loadings and risk premia, the deep learning augmented models are still robust and outperform all other models. Again,

6. Consistent to the findings of Lewellen et al. (2010), we find CAPM and the Fama-French 3-factor model perform well in characteristics-sorted portfolios but have weak performance in 49 industry portfolios.

the Fama-French 3-factor model has good performances in characteristics-sorted portfolios but does not perform at the same level for the 49 industry portfolios. On the contrary, the deep learning augmented models have robustly positive performances to all test portfolios. For the 49 industry portfolios, we find at least 10% Pricing Error R^2 increases for all deep learning structures.

We also provide the Cross-Sectional R^2 in Table 3.4. This traditional asset pricing model measure explains the cross-sectional variation in average returns by the model-implied expected returns. If the asset pricing model is sufficiently good, the model implied expected return should line up with the average returns. Both RP-PCA and IPCA outperform the Fama-French 3-factor model for the in-sample analysis but underperform for the out-of-sample analysis. PCA's drawbacks include the fixed PC loadings in the training sample, which deteriorate in the test sample.

In Table 3.4, two deep learning augmented models even achieve higher Cross-Sectional R^2 in most cases for all different test portfolios. Regarding either the weak or strong benchmark model, adding deep factors boosts the cross-sectional variation explained in both in-sample and out-of-sample analysis. For the out-of-sample example of 49 industry portfolios, we find about 30% Cross-Sectional R^2 increases for all deep learning structures. This pure out-of-sample economic evidence demonstrates the asset pricing improvement for the deep learning augmented models.

3.4.5 Investing Deep Factors

We present how to use our deep factors and build a factor investing portfolio. We try to show the deep learning augmented factor model helps improve the portfolio performance. Kozak et al. (2020) show the portfolio performance for SDF coefficients on factors, which is

equivalent to the mean-variance efficient portfolio weights:

$$b = \Sigma_F^{-1}\mu_F,$$

where $F_t = \{f_t, g_t\}$. The efficient portfolio is simply $\{F_t b\}$ before the standardization.

The results for annualized Sharpe ratios are listed in the bottom panel of Table 3.5. Similar to the factor loadings, we estimate b using a rolling-window of the past 60 months. The annualized numbers for CAPM and FF3 are listed for the comparison. We find consistent numbers for RP-PCA and IPCA in the in-sample analysis. IPCA continues to produce a high Sharpe ratio consistently in the recent decade, and so do our models. Ours also have consistently high performances in both the training and test samples.

For the Sharpe ratio improvement in the nested model, Barillas et al. (2020) shows it is possible to apply a simple squared Sharpe ratio test for the null hypothesis $H_0 : SR_F = SR_g$. This model diagnostic test aims to evaluate the asset pricing model fitness improvement by adding f_t on the benchmark factors g_t. We have included the details for the test in Appendix 3.6.2. We only include the test significances in the bottom panel of Table 3.5. We find almost all deep learning augmented models over either CAPM or FF3 achieve improvement in the 1% significance level. This is another strong economic evidence to show adding our deep factors help asset pricing models for this significant improvement in the efficient portfolio.

3.4.6 Interpreting Deep Characteristics

Finally, for the interpretation of deep characteristics, one benefit of our non-reduced-form mechanism over a factor model is we can visualize the underlying nonlinear relationship among characteristics. The nonlinear activation of the neural network directly transforms raw characteristics into deep characteristics. To interpret such a "black box" model, we propose to visualize the nonlinear relationships through fitted smooth splines in Figure 3.8.

These are unique empirical outcomes by applying interaction or nonlinear transformation on characteristics instead of factors.

We take the example of the selected 1-layer model augmented by Fama-French three factors (five additional deep factors) and plot the fitted smooth splines and 95% confidence intervals to the firm characteristics of the Fama-French five factors. Even after controlling SMB and HML, we still find the deep characteristics plotted with a nonlinear relationship with size, value, (operating) profitability, and investment (asset growth). Notice that our characteristics data are standardized into the uniform distribution from -1 to 1. These plots illustrate the nonlinear patterns picked up in our deep learning augmented model to help achieve the asset pricing improvement.

Though the constructed deep characteristics are not like ordered principal components and lack the factor loading interpretation, we can evaluate the raw characteristics' contribution to each construction. We run the Fama-Macbeth cross-sectional regressions with deep characteristics on all raw characteristics. Given data of 540 months, we calculate the average explained variation. We provide the normalized explained variation in Table 3.6 to demonstrate the variable importance.

$$\text{deepchar}_{i,t} = a_t + b_{1,t}\text{char}_{1,i,t} + \cdots + b_{62,t}\text{char}_{61,i,t} + \epsilon_{i,t} \tag{3.24}$$

We find Dividend-to-Price has more than 10% weights on the 1st and 5th deep characteristics. Besides, Corporate Investment has a 24.69% weight on the 3rd deep characteristics. This is a second empirical outcome by applying dimension reduction on characteristics instead of factors. By controlling Fama-French three factors, the most important characteristics that reduce the pricing error are Dividend-to-Price and Corporate Investment. Unlike PCA to maximize variation in the data, our new perspective is directly related to the objective function of the non-arbitrage restriction. Dividend-to-Price and Corporate Investment might be nonlinear signals such that sorting securities on original data is not an optimal

case, where our deep learning model provides a solution.

There are additional useful characteristics, such as Abnormal Returns around Earning Dates (9.11% on 5th char.), Change in Tax Expense (7.24% on 2nd and 3rd char.), Net Operating Assets (8.17% on 2nd char.), and three volatility measures: residual variance w.r.t FF3, residual variance w.r.t. CAPM, and stock total variance (around 10% variation combined). These sparse signals are useful for the characteristics-sorted factor model by the deep learning transformation.

3.5 Summary and Discussion

In short, our goal is to introduce deep learning into the field of asset pricing. Most people view a deep neural network as a "black box" model. However, we adopt the deep learning framework with a bottom-up approach, which provides a non-reduced-form mechanism for the characteristics-sorted factor model. With a non-arbitrage objective to minimize pricing errors, we train a deep learning model using firm characteristics *[inputs]*, and generate risk factors *[intermediate features]* to fit the cross-section of security returns *[outputs]*. Our algorithm provides deep learning generated factors that reduce the pricing errors and show significant improvement in the efficient portfolio.

This paper is the first to provide a unified framework to implement the characteristics-sorted factor model to the best of our knowledge. We want to emphasize that our paper is not directly related to the literature on predicting asset returns using machine learning. The current prediction literature studies the time series predictive performance between firm characteristics *[inputs]* and security returns *[outputs]*, and skips the intermediate channel involved with risk factors *[intermediate features]*. Our bottom-up approach fills in this missing piece.

Moreover, we design the softmax activation on the technical side to approximate the long-short portfolio weights for factor generation. This procedure generalizes the "rank

weighting" scheme of Frazzini and Pedersen (2014) and Novy-Marx and Velikov (2018). Though equal- and value-weighted portfolios are widely used procedures, the cross-sectional distribution properties for different characteristics are largely omitted. When evaluating the long-short portfolio for a characteristic, the discussion of the long and short portfolio weights is necessary. Our method provides an alternative view on security sorting as well as factor generation.

Prediction and pattern matchings are essential applications for machine learning and deep learning. However, our paper shows the flexible optimization framework is also useful to researchers. If the current empirical test procedure always rejects the asset pricing test, stepping out of the comfort zone to look for new technologies could be beneficial. We have a chance to modify the objective function with an economic goal (minimizing pricing errors). We also can build up a non-reduced-form neural network to link together different pieces from square one.

3.6 Appendix

3.6.1 Optimization Details

This section shows how we minimize our objective function to train the deep learner. The common techniques include stochastic gradient descent (SGD), dropout, and ensemble learning. In the model training, we only apply SGD.

The new technology for deep learning that allows us to train such a complex bottom-up system is, the structure of the deep learner makes its objective function differentiable with respect to its parameters. The first-order derivative information is directly available by carefully applying the backward-chain rule. The TensorFlow library performs automatic derivative calculation for practitioners, allowing us to train the model using SGD.[7] Let the

7. See Robbins and Monro (1951), Kiefer and Wolfowitz (1952).

Table 3.2: Asset Pricing Explanatory Power: Total R^2

This table provides results for the total R^2, which is listed in equation 3.20. We show the in-sample and out-of-sample results in the top and bottom panels. We offer results for different layers of deep learning models augmented by CAPM and Fama-French three factors in each panel. The first row indicates the corresponding portfolios for calculating the total R^2. Also, we add results for RP-PCA of Lettau and Pelger (2020), IPCA of Kelly et al. (2019), CAPM, and Fama-French three factors for comparison.

In-Sample							
Portfolios	RP-PCA	IPCA	CAPM	$L=1$	$L=2$	$L=3$	$L=4$
Bivariate-Sorted	0.767	0.746	0.682	0.884	0.873	0.887	0.882
Univariate-Sorted	0.682	0.649	0.810	0.871	0.861	0.870	0.863
Fama & French 25	0.662	0.684	0.716	0.826	0.809	0.812	0.820
Industry 49	0.414	0.443	0.509	0.552	0.542	0.551	0.539
Portfolios	RP-PCA	IPCA	FF3	$L=1$	$L=2$	$L=3$	$L=4$
Bivariate-Sorted	0.767	0.746	0.847	0.904	0.907	0.906	0.895
Univariate-Sorted	0.682	0.649	0.869	0.891	0.888	0.890	0.887
Fama & French 25	0.662	0.684	0.909	0.920	0.920	0.918	0.915
Industry 49	0.414	0.443	0.559	0.579	0.578	0.580	0.575
Out-of-Sample							
Portfolios	RP-PCA	IPCA	CAPM	$L=1$	$L=2$	$L=3$	$L=4$
Bivariate-Sorted	0.675	0.662	0.775	0.892	0.907	0.907	0.903
Univariate-Sorted	0.570	0.622	0.854	0.881	0.890	0.892	0.885
Fama & French 25	0.617	0.642	0.784	0.848	0.860	0.863	0.866
Industry 49	0.346	0.386	0.517	0.513	0.528	0.516	0.498
Portfolios	RP-PCA	IPCA	FF3	$L=1$	$L=2$	$L=3$	$L=4$
Bivariate-Sorted	0.675	0.662	0.908	0.929	0.928	0.928	0.919
Univariate-Sorted	0.570	0.622	0.889	0.900	0.888	0.895	0.899
Fama & French 25	0.617	0.642	0.922	0.928	0.928	0.925	0.925
Industry 49	0.346	0.386	0.525	0.533	0.528	0.510	0.538

Table 3.3: Asset Pricing Explanatory Power: Pricing Error R^2

This table provides results for the pricing error R^2, which is listed in equation 3.22. We show the in-sample and out-of-sample results in the top and bottom panels. We offer results for different layers of deep learning models augmented by CAPM and Fama-French three factors in each panel. The first column indicates the corresponding portfolios used in the calculation. Also, we add results for RP-PCA of Lettau and Pelger (2020), IPCA of Kelly et al. (2019), CAPM, and Fama-French three factors for comparison.

In-Sample							
Portfolios	RP-PCA	IPCA	CAPM	$L=1$	$L=2$	$L=3$	$L=4$
Bivariate-Sorted	0.927	0.233	0.728	0.856	0.824	0.838	0.801
Univariate-Sorted	0.951	-0.477	0.769	0.802	0.781	0.787	0.734
Fama & French 25	0.901	0.311	0.691	0.826	0.780	0.780	0.724
Industry 49	0.601	-0.005	0.732	0.830	0.769	0.773	0.693
Portfolios	RP-PCA	IPCA	FF3	$L=1$	$L=2$	$L=3$	$L=4$
Bivariate-Sorted	0.927	0.233	0.845	0.844	0.838	0.850	0.852
Univariate-Sorted	0.951	-0.477	0.806	0.812	0.791	0.797	0.803
Fama & French 25	0.901	0.311	0.869	0.879	0.857	0.864	0.863
Industry 49	0.601	-0.005	0.808	0.821	0.785	0.795	0.791
Out-of-Sample							
Portfolios	RP-PCA	IPCA	CAPM	$L=1$	$L=2$	$L=3$	$L=4$
Bivariate-Sorted	0.313	-0.476	0.807	0.958	0.943	0.838	0.939
Univariate-Sorted	0.361	-0.244	0.921	0.981	0.982	0.977	0.967
Fama & French 25	0.492	-0.320	0.922	0.967	0.969	0.935	0.974
Industry 49	0.441	-0.035	0.679	0.860	0.871	0.823	0.794
Portfolios	RP-PCA	IPCA	FF3	$L=1$	$L=2$	$L=3$	$L=4$
Bivariate-Sorted	0.313	-0.476	0.918	0.963	0.951	0.951	0.962
Univariate-Sorted	0.361	-0.244	0.958	0.986	0.981	0.979	0.985
Fama & French 25	0.492	-0.320	0.981	0.972	0.973	0.974	0.979
Industry 49	0.441	-0.035	0.732	0.869	0.848	0.878	0.893

Table 3.4: Asset Pricing Explanatory Power: Cross-Sectional R^2

This table provides results for the cross-sectional R^2, which is listed in equation 3.23. We show the in-sample and out-of-sample results in the top and bottom panels. We offer results for different layers of deep learning models augmented by CAPM and Fama-French three factors in each panel. The first column indicates the corresponding portfolios used in the calculation. Also, we add results for RP-PCA of Lettau and Pelger (2020), IPCA of Kelly et al. (2019), CAPM, and Fama-French three factors for comparison.

In-Sample							
Portfolios	RP-PCA	IPCA	CAPM	$L=1$	$L=2$	$L=3$	$L=4$
Bivariate-Sorted	0.540	0.594	0.041	0.368	0.275	0.362	0.242
Univariate-Sorted	0.630	0.536	0.158	0.321	0.198	0.310	0.184
Fama & French 25	0.873	0.859	0.126	0.732	0.643	0.782	0.749
Industry 49	0.187	0.222	0.015	0.090	0.096	0.087	0.107
Portfolios	RP-PCA	IPCA	FF3	$L=1$	$L=2$	$L=3$	$L=4$
Bivariate-Sorted	0.540	0.594	0.225	0.330	0.324	0.275	0.278
Univariate-Sorted	0.630	0.536	0.188	0.327	0.248	0.237	0.222
Fama & French 25	0.873	0.859	0.739	0.827	0.840	0.904	0.839
Industry 49	0.187	0.222	0.063	0.101	0.116	0.248	0.175
Out-of-Sample							
Portfolios	RP-PCA	IPCA	CAPM	$L=1$	$L=2$	$L=3$	$L=4$
Bivariate-Sorted	0.225	0.262	0.335	0.428	0.438	0.445	0.482
Univariate-Sorted	0.152	0.138	0.081	0.494	0.451	0.508	0.515
Fama & French 25	0.221	0.195	0.009	0.648	0.368	0.466	0.553
Industry 49	0.299	0.257	0.009	0.572	0.632	0.535	0.621
Portfolios	RP-PCA	IPCA	FF3	$L=1$	$L=2$	$L=3$	$L=4$
Bivariate-Sorted	0.225	0.262	0.347	0.529	0.550	0.535	0.510
Univariate-Sorted	0.152	0.138	0.347	0.436	0.452	0.434	0.447
Fama & French 25	0.221	0.195	0.409	0.603	0.682	0.557	0.561
Industry 49	0.299	0.257	0.394	0.648	0.670	0.639	0.659

Table 3.5: Asset Pricing Predictive Power: Predictive R^2 and Sharpe Ratio

This table provides results for the predictive R^2, listed in equation 3.21, and the Sharpe Ratio for the efficient portfolio. We offer results for different layers of deep learning models augmented by CAPM and Fama-French three factors in each panel. The first column indicates the corresponding portfolios used in the calculation. In addition, we add results for RP-PCA of Lettau and Pelger (2020), IPCA of Kelly et al. (2019), CAPM, and Fama-French three factors for comparison. We also conduct a squared Sharpe ratio test of Barillas et al. (2020) to show the significances of nested asset pricing model improvement. Respectively, $***$ is 1%, $**$ is 5%, and $*$ is 10%.

Predictive R2 (Out-of-Sample)							
Portfolios	RP-PCA	IPCA	CAPM	$L=1$	$L=2$	$L=3$	$L=4$
Bivariate-Sorted	0.014	-0.028	0.005	0.016	0.014	0.009	0.015
Univariate-Sorted	0.023	-0.027	0.021	0.026	0.025	0.025	0.025
Fama & French 25	0.025	-0.026	0.008	0.015	0.013	0.010	0.015
Industry 49	0.009	-0.018	0.001	0.005	0.006	0.002	0.006
Portfolios	RP-PCA	IPCA	FF3	$L=1$	$L=2$	$L=3$	$L=4$
Bivariate-Sorted	0.014	-0.028	0.012	0.016	0.012	0.015	0.015
Univariate-Sorted	0.023	-0.027	0.023	0.026	0.025	0.026	0.025
Fama & French 25	0.025	-0.026	0.015	0.015	0.015	0.015	0.014
Industry 49	0.009	-0.018	0.003	0.005	0.001	0.009	0.005
Annualized Sharpe Ratio							
	RP-PCA	IPCA	CAPM	$L=1$	$L=2$	$L=3$	$L=4$
In-Sample	4.132	4.757	0.453	1.264***	0.965***	0.791**	1.046***
Out-of-Sample	1.896	2.478	0.708	1.265***	0.773**	1.807***	0.551
	RP-PCA	IPCA	FF3	$L=1$	$L=2$	$L=3$	$L=4$
In-Sample	4.132	4.757	0.824	1.053**	1.394***	1.054**	1.052**
Out-of-Sample	1.896	2.478	0.754	0.812**	1.031**	1.014***	0.954**

Figure 3.8: Deep Characteristics v.s. Fama-French Characteristics

This figure provides the marginal fitted smooth splines for our constructed deep characteristics (1-layer model augmented by Fama-French three factors) to four Fama-French characteristics. These smooth splines are plotted at the firm-level in the cross-section, and dotted lines are the corresponding 95% confidence intervals.

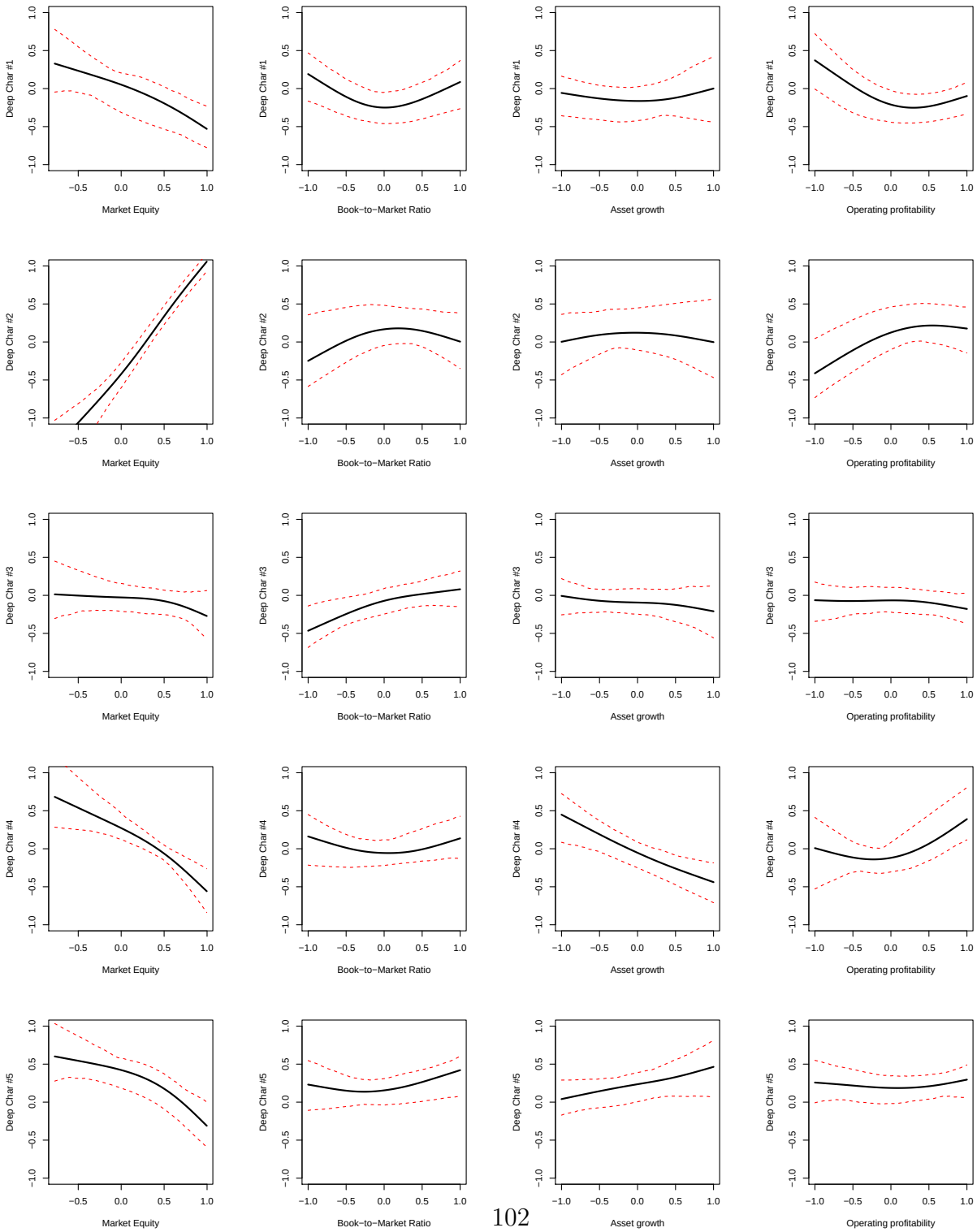


Table 3.6: Explained Variation for Deep Characteristics

Variable	Char #1	Char #2	Char #3	Char #4	Char #5
Abnormal Returns around Earnings Dates	4.10%	4.55%	1.38%	4.93%	9.11%
Working Capital Accruals	0.87%	2.65%	3.92%	4.12%	0.14%
Advertising Expense-to-market	0.27%	2.02%	2.98%	0.38%	0.02%
Asset Growth	2.18%	1.22%	1.11%	1.34%	0.16%
Asset Liquidity	0.44%	0.67%	2.55%	2.31%	0.19%
Asset Turnover	7.31%	0.06%	0.60%	2.98%	1.36%
Bid-Ask Spread (3m)	0.09%	0.03%	2.63%	0.08%	2.65%
Beta (3m)	0.03%	0.05%	2.26%	0.05%	3.90%
Book to Market	0.08%	0.10%	3.11%	0.44%	2.48%
Industry-adjusted Book to Market	3.02%	1.95%	7.47%	3.82%	0.04%
Cash Holdings	0.73%	0.87%	0.28%	0.03%	0.64%
Cash Flow to Debt	1.10%	0.20%	0.65%	0.81%	0.88%
Cash Flow to Price ratio	0.40%	0.46%	0.20%	2.76%	0.66%
Change in Shares Outstanding	0.64%	0.15%	2.39%	1.21%	0.13%
Change in Profit Margin	0.57%	1.85%	1.28%	0.76%	0.92%
Change in Tax Expense	1.81%	7.24%	7.24%	0.03%	4.20%
Corporate Investment	0.80%	0.30%	24.69%	1.29%	0.81%
Depreciation / PP&E	0.01%	0.26%	1.62%	5.52%	0.02%
Dollar Trading Volume	2.76%	6.57%	0.20%	1.71%	3.28%
Dividend to Price	11.03%	1.83%	0.05%	0.39%	10.74%
Earnings to Price	0.57%	1.71%	0.19%	3.41%	0.13%
Gross Profitability	1.80%	3.81%	0.21%	2.79%	0.15%
Growth in Long-Term Net Operating Assets	4.66%	0.94%	0.02%	3.32%	0.62%
Industry Concentration	0.79%	0.92%	2.14%	1.97%	0.37%
Employee Growth Rate	3.68%	0.52%	0.07%	0.15%	1.22%
Illiquidity (3m)	1.07%	2.59%	5.48%	0.49%	1.97%
Leverage	1.28%	2.33%	0.17%	1.25%	0.17%
Growth in Long-Term Debt	3.31%	0.06%	0.03%	0.55%	0.16%
Maximum Daily Return	0.73%	0.17%	0.73%	0.75%	4.99%
Market Equity	0.55%	3.18%	0.29%	0.28%	1.00%
Industry-Adjusted Market Equity	6.00%	2.48%	0.08%	0.98%	0.53%
Momentum (2-12 month)	0.12%	1.41%	0.88%	1.58%	1.62%
Short-Term Reversal (1-1 month)	2.38%	0.34%	0.90%	0.02%	2.19%
Momentum 36m (13-36 month)	1.46%	0.08%	0.09%	1.79%	0.72%
Long-Term Reversal (13-60 month)	0.02%	1.88%	0.17%	0.96%	0.05%
Momentum 6m (2-6 month)	1.39%	0.49%	0.08%	1.69%	0.49%
Net Stock Issues	0.12%	0.74%	0.07%	4.29%	0.37%
Number of Earnings Increases	0.32%	2.27%	0.69%	0.80%	0.34%
Net Operating Assets	1.88%	8.17%	1.02%	5.77%	1.10%
Operating Profitability	1.68%	4.63%	0.45%	6.93%	3.39%
Percent Accruals	0.27%	0.19%	0.06%	0.14%	0.10%
Profit Margin	1.53%	0.21%	0.55%	0.18%	0.07%
Performance Score	1.02%	0.03%	0.92%	0.20%	0.03%
R&D to Sales	0.58%	0.65%	3.60%	0.05%	0.70%
R&D to Market Capitalization	0.68%	1.85%	4.74%	0.56%	0.07%
Revisions in Analysts's Earnings Forecasts	4.10%	1.19%	0.93%	2.27%	0.05%
Return on Net Operating Assets	0.12%	1.82%	0.30%	0.53%	0.18%
Return on Assets	0.30%	0.11%	2.63%	0.15%	0.52%
Return on Equity	1.24%	0.46%	0.09%	2.66%	0.43%
Revenue Surprise	0.69%	0.14%	0.52%	1.00%	3.02%
CAPM Residual Variance (3 month)	3.70%	4.73%	4.39%	8.72%	5.45%
FF3 Residual Variance (3 month)	3.68%	3.42%	2.58%	2.59%	3.01%
Return Variance (3 month)	3.29%	1.54%	1.19%	0.87%	1.19%
Seasonality	0.09%	0.02%	0.49%	1.60%	0.30%
Sales Growth	0.59%	0.40%	0.12%	4.39%	0.45%
Sales to Price	0.42%	1.17%	2.26%	3.95%	0.63%
Std of Dollar Trading Volume (3 month)	1.30%	0.78%	0.44%	3.42%	0.09%
Std of Share Turnover (3 month)	2.20%	0.10%	0.23%	1.40%	0.59%
Unexpected Quarterly Earnings	4.22%	1.88%	1.48%	0.69%	6.20%
Shares Turnover	0.11%	0.17%	1.40%	2.37%	1.53%
Number of Zero-Trading Days (3 month)	0.56%	0.07%	0.39%	1.02%	0.03%

superscript (t) denote the t-th iterate. SGD updates the parameters by

$$\begin{bmatrix} \hat{A}^{(t+1)} \\ \hat{b}^{(t+1)} \\ \hat{\beta}^{(t+1)} \\ \hat{\gamma}^{(t+1)} \end{bmatrix} \longleftarrow \begin{bmatrix} \hat{A}^{(t)} \\ \hat{b}^{(t)} \\ \hat{\beta}^{(t)} \\ \hat{\gamma}^{(t)} \end{bmatrix} - \eta^{(t+1)} \nabla \mathcal{L}_{\lambda}^{(t)} \tag{3.25}$$

until convergence, where η is the step size, and the gradient is evaluated at $(\hat{A}^{(t)}, \hat{b}^{(t)}, \hat{\beta}^{(t)}, \hat{\gamma}^{(t)})$. At each iterate, the loss $\mathcal{L}_{\lambda}^{(t)}$ only involves a random subset of data, $\mathcal{B} \subset \{1, 2, ..., T\}$, called mini-batch,

$$\mathcal{L}_{\lambda}^{(t)}(A, b, \beta, \gamma) = \frac{1}{N|\mathcal{B}|} \sum_{t \in \mathcal{B}} \sum_{i=1}^{N} \left(R_{i,t} - \widehat{R}_{i,t} \right)^2 + \frac{\lambda_1}{N} \sum_{i=1}^{N} \left(\frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} (R_{i,t} - \widehat{R}_{i,t}) \right)^2 \tag{3.26}$$

$$+ \lambda_2 \sum_{l=1}^{L-1} \sum_{i \neq j} |A_{i,j}^{[l]}|, \tag{3.27}$$

where $|\mathcal{B}| < T$, and in practice we set $|\mathcal{B}| = 120$; namely, we use a batch of 120 months for training. This mini-batch setting on the time dimension is reasonable for the asset pricing factor model, which we usually assume with no serial correlation.

Also, we set the number of epochs (roughly the number of times SGD explores the whole training set) to be 300, because the objective function has stopped decreasing significantly. Adding too many epochs for model training can cause over-fitting. In our study, we consider 300 epochs a reasonable number.

3.6.2 Squared Sharpe Ratio Test in Barillas et al. (2020)

Since the benchmark model g is nested in the augmented model F, the Sharpe ratio of F is greater or equal to that of g. The improvement in the squared Sharpe ratio is a quadratic form as shown by Equation (2) in Barillas et al. (2020),

$$SR_F^2 - SR_g^2 = \alpha_f^{\intercal} \Sigma_{\epsilon}^{-1} \alpha_f$$

where α_f is the $N \times 1$ intercept vector from the pricing model

$$f_t = \alpha_f + \tilde{\beta} g_t + \epsilon_t,\ t = 1, 2..., T$$

and Σ_ϵ is covariance matrix of ϵ_t. Therefore, the simple test of equality in the Sharpe ratios is actually the GRS test, with the tradable deep factors f_t serving as left-hand-side test assets on the right-hand-side benchmark g_t.

Under the null hypothesis $H_0 : SR_F = SR_g$, i.e. $\alpha_f^\intercal \Sigma_\epsilon^{-1} \alpha_f = 0$, the GRS test statistic is proportional to the difference in squared sample Sharpe ratios divided by one plus the squared sample Sharpe ratio of g,

$$\left(\frac{T}{P}\right)\left(\frac{T-P-D}{T-D-1}\right)\frac{\widehat{SR}_F^2 - \widehat{SR}_g^2}{1+\widehat{SR}_g^2} \sim F(P, T-P-D)$$

$$\widehat{SR}_F^2 - \widehat{SR}_g^2 = \hat{\alpha}_f^\intercal \hat{\Sigma}_\epsilon^{-1} \hat{\alpha}_f$$

$$\widehat{SR}_g^2 = \bar{g}^\intercal \hat{\Sigma}_g^{-1} \bar{g},$$

where $\bar{g}, \hat{\Sigma}_g$ are the sample mean and covariance matrix of benchmark factors g_t.

REFERENCES

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Aktekin, T., N. Polson, and R. Soyer (2018). Sequential bayesian analysis of multivariate count data. *Bayesian Analysis 13*(2), 385–409.

Andrews, D. F. and C. L. Mallows (1974). Scale mixtures of Normal distributions. *Journal of the Royal Statistical Society, B*, 99–102.

Antoniak, C. E. (1974, 11). Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *Ann. Statist. 2*(6), 1152–1174.

Arnold, B. C. and S. J. Press (1989). Bayesian estimation and prediction for pareto data. *Journal of the American Statistical Association 84*(408), 1079–1084.

Arnold, T. B. and R. J. Tibshirani (2014). genlasso: Path algorithm for generalized lasso problems. *R package version 1*(3).

Bacák, M. (2014). *Convex analysis and optimization in Hadamard spaces*, Volume 22. Walter de Gruyter GmbH & Co KG.

Bansal, R., D. A. Hsieh, and S. Viswanathan (1993). A new approach to international arbitrage pricing. *The Journal of Finance 48*(5), 1719–1747.

Bansal, R. and S. Viswanathan (1993). No arbitrage and arbitrage pricing: A new approach. *The Journal of Finance 48*(4), 1231–1262.

Barbe, P. and P. Bertail (2012). *The weighted bootstrap*, Volume 98. Springer Science & Business Media.

Barillas, F., R. Kan, C. Robotti, and J. Shanken (2020). Model comparison with sharpe ratios. *Journal of Financial and Quantitative Analysis 55*(6), 1840–1874.

Barndorff-Nielsen, O., P. Blaesild, and V. Seshadri (1992). Multivariate distributions with generalized inverse gaussian marginals, and associated poisson mixtures. *The Canadian Journal of Statistics/La Revue Canadienne de Statistique*, 109–120.

Barndorff-Nielsen, O. and C. Halgreen (1977). Infinite divisibility of the hyperbolic and generalized inverse gaussian distributions. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete 38*(4), 309–311.

Barndorff-Nielsen, O., J. Kent, and M. Sørensen (1982). Normal variance-mean mixtures and Z distributions. *International Statistical Review/Revue Internationale de Statistique*, 145–159.

Barndorff-Nielsen, O. E. and N. Shephard (2012). *Basics of Lévy processes*. Citeseer.

Beaumont, M. A., J.-M. Cornuet, J.-M. Marin, and C. P. Robert (2009). Adaptive approximate bayesian computation. *Biometrika 96*(4), 983–990.

Bhattacharya, A., A. Chakraborty, and B. K. Mallick (2016). Fast sampling with Gaussian scale mixture priors in high-dimensional regression. *Biometrika*.

Bianchi, D., M. Büchner, and A. Tamoni (2020). Bond risk premia with machine learning. *The Review of Financial Studies, Forthcoming*.

Black, F., M. C. Jensen, and M. Scholes (1972). *The Capital Asset Pricing Model: Some Empirical Tests*. Praeger Publishers Inc.

Blei, D. M., A. Kucukelbir, and J. D. McAuliffe (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association 112*(518), 859–877.

Blei, D. M. and J. D. Lafferty (2006). Dynamic topic models. In *Proceedings of the 23rd international conference on Machine Learning*. ACM.

Blei, D. M., A. Y. Ng, and M. I. Jordan (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research 3*, 993–1022.

Bollerslev, T., R. F. Engle, and J. M. Wooldridge (1988). A capital asset pricing model with time-varying covariances. *Journal of political Economy 96*(1), 116–131.

Bondesson, L. (1982). On simulation from infinitely divisible distributions. *Advances in Applied Probability 14*(4), 855–869.

Bondesson, L. (1992). Generalized Gamma Convolutions and related classes of distributions and densities. *Lecture Notes in Statistics 76*.

Boyd, S., S. P. Boyd, and L. Vandenberghe (2004). *Convex optimization*. Cambridge university press.

Breiman, L. (2001). Random forests. *Machine learning 45*(1), 5–32.

Breiman, L. (2003). Statistical modeling: The two cultures. *Quality Control and Applied Statistics 48*(1), 81–82.

Brennan, M. J., T. Chordia, and A. Subrahmanyam (1998). Alternative factor specifications, security characteristics, and the cross-section of expected stock returns. *Journal of Financial Economics 49*(3), 345–373.

Campbell, J. Y. and S. B. Thompson (2007). Predicting excess stock returns out of sample: Can anything beat the historical average? *The Review of Financial Studies 21*(4), 1509–1531.

Carlin, B. P. and N. G. Polson (1991). Inference for Non-Conjugate Bayesian models using the Gibbs sampler. *Canadian Journal of Statistics 19*(4), 399–405.

Carlin, B. P., N. G. Polson, and D. S. Stoffer (1992). A Monte Carlo approach to Non-normal and nonlinear state-space modeling. *Journal of the American Statistical Association 87*(418), 493–500.

Chamberlain, G. and M. Rothschild (1983). Arbitrage, factor structure, and mean-variance analysis on large asset markets. *Econometrica: Journal of the Econometric Society*, 1281–1304.

Chen, J., J. Zhu, Z. Wang, X. Zheng, and B. Zhang (2013). Scalable inference for logistic-normal Topic Models. In *Advances in Neural Information Processing Systems*, pp. 2445–2453.

Chen, L., M. Pelger, and J. Zhu (2019). Deep learning in asset pricing. Technical report, Stanford University.

Chinco, A. M., A. D. Clark-Joseph, and M. Ye (2017). Sparse signals in the cross-section of returns. Technical report, National Bureau of Economic Research.

Chollet, F. et al. (2015). Keras.

Cochrane, J. H. (2008). The dog that did not bark: A defense of return predictability. *The Review of Financial Studies 21*(4), 1533–1575.

Cochrane, J. H. (2009). *Asset pricing: Revised edition*. Princeton university press.

Cochrane, J. H. (2011). Presidential address: Discount rates. *The Journal of Finance 66*(4), 1047–1108.

Connor, G. and R. A. Korajczyk (1986). Performance measurement with the arbitrage pricing theory: A new framework for analysis. *Journal of Financial Economics 15*(3), 373–394.

Connor, G. and R. A. Korajczyk (1988). Risk and return in an equilibrium apt: Application of a new test methodology. *Journal of Financial Economics 21*(2), 255–289.

Damien, P., P. W. Laud, and A. F. Smith (1995). Approximate random variate generation from infinitely divisible distributions with applications to Bayesian inference. *Journal of the Royal Statistical Society B*, 547–563.

Damsleth, E. (1975). Conjugate classes for Gamma distributions. *Scandinavian Journal of Statistics*, 80–84.

Daniel, K. and S. Titman (1997). Evidence on the characteristics of cross sectional variation in stock returns. *The Journal of Finance 52*(1), 1–33.

DeMiguel, V., A. Martin-Utrera, F. J. Nogales, and R. Uppal (2020). A transaction-cost perspective on the multitude of firm characteristics. *The Review of Financial Studies 33*(5), 2180–2222.

Diebold, F. X. and R. S. Mariano (2002). Comparing predictive accuracy. *Journal of Business & economic statistics 20*(1), 134–144.

Diebolt, J., M.-A. El-Aroui, M. Garrido, and S. Girard (2005). Quasi-conjugate bayes estimates for gpd parameters and application to heavy tails modelling. *Extremes 8*(1-2), 57–78.

Escobar, M. D. and M. West (1995). Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association 90*(430), 577–588.

Fabozzi, F. J. and J. C. Francis (1977). Stability tests for alphas and betas over bull and bear market conditions. *The Journal of Finance 32*(4), 1093–1099.

Fama, E. F. (2015). Cross-section versus time-series tests of asset pricing models. Technical report, University of Chicago.

Fama, E. F. and K. R. French (1992). The cross-section of expected stock returns. *The Journal of Finance 47*(2), 427–465.

Fama, E. F. and K. R. French (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics 33*(1), 3–56.

Fama, E. F. and K. R. French (1996). Multifactor explanations of asset pricing anomalies. *The Journal of Finance 51*(1), 55–84.

Fama, E. F. and K. R. French (2015). A five-factor asset pricing model. *Journal of Financial Economics 116*(1), 1 – 22.

Fama, E. F. and K. R. French (2016). Dissecting anomalies with a five-factor model. *The Review of Financial Studies 29*(1), 69–103.

Fama, E. F. and K. R. French (2018). Choosing factors. *Journal of Financial Economics 128*(2), 234–252.

Fama, E. F. and J. D. MacBeth (1973). Risk, return, and equilibrium: Empirical tests. *Journal of Political Economy 81*(3), 607–636.

Fan, J., Y. Liao, and W. Wang (2016). Projected principal component analysis in factor models. *Annals of statistics 44*(1), 219.

Fan, J., L. Xue, and J. Yao (2017). Sufficient forecasting using factor models. *Journal of Econometrics* (2), 292–306.

Feng, G., S. Giglio, and D. Xiu (2020). Taming the factor zoo: A test of new factors. *The Journal of Finance 75*(3), 1327–1370.

Feng, G. and J. He (2019). Factor investing: Hierarchical ensemble learning. Technical report, City University of Hong Kong.

Feng, G., J. He, and N. Polson (2019). Deep learning for predicting asset returns. Technical report, City University of Hong Kong.

Feng, G., N. Polson, and J. Xu (2020). Deep learning in characteristics-sorted factor models. Technical report, City University of Hong Kong.

Feng, G. G., A. Fulop, and J. Li (2020). Real-time macro information and bond return predictability: Does deep learning help? Technical report, City University of Hong Kong.

Ferguson, T. S. (1973, 03). A bayesian analysis of some nonparametric problems. *Ann. Statist. 1*(2), 209–230.

Ferguson, T. S. and M. J. Klass (1972). A representation of independent increment processes without Gaussian components. *Annals of Mathematical Statistics 43*(5), 1634–1643.

Frazzini, A. and L. H. Pedersen (2014). Betting against beta. *Journal of Financial Economics 111*(1), 1–25.

Freyberger, J., A. Neuhierl, and M. Weber (2020). Dissecting characteristics nonparametrically. *The Review of Financial Studies 33*(5), 2326–2377.

Friedman, J., T. Hastie, and R. Tibshirani (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software 33*(1), 1.

Gallant, A. R. and H. White (1988a). There exists a neural network that does not make avoidable mistakes. In *Proceedings of the Second Annual IEEE Conference on Neural Networks, San Diego, CA, I.*

Gallant, A. R. and H. White (1988b). *A unified theory of estimation and inference for nonlinear dynamic models*. Blackwell.

Gelman, A. and X.-L. Meng (1998). Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical science*, 163–185.

Gibbons, M. R., S. A. Ross, and J. Shanken (1989). A test of the efficiency of a given portfolio. *Econometrica: Journal of the Econometric Society*, 1121–1152.

Giglio, S. and D. Xiu (2017). Asset pricing with omitted factors. Technical report, Yale University.

Glynn, C. and E. B. Fox (2019). Dynamics of homelessness in urban America. *Annals of Applied Statistics 13*(1), 133–165.

Glynn, C., J. He, N. G. Polson, and J. Xu (2018). Scalable Logistic modeling via Pólya-Gamma sampling. *Technical report, University of Chicago*.

Glynn, C., S. T. Tokdar, B. Howard, and D. L. Banks (2019, 03). Bayesian analysis of dynamic linear topic models. *Bayesian Anal. 14*(1), 53–80.

Goodfellow, I., Y. Bengio, A. Courville, and Y. Bengio (2016). *Deep learning*, Volume 1. MIT press Cambridge.

Gordon, N. J., D. J. Salmond, and A. F. Smith (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, Volume 140, pp. 107–113. IET.

Green, J., J. R. Hand, and X. F. Zhang (2017). The characteristics that provide independent information about average us monthly stock returns. *The Review of Financial Studies 30*(12), 4389–4436.

Green, P. J., K. Łatuszyński, M. Pereyra, and C. P. Robert (2015). Bayesian computation: a summary of the current state, and samples backwards and forwards. *Statistics and Computing 25*(4), 835–862.

Gribonval, R. and P. Machart (2013). *Reconciling" priors" &" priors" without prejudice?* Ph. D. thesis, INRIA.

Griffin, J. E. and P. J. Brown (2010, 03). Inference with normal-gamma prior distributions in regression problems. *Bayesian Anal. 5*(1), 171–188.

Gu, S., B. Kelly, and D. Xiu (2020a). Autoencoder asset pricing models. *Journal of Econometrics*.

Gu, S., B. Kelly, and D. Xiu (2020b). Empirical asset pricing via machine learning. *The Review of Financial Studies 33*(5), 2223–2273.

Han, Y., A. He, D. Rapach, and G. Zhou (2018). What firm characteristics drive us stock returns? Technical report, Washington University in St. Louis.

Hartman, P. (1976). Completely monotone families of solutions of n-th order linear differential equations and infinitely divisible distributions. *Annali della Scuola Normale Superiore di Pisa-Classe di Scienze 3*(2), 267–287.

Harvey, C. R. and Y. Liu (2017). Lucky factors. Technical report, Duke University.

Harvey, C. R., Y. Liu, and H. Zhu (2016). ... and the cross-section of expected returns. *The Review of Financial Studies 29*(1), 5–68.

Hastie, T., B. Efron, and M. T. Hastie (2013). Package lars.

Hastie, T., R. Tibshirani, and M. Wainwright (2019). *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC.

Heaton, J., N. Polson, and J. H. Witte (2017). Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry 33*(1), 3–12.

Hinton, G. E. and R. R. Salakhutdinov (2006). Reducing the dimensionality of data with neural networks. *Science 313*(5786), 504–507.

Holmes, C. C. and L. Held (2006, 03). Bayesian auxiliary variable models for binary and multinomial regression. *Bayesian Analysis 1*, 145–168.

Hornik, K., M. Stinchcombe, and H. White (1989). Multilayer feedforward networks are universal approximators. *Neural Networks 2*(5), 359–366.

Hou, K., C. Xue, and L. Zhang (2020). Replicating anomalies. *The Review of Financial Studies 33*(5), 2019–2133.

Huang, D., J. Li, and G. Zhou (2018). Shrinking factor dimension: A reduced-rank approach. Technical report, Washington University in St. Louis.

Jacobs, B. I. and K. N. Levy (1993). Long/short equity investing. *The Journal of Portfolio Management 20*(1), 52–63.

Jacobs, B. I., K. N. Levy, and D. Starer (1999). Long-short portfolio management: An integrated approach. *The Journal of Portfolio Management 25*(2), 23–32.

Johnstone, I. M. (2010). High dimensional bernstein-von mises: simple examples. *Institute of Mathematical Statistics Collections 6*, 87.

Jones, C. S. (2006). A nonlinear factor analysis of s&p 500 index option returns. *The Journal of Finance 61*(5), 2325–2363.

Kan, R. and C. Robotti (2008). Model comparison using the hansen-jagannathan distance. *The Review of Financial Studies 22*(9), 3449–3490.

Kan, R., C. Robotti, and J. Shanken (2013). Pricing model performance and the two-pass cross-sectional regression methodology. *The Journal of Finance 68*(6), 2617–2649.

Kelly, B. T., D. Palhares, and S. Pruitt (2020). Modeling corporate bond returns. *Available at SSRN*.

Kelly, B. T., S. Pruitt, and Y. Su (2019). Characteristics are covariances: A unified model of risk and return. *Journal of Financial Economics 134*(3), 501–524.

Kiefer, J. and J. Wolfowitz (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 462–466.

Kim, S., R. A. Korajczyk, and A. Neuhierl (2018). Arbitrage portfolios in large panels. Technical report, Georgia Institute of Technology.

Kleijn, B. J., A. W. van der Vaart, et al. (2012). The bernstein-von-mises theorem under misspecification. *Electronic Journal of Statistics 6*, 354–381.

Knight, K. and W. Fu (2000). Asymptotics for lasso-type estimators. *Annals of statistics*, 1356–1378.

Kolmogorov, A. N. (1963). On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *American Mathematical Society Translation 28*(2), 55–59.

Kozak, S., S. Nagel, and S. Santosh (2018). Interpreting factor models. *The Journal of Finance 73*(3), 1183–1223.

Kozak, S., S. Nagel, and S. Santosh (2020). Shrinking the cross-section. *Journal of Financial Economics 135*(2), 271–292.

Krishnamoorthy, K., T. Mathew, and S. Mukherjee (2008). Normal-based methods for a gamma distribution: Prediction and tolerance intervals and stress-strength reliability. *Technometrics 50*(1), 69–78.

Kuan, C.-M. and H. White (1994). Artificial neural networks: an econometric perspective. *Econometric Reviews 13*(1), 1–91.

Lange, K. (2016). *MM optimization algorithms*. SIAM.

LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *Nature 521*(7553), 436.

LeCun, Y. and C. Cortes (2010). MNIST handwritten digit database.

Lettau, M. and S. Ludvigson (2001). Resurrecting the (c) capm: A cross-sectional test when risk premia are time-varying. *Journal of Political Economy 109*(6), 1238–1287.

Lettau, M. and M. Pelger (2020). Estimating latent asset-pricing factors. *Journal of Econometrics 218*(1), 1–31.

Lewellen, J. and S. Nagel (2006). The conditional capm does not explain asset-pricing anomalies. *Journal of Financial Economics 82*(2), 289–314.

Lewellen, J., S. Nagel, and J. Shanken (2010). A skeptical appraisal of asset pricing tests. *Journal of Financial Economics 96*(2), 175–194.

Liang, M. and X. Hu (2015). Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3367–3375.

Light, N., D. Maslov, and O. Rytchkov (2017). Aggregation of information about the cross section of stock returns: A latent variable approach. *The Review of Financial Studies 30*(4), 1339–1381.

Lijoi, A., P. Muliere, I. Prünster, and F. Taddei (2016). Innovation, growth and aggregate volatility from a bayesian nonparametric perspective. *Electronic Journal of Statistics 10*(2), 2179–2203.

Linderman, S., M. Johnson, and R. P. Adams (2015). Dependent multinomial models made easy: Stick-breaking with the pólya-gamma augmentation. In *Advances in Neural Information Processing Systems*, pp. 3456–3464.

Liu, C., R. Martin, and N. Syring (2013). Simulating from a Gamma distribution with small shape parameter. *arXiv preprint arXiv:1302.1884*.

Lopes, H. F., N. G. Polson, and C. M. Carvalho (2012). Bayesian statistics with a smile: A resampling–sampling perspective. *Brazilian Journal of Probability and Statistics 26*(4), 358–371.

Lyddon, S., C. Holmes, and S. Walker (2019). General bayesian updating and the loss-likelihood bootstrap. *Biometrika 106*(2), 465–478.

Maheu, J. M. and T. H. McCurdy (2007). Components of market risk and return. *Journal of Financial Econometrics 5*(4), 560–590.

Mallick, B. K., D. Ghosh, and M. Ghosh (2005). Bayesian classification of tumours by using gene expression data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67*(2), 219–234.

Martino, L. and D. Luengo (2013). Extremely efficient generation of Gamma random variables for\alpha¿= 1. *arXiv preprint arXiv:1304.3800*.

Matthew, T., C.-S. Chen, J. Yu, and M. Wyle (2015). Bayesian and empirical bayesian forests. In *International Conference on Machine Learning*, pp. 967–976. PMLR.

McLeish, D. (2014). Simulating random variables using moment-generating functions and the saddlepoint approximation. *Journal of Statistical Computation and Simulation 84*(2), 324–334.

Merton, R. C. (1973). An intertemporal capital asset pricing model. *Econometrica: Journal of the Econometric Society*, 867–887.

Miller, J. W. (2018). Fast and accurate approximation of the full conditional for Gamma shape parameters. *arXiv:1802.01610*.

Minka, T. (2000). Estimating a Dirichlet distribution. *Technical report, MIT*.

Minsker, S., S. Srivastava, L. Lin, and D. Dunson (2014). Scalable and robust bayesian inference via the median posterior. In *International conference on machine learning*, pp. 1656–1664. PMLR.

Mitchell, A. F. (1995). A note on posterior moments for a normal mean with double-exponential prior. *Journal of the Royal Statistical Society: Series B (Methodological) 57*(2), 471–471.

Moritz, B. and T. Zimmermann (2016). Tree-based conditional portfolio sorts: The relation between past and future stock returns.

Muliere, P. and P. Secchi (1996). Bayesian nonparametric predictive inference and bootstrap techniques. *Annals of the Institute of Statistical Mathematics 48*(4), 663–673.

Narisetty, N. N., X. He, et al. (2014). Bayesian variable selection with shrinking and diffusing priors. *Annals of Statistics 42*(2), 789–817.

Newton, M. A. and A. E. Raftery (1994). Approximate bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society: Series B (Methodological) 56*(1), 3–26.

Nocedal, J. and S. Wright (2006). *Numerical optimization*. Springer Science & Business Media.

Novy-Marx, R. and M. Velikov (2018). Betting against betting against beta. Technical report, University of British Columbia.

Parikh, N. and S. Boyd (2014). Proximal algorithms. *Foundations and Trends in optimization 1*(3), 127–239.

Park, T. and G. Casella (2008). The bayesian lasso. *Journal of the American Statistical Association 103*(482), 681–686.

Pillow, J. and J. Scott (2012). Fully bayesian inference for neural models with negative-binomial spiking. *Advances in neural information processing systems 25*, 1898–1906.

Poggio, T. and F. Girosi (1990). Networks for approximation and learning. *Proceedings of the IEEE 78*(9), 1481–1497.

Polson, N. G. and J. G. Scott (2013). Data augmentation for Non-Gaussian regression models using variance-mean mixtures. *Biometrika 100*(2), 459–471.

Polson, N. G. and J. G. Scott (2016). Mixtures, envelopes and hierarchical duality. *Journal of the Royal Statistical Society: Series B: Statistical Methodology*, 701–727.

Polson, N. G., J. G. Scott, B. T. Willard, et al. (2015). Proximal algorithms in statistics and machine learning. *Statistical Science 30*(4), 559–581.

Polson, N. G., J. G. Scott, and J. Windle (2013). Bayesian inference for logistic models using Pólya–Gamma latent variables. *Journal of the American Statistical Association 108*(504), 1339–1349.

Polson, N. G. and S. L. Scott (2011). Data augmentation for Support Vector Machines. *Bayesian Analysis 6*(1), 1–23.

Polson, N. G., V. Sokolov, et al. (2017). Deep learning: A bayesian perspective. *Bayesian Analysis 12*(4), 1275–1304.

Polson, N. G. and L. Sun (2019). Bayesian l0-regularized least squares. *Applied Stochastic Models in Business and Industry 35*(3), 717–731.

Rapach, D. E., J. K. Strauss, and G. Zhou (2010). Out-of-sample equity premium prediction: Combination forecasts and links to the real economy. *The Review of Financial Studies 23*(2), 821–862.

Robbins, H. and S. Monro (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 400–407.

Roll, R. (1977). A critique of the asset pricing theory's tests part i: On past and potential testability of the theory. *Journal of Financial Economics 4*(2), 129–176.

Rosenberg, B. and W. McKibben (1973). The prediction of systematic and specific risk in common stocks. *Journal of Financial and Quantitative Analysis 8*(2), 317–333.

Rosiński, J. (2001). Series representations of lévy processes from the perspective of point processes. In *Lévy processes*, pp. 401–415. Springer.

Ross, S. A. (1976). The arbitrage theory of capital asset pricing. *Journal of economic theory 13*(3), 341–360.

Rossell, D. (2009). GaGa: a parsimonious and flexible model for differential expression analysis. *The Annals of Applied Statistics 3*(3), 1035–1051.

Roynette, B. and M. Yor (2005). Couples de Wald indéfiniment divisibles. Exemples liés à la fonction gamma d'Euler et à la fonction zeta de Riemann. *Annales de l'institut Fourier 55*(4), 1219–1284.

Rubin, D. B. (1981). The bayesian bootstrap. *The annals of statistics*, 130–134.

Srivastava, N., G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research 15*(1), 1929–1958.

Stambaugh, R. F. and Y. Yuan (2016). Mispricing factors. *The Review of Financial Studies 30*(4), 1270–1315.

Stefanski, L. A. (1991). A normal scale mixture representation of the logistic distribution. *Statistics & Probability Letters 11*(1), 69–70.

Strawderman, R. L., M. T. Wells, and E. D. Schifano (2013). Hierarchical bayes, maximum a posteriori estimators, and minimax concave penalized likelihood estimation. *Electronic Journal of Statistics 7*, 973–990.

Tibshirani, R. J. (2014). Adaptive piecewise polynomial estimation via trend filtering. *The Annals of Statistics 42*(1), 285 – 323.

Van Binsbergen, J. H. and R. S. Koijen (2010). Predictive regressions: A present-value approach. *The Journal of Finance 65*(4), 1439–1471.

van der Pas, S. L., B. J. K. Kleijn, and A. W. van der Vaart (2014). The horseshoe estimator: Posterior concentration around nearly black vectors. *Electronic Journal of Statistics 8*(2), 2585 – 2618.

Walker, S. and P. Damien (2000). Representations of Lévy processes without Gaussian components. *Biometrika 87*(2), 477–483.

Wang, Y. and L. Swiler (2017). Special Issue on Uncertainty Quantification in Multiscale System Design and Simulation. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B Mechanical Engineering 4*.

Welch, I. and A. Goyal (2007). A comprehensive look at the empirical performance of equity premium prediction. *The Review of Financial Studies 21*(4), 1455–1508.

Welling, M. and Y. W. Teh (2011). Bayesian learning via stochastic gradient Langevin dynamics. *In Proceedings of the 28th international conference on machine learning (ICML-11)*, 681 – 688.

Wellner, J. and T. Zhang (2012). Introduction to the special issue on sparsity and regularization methods. *Statistical Science*, 447 – 449.

West, M. (1987). On scale mixtures of Normal distributions. *Biometrika 74*(3), 646–648.

West, M. (1992). Hyperparameter estimation in Dirichlet process mixture models. *Working paper, Duke University*.

White, H. (1988). Economic prediction using neural networks: The case of ibm daily stock returns. Technical report, University of California, San Diego.

Windle, J., N. G. Polson, and J. G. Scott (2014). Sampling Polya-Gamma random variates: alternate and approximate techniques. *arXiv:1405.0506*.

Wolfram Research, I. Mathematica, Version 12.3. Champaign, IL, 2021.

Wright, S. J. (2015). Coordinate descent algorithms. *Mathematical Programming 151*(1).

Zhang, Z., S. Wang, D. Liu, M. I. Jordan, and N. Lawrence (2012). Ep-gig priors and applications in bayesian sparse learning. *Journal of Machine Learning Research 13*(6).

Zhou, M., L. Hannah, D. Dunson, and L. Carin (2012). Beta-negative binomial process and poisson factor analysis. In *Artificial Intelligence and Statistics*.