S&DS 365 / 565

Data Mining and Machine Learning

Cross Validation

Yale

Model Selection

For purposes of prediction, **minimizing test error** is priority.

Recall our two error metrics for evaluating predictions $\widehat{f}(x_i)$:

(empirical risk)

- Regression:

mean squared error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}(x_i))^2$$

- Classification: misclassification

$$Err = \frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\left\{\widehat{f}(x_i) \neq y_i\right\}$$

Bias-Variance Tradeoff

f^* true f never know
$\hat{f}$ estimated f } we know
$\bar{f}$ optimal f

eg (Classification)

pop risk

Define $R(f) = \mathbb{E}\mathbb{1}(f(X) \neq Y) = \mathbb{P}(f(X) \neq Y)$

estimated f

Suppose that $\hat{f}$ comes from some model class $\mathcal{M}$

true f

Let $f^* \in \arg\min R(f)$ such that $f \in \mathcal{M}$

optimal f: $\bar{f}$ (f bar) $= \mathbb{1}[P(y=1|x) > P(y=0|x)]$

optimal error $f \in \mathcal{M}$ statistical error

$$R(\hat{f}) - R(\bar{f}) = \underbrace{R(\hat{f}) - R(f^*)}_{\text{variance}} + \underbrace{R(f^*) - R(\bar{f})}_{\text{bias}}$$

Excess Risk

- proper learning: assum $\hat{f} \in \mathcal{M}$
- inproper learning: $\hat{f} \in$ bigger function class

variance↑ error↑
bias↑ error↑

bias: ① bias on function class
I think my data fit into a linear model
hope bias↓ when degree of polynomial↑
② KNN when k=1, no bias
when k↑ bias↑ function more smooth

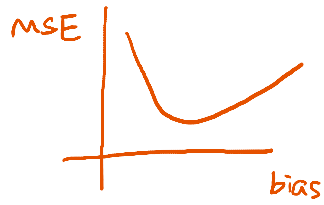
bias is NOT random

variance: how much your function $\hat{f}$ fluctuate from one random data sample to another
eg ① when k=1, change sample → $\hat{f}$ change dramatically
when k↑ $\hat{f}$ change smaller

variance is random coz $\hat{f}$ is random we generate a $\hat{f}$ from a random sample

Bias-Variance Tradeoff

often bias↑ variance↓



Eg (Regression case)

Model

Given $Y = f(X) + \epsilon$, where $E(\epsilon) = 0$ and $Var(\epsilon) = \sigma^2$, consider a predictor $\hat{f}$.

Assumption for calculate bias

Expected MSE for predicting a new Y at $X = x$ can be decomposed into:

$$E(MSE) = E[(Y - \hat{f}(x))^2] = Var(\hat{f}(x)) + [Bias(\hat{f}(x))]^2 + \sigma^2$$

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

excess risk

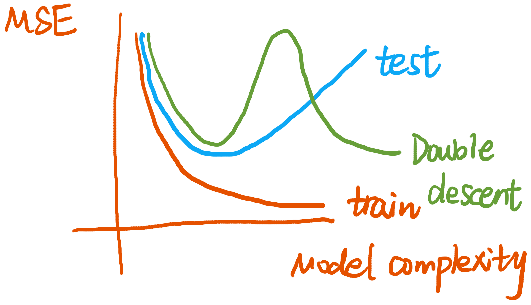
↓ noise

2 random things, other are numbers

ε is independent of other thing, pull out ε get σ^2

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

$$Bias\,\hat{f}(x) = [f(x) - E\hat{f}(x)]^2$$



Bias-Variance Tradeoff

$$E[(Y - \widehat{f}(x))^2] = Var(\widehat{f}(x)) + [Bias(\widehat{f}(x))]^2 + \sigma^2$$

- $Var(\widehat{f})$ is the amount of variability in our predictor with different training set.
- $Bias(\widehat{f})$ is the systematic error introduced by model approximation.
- σ^2 is *irreducible error*, inherent in the error term ϵ.

Bias-Variance Tradeoff

$$E[(Y - \widehat{f}(x))^2] = Var(\widehat{f}(x)) + [Bias(\widehat{f}(x))]^2 + \sigma^2$$

- $Var(\widehat{f})$ is the amount of variability in our predictor with different training set. Increases with increasing model flexibility. (complexity)
- $Bias(\widehat{f})$ is the systematic error introduced by model approximation. Decreases with increasing model flexibility.
- σ^2 is *irreducible error*, inherent in the error term ϵ. Cannot get rid of this!

Need to balance bias and variance.

Bias-Variance?

Let's take a step back. Compare this equation

$$E[(Y - \widehat{f}(x))^2] = Var(\widehat{f}(x)) + [Bias(\widehat{f}(x))]^2 + \sigma^2$$

with what we are more much more familiar with: if $Y \sim N(\theta, \sigma^2)$, then with $\widehat{\theta} = Y$, we have

$$Var(\widehat{\theta}) = \sigma^2, Bias(\widehat{\theta}) = E(\hat{\theta}) - \theta$$

KNN $k=1 \Rightarrow$ Bias $= \theta - \theta = 0$

Bias-Variance?

Let's take a step back. Compare this equation

$$E[(Y - \widehat{f}(x))^2] = Var(\widehat{f}(x)) + [Bias(\widehat{f}(x))]^2 + \sigma^2$$

with what we are more much more familiar with: if $Y \sim N(\theta, \sigma^2)$, then with $\widehat{\theta} = Y$, we have

take $\hat{\theta}$ is very small $\hat{\theta} = 0$

$Var(\hat{\theta}) = 0$

$Bias(\hat{\theta}) = \theta - 0 = \theta$

$$Var(\widehat{\theta}) = \sigma^2, Bias(\widehat{\theta}) = \theta$$

What does it mean to talk about bias/variance of $\widehat{f}$?

Bias-Variance

How does this apply to the k-NN algorithm?

$k=1$, no bias

$k \uparrow$ bias $\uparrow$ variance $\downarrow$

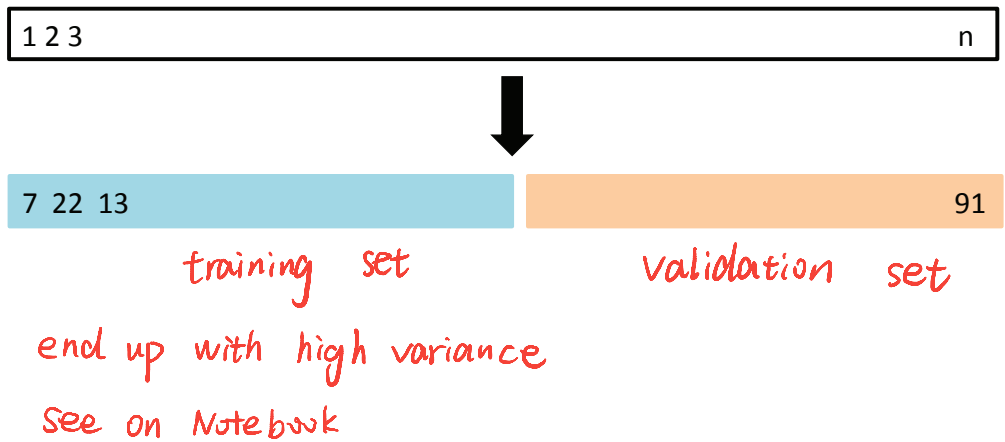
Cross-Validation

Cross-validation is an intuitive, widely-applicable approach for:

- model assessment
- model selection

Validation Sets

We've been doing this:



1. Divide dataset randomly into a training set and a validation set.
2. Fit the model on the training set.
3. Use the validation set to obtain estimated test error.
4. Repeat!

Validation Sets

- highly variable validation error
- only uses a fraction of the training set

Leave-One-Out Cross-Validation

How do we use more data to train with?

- Use a tiny validation set (e.g. (x_1, y_1)) one sample
- Train with the rest (e.g. $\{(x_2, y_2), \ldots, (x_n, y_n)\}$)

How do we feel about error rate evaluated on a single observation?

Leave-One-Out Cross-Validation

How do we use more data to train with?

- Use a tiny validation set (e.g. (x_1, y_1))
- Train with the rest (e.g. $\{(x_2, y_2), \ldots, (x_n, y_n)\}$)

How do we feel about error rate evaluated on a single observation?

Not good, but we can iterate through the dataset, each time using a different (x_i, y_i) as the validation set and obtaining an error MSE_i.

train on rest samples (n-1)

Leave-One-Out Cross-Validation

	Iteration					
Obs	1	2	3	4	...	n
1	valid	train	train	train	...	train
2	train	valid	train	train	...	train
3	train	train	valid	train	...	train
4	train	train	train	valid	...	train
...	...	...	...	...	...	...
n	train	train	...	...	...	valid
MSE	MSE_1	MSE_2	MSE_3	MSE_4	...	MSE_n

Leave-One-Out Cross-Validation

n-fold cross validation

n is iteration times = sample size

if sample size is small, do that

LOOCV estimate of test error is given by:

$$CV_{(n)} = \frac{1}{n} \sum_i MSE_i$$

k-fold Cross-Validation

A potentially faster approach: if sample size is large

- Randomly divide the dataset into k *folds*.
- For $b = 1, \ldots, k$:
  - Use b-th fold ("batch") as validation set.
  - Use everything else as training set.
  - Compute validation error on b-th fold.
- Estimate test error using:

$$CV_{(k)} = \sum_b \frac{n_b}{n} MSE_b,$$

usually n_b is same for k folds

where n_b is the total # observations in the b-th fold, and n is the total # observations in the entire dataset.

k-fold Cross-Validation

	Iteration						
Obs	1	2	3	4	...	k	
1	valid	train	train	train	...	train	fold 1
2	valid	train	train	train	...	train	
3	valid	train	train	train	...	train	
4	train	valid	train	train	...	train	
...	...	...	...	...	...	...	
$n-2$	train	train	...	...	...	valid	fold k
$n-1$	train	train	...	...	...	valid	
n	train	train	...	...	...	valid	
MSE	MSE_1	MSE_2	MSE_3	MSE_4	...	MSE_k	

k-fold Cross-Validation

	Iteration						
Obs	1	2	3	4	...	k	
1	valid	train	train	train	...	train	fold 1
2	valid	train	train	train	...	train	
3	valid	train	train	train	...	train	
4	train	valid	train	train	...	train	
...	...	...	...	...	...	...	
$n-2$	train	train	...	...	...	valid	fold v
$n-1$	train	train	...	...	...	valid	
n	train	train	...	...	...	valid	
MSE	MSE_1	MSE_2	MSE_3	MSE_4	...	MSE_k	

n-fold CV is just LOOCV.

A slick shortcut

Suppose that the fitted values can be written $\widehat{Y} = LY$ where L is an $n \times n$ matrix.

$\widehat{Y} = X(X^TX)^{-1}X^TY$

L

This is the only case for least squares regression

Then the leave-one-out-cross-validation error is (Risk)

$$R_{LOO} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_{(-i)})^2$$

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \widehat{Y}_i}{1 - L_{ii}}\right)^2$$

where L_{ii} is the ith diagonal entry.

A slick shortcut

Suppose that the fitted values can be written $\widehat{Y} = LY$ where L is an $n \times n$ matrix.

This is the case for least squares regression

Then the leave-one-out-cross-validation error is

Empirical Risk = Mean squared error

$$R_{LOO} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_{(-i)})^2$$

(Y_i: true Y of ith observ.; $\widehat{Y}_{(-i)}$: estimate of ith observ. trained on excluding ith observ.)

$$= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \widehat{Y}_i}{1 - L_{ii}}\right)^2$$

($\widehat{Y}$ trained on full data set)

where L_{ii} is the ith diagonal entry.

So, no need to fit n regressions!

only need 1 regression

Model Selection

So far, we've used it to estimate the test error. It's also useful for model selection.

Suppose we are interested in comparing the following 3 models:

Model 1: linear model

$$\widehat{medv} = \widehat{\beta}_0 + \widehat{\beta}_1 lstat$$

Model 2: log model

$$\widehat{log(medv)} = \widehat{\beta}_0 + \widehat{\beta}_1 lstat$$

Model 3: quadratic model

$$\widehat{medv} = \widehat{\beta}_0 + \widehat{\beta}_1 lstat + \widehat{\beta}_2 lstat^2$$

Model Selection

So far, we've used it to estimate the test error. It's also useful for model selection.

Suppose we are interested in comparing the following 3 models:

Model 1:

$$\widehat{medv} = \widehat{\beta}_0 + \widehat{\beta}_1 lstat$$

Model 2:

$$\widehat{log(medv)} = \widehat{\beta}_0 + \widehat{\beta}_1 lstat$$

Model 3:

$$\widehat{medv} = \widehat{\beta}_0 + \widehat{\beta}_1 lstat + \widehat{\beta}_2 lstat^2$$

We can use cross-validation to estimate the test error for each of these models, and select the model with the lowest test error.

Model Selection: Workflow

In order to do model selection, you need a set of models $\{M_1, \ldots, M_m\}$:

1. for each model M_i:
    1. calculate k-fold CV validation error learn $m \times k$ models
2. choose model with the lowest k-fold CV validation error M_{min}
3. apply M_{min} to the full dataset

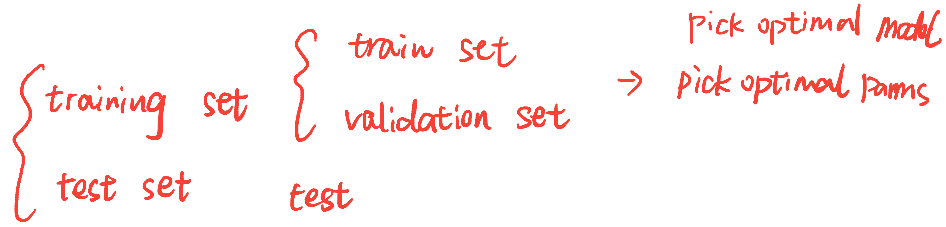
Model Selection: Workflow

In order to do model selection, you need a set of models $\{M_1, \ldots, M_m\}$:

1. for each model M_i:
   1. calculate k-fold CV validation error
2. choose model with the lowest k-fold CV validation error M_{min}
3. apply M_{min} to the full dataset — to learn params of M_{min}
   can't report this lowest CV error as model performance measure
4. Don't necessarily pick the lowest always. If have a lot of models that are similar, might pick the lowest complexity model that's close enough to the smallest error. For KNN low-complexity means higher k. For polynomials, lower degree
   coz you use validation set to pick model and still use validation set for optimization
   may cause underreport of error (over optimistic about models)
   pick model with 1 standard deviation of lowest CV error

A safer way

We have split into two datasets.



If you want an accurate assessment of the true test error need to split off a third set that is the **true** test set.

The result of model selection based on just validation will be over optimistic

To have an unbiased estimate of the final model error, need to use third **untouched** test dataset.

untouched means the dataset was not ever used to pick a model

With experience can develop a sense to connect validation set error to true error and can just ignore the test set