# Machine Learning for Finance (FIN 570) Hyperparameter Tuning: Bias-Variance Tradeoff, Cross-Validation, and Evaluation Metric

Instructor: Jaehyuk Choi

Peking University HSBC Business School, Shenzhen, China

2090-20 Module 3 (Spring 2020)

# Regularization L-1 vs L-2

Give a penalty for complexity or overfitting. The cost function to minimize:

$$J(\boldsymbol{w}) = J_0(\boldsymbol{w}) + \lambda\, R(\boldsymbol{w}) \quad (= C\, J_0(\boldsymbol{w}) + R(\boldsymbol{w})),$$

where $J_0(\boldsymbol{w})$ is the un-regularized cost function, e.g., log-likelihood (logistic), RSS (linear) or slack variable sum (SVM).

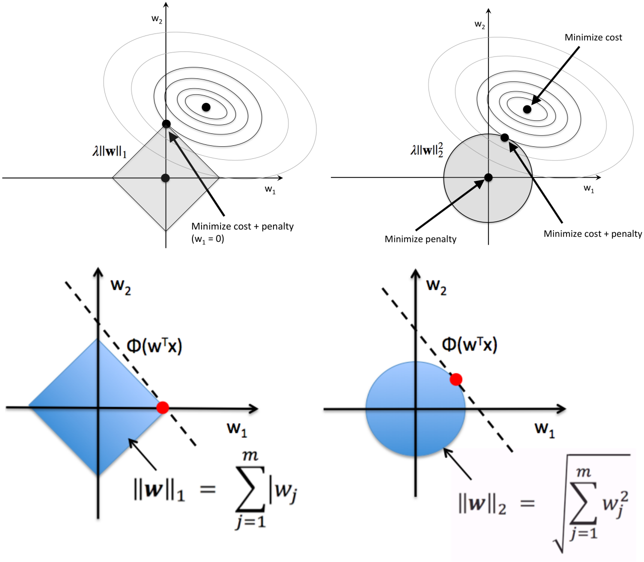
## L-2 Regularization

- $R(\boldsymbol{w}) = ||\boldsymbol{w}||_2^2 = \sum_j w_j^2$
- $N$-sphere boundary (e.g., circle or sphere). Easy to locate the minimum.

## L-1 Regularization

- $R(\boldsymbol{w}) = ||\boldsymbol{w}||_1 = \sum_j |w_j|$
- 'Diamond' boundary: leads to sparse vector (many zero components)
- Effectively works as feature selection

# Regularization L-1 vs L-2



# Measuring quality of ML method

Given a ML method, we want to minimize the mean squared error (MSE) on **test data set** (expected test MSE).

$$E\Big(y - \hat{f}(x)\Big)^2 = \text{Bias}(\hat{f}(x))^2 + \text{Var}(\hat{f}(x)) + \text{Var}(\varepsilon)$$

$$\text{where,} \quad y = f(x) + \varepsilon \quad \text{(true pattern)}$$

- By *a given ML method*, we mean that model (LR, SVM, etc) and hyper-parameter ($C$, $\gamma$, reduced dimension $k$ for PCA/LDA, etc) are fixed. However fitted model parameters (i.e., $\hat{f}$) can change over training set.
- The expectation is made over repeatedly selecting different training vs test dataset. Therefore, the expectation is over $\hat{f}$ as well as $x$.
- We need to minimize Bias$(\hat{f}(x))^2$ and Var$(\hat{f}(x))$ together while Var$(\varepsilon)$ is fundamentally irreducible.

# Bias and Variance

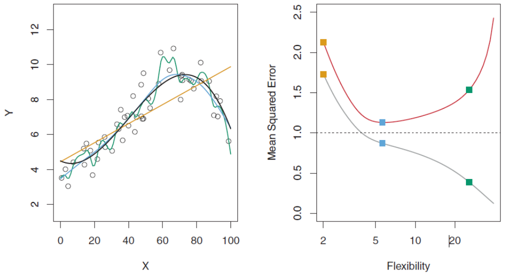
## Bias

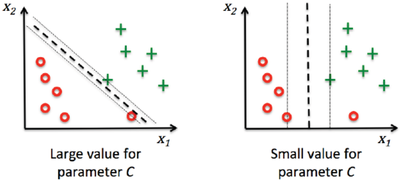
- Error from $\hat{f}$ not correctly representing the true $f$ (e.g. linear regression on non-linear data).
- A model has **high bias** when $\hat{f}$ overly simply $f$ (under-fitting), i.e., the used parameters are too few.

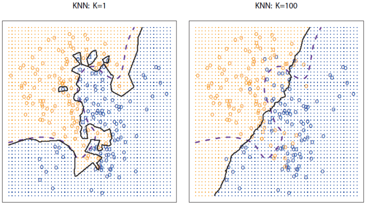
## Variance

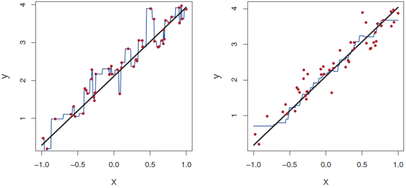
- Error from variability or sensitivity (vs consistency) of the trained model $\hat{f}$ against the selection of training dataset.
- A model has **high variance** when the model is too flexible (overfitting), i.e., there are too many parameters, e.g. KNN with $K = 1$, high-order polynomial regression, SVM/LR with large $C$ (small $\lambda$), decision tree with many leaves, etc.

# Bias and Variance (examples)



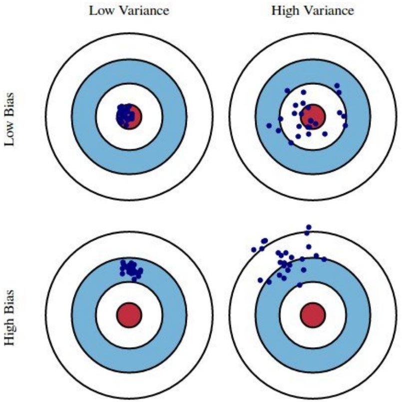



- Grey line: Bias vs the number of parameters
- Red line: MSE measured with the true $f$ (black line).

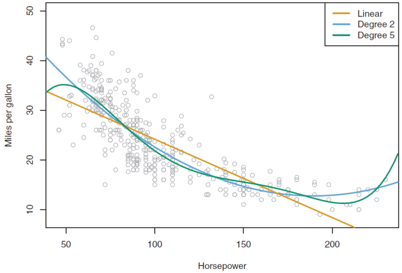
# Bias-Variance Tradeoff

- It is hard to reduce both bias and variance.
- As the model flexibility increase, bias decreases but variance increases. It is important to find a right trade-off.
- Bias-variance tradeoff is one of the most important theme in ML (and other fields!).
- In real problems, the true pattern $f$ is unknown and the dataset size is limited. How can we efficiently measure the expected test MSE?

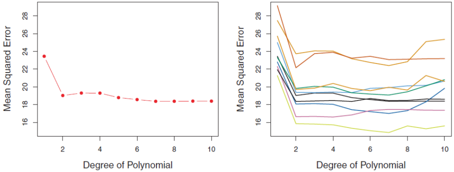


# Cross-Validation(CV): Validation Set (Hold-out set)

- Divide observations into a **training set** and a **validation (hold-out) set**.
- Fit model on the training set and measure error on the validation set.
- Error rate is highly variable (sensitive to division) and over-estimated than the true test error rate as the model is trained on fewer observations.
- Training set is further divided into **training** and **validation** sets. Validation set is used for model selection and hyper-parameter funning.

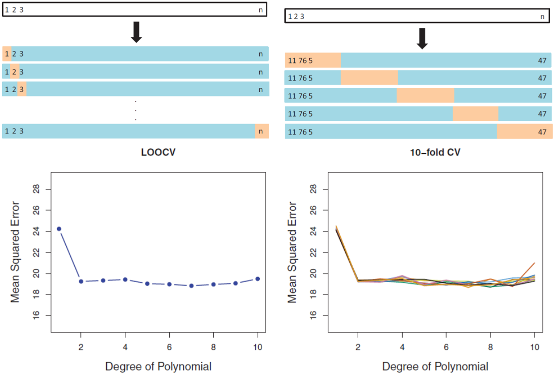






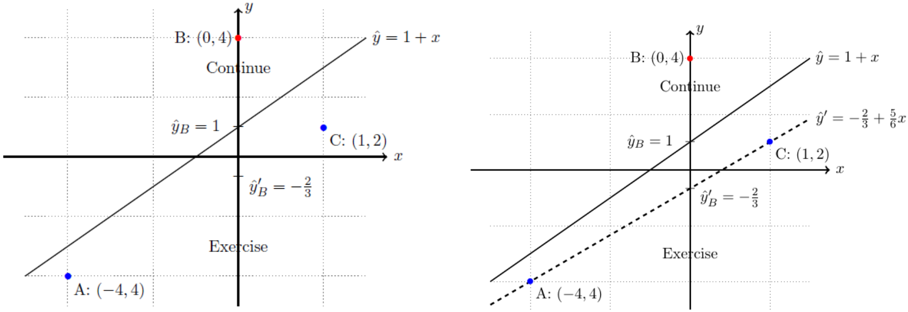
# Cross-Validation: Leave-One-Out (LOOCV) and $k$-Fold CV

- LOOCV: train model with one sample left out and measure the error on the sample. Error is close to the true test rate but computation is heavy (train $n$ times).
- $k$-fold CV: divide the samples into $k$ (typically 5 or 10) folds. Train model on $k-1$ **training** folds and measure error on the remaining **test** fold.



# LOOCV in linear regression (1/3)

- LOOCV can be computed analytically in linear regression.
- An example with 3 data points:



# LOOCV in linear regression (2/3)

The multivariate regression $Y \sim X\beta$:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{H}\boldsymbol{y}, \quad \text{where} \quad \boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y},\ \boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$$

$$\boldsymbol{y} = \begin{bmatrix} \vdots \\ y_j \\ \vdots \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} \vdots \\ -\boldsymbol{x}_j- \\ \vdots \end{bmatrix} \begin{matrix} (M \times N) \\ (M \ll N) \end{matrix}, \qquad \begin{matrix} \boldsymbol{x}_j : j\text{-th row vector of } \boldsymbol{X} \\ y_j : j\text{-th value of } \boldsymbol{y} \end{matrix}$$

Let $\boldsymbol{X}_{\text{-}j}$ and $\boldsymbol{y}_{\text{-}j}$ be $\boldsymbol{X}$ and $\boldsymbol{y}$ with $j$-th row removed, respectively.
To compute the regression coefficients $\boldsymbol{\beta}_{\text{-}j}$ from $\boldsymbol{X}_{\text{-}j}$ and $\boldsymbol{y}_{\text{-}j}$, we use

$$\boldsymbol{X}_{\text{-}j}^T\boldsymbol{X}_{\text{-}j} = \boldsymbol{X}^T\boldsymbol{X} - \boldsymbol{x}_j^T\boldsymbol{x}_j, \quad \boldsymbol{X}_{\text{-}j}^T\boldsymbol{y}_{\text{-}j} = \boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{x}_j^T y_j,$$

and the Sherman-Morrison formula, (intuition: $\frac{1}{X-\varepsilon} \approx \frac{1}{X} + \frac{\varepsilon}{X^2}$)

$$(\boldsymbol{X}_{\text{-}j}^T\boldsymbol{X}_{\text{-}j})^{-1} = (\boldsymbol{X}^T\boldsymbol{X})^{-1} + \frac{(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{x}_j^T\boldsymbol{x}_j(\boldsymbol{X}^T\boldsymbol{X})^{-1}}{1-h_j},$$

where $\boldsymbol{h}$ be the diagonal vector of the hat matrix $\boldsymbol{H}$:

$$\boldsymbol{h} = \text{diag}\left(\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\right) \quad \text{or} \quad h_j = \boldsymbol{x}_j(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{x}_j^T$$

# LOOCV in linear regression (3/3)

- The regression coefficients $\hat{\boldsymbol{\beta}}_{\text{-}j}$ (the $j$-th sample removed) is

$$\begin{aligned}
\hat{\boldsymbol{\beta}}_{\text{-}j} &= (\boldsymbol{X}_{\text{-}j}^T \boldsymbol{X}_{\text{-}j})^{-1} \boldsymbol{X}_{\text{-}j}^T \boldsymbol{y}_{\text{-}j} \\
&= \left( (\boldsymbol{X}^T\boldsymbol{X})^{-1} + \frac{(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{x}_j^T\boldsymbol{x}_j(\boldsymbol{X}^T\boldsymbol{X})^{-1}}{1-h_j} \right) (\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{x}_j^T y_j) \\
&= \hat{\boldsymbol{\beta}} - (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{x}_j^T \frac{e_j}{1-h_j} \quad \text{for the prediction error } \boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}}.
\end{aligned}$$

- Moreover, the corrected estimation values $\hat{\boldsymbol{y}}'$ and the new prediction errors $\boldsymbol{e}'$ for all points are obtained in one go as

$$\hat{\boldsymbol{y}}' = \hat{\boldsymbol{y}} - \frac{\boldsymbol{h}\cdot\boldsymbol{e}}{\boldsymbol{1}-\boldsymbol{h}} \quad \text{or} \quad \boldsymbol{e}' = \frac{\boldsymbol{e}}{\boldsymbol{1}-\boldsymbol{h}},$$

  where $\cdot$ and the fraction are the element-wise operations.
- Given that $0 < h_j < 1$, the correction is always in the direction of reducing over-fitting or increasing the prediction error.
- Little extra computation: $\boldsymbol{h}$ is a byproduct of the regression.

$$\boldsymbol{h} = \text{row sum}(\boldsymbol{X}\cdot\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}) \Leftarrow \boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

# Evaluation Metrics

## Confusion Matrix

| Credit card Default | | Predicted | | |
|---|---|---|---|---|
| | | $P^*$ | $N^*$ | Total |
| Actual | $P$ | 40 | 40 | 80 |
| | $N$ | 10 | 910 | 920 |
| | Total | 50 | 950 | 1000 |

| | | **Predicted class** | |
|---|---|---|---|
| | | $P^*$ | $N^*$ |
| **Actual class** | $P$ | True positives (TP) | False negatives (FN) |
| | $N$ | False positives (FP) | True negatives (TN) |

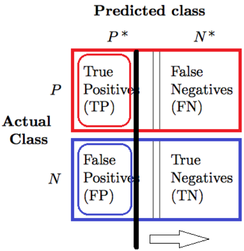
- Accuracy (ACC) $= \dfrac{\text{TP} + \text{TN}}{\text{ALL}} = \dfrac{40 + 910}{1000} = 95\%$
- Error (ERR) $= 1 - \text{ACC} = \dfrac{\text{FP} + \text{FN}}{\text{ALL}} = \dfrac{10 + 40}{1000} = 5\%$
- However, accuracy/error may be misleading!

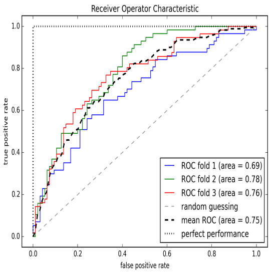
# Evaluation Metrics

| | | Predicted | |
|---|---|---|---|
| | | $P^*$ | $N^*$ |
| Actual | $P$ | TP (40) | FN (40) |
| | $N$ | FP (10) | TN (910) |

- Precision $(\texttt{PRE}) = \dfrac{\text{TP}}{P^*} = \dfrac{\text{TP}}{\text{TP}+\text{FP}} = \dfrac{40}{50} = 80\%$
  Case: Spam mail filter (minimize FP)
- Recall $(\texttt{REC}) = \dfrac{\text{TP}}{P} = \dfrac{\text{TP}}{\text{TP}+\text{FN}} = \dfrac{40}{80} = 50\%$
  Case: Credit approval, Cancer diagnosis (minimize FN)
- F1-Score $(\texttt{F1}) = \dfrac{2\,\text{PRE}\times\text{REC}}{\text{PRE}+\text{REC}} = 61.5\% \quad \left(\dfrac{2}{\text{F1}} = \dfrac{1}{\text{PRE}} + \dfrac{1}{\text{REC}}\right)$
  The harmonic average of PRE and REC to ensure $0 \le \text{F1} \le 1$
  A widely used accuracy for binary classification with imbalanced sample.

# Receiver Operator Characteristic (ROC) Curve




- True Positive Rate ($\texttt{TPR}=\texttt{REC}$) = TP$/P = 50\%$
- False Positive Rate ($\texttt{FPR}$) = FP$/N = 10/920 = 1.1\%$
- ROC Curve: graph of (FPR, TPR) for varying classification threshold of the binary classification.
- Area Under Curve (AUC) give an overall accuracy of a classifier, summarizing over all possible threshold
- The diagonal line is from random-guessing: ROC AUC = 0.5
  A model with lower AUC than 0.5 is worthless.
- A perfect classifier ($\Gamma$-shaped lines): ROC AUC = 1.