Day 10 - Bias Variance Tradeoff

Agenda:

- Statistical learning framework
- Bias variance tradeoff - Classical Story
- Bias variance decomposition
- Bias variance tradeoff - Modern Story

CS 6140: Machine Learning — Fall 2021— Paul Hand

HW 4 Revised (corrected hyperlinks)
Due: Wednesday October 13, 2021 at 2:30 PM Eastern time via Gradescope.

Names: [Put Your Name(s) Here]

You can submit this homework either by yourself or in a group of 2. You may consult any and all resources. You may submit your answers to this homework by directly editing this tex file (available on the course website) or by submitting a PDF of a Jupyter or Colab notebook. When you upload your solutions to Gradescope, make sure to tag each problem with the correct page.

Question 1. *In this problem, you will use logistic regression for heart attack prediction.*

Download the dataset at this Kaggle site. It is a CSV file of 14 attributes of 294 people. The meaning of the attributes is as follows:

- age: age in years
- sex: sex (1 = male; 0 = female)
- cp: chest pain type – Value 1: typical angina – Value 2: atypical angina – Value 3: non-anginal pain – Value 4: asymptomatic
- trestbps: resting blood pressure (in mm Hg on admission to the hospital)
- chol: serum cholestoral in mg/dl
- fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
- restecg: resting electrocardiographic results – Value 0: normal – Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV) – Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
- thalach: maximum heart rate achieved
- exang: exercise induced angina (1 = yes; 0 = no)
- oldpeak = ST depression induced by exercise relative to rest
- slope: ignore
- ca: ignore
- thal: ignore
- num: diagnosis of heart disease (angiographic disease status) – Value 0: $< 50\%$ diameter narrowing – Value 1: $> 50\%$ diameter narrowing

You will train binary classifiers to predict the diagnosis of heart disease using some or all of the following features: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak. You may find helpful the following step-by-step guide.

(a) Randomly select a test dataset consisting of 20% of the examples with num= 0 and 20% of the examples with num= 1. The remaining examples will constitute the training data. Plot histograms of each of the features for the training data and the test data.

Response:

(b) Use logistic regression to learn a binary classifier that predicts the diagnosis of heart disease using only the features: age, sex, cp, chol. Plot the ROC curve and the precision-recall curve for your classifier. Remove from your training set any example for which any of these features is missing. You may use an existing computer package that computes the logistic regression, such as scikit-learn. You may choose to follow other data cleaning steps in the step-by-step guide

Response:

(c) Same as part (b), but use the following features: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak.

Response:

(d) Compare your two classifiers. Which would you argue is better for deployment in practice?

Response:

(e) Solve the logistic regression in part (b) using gradient descent and a cross-entropy loss. Plot the ROC curve and the precision-recall curve.

Response:

(f) Solve the logistic regression in part (b) using gradient descent and a square loss. Plot the ROC curve and the precision-recall curve. How does your classifier compare to that from part (e)?

Response:

Statistical Framework for ML (supervised)

Assume:

- (x, y) are sampled from a joint probability distribution
- Training data $D = \{(x_i, y_i)\}_{i=1 \cdots n}$ are iid samples
- Test data are also iid samples OF THE SAME DISTRIBUTION!

Can estimate the model/predictor by maximum likelihood estimation

Results (usually) in an optimization problem

$$\hat{f} = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$$

"empirical risk minimization"

where

ℓ ~ loss function eg $\ell(\hat{y}, y) = |\hat{y} - y|^2$

$\mathcal{H}$ - hypothesis class eg degree d polynomial

Evaluate performance on test data $\{(x_i, y_i)\}_{i=1 \cdots m}$

$$\frac{1}{m} \sum_{i=1}^{m} \ell(y_i, \hat{f}(x_i))$$

Formalism for Statistical Framework for ML (supervised)

Domain Set - $\mathcal{X}$ - arbitrary set of objects/instances that could be labelled
- usually represented as a feature vector in $\mathbb{R}^d$
- could be infinite dimensional

Label Set - $\mathcal{Y}$ - set of possible labels

eg • $\mathbb{R}^d$ for regression
• $\{1, 0\}$ for binary classification
• Finite set for multiclass classification

Training data - $S = \{(x_i, y_i)\}_{i=1 \cdots n}$

n points in $\mathcal{X} \times \mathcal{Y}$

Predictor/hypothesis - any function $f: \mathcal{X} \to \mathcal{Y}$ that
$x \mapsto y$

outputs a prediction y for any instance x

Hypothesis Class - $\mathcal{H}$ a set of predictors/hypotheses that are being considered

eg $\mathcal{H} = \{\text{degree } d \text{ polynomials}\}$

Data generation model

Simple version

- Assume $x \sim D$, where D is a probability distribution over $\mathcal{X}$
- Each sample is independent
- $y = f^*(x)$ for a "correct" function f^*.

Realistic version

- Assume $(x, y) \sim D$, a joint probability distribution over $\mathcal{X} \times \mathcal{Y}$

There is some marginal distribution of $\mathcal{X}$, $D_{\mathcal{X}}$.

For any x, there is a conditional distribution over y $D_{y|x}$

Loss — how bad is the prediction of an instance relative to its label

$\ell(y, \hat{y}) \in \mathbb{R}$

label / prediction

Examples

- Square loss $\ell(y,\hat{y}) = \|y - \hat{y}\|^2$ if $y, \hat{y} \in \mathbb{R}^d$

- log loss $\ell(y,\hat{y}) = \sum_{i=1}^{k} y_i \log \hat{y}_i$ if $y \in \mathbb{R}^k$ are one-hot encodings & $\hat{y} \in \mathbb{R}^k$ is a probability dist over k labels

- 0-1 loss $\ell(y,\hat{y}) = \begin{cases} 0 & \text{if } \hat{y} = y \\ 1 & \text{if o'wise} \end{cases}$

Risk – expected loss of a predictor for new data samples

$$R(f) = \mathbb{E}_{(x,y)\sim D}\, \ell(y, f(x))$$

aka "generalization error"
"error"
"population error"
"test error"

Generalization – ability to perform well on new data

Goal of learning:

To find a f such that $R(f)$ is minimal. Want to solve

$$\arg\min_{f \in \mathcal{H}} R_D(f)$$

Challenge: We don't know D. We only have samples S

Empirical Risk Minimization — approximation of risk based on training data S

$$\hat{f} = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \underbrace{\overbrace{\frac{1}{n}\sum_{i=1}^{n}}^{\text{empirical mean}} \ell(y_i, f(x_i))}_{\text{Empirical risk}}$$

Test Error — Use a finite test set to assess generalization

$$\frac{1}{m}\sum_{i=1}^{m} \ell(y_i^{test}, \hat{f}(x_i^{test}))$$

$$\approx \mathbb{E}_{(x^{test}, y^{test}) \sim \mathcal{D}} \ell(y_i^{test}, \hat{f}(x_i^{test}))$$

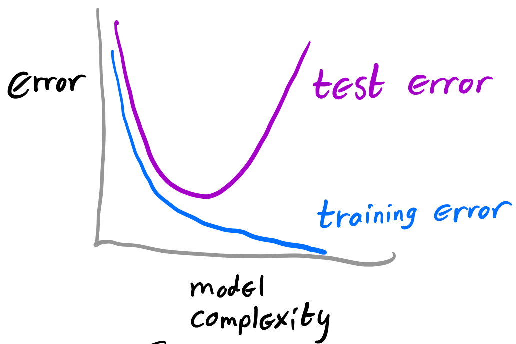
Model complexity — Cardinality or dimensionality of hypothesis set $\mathcal{H}$

\# unknown parameters

Bias - Variance Tradeoff

What class of hypotheses should you search over?

Standard Statistical ML story:



higher complexity models have lower bias but higher variance

If complexity is too high, it overfits data, variance term dominates test error

after a certain threshold, "larger models are worse"

degree of polynomial you are learning

Why is training error monotonically decreasing? wrt complexity

In training, you are minimizing (empirical) risk
Higher complexity model results in a larger search space.
Minimizing the same function over a larger search space gets you at least as good of an answer

Why is test error initially decreasing?

For low complexity models, you are "underfitting". Additional complexity allows you to better represent the "true" response.

Why does test error start increasing after a given point?
You begin over fitting. You begin estimating parameters in order to fit to the noise

If you have 10^3 data samples, how complex of a data model would you consider? $\{(x_i, y_i)\}$ w/ $y_i = f(x_i) + \varepsilon_i$

If I used a model with 1000 parameters, I would expect to fit the noise.

100?

Why does understanding this tradeoff matter?

It helps you train models.
Allow you to choose reasonable value of complexity for your problems.
Tells you: check for overfitting

If you fit all of your training data perfectly, that is bad (in the classical ML perspective)

Why shouldn't you use test data to estimate model parameters? Wouldn't more data lead to a better model?

With more data, you would get a better model. With no test data, you won't know how good your model is.

We want to know how good our model is? Without test data, we have no assessment

DON'T TEST ON THE TRAINING DATA. DON'T TRAIN ON THE TESTING DATA

What is the role of validation data and test data?

Bias - Variance Decomposition

Consider regression model

$$y = f(x) + \varepsilon \qquad \text{w/} \quad \mathbb{E}[\varepsilon \mid x] = 0$$

Let $S = \{(x_i, y_i)\}_{i=1\cdots n}$ be iid samples

Estimate f by an algorithm producing $\hat{f}_S$

Evaluate $\hat{f}_S$ by expected loss on a new sample

$$\underbrace{R}_{\text{risk}}(\hat{f}_S) = \underbrace{\mathbb{E}_{x,y}}_{\text{test sample}} \underbrace{(\hat{f}_S(x) - y)^2}_{\text{square loss}}$$

Performance will vary based on S. Take expectation over S.

$$\mathbb{E}_S\, R(\hat{f}_S) = \mathbb{E}_{x,y,S}(\hat{f}_S(x) - y)^2$$

We will decompose into 3 effects: bias, variance, irreducible error

$$\mathbb{E}_S\, R(\hat{f}_S) = \mathbb{E}_{x,y,S}\left[\left((\hat{f}_S(x) - f(x)) - \varepsilon\right)^2\right]$$

$$= \mathbb{E}_{x,y,S}(\hat{f}_S(x) - f(x))^2 - \cancelto{0}{2\,\mathbb{E}\left[(\hat{f}_S(x) - f(x))\varepsilon\right]} + \underbrace{\mathbb{E}[\varepsilon^2]}_{Var(\varepsilon)}$$

$$= \mathbb{E}_{x,y,S}(\hat{f}_S(x) - f(x))^2 + Var(\varepsilon)$$

Evaluating the first term, Conditioning on x,

$$\mathbb{E}_S(\hat{f}_S(x) - f(x))^2 = \mathbb{E}_S\left[\left((\hat{f}_S(x) - \mathbb{E}_S \hat{f}_S(x)) + (\mathbb{E}_S \hat{f}_S(x) - f(x))\right)^2\right]$$

$$= \mathbb{E}_S\left((\hat{f}_S(x) - \mathbb{E}_S \hat{f}_S(x))\right)^2 + 2\,\mathbb{E}_S \underbrace{\cancelto{0}{(\hat{f}_S(x) - \mathbb{E}_S \hat{f}_S(x))}}_{\text{0 in expectation in } S} \underbrace{(\mathbb{E}_S \hat{f}_S(x) - f(x))}_{\text{does not depend on } S} + \mathbb{E}_S(\mathbb{E}_S \hat{f}_S(x) - f(x))^2$$

$$= \underbrace{\mathbb{E}_S\left(\hat{f}_S(x) - \mathbb{E}_S\hat{f}_S(x)\right)^2}_{\text{Variance of } \hat{f}_S(x)} + \underbrace{\left(\mathbb{E}_S\left(\hat{f}_S(x) - f(x)\right)\right)^2}_{\text{squared bias}}$$

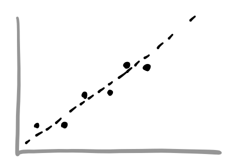
So,

$$\mathbb{E}_S R(\hat{f}_S) = \mathbb{E}_{x,y}\left[\mathbb{E}_S\left(\hat{f}_S(x) - \mathbb{E}_S\hat{f}_S(x)\right)^2 + \left(\mathbb{E}_S\left(\hat{f}_S(x) - f(x)\right)\right)^2\right] + Var(\varepsilon)$$

$$= \underbrace{\mathbb{E}_x\left(f(x) - \mathbb{E}_S\hat{f}_S(x)\right)^2}_{\text{expected squared bias of estimate}} + \underbrace{\mathbb{E}_x Var_S \hat{f}_S(x)}_{\text{expected variance of estimate}} + \underbrace{Var(\varepsilon)}_{\text{irreducible error}}$$

Illustration of bias variance tradeoff

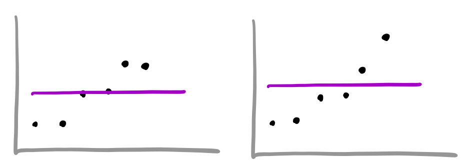
Suppose $y = x + \varepsilon$



Low complexity model : $y = c$

$\mathbb{E}_x\left(f(x) - \mathbb{E}_S\hat{f}_S\right)^2$ is high

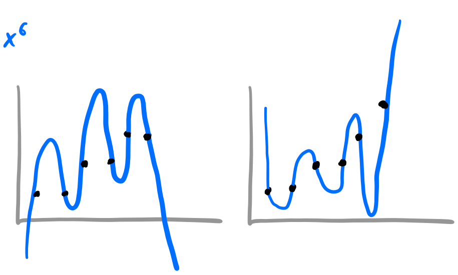
$\mathbb{E}_x Var_S \hat{f}_S(x)$ is low



High complexity model : $y = c_0 + c_1x + c_2x^2 + \cdots c_6x^6$

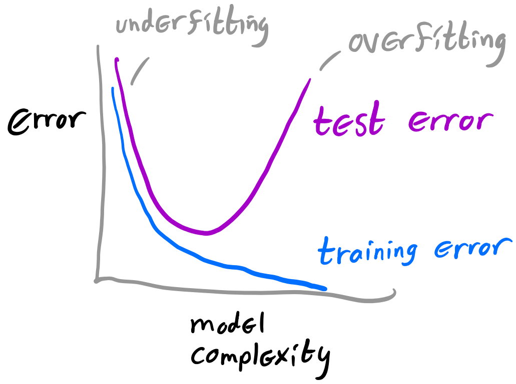
$\mathbb{E}_x\left(f(x) - \mathbb{E}_S\hat{f}_S\right)^2$ is low

$\mathbb{E}_x Var_S \hat{f}_S(x)$ is high



Bias - Variance Tradeoff

Standard Statistical ML story:

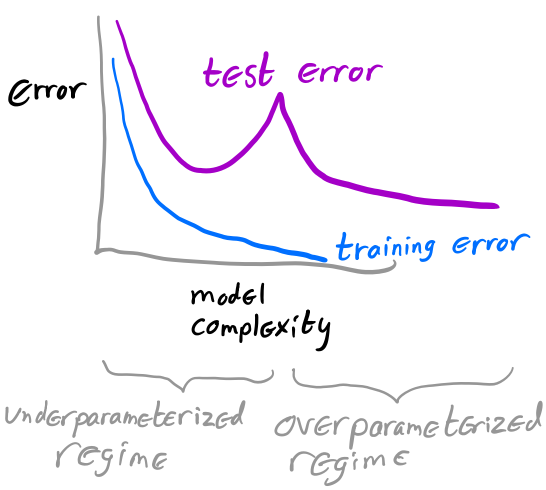


higher complexity models have lower bias but higher variance

If complexity is too high, it overfits data, variance term dominates test error

after a certain threshold, "larger models are worse"

Modern Story based on Neural Nets:



Test error can decrease as model complexity continues increasing.

And it can be lower than in underparameterized regime

Phenomenon: double descent

"larger models are better"

If you have 10^3 data samples, how complex of a data model would you consider?

Why is being critically parametrized bad for generalization?

In the overparameterized regime, do all models with 0 training error generalize well?

How is good generalization possible in the overparameterized regime?

Why does understanding this tradeoff matter?