Machine Learning in Finance

Semyon Malamud

EPFL

Table of Contents

- **Data** : A vector $x \in \mathbb{R}^d$.

- **A (Machine Learning) Model = family of functions**

$$Function(\textbf{Data}; \textbf{Parameters}) \;=\; f(x; \theta) \qquad (1)$$

where

$$\textbf{Parameters} \;=\; (Parameter_1, \cdots, Parameter_P) \;=\; \theta \;=\; (\theta_1, \cdots, \theta_P) \;\in\; \mathbb{R}^P \quad (2)$$

- $P = $ **parametrization**
- **large $P = $ rich parametrization**

Optimizing a model = **selecting a "good" parameter vector $\theta_* \in \mathbb{R}^P$**

Examples:

- Linear functions family: $x \in \mathbb{R}^1$, $\theta = (\theta_0, \theta_1) \in \mathbb{R}^2$:

$$\text{Function}(\textbf{\textcolor{green}{Data}}; \textbf{\textcolor{red}{Parameters}}) = f(x; \theta) = \theta_0 + \theta_1 x \tag{3}$$

- 2-dimensional linear functions family: $x = (x_1, x_2) \in \mathbb{R}^2$, $\theta = (\theta_0, \theta_1, \theta_2) \in \mathbb{R}^3$:

$$f(x; \theta) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = \underbrace{(\theta_0, \theta_1, \theta_2) \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}}_{inner\ product} = \underbrace{\theta' \begin{pmatrix} 1 \\ x \end{pmatrix}}_{matrix\ multiplication} \ , \tag{4}$$

where we have used the notation

$$\theta = \underbrace{\begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{pmatrix}}_{vertical\ vector} \ , \quad \underbrace{\theta'}_{transposition} = \underbrace{(\theta_0, \theta_1, \theta_2)}_{horizontal\ vector} \tag{5}$$

Parametric Families and Neural Networks ii

▶ when $x = (x_1, \cdots, x_d)$ is d-dimensional, linear family always has **parametrization**
$P = d + 1$: **linear families have few(er) parameters**

▶ Simplest **non-linear** family: Pick a function $h(x)$ and create

$$f(x; \theta) = h(\theta' x) \tag{6}$$

This function family is *non-linear*. E.g., a popular choice is

$$ReLu(x) = \max(x, 0) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

▶ We can now take **linear combinations** of such functions. E.g.,

$$
\begin{aligned}
f_1(x) &= h(\omega_{1,1} x_1 + \omega_{2,1} x_2) \\
f_2(x) &= h(\omega_{1,2} x_1 + \omega_{2,2} x_2) \\
f_3(x) &= h(\omega_{1,3} x_1 + \omega_{2,3} x_2) \\
f(x; \theta) &= W_1 f_1(x) + W_2 f_2(x) + W_3 f_3(x) \\
\theta &= (W_1, W_2, W_3, \omega_{1,1}, \omega_{2,1}, \omega_{1,2}, \omega_{2,2}, \omega_{1,3}, \omega_{2,3})
\end{aligned} \tag{7}
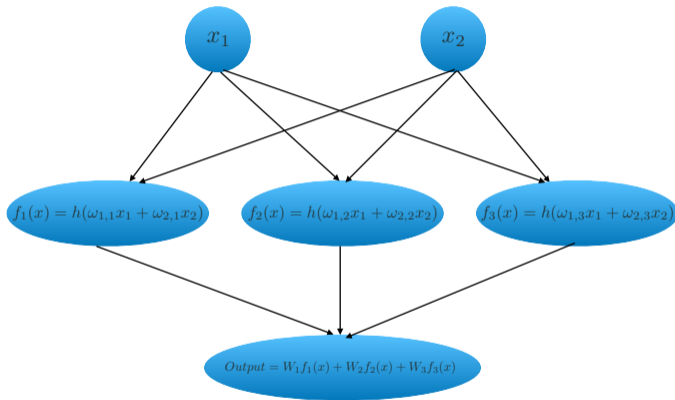$$

$$\underbrace{P = 9}_{\text{parametrization}} \; > \; \underbrace{d}_{\text{data dimension}} \; +1 = \; 3$$

▶ **Observation: Non-linear Models = Rich parametrization**

▶ Note: **Human Beings Always construct recursive functions.** Perhaps this is how our brains function. In the example above:
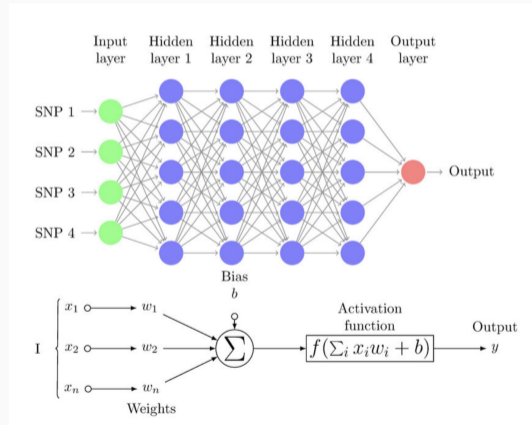
$$Linear \; Combinations \; \rightarrow \; Non-linearity \; h(x) \; \rightarrow \; Linear \; Combinations \qquad (8)$$

Parametric Families and Neural Networks v

Deep Neural Networks: Huge P

Figure: Linear family $f(x; \alpha, \beta) = \alpha + \tan(\beta)x$. **Error** $= y - f(x; \alpha, \beta)$

Ingredients:

▶ **A (Machine Learning) Model = family of functions**
  $Function(Data; Parameters) \; = \; f(x; \theta)$

▶ Stuff to be modelled/Explained = **labels** = y

▶ *Training* **Data**: T observations on which the model learns (=In-Sample Data); and
  Testing **Data**: observations on which the model is tested Out-Of-Sample

$$Training \; \mathbf{Data} \; = \; (label(t), \; \mathbf{Data}(t))_{t=1,\cdots,T} \; = \; (\; \underbrace{y(t), \; x(t)}_{observation \; number \; t} \;)_{t=1}^{T} \qquad (9)$$

▶ **Errors**

$$\mathbf{Error}(label, \; \mathbf{Data}, \; \mathbf{Parameters}) \; = \; label \; - \; Function(\mathbf{Data}; \mathbf{Parameters}) \qquad (10)$$

▶ **Objective function.** E.g., distance between labels and decisions: and

$$Objective(\theta; \text{ Training } \textbf{Data}) = Mean \text{ } Squared \text{ } Error$$

$$= \frac{1}{T} Sum_{\textbf{Data}(t) \in \text{ Training } \textbf{Data}} (\textcolor{red}{\textbf{Error}}(label(t), \textbf{Data}(t), \textcolor{red}{\textbf{Parameters}}))^2 \tag{11}$$

$$= \frac{1}{T} \sum_{t=1}^{T} (\underbrace{y(t) - f(x(t); \theta)}_{Error \text{ } on \text{ } t-th \text{ } observation})^2$$

▶ But in finance (asset pricing), there are **other, more natural objectives (more on this later)**

▶ A **learning algorithm** tries to solve

$$\theta_* = \arg\min_{\theta} \text{ } Objective(\theta; \text{ Training } \textbf{Data}) \tag{12}$$

▶ But what we really care about is the out-of-sample performance of the model: We want the model to work for $\tau > T$ so that

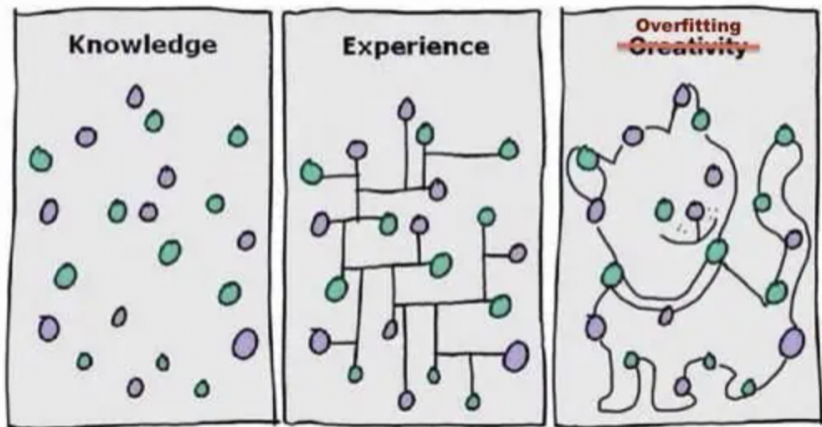$$Test\ Error\ =\ Average((\mathbf{Error}(Testing\ label, Testing\ \mathbf{Data}, \mathbf{Parameters}))^2)$$

$$=\ \frac{1}{T_{test}} \sum_{\tau=T+1}^{T+T_{test}} \Big(\ \underbrace{y_\tau}_{out-of-sample\ label}\ -\ \underbrace{f(x_\tau, \theta_*)}_{out-of-sample\ predictions}\ \Big)^2 \qquad (13)$$

to be **small**.

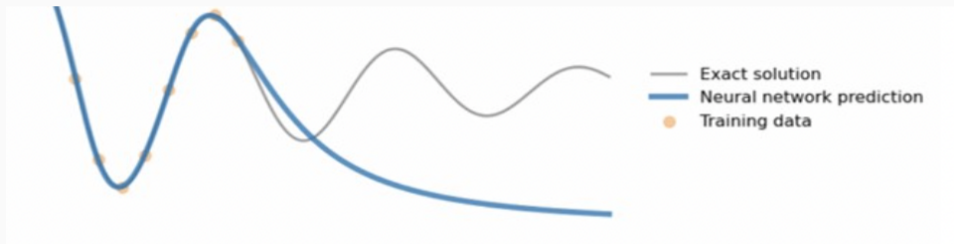▶ Conventional wisdom if we have n observations, we should have $P << n$.

▶ **Intuition: To avoid Overfit**

▶ P parameters can be used to **perfectly fit** P observations (one parameter per observation).

Exact solution
Neural network prediction
Training data

Table of Contents

So, How do we Choose a Model? i

▶ Consider a classical problem of **predicting returns on the US stock market**

▶ In this case, **labels** are returns R_{t+1}.

▶ A comprehensive analysis presented in Goyal and Welch (2008) suggests the following 12 indicators to predict the next month's stock market return: $X_t \in \mathbb{R}^{12}$:

▶ Stock Variance (**svar**), Cross-Sectional Premium (**csp**), Book-to-Market Ratio (**b/m**), Net Equity Expansion (**ntis**), Percent Equity Issuing (**eqis**), Treasury Bills (**tbl**), Long Term Yield (**lty**), The Term Spread (**tms**), Default Yield Spread (**dfy**), The Default Return Spread (**dfr**), Inflation (**infl**), Investment to Capital Ratio (**i/k**), and

$$X_t = \begin{pmatrix} \mathbf{SVAR}_t \\ \mathbf{CSP}_t \\ (\mathbf{B/M})_t \\ \vdots \\ (\mathbf{I/K})_t \end{pmatrix} \tag{14}$$

▶ A simple model would be, say, a linear model with $P = d + 1$:

$$R_{t+1} = \theta_{svar}\textbf{SVAR}_t + \theta_{csp}\textbf{CSP}_t + \theta_{b/m}(\textbf{B/M})_t + \cdots + \theta_{i/k}(\textbf{I/K})_t \qquad (15)$$

▶ However, we could also try to build a wide neural net.

▶ We generate a bunch of random weight vectors $W_k \in \mathbb{R}^d, \; k = 1, \cdots, P$

▶ We generate a bunch of **random features** $\phi(X_t'W_k)$ and build

$$f(X_t; \theta) = \sum_{k=1}^{P} \phi(X_t'W_k)\theta_k$$

E.g., if $W_1 = (0.1, -0.5, 0.7, \cdots, 3.1)'$ and $\phi(x) = ReLu(x) = \max(x, 0)$, then

$$Feature1_t = \phi(X_t'W_1)$$
$$= \max\left(0.1\,\textbf{SVAR}_t - 0.5\,\textbf{CSP}_t + 0.7\,(\textbf{B/M})_t + \cdots + 3.1\,(\textbf{I/K})_t, \; 0\right) \qquad (16)$$

▶ Now, we can again run a linear regression

$$R_{t+1} = \underbrace{\theta_{feature1} \, Feature1_t \; + \; \theta_{feature2} \, Feature2_t \; + \; \cdots \; + \; \theta_{featureP} \, FeatureP_t}_{\text{Prediction} \; = \; \text{combination of features}} \; + \; \varepsilon_{t+1}$$

where

$$Featurek_t \; = \; \phi(X_t' W_k)$$

▶ E.g., $\theta_{feature1} = 7.3$, $\theta_{feature2} = -12.6$, and

Prediction $= 7.3 \max(0.1\,\textbf{SVAR}_t \; - \; 0.5\,\textbf{CSP}_t \; + \; 0.7\,(\textbf{B/M})_t \; + \; \cdots \; + \; 3.1\,(\textbf{I/K})_t, \; 0)$

$\qquad\qquad - \; 12.6 \max(0.4\,\textbf{SVAR}_t \; - \; 0.7\,\textbf{CSP}_t \; - \; 0.9\,(\textbf{B/M})_t \; + \; \cdots \; - \; 4.2\,(\textbf{I/K})_t, \; 0)$

$$\tag{17}$$

▶ Which P do we choose?

$$x \in \mathbb{R}^d$$

$$W = \left[\begin{array}{ccccc} | & \cdots & | & \cdots & | \\ W_1 & \cdots & W_k & \cdots & W_P \\ | & \cdots & | & \cdots & | \end{array} \right]_{d \times P}$$

$$\phi(x'W)$$

$$\theta \in \mathbb{R}^P$$

$$f(x;\theta) = f(x;\theta;W) = \sum_{k=1}^{P} \phi(x'W_k)\theta_k$$

$f(x;\theta;W)$

Figure: $R_{t+1} = f(X_t; \theta) + \varepsilon_{t+1}$

Bias-Variance Tradeoff ii
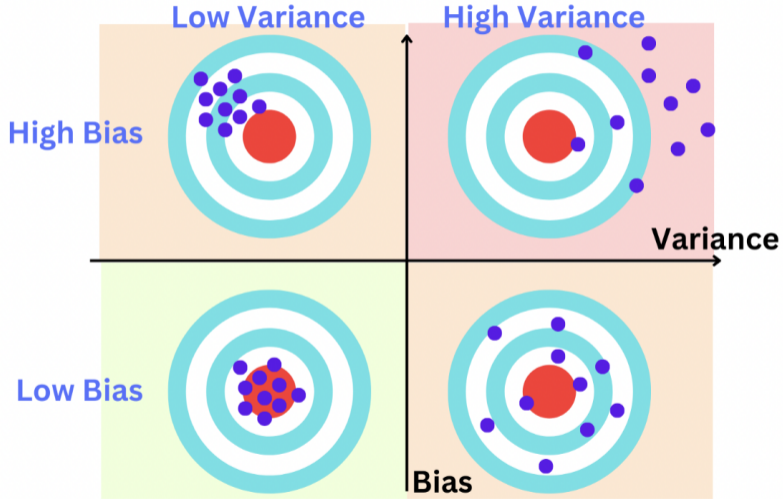
- $P =$ **model complexity**
- **Large** P $\rightarrow$ **low bias** because a complex model can approximate **any ground truth. Always true!**
- **Large** P $\rightarrow$ **high variance**: **Used to be true in 20th century.** Intuition:

$$Variance \;=\; \mathrm{Var}[\theta_{feature1}] \;+\; \cdots \;+\; \mathrm{Var}[\theta_{featureP}] \tag{18}$$

▶ As we increase complexity,

$$
\begin{aligned}
\textit{Train Error} \;\to\; & 0 \\
\textit{Test Error} \;\not\to\; & 0
\end{aligned}
\tag{19}
$$

Statistical Wisdom and Overfitting
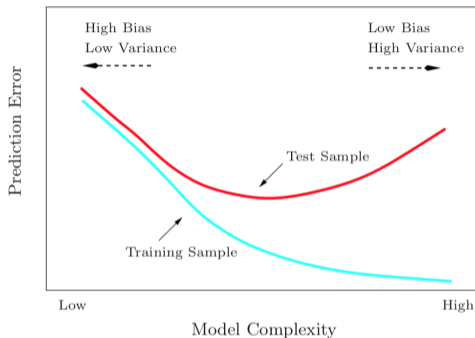
FIGURE 2.11. *Test and training error as a function of model complexity.*

Figure 2.11 shows the typical behavior of the test and training error, as model complexity is varied. The training error tends to decrease whenever we incre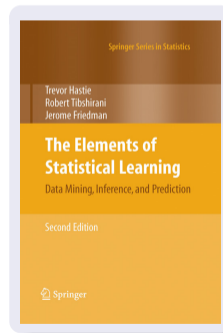ase the model complexity, that is, whenever we fit the data harder. However with too much fitting, the model adapts itself too closely to the training data, and will not generalize well (i.e., have large test error). In

Statistical Wisdom and Overfitting

22   2. How to Construct Nonparametric Regression Estimates?
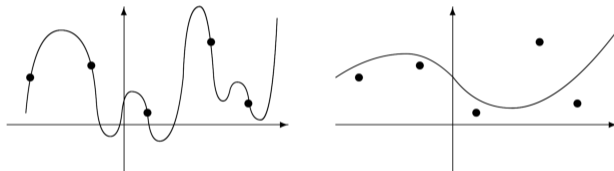
Figure 2.3. The estimate on the right seems to be more reasonable than the estimate on the left, which interpolates the data.

over $\mathcal{F}_n$. Least squares estimates are defined by minimizing the empirical L_2 risk over a general set of functions $\mathcal{F}_n$ (instead of (2.7)). Observe that it doesn't make sense to minimize (2.9) over all (measurable) functions f, because this may lead to a function which interpolates the data and hence is

Model Size (ResNet18 Width)

Classical Regime
Bias-Variance Tradeoff

Modern Regime
Larger Model is Better

Critical Regime

Interpolation Threshold

Test / Train Error

ResNet18 Width Parameter
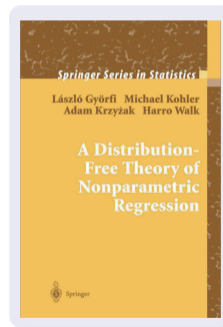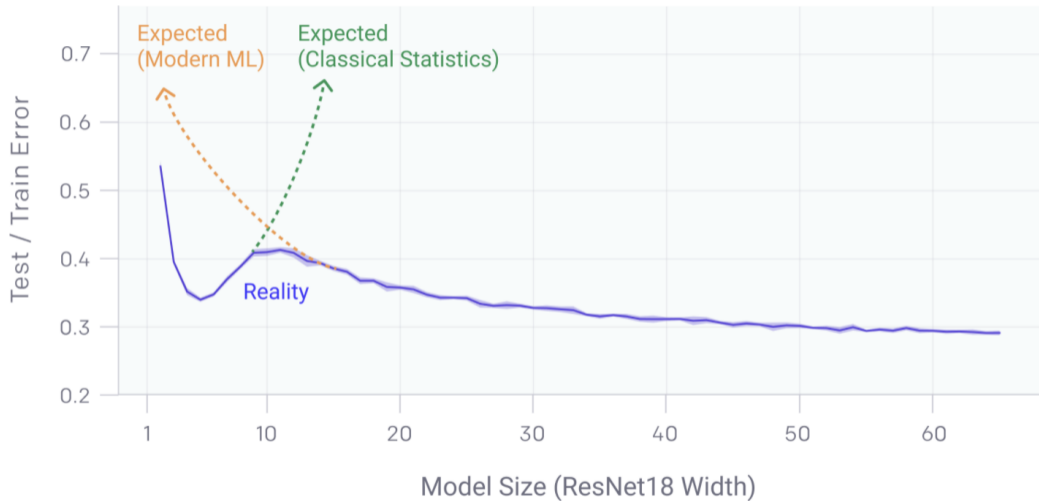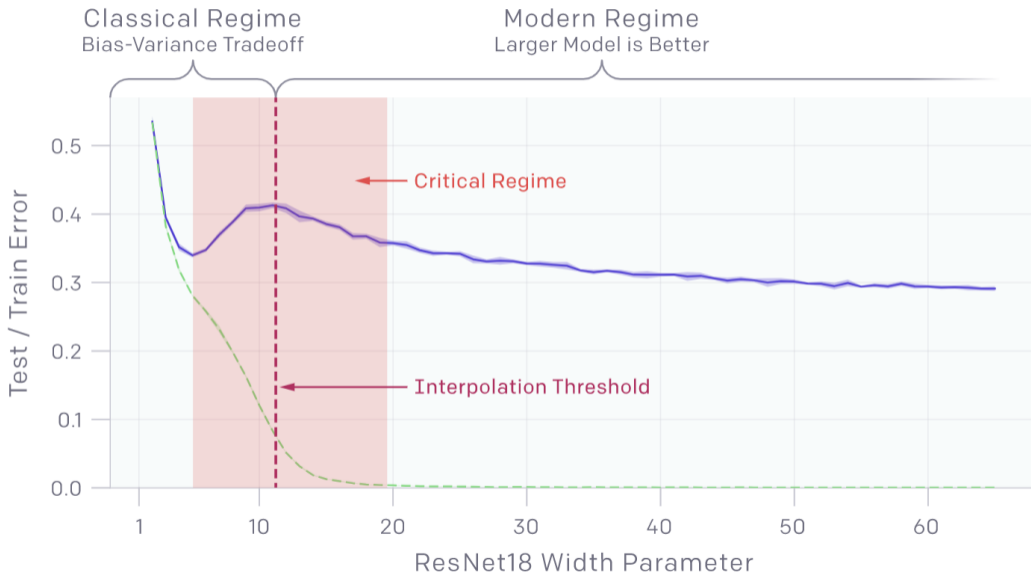
DEEP DOUBLE DESCENT:
WHERE BIGGER MODELS AND MORE DATA HURT

Preetum Nakkiran[*]
Harvard University

Gal Kaplun[†]
Harvard University

Yamini Bansal[†]
Harvard University

Tristan Yang
Harvard University

Boaz Barak
Harvard University

Ilya Sutskever
OpenAI

ABSTRACT

We show that a variety of modern deep learning tasks exhibit a "double-descent" phenomenon where, as we increase model size, performance first gets *worse* and then gets better. Moreover, we show that double descent occurs not just as a function of model size, but also as a function of the number of training epochs. We unify the above phenomena by defining a new complexity measure we call the *effective model complexity* and conjecture a generalized double descent with respect to this measure. Furthermore, our notion of model complexity allows us to

Scaling Laws for Neural Language Models

Jared Kaplan [*]

Johns Hopkins University, OpenAI

jaredk@jhu.edu

Sam McCandlish[*]

OpenAI

sam@openai.com

Tom Henighan

OpenAI

henighan@openai.com

Tom B. Brown

OpenAI

tom@openai.com

Benjamin Chess

OpenAI

bchess@openai.com

Rewon Child

OpenAI

rewon@openai.com

Scott Gray

OpenAI

scott@openai.com

Alec Radford

OpenAI

alec@openai.com

Jeffrey Wu

OpenAI

jeffwu@openai.com

Dario Amodei
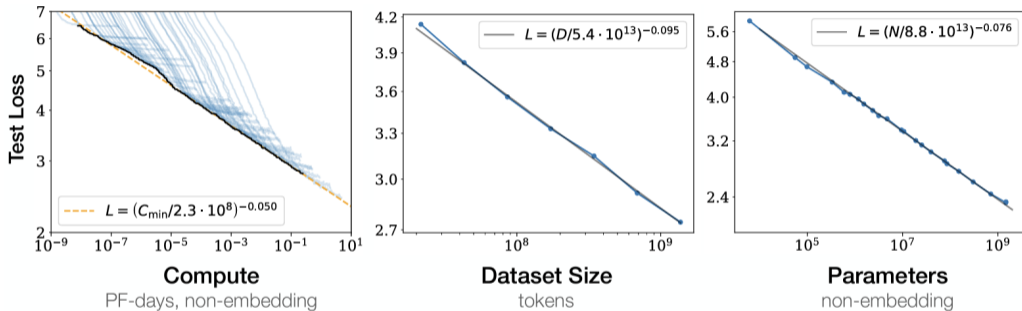
OpenAI

damodei@openai.com

Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Table of Contents

▶ modify objective **to reduce the gap** between **Train Error** and **Test Error**, we use **ridge penalty**:

$$Objective \ = \ \frac{1}{T} \sum_t (R_{t+1} \ - \ f(X_t; \theta))^2 \ + \ \underbrace{z}_{ridge \ penalty} \|\theta\|^2 \tag{20}$$

▶ It penalizes large norm of the coefficients,

$$\|\theta\|^2 \ = \ \sum_k \theta_k^2$$

▶ minimizing the objective gives **Prediction**$_t$

$$\pi_t(z) \ = \ \textbf{Prediction}_t \ = \ \theta_{feature1} \ Feature1_t + \theta_{feature2} \ Feature2_t + \cdots + \theta_{featureP} \ FeatureP_t$$

▶ we then build **managed returns**

$$R_{t+1}^{\pi(z)} \ = \ \pi_t(z) \ R_{t+1}$$

Sharpe Ratios and Timing Strategies ii

▶ we want these returns to have high mean **out-of-sample (OOS)** (i.e., on the *Test* **Data**)

$$\text{OOS Mean} = \frac{1}{T_{OOS}} \sum_{OOS \ \tau} R_{\tau+1}^{\pi(z)}$$

▶ we want these returns to have low variance **out-of-sample (OOS)** (i.e., on the *Test* **Data**)

$$\text{OOS Variance} = \frac{1}{T_{OOS}} \sum_{OOS \ \tau} \left(R_{\tau+1}^{\pi(z)} - \text{OOS Mean} \right)^2$$

▶ we commonly measure the quality of a strategy is through its **Sharpe Ratio (S'R)**

$$S'R = \frac{\text{OOS Mean}}{\sqrt{\text{OOS Variance}}} \tag{21}$$

Virtue of Complexity for the SR
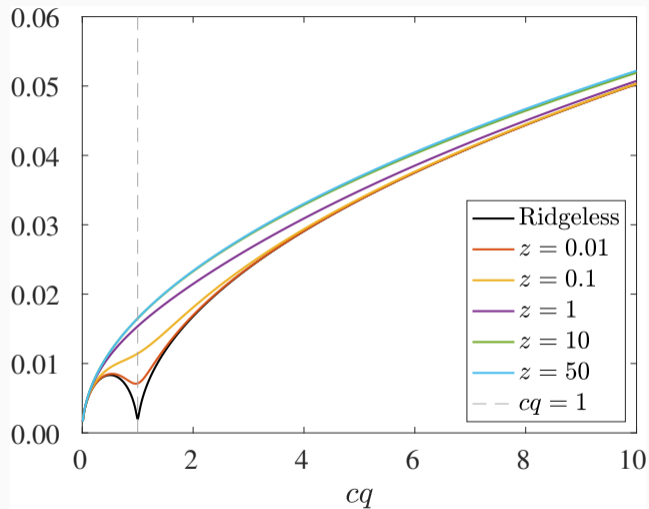
Figure: Expected Out-of-sample Timing Strategy Sharpe Ratio

Conditional Expectations

$$Objective \ = \ \frac{1}{T} \sum_t (R_{t+1} \ - \ f(X_t; \theta))^2 \qquad (22)$$

Why is this the right objective?

▶ We would like to incorporate conditional information. But how?

► Conditional expectation

$$E_t[X] = \sum x \underbrace{P(X = x | \textit{Information at time } t)}_{\textit{conditional probability}}$$

► Practitioners often use rolling window estimates

$$\textit{Practitioner} E_t[R_{t+1}] = \frac{1}{T} \sum_{\tau=t-T}^{t} R_\tau, \ T = \textit{Rolling Window}$$

► But this is a very naive approach. What if some other signals are informative about stock returns?

► Here, we will need the notion of conditional expectation. Formally,

$$\underbrace{E_t[X_{t+1}]}_{\textit{conditional expectation}} = \textbf{best prediction of } \mathbf{X_{t+1}} \textbf{ given time} - \textbf{t information} \quad (23)$$

▶ Formally, if $S_t \in \mathbb{R}^P$ for some huge $P!$ is all our relevant information set at time t,

$$\underbrace{E_t[X_{t+1}]}_{conditional\ expectation} \quad = \; F_*(S_t) \; = \; \textbf{best non} - \textbf{linear function of } \mathbf{S_t} \textbf{ that predicts } \mathbf{X_{t+1}}$$

$$(24)$$

▶ Even more formally,

$$F_* \; = \; \arg\min_F E[\|X_{t+1} - F(S_t)\|^2] \tag{25}$$

▶ The reality is that we still cannot compute $E[\cdot]$ because we do not have enough data (see the picture curse of dimensionality)

▶ So, we will still be doing

$$\hat{F}_* \; = \; \arg\min_F \frac{1}{T} \sum_t |X_{t+1} - F(S_t)|^2 \tag{26}$$

Understanding Conditional Expectation (Good to Understand it Even we Cannot Estimate It) i

Theorem

If $F(S_t)$ is the true conditional expectation, then

$$X_{t+1} = F(S_t) + \varepsilon_{t+1} = \underbrace{E_t[X_{t+1}]}_{\text{predictable part}} + \underbrace{\varepsilon_{t+1}}_{\text{unpredictable noise}} \tag{27}$$

and we have the variance decomposition

$$\underbrace{\mathrm{Var}[X_{t+1}]}_{\text{total variation}} = \underbrace{\mathrm{Var}[E_t[X_{t+1}]]}_{\text{predictable variation}} + \underbrace{\mathrm{Var}[\varepsilon_{t+1}]}_{\text{unpredictable variation}} \tag{28}$$

Proof:

$$E[\varepsilon_{t+1}] = 0, \ E[\varepsilon_{t+1} E_t[X_{t+1}]] = 0, \ \mathrm{Cov}[E_t[X_{t+1}], \varepsilon_{t+1}] = 0 \tag{29}$$

and hence

$$\text{Var}[X_{t+1}] = \text{Var}[E_t[X_{t+1}] + \varepsilon_{t+1}] = \text{Var}[E_t[X_{t+1}]] + \text{Var}[\varepsilon_{t+1}] + 2\underbrace{\text{Cov}[E_t[X_{t+1}], \varepsilon_{t+1}]}_{=0} \tag{30}$$

Why is this important?

▶ We would like to find the true amount of predictability:

$$R^2 = \frac{\text{Var}[E_t[X_{t+1}]]}{\text{Var}[X_{t+1}]} = 1 - \frac{E[\varepsilon_{t+1}^2]}{\text{Var}[X_{t+1}]} \tag{31}$$

and the true residuals

$$\varepsilon_{t+1} = X_{t+1} - F(S_t). \tag{32}$$

Examples: True Expectations

▶ ARMA process:

$$R_{t+1} = \frac{1}{T} \sum_{\tau=t-T}^{t} R_\tau + \varepsilon_{t+1}, \; E_t[R_{t+1}] = \frac{1}{T} \sum_{\tau=t-T}^{t} R_\tau \qquad (33)$$

Here,

$$S_t = (R_\tau)_{\tau=t-T}^{t}, \; F(S_t) = \frac{1}{T} \sum_{\tau=t-T}^{t} R_\tau \qquad (34)$$

▶ Linear prediction:

$$R_{t+1} = \sum_{j=1}^{P} \beta_j S_{j,t} + \varepsilon_{t+1}, \; E_t[R_{t+1}] = \sum_{j=1}^{P} \beta_j S_{j,t} \qquad (35)$$

Here,

$$F(S_t) = \sum_{j=1}^{P} \beta_j S_{j,t} \qquad (36)$$

is linear

Table of Contents

Learning Is Impossible Without Some Kind of Bias

The ugly duckling theorem. Classification is not really possible without some sort of bias.

"Suppose that one is to list the attributes that plums and lawnmowers have in common in order to judge their similarity. It is easy to see that the list could be infinite: Both weigh less than 10,000 kg (and less than 10,001 kg), both did not exist 10,000,000 years ago (and 10,000,001 years ago), both cannot hear well, both can be dropped, both take-up space and so on. Likewise, the list of differences could be infinite... any two entities can be arbitrarily similar or dissimilar by changing the criterion of what counts as a relevant attribute."

Inductive Biases

So how do we decide which attributes (features; characteristics) matter?

We use **Inductive Biases!**

Definition. *Inductive bias is the set of assumptions a learning algorithm makes to generalize beyond the training data. In other words, it's what allows the algorithm to "induce" a hypothesis from a limited amount of data, guiding it on how to predict unseen examples. For instance, a linear regression model assumes that the underlying relationship is approximately linear—even if the true relationship is more complex—so it is biased towards linear solutions. This bias is essential: without it, an algorithm would have no basis for generalization and might simply memorize the training data without making any useful predictions on new data.*

The Cat-astrophe Bias

Imagine a photo recognition AI trained exclusively on images of cats. Its inductive bias is that all important entities in the world are cats. Show it a picture of a dog, and it's perplexed. "Is this some kind of weird cat?" it wonders. A car? "Strange cat with wheels!" A tree? "Clearly, a tall, wooden cat!"
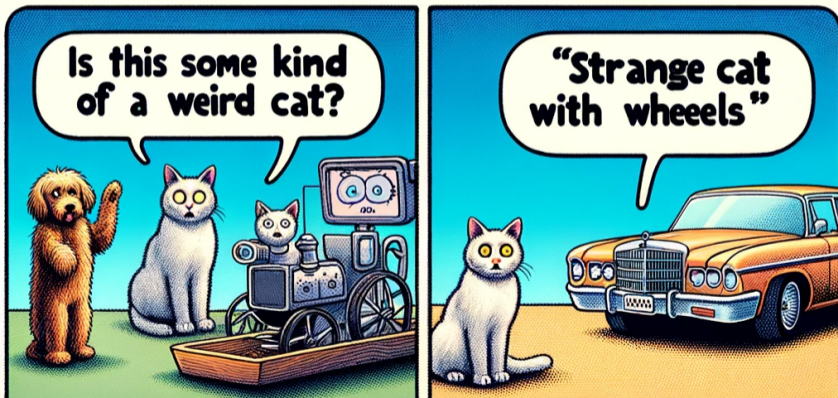
An Analyst with Extrapolative Expectations

▶ I believe past returns predict future returns and I will use

$$E_t[R_{t+1}] = \frac{1}{T} \sum_{\tau=t-T}^{t} R_\tau$$

- Fantastic bias if

$$R_{t+1} = \frac{1}{T} \sum_{\tau=t-T}^{t} R_\tau + \varepsilon_{t+1}, \ E_t[R_{t+1}] = \frac{1}{T} \sum_{\tau=t-T}^{t} R_\tau \qquad (39)$$

- OK bias if

$$R_{t+1} = R_t + \varepsilon_{t+1}, \ E_t[R_{t+1}] = R_t \qquad (40)$$

  will be positively related to the analyst's guess

- Horrible bias if returns are mean-reverting:

$$R_{t+1} = -R_t + \varepsilon_{t+1}, \ E_t[R_{t+1}] = -R_t \qquad (41)$$

An Economist Who Believes in Linearity

▶ I believe that

$$E_t[R_{t+1}] = \sum_{j=1}^{P} \beta_j S_{j,t}$$

- Fantastic bias if

$$R_{t+1} = \sum_{j=1}^{P} \beta_j S_{j,t} + \varepsilon_{t+1} \tag{42}$$

- OK bias if

$$R_{t+1} = \underbrace{S_{1,t} + 5S_{17,t}}_{\text{sparse: only 1 and 17 matter}} + \varepsilon_{t+1}, \; E_t[R_{t+1}] = R_t \tag{43}$$

- Horrible bias if non-linear:

$$R_{t+1} = S_{15,t}^2 + \varepsilon_{t+1} \tag{44}$$

More Complex Biases

► We believe

$$E_t[X_{t+1}] = \theta' S_t$$

but $\|\theta\|$ is small (small $\|\theta\|$ inductive bias). Implementation: ridge penalty

$$\min_\theta \|R - \theta' S\|^2 + \underbrace{z}_{ridge\ penalty} \|\theta\|^2, \ \|\theta\|^2 = \sum_i |\theta_i|^2$$

► We believe θ is sparse: only a few coordinates are non-zero. The sparsity bias. Implementation: LASSO

$$\min_\theta \|R - \theta' S\|^2 + \underbrace{z}_{ridge\ penalty} \|\theta\|_1, \ \|\theta\|_1 = \sum_i |\theta_i|$$

Deep Learning Trained by Gradient Descent: The Most Powerful Inductive Bias Human Beings Have Discovered So Far i

▶ **Inductive Bias of DNNs:** The world has a smooth, hierarchical structure

Deep Learning Trained by Gradient Descent: The Most Powerful Inductive Bias Human Beings Have Discovered So Far ii
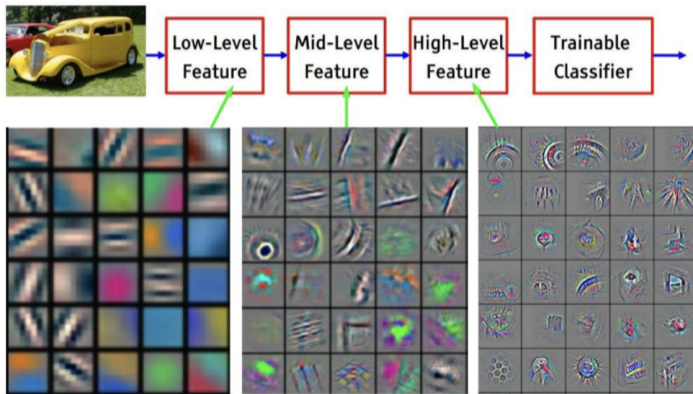
Learning Hierarchical Concepts

LLM concepts by combining nonlinear predictors over multiple layers

Table of Contents

OLS i

▶ Predictors' vector $S_t \in \mathbb{R}^P$

▶ We organize them like an excel spreadsheet, $S = (S_t)_{t=1}^T \in \mathbb{R}^{T \times P}$.

▶ OLS (linear regression, Carl Friedrich Gauss):

$$\hat{\beta}_{OLS} \;=\; \arg\min_\beta \| \underbrace{R}_{T \times 1} - \underbrace{S}_{T \times P} \underbrace{\beta}_{P \times 1} \|^2 \;=\; \arg\min_\beta \sum_t \left(R_{t+1} - \sum_j S_{j,t}\beta_j \right)^2$$

and

$$\hat{\beta}_{OLS} \;=\; \underbrace{\hat{\Psi}_T^{-1}}_{P \times P} T^{-1} \sum_t \underbrace{S_t}_{P \times 1} \underbrace{R_{t+1}}_{1 \times 1} \;\in\; \mathbb{R}^P$$

▶

$$\hat{\Psi}_T \;=\; T^{-1}\sum_t S_t S_t' \;=\; \underbrace{T^{-1}S'S}_{(P\times T)\times(T\times P)} \;\in\; \mathbb{R}^{P\times P} \tag{45}$$

is the sample covariance matrix

▶ Defining $R = (R_t)_{t=1}^{T} \in \mathbb{R}^T$, we can rewrite

$$\hat{\beta}_{OLS} \;=\; (T^{-1}S'S)^{-1}T^{-1}S'R. \tag{46}$$

$$\hat{\beta} = \arg\min_{\beta}(T^{-1}\|R - S\beta\|^2 + z\|\beta\|^2)$$

gives

$$\hat{\beta}(z) = \left(zI + T^{-1}\sum_t S_t S_t'\right)^{-1} T^{-1}\sum_t S_t R_{t+1} = (zI + T^{-1}S'S)^{-1}T^{-1}S'R.$$

Ridgeless Limit and the Moore-Pensore Quasi-Inverse

When P moves beyond T, there are more parameters than observations, and the least squares problem has multiple solutions. A particularly interesting solution invokes the Moore-Penrose pseudo-inverse, which can be computed as follows:

$$A = U \operatorname{diag}(D) U' \;\Rightarrow\; A^+ = U \operatorname{diag}((D^{-1} \text{ if } D > 0 \text{ else } 0)) U' \tag{47}$$

As we show below, as the shrinkage parameter approaches zero, we converge to the **Ridgeless Estimator**:

$$\hat{\beta}(0^+) = \lim_{z \to 0+} \hat{\beta}(z) = \left(T^{-1} \sum_t S_t S_t' \right)^+ \frac{1}{T} \sum_t S_t R_{t+1}.$$

Proving this is non-trivial and requires a bit of algebra.

The Magic of Ridge: Playing with Dimension i

Lemma (Swapping Dimension)

Let $S \in \mathbb{R}^{T \times P}$. Then,

$$(zI + \underbrace{S'S}_{T \times T})^{-1}S' = S'(zI + \underbrace{SS'}_{P \times P})^{-1} \tag{48}$$

Proof.

We have

$$
\begin{aligned}
(zI + S'S)^{-1}S' &= S'(zI + SS')^{-1} \Leftrightarrow \\
S'(zI + SS') &= (zI + S'S)S' \Leftrightarrow \\
S'z + S'SS' &= zS' + S'SS'
\end{aligned}
\tag{49}
$$

∎

We use this in the Jupyter Notebook to get fast regression in very high dimensions

The Magic of Ridge: Playing with Dimension ii

Corollary

$$\hat{\beta}(z) = \left(zI + T^{-1}\underbrace{S'S}_{P \times P}\right)^{-1}\frac{1}{T}S'R = S'\left(zI + T^{-1}\underbrace{SS'}_{T \times T}\right)^{-1}\frac{1}{T}R \qquad (50)$$

and

$$\hat{\beta}(0+) = \underbrace{S'}_{P \times T}\left(\underbrace{SS'}_{T \times T}\right)^{-1}\underbrace{R}_{T \times 1} \qquad (51)$$

Let us now prove that

$$\hat{\beta}(0+) = (S'S)^+ S'R. \qquad (52)$$

Indeed,

$$\hat{\beta}(0+) = S'(SS')^{-1}R = (S'S)^+ S'R \qquad (53)$$
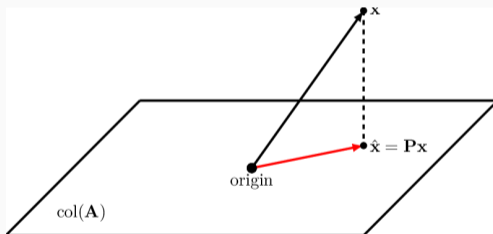
because $Im(S') \perp \ker(S) = \ker(S'S)$.

$\hat{\beta}(0+)$ **is an interpolator:**

$$S\hat{\beta}(0+) \; = \; SS'\left(SS'\right)^{-1} R \; = \; R \tag{54}$$

Given a vector x and a subspace S of $\mathbb{R}^n$, given by the span of columns of a matrix A (denoted by $col(A)$), the **orthogonal projection** P of x onto S, is the closest vector $\hat{x}$ in S to x.

Lemma

Let $\Pi = (AA')^+ AA'$. Then, Π is the orthogonal projection onto the $\text{col}(A)$.

Proof.

First, Π is symmetric and positive definite. Second, $AA' = U \text{diag}(D)U'$, and hence

$$(AA')^+ AA' = U(D(D^{-1}\mathbf{1}_{D>0}))U' = U \text{diag}(\mathbf{1}_{D>0})U' \tag{55}$$

and hence Π is the projection onto the image of AA', while it kills the kernel of AA'. But $AA'x = 0$ means $x'AA'x = 0$ means $A'x = 0$ and we know

$$\ker A' = (ImA)^{\perp}. \tag{56}$$

∎

Any solution $\hat{\beta} \in \mathbb{R}^P$ to the system

$$\hat{\beta}' S_t = R_{t+1}, \ t = 1, \cdots, T \tag{57}$$

of T equations and P variables is called an interpolator.

Lemma

Suppose that $R \in \mathrm{span}(S)$, so that (57) has a solution. Then, any solution β to (57) can be written down as

$$\beta = \underbrace{\hat{\beta}(0^+)}_{\in \ \mathrm{span}(S)} + \underbrace{\beta^{\perp}}_{orthogonal \ to \ \mathrm{span}(S)} \tag{58}$$

In particular, $\hat{\beta}(0^+)$ gives the minimal norm interpolator:

$$\hat{\beta}(0^+) = \arg\min\{\|\beta\|^2 : \ \beta \ satisfies \ (57)\} \tag{59}$$

What does the Ridgeless Regression Do? ii

Proof.

We have

$$\hat{\beta}(0^+) = (SS')^+ S'R \tag{60}$$

Let

$$S'\beta = R \tag{61}$$

be some other interpolator. Then,

$$SS'\beta = S'R \Rightarrow (SS')^+ SS'\beta = (SS')^+ S'R = \hat{\beta}(0^+). \tag{62}$$

Let $\Pi = (SS')^+ SS'$. Then, Π is the orthogonal projection onto the $\mathrm{span}(S)$. Thus,

$$\Pi\beta = \hat{\beta}(0^+) \tag{63}$$

and the claim follows. $\blacksquare$

Summarizing, we can formulate the complexity principle.

Theorem (heuristic)

$\|\hat{\beta}(0+)\|$ *is monotone decreasing in* P. *Thus, if the small norm of* β *is a good* **inductive bias**, *ridgeless regression generalizes better when* P *gets larger.*

Generalizes better= has lower test (out-of-sample) error

Experiments with High Dimensional Regressions

Please click on the link:

Understanding High-Dimensional Regressions

Random Features for Goyal-Welch