

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

2021-22 Module 1 (Fall 2021)

Regularization L-1 vs L-2

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

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

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

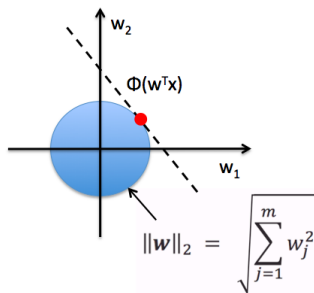
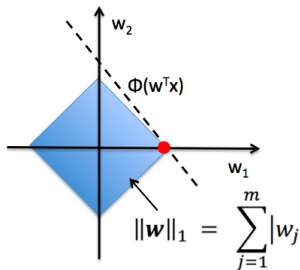
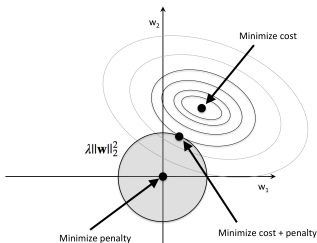
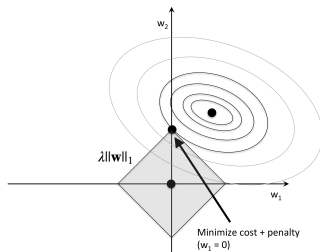
L-2 Regularization

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

L-1 Regularization

- $R(\mathbf{w}) = \|\mathbf{w}\|_1 = \sum_j |w_j|$
- '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).

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

where $y = f(x) + \varepsilon$ (true pattern)

- By “given a ML method”, we mean that model (LR, SVM, etc) and hyper-parameter (C , γ , 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 $\text{Bias}(\hat{f}(x))^2$ and $\text{Var}(\hat{f}(x))$ together while $\text{Var}(\varepsilon)$ is fundamentally irreducible.

Bias and Variance

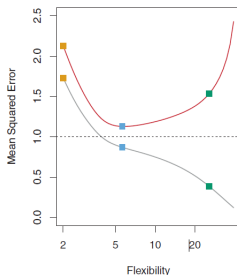
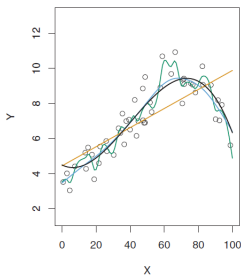
Bias

- 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.

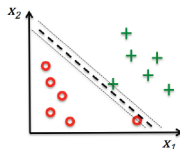
Variance

- 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 λ), decision tree with many leaves, etc.

Bias and Variance (examples)

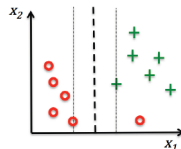


- Grey line: Bias vs the number of parameters
- Red line: MSE measured with the true f (black line).



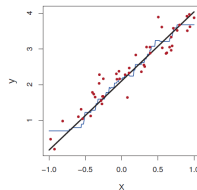
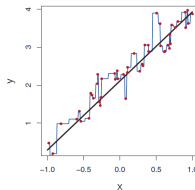
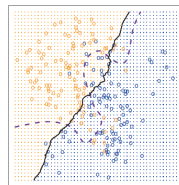
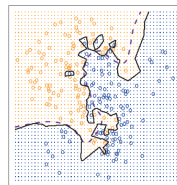
Large value for parameter C

KNN: $K=1$



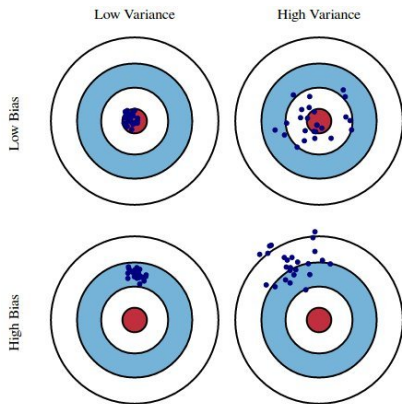
Small value for parameter C

KNN: $K=100$



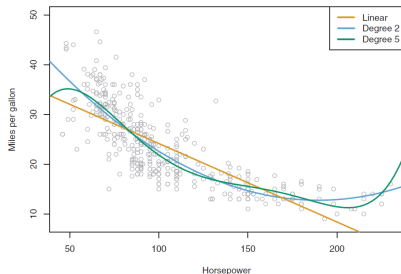
Bias-Variance Trade-off

- It is hard to reduce bias and variance simultaneously (but easy to have high bias and variance simultaneously).
- As model flexibility increases, bias decreases but variance increases. It is important to find a right trade-off.
- Bias-variance trade-off is one of the most important theme in ML (and other fields!).
- In real problems, the true pattern $f(x)$ 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)

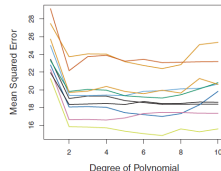
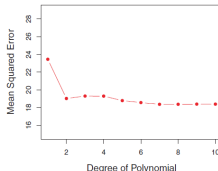
- Divide observations into a **training set** and a **validation (hold-out) set**.
- Fit model on the training set and measure error on the validation set.
- Error rate is highly variable (sensitive to division) and over-estimated than the true test error rate as the model is trained on fewer observations.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



1 2 3 n

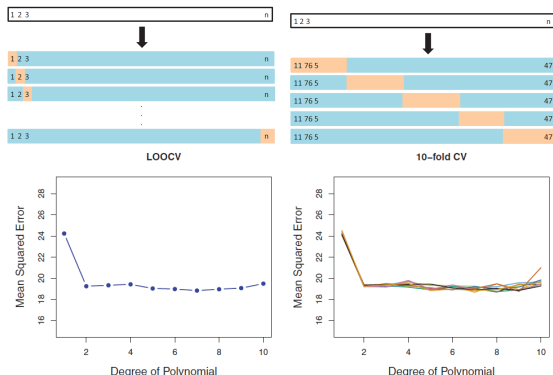


7 22 13 91



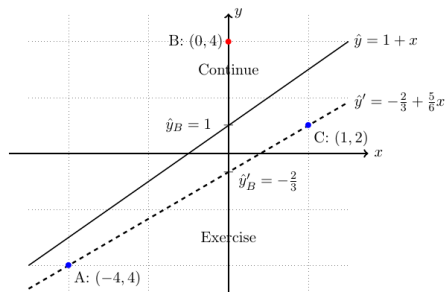
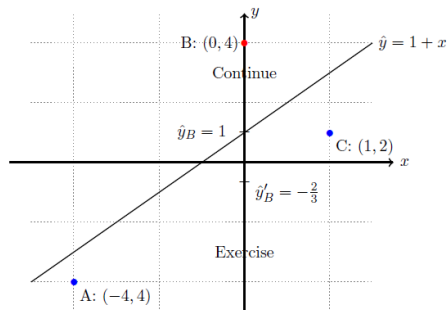
Cross-Validation: Leave-One-Out (LOOCV) and k -Fold CV

- 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 $Y \sim X\beta$:

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

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

Let \mathbf{X}_{-j} and \mathbf{y}_{-j} be \mathbf{X} and \mathbf{y} with j -th row removed, respectively.

To compute the regression coefficients β_{-j} from \mathbf{X}_{-j} and \mathbf{y}_{-j} , we use

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

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

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

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

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

LOOCV in linear regression (3/3)

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

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

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

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

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: \mathbf{h} is a byproduct of the regression.

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

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

Evaluation Metrics

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

- Precision (PRE) = $\frac{TP}{P^*} = \frac{TP}{TP + FP} = \frac{40}{50} = 80\%$

Case: Spam mail filter (minimize FP)

- Recall (REC) = $\frac{TP}{P} = \frac{TP}{TP + FN} = \frac{40}{80} = 50\%$

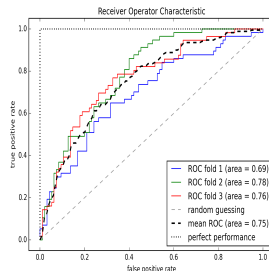
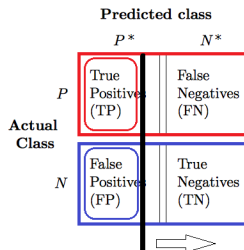
Case: Credit approval, Cancer diagnosis (minimize FN)

- F1-Score (F1) = $\frac{2 \text{ PRE} \times \text{REC}}{\text{PRE} + \text{REC}} = 61.5\% \quad \left(\frac{2}{F1} = \frac{1}{\text{PRE}} + \frac{1}{\text{REC}} \right)$

The harmonic average of PRE and REC to ensure $0 \leq F1 \leq 1$

A widely used accuracy for binary classification with imbalanced sample.

Receiver Operator Characteristic (ROC) Curve



- True Positive Rate ($\text{TPR}=\text{REC}$) = $\text{TP}/P = 50\%$
- False Positive Rate (FPR) = $\text{FP}/N = 10/920 = 1.1\%$
- ROC curve is the set of (FPR, TPR) points for different classification binary classification threshold p from 0 to 1.
- Area under the curve (AUC) gives an accuracy of a classifier summarizing over all possible threshold
 - Perfect model: ROC AUC = 1 (Γ -shaped lines)
 - Random guess: ROC AUC = 0.5 ($y = x$ line, $\text{TPR}=\text{FPR}=1 - p$)
 - A model with ROC AUC < 0.5 is worthless.

Dealing with Class Imbalance

- Class imbalance is common, causing the model biased to the majority class.
- Use alternative metrics (e.g., PRE, REC, F1) in model selection.
- Use `class_weight='balanced'` option in sklearn. If r is the ratio of the minority class,

$$\log L = \sum_i \frac{y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)}{w_i}, \quad w_i = \begin{cases} 2r & \text{if } y_i = 1 \\ 2(1 - r) & \text{if } y_i = 0 \end{cases}$$

- Undersampling: add copies of minority class samples
- Oversampling: delete majority class samples
- Synthetic Minority Oversampling Technique (SMOTE)

