DDA3020 Machine Learning: Lecture 12 Performance Evaluation

Baoyuan Wu
School of Data Science, CUHK-SZ

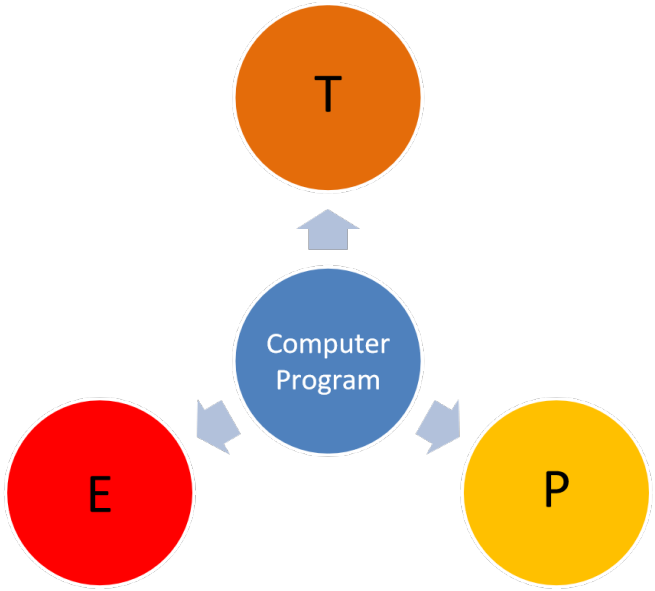
April 3/8, 2024

Outline

Machine Learning

Recall the definition of machine learning we introduced at the first lecture:

- Tom Mitchell: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."
- Experience E corresponds to the training data
- Task T corresponds to classification or regression in supervised learning.
- How about performance measure P?



Work Flow of Machine Learning

Building Blocks of a Learning Algorithm

You may have noticed that those learning algorithms we saw thus far consisted of three parts:

1. a loss function (*e.g.*, mean squared error (MSE), $\frac{1}{n}\sum_i^n(f(\mathbf{x}_i;\mathbf{w},b)-y_i)^2$);
2. an objective function/optimization criterion based on the loss function (*e.g.*, the minimization of the above MSE loss)
3. an optimization routine leveraging training data to find a solution to the optimization criterion (*e.g.*, gradient descent algorithm)

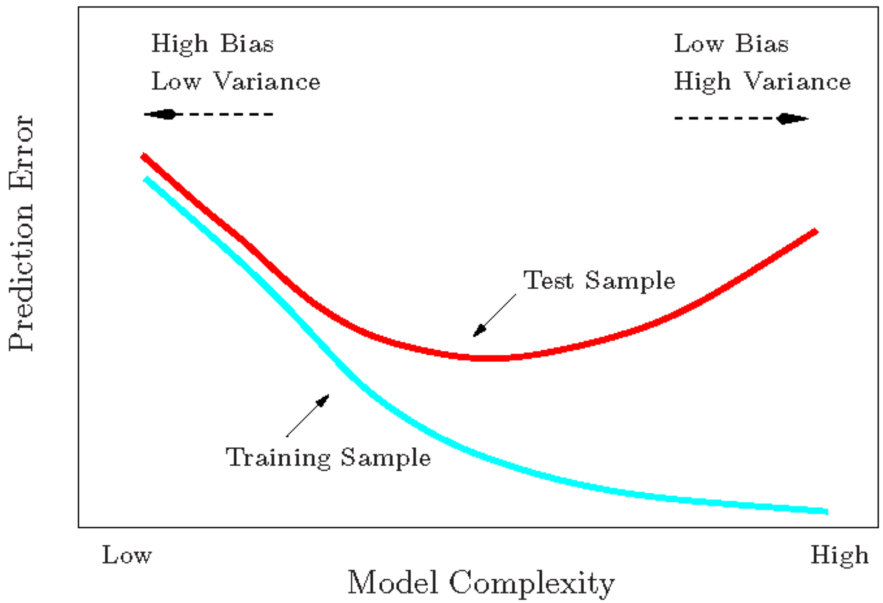
However, this is just one part in the work flow of machine learning!

Work Flow of Machine Learning

- After learning, we certainly want to know about how the algorithm/the learned model performs.
- Essentially, we want to know about the accuracy of the algorithm in terms of predicting novel data based on the limited data we have.
- To answer this question, we need a way to evaluate the algorithm using the limited data we have.
- This forms the motivation for this lecture, where we shall start with the evaluation protocol in the work flow.

1. Motivation: performance evaluation of a learning algorithm
2. Cross-validation
   - Hyper-parameter tuning
   - Cross-validation
3. Evaluation metrics for regression
4. Evaluation metrics for classification
   - Confusion matrix
   - ROC curve and AUC
5. Computational and maintainability (optional)

Recall: Bias-variance Trade-off



In last lecture, we have introduced the bias-variance trade-off, which is related to overfitting/underfitting:

- When the model complexity is **low**: the test prediction error is high, with high bias and low variance, corresponding to underfitting;
- When the model complexity is **high**: the test prediction error is high, with low bias and high variance, corresponding to overfitting.

How to find a model with suitable complexity? This is called model selection problem.

Hyper-parameter tuning

The model selection problem is just a special one of the more general problem in machine learning, hyper-parameter tuning.
The difference between hyper-parameters and parameters is:

- hyper-parameters: they are determined typically outside the learning algorithms (*e.g.*, choosing which model, logistic regression or SVM);
- parameters: they are learned by a learning algorithm based on the training set (*e.g.*, $\mathbf{w}, b$ in linear models $y = \mathbf{w}^\top \mathbf{x} + b$).

There are lots of hyper-parameters in the workflow of a machine learning task, such as

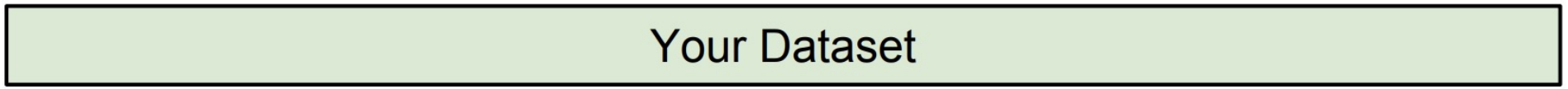
- In linear models with polynomial hypothesis function, the polynomial order is hyper-parameter;
- In decision tree, the maximum tree depth is hyper-parameter; in random forests, the number of decision trees is hyper-parameter;
- In SVM with slack variable's, the trade-off parameter C in $\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i^m \xi_i$ is hyper-parameter;
- In gradient descent algorithm, the learning rate is hyper-parameter.

How to tune these hyper-parameters?

Cross-validation

How to tune these hyper-parameters?

Idea 1: Choose hyper-parameters that work best on the data, *i.e.*, leading to the lowest error. Is it a good idea?



The system will perform perfectly on the whole data, but it may perform poor on testing data. Hence, it is not a good idea.

Cross-validation

How to tune these hyper-parameters?

Idea 2: Split data into train and test, choose hyper-parameters that work best on test data Is it a good idea?

There is no idea how the algorithm will perform on unseen data. Hence, it is not a good idea either.

Cross-validation

How to tune these hyper-parameters?

Idea 3: Split data into train, val, and test; choose hyper-parameters on val and evaluate on test. Is it a good idea?

It is better, because the test set is not involved in the training process. However, the performance will be influenced by the random split of train and validation set.

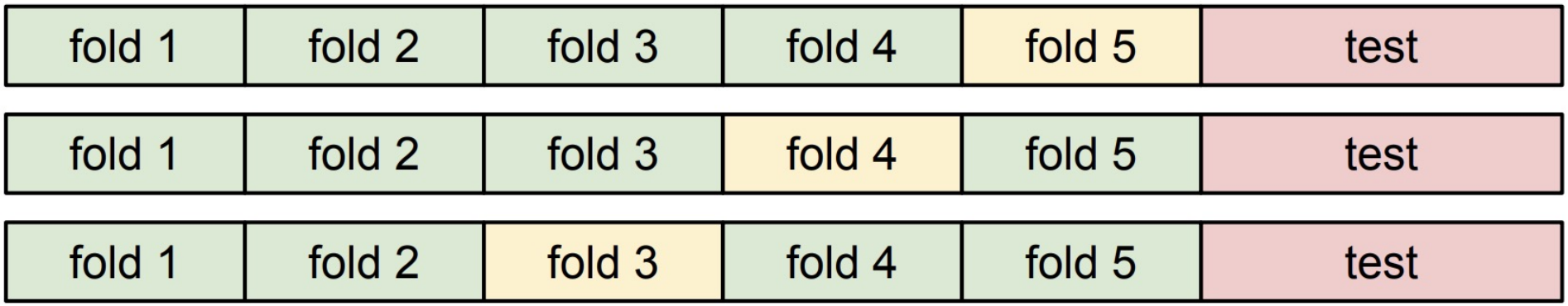
K-fold Cross-validation

How to tune these hyper-parameters?

Idea 4:

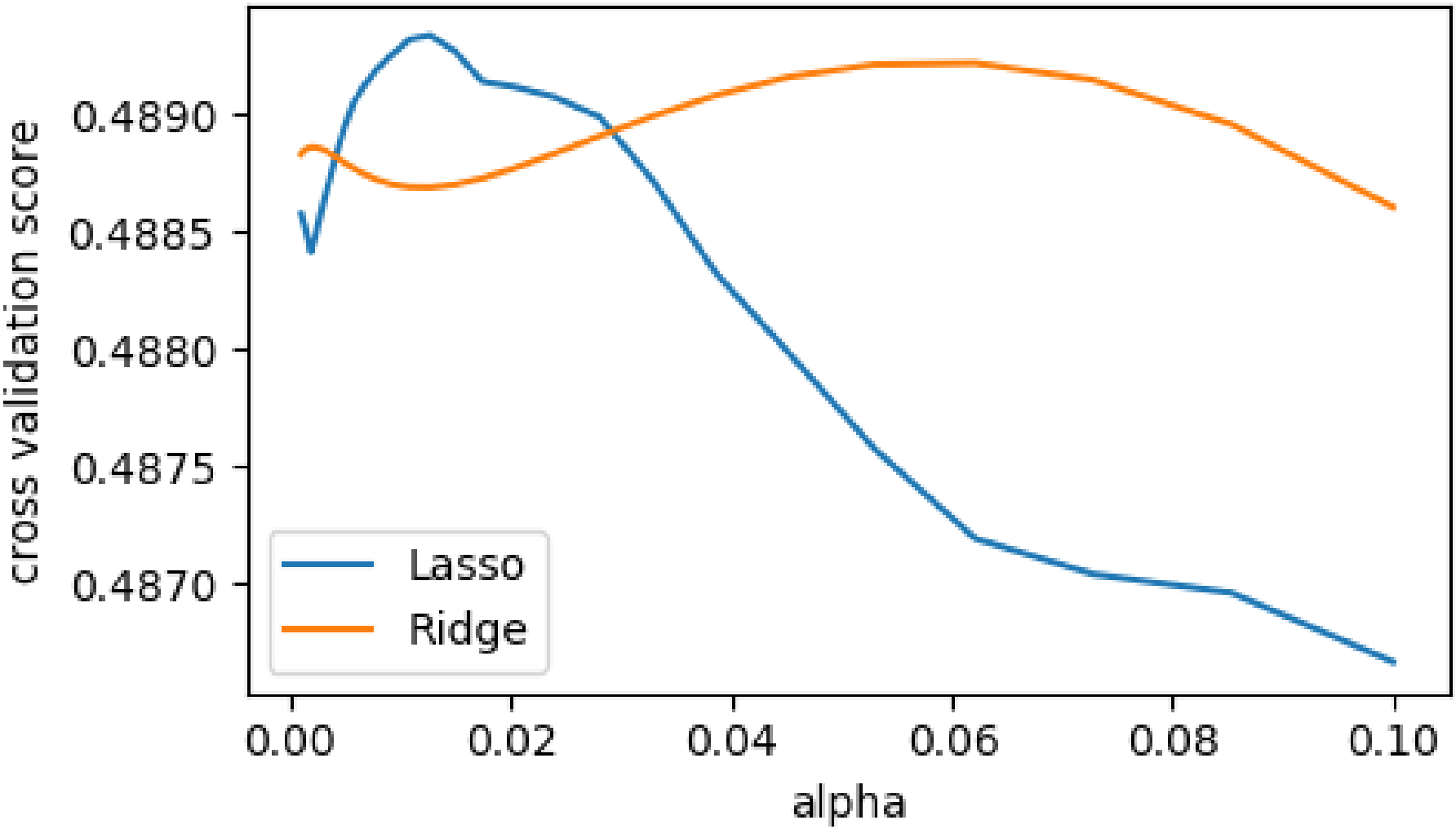
- split the train data into K folds
- try each fold as validation, while other folds as train
- train the model on the train folds, and evaluate the performance on the validation fold
- calculate the average results on all validation folds across all trials, pick the hyper-parameters with the best average result.

This is called K-fold cross validation. It is a widely used method to tune the hyper-parameters.



K-fold Cross-validation

Using K-fold cross validation to determine the trade-off hyper-parameter α in Lasso/Ridge regression models.



K-fold Cross-validation

Remaining issues of K-fold cross validation:

- There is also a hyper-parameter K.
  - If K is too large, then the validation set is too small, and the training sets across different trials are too close to each other, causing overfitting
  - If K is too small, then the training data may be insufficient, causing underfitting
  - In practice, K is set to 5 to 10.
- The computational cost is large, which restricts its application to large-scale data. This is why it is rarely used in deep learning, which requires very large-scale data.

K-fold Cross-validation

- Cross-validation is a statistical method of evaluating generalization performance that is more stable and thorough than using a split into a training and a test set.
- Note that hyper-parameter tuning is just one usage of cross validations.
- It can be also used to estimate the generalization performance of the learned model based on the provided data.
- More details can be found in Chapter 5 of the Book 2 "Introduction to Machine Learning with Python".

1. Motivation: performance evaluation of a learning algorithm

2. Cross-validation
   - Hyper-parameter tuning
   - Cross-validation

3. Evaluation metrics for regression

4. Evaluation metrics for classification
   - Confusion matrix
   - ROC curve and AUC

5. Computational and maintainability (optional)

Evaluation metrics for regression

After the evaluation protocol setup, we need an evaluation metric to provide a measure of the accuracy.
For regression, the main metrics for evaluation are the MSE and the MAE.

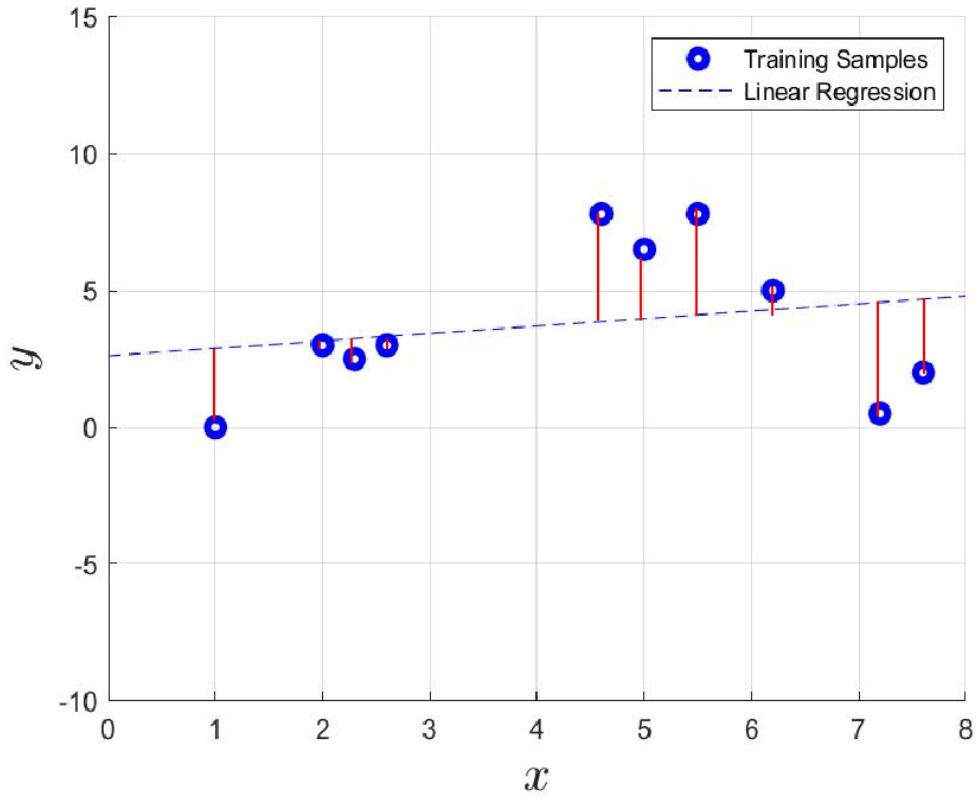
Mean Square Error

$$\text{MSE} = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}.$$

Mean Absolute Error

$$\text{MAE} = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n},$$

where y_i denotes the target output and $\hat{y}_i$ denotes the predicted output for sample i.



1. Motivation: performance evaluation of a learning algorithm

2. Cross-validation
   - Hyper-parameter tuning
   - Cross-validation

3. Evaluation metrics for regression

4. Evaluation metrics for classification
   - Confusion matrix
   - ROC curve and AUC

5. Computational and maintainability (optional)

1. Motivation: performance evaluation of a learning algorithm
2. Cross-validation
   - Hyper-parameter tuning
   - Cross-validation
3. Evaluation metrics for regression
4. Evaluation metrics for classification
   - Confusion matrix
   - ROC curve and AUC
5. Computational and maintainability (optional)

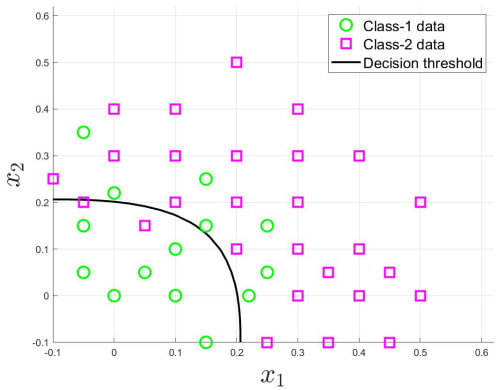
Evaluation metrics for classification

For classification, we define the confusion matrix here before introducing the accuracy.

In this example, we classify some data to predict whether they belong to class-1 or class-2.

- From this figure, we see 7 out of 14 green samples are classified correctly. Also, 25 out of 27 pink samples are classified correctly. The remaining are incorrectly classified.
- There are two types of erroneous classification here: class-1 classified as class-2, and vice versa.
- If we call class-1 as positive-class and class-2 as negative-class, then these two types of errors are called false-positive and false-negative (also known as type-I and type-II errors, respectively).

	Class-1 (predicted)	Class-2 (predicted)
Class-1 (actual)	7 (TP)	7 (FN)
Class-2 (actual)	2 (FP)	25 (TN)



Confusion matrix for binary classification

Let's call class-1 data as positive-class, and class-2 as negative class.
A confusion matrix consists of four entries:

- True Positive (TP): a positive sample is classified (predicted) as positive sample.
- True Negative (TN): a negative sample is classified (predicted) as negative sample.
- False Negative (FN): a positive sample is classified (predicted) as negative sample.
- False Positive (FP): a negative sample is classified (predicted) as positive sample.

	$\widehat{P}$ (predicted)	$\widehat{N}$ (predicted)	
P (actual)	TP	FN	Recall TP/(TP+FN)
N (actual)	FP	TN	
	Precision TP/(TP+FP)	Accuracy (TP+TN)/(TP+TN+FP+FN)	

Evaluation metrics for binary classification

- Accuracy: the total number of correctly classified samples over **ALL** samples under evaluation:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- In imbalanced classification or document retrieval, it is also common to use performance metrics such as precision and recall.

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}$$

	$\widehat{\text{P}}$ (predicted)	$\widehat{\text{N}}$ (predicted)	
P (actual)	TP	FN	Recall TP/(TP+FN)
N (actual)	FP	TN	
	Precision TP/(TP+FP)		Accuracy (TP+TN)/(TP+TN+FP+FN)

Evaluation metrics for binary classification

- In practice, different classes may have different importance. For example, in the medical test task, the cost of predicting a malignant tumor as benign will be much higher than predicting a benign one as malignant. In such situation, the accuracy is no longer a good metric.
- For dealing with such situation, a useful metric is cost-sensitive accuracy:
  - To compute a cost-sensitive accuracy, you first assign a cost (a positive number) to both types of mistakes: FP and FN, or a cost matrix.
  - Then, compute the cost-sensitive accuracy as follows:

$$\frac{C_{p,p} \cdot TP + C_{n,n} \cdot TN}{C_{p,p} \cdot TP + C_{n,n} \cdot TN + C_{p,n} \cdot FN + C_{n,p} \cdot FP}$$

	$\hat{P}$ (predicted)	$\hat{N}$ (predicted)
P (actual)	$C_{p,p}$TP	$C_{p,n}$FN
N (actual)	$C_{n,p}$FP	$C_{n,n}$TN

Evaluation metrics for binary classification

Before we move on, let's normalize those TP, TN, FP and FN values to within [0,1], as follows:

- (True Positive Rate) TPR = TP/(TP+FN)
- (False Negative Rate) FNR = FN/(TP+FN)
- (True Negative Rate) TNR = TN/(FP+TN)
- (False Positive Rate) FPR = FP/(FP+TN)

They satisfy the following conditions:

- TPR + FNR = 1 (100% of positive-class data)
- TNR + FPR = 1 (100% of negative-class data)

If positive and negative classes are balanced, then the accuracy could be computed by

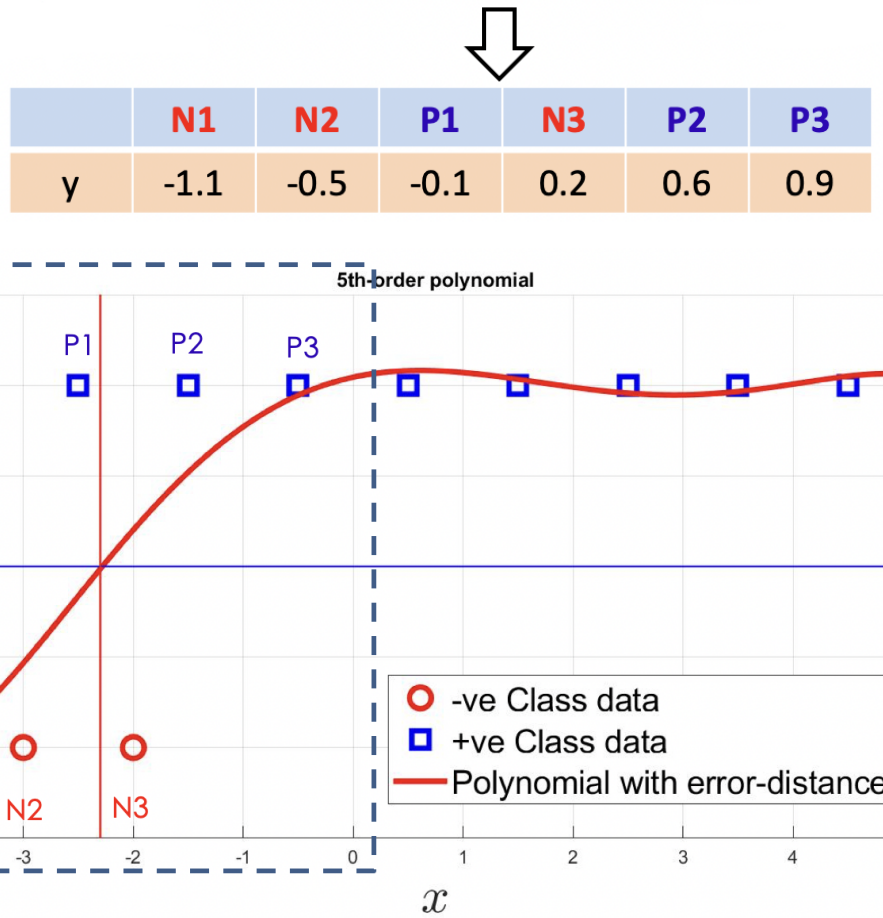
$$\text{Accuracy} = \frac{TPR + TNR}{2} = 1 - \frac{FPR + FNR}{2}$$

Please refer to the following wiki page for more details about confusion matrix and evaluation metrics:

`https://en.wikipedia.org/wiki/Confusion_matrix`

Evaluation metrics for binary classification

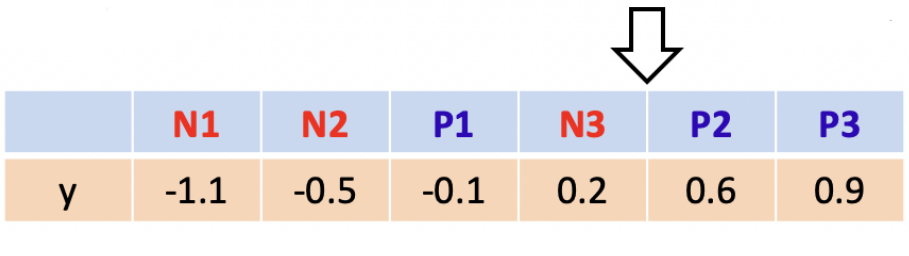
- Suppose you have trained a classifier on some data points as shown blue and red points.
- This plot shows the prediction error (2 error count) if the threshold is set at default value of 0.



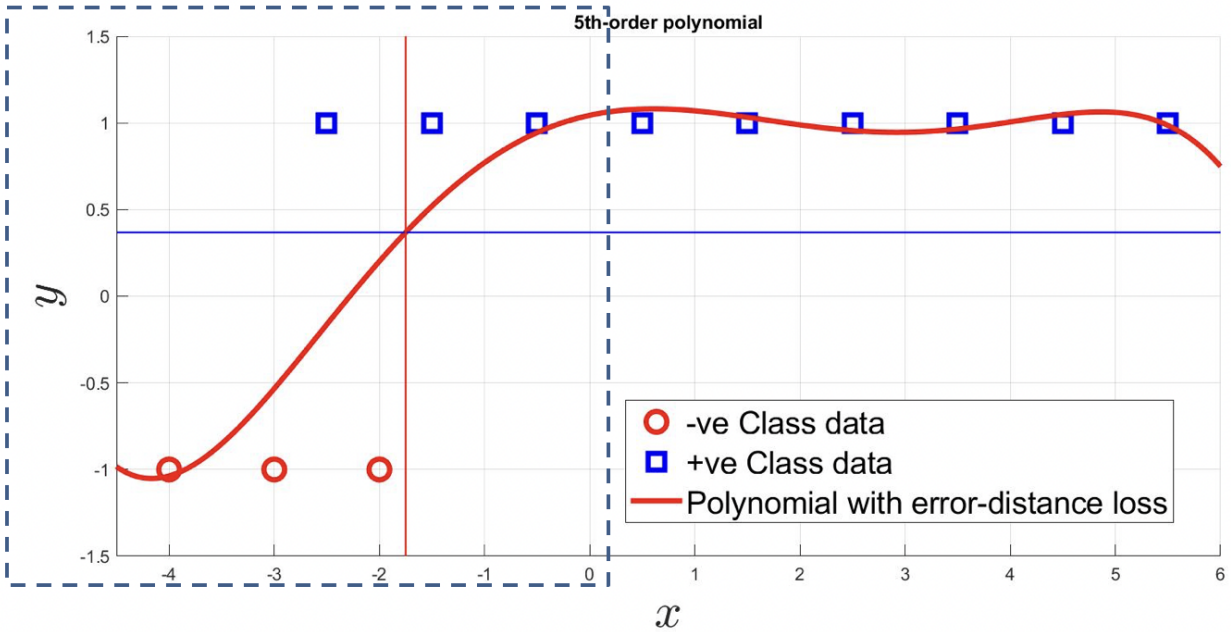
	N1	N2	P1	N3	P2	P3
y	-1.1	-0.5	-0.1	0.2	0.6	0.9

Evaluation metrics for binary classification

Suppose the threshold is moved upward, then the error count is different.



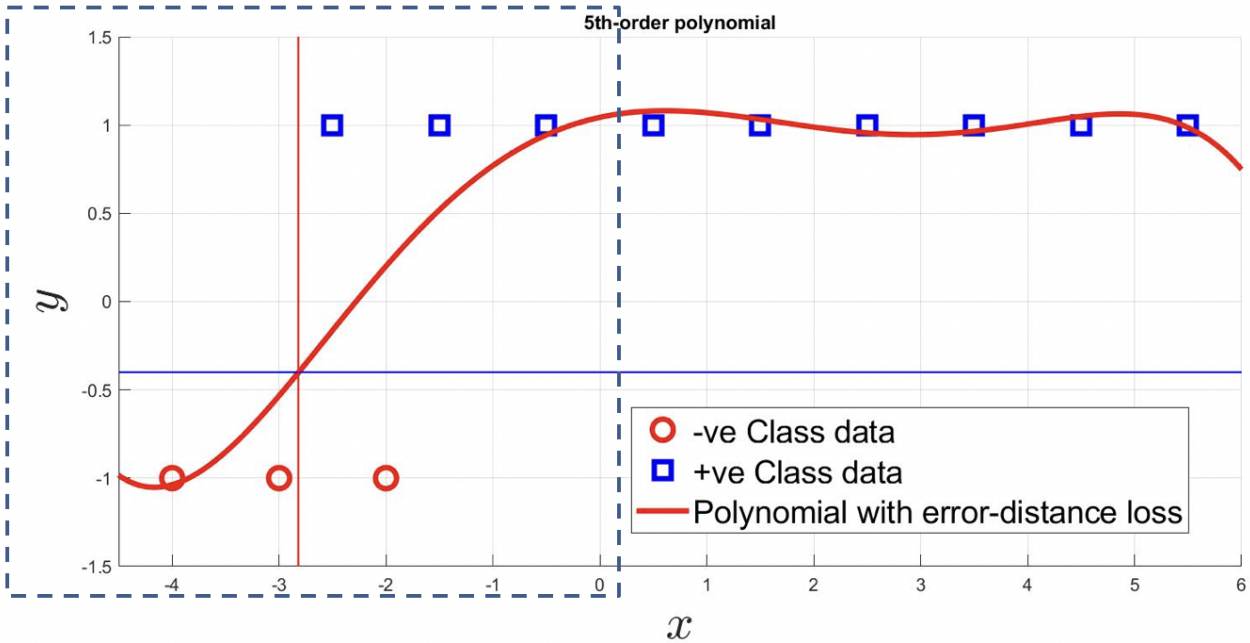
	N1	N2	P1	N3	P2	P3
y	-1.1	-0.5	-0.1	0.2	0.6	0.9



Evaluation metrics for binary classification

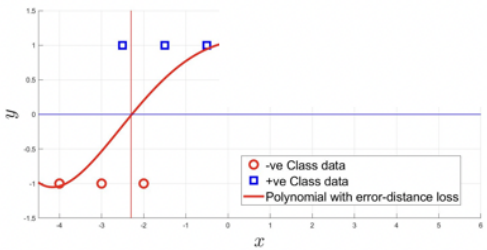
Suppose the threshold is moved downward, then the error count is different.

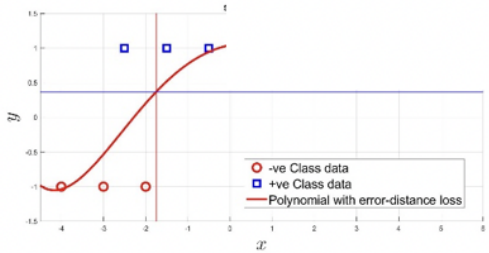
	N1	N2	P1	N3	P2	P3
y	-1.1	-0.5	-0.1	0.2	0.6	0.9

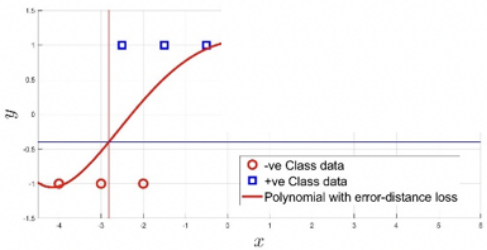


Evaluation metrics for binary classification

Let's check the confusion matrix when setting different thresholds





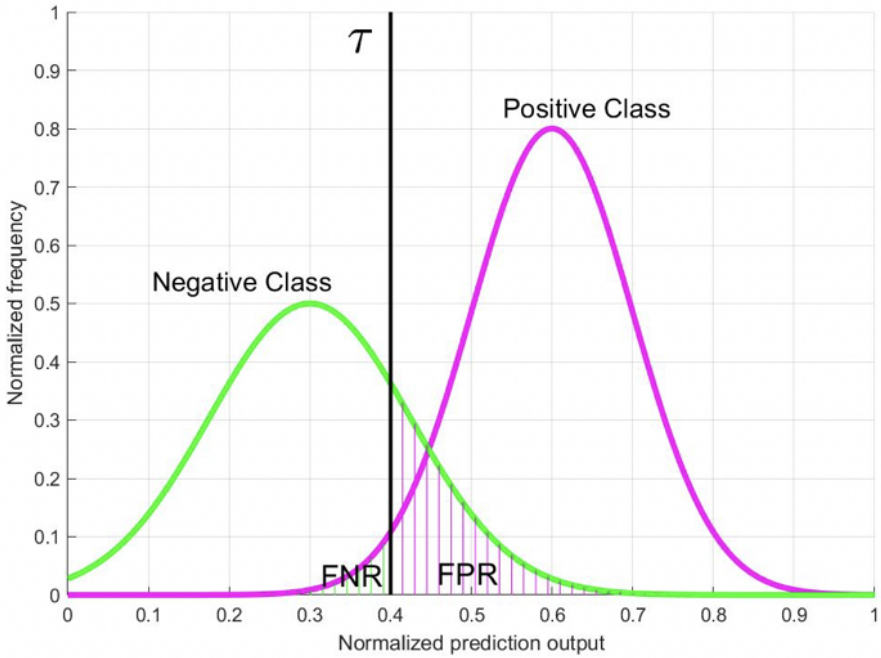


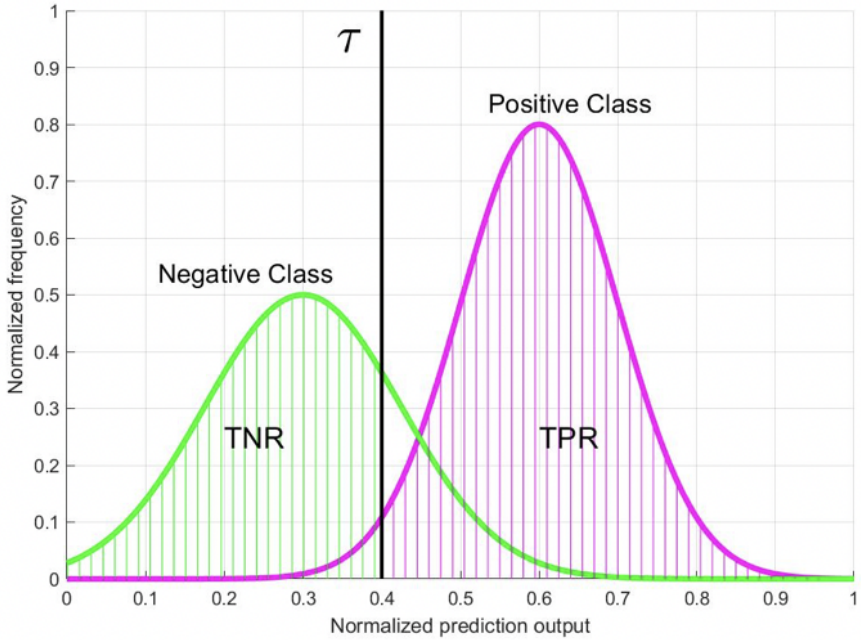
	$\widehat{\mathbf{P}}$ (predicted)	$\widehat{\mathbf{N}}$ (predicted)
P (actual)	$TP = 2$	$FN = 1$
N (actual)	$FP = 1$	$TN = 2$

	$\widehat{\mathbf{P}}$ (predicted)	$\widehat{\mathbf{N}}$ (predicted)
P (actual)	$TP = 2$	$FN = 1$
N (actual)	$FP = 0$	$TN = 3$

	$\widehat{\mathbf{P}}$ (predicted)	$\widehat{\mathbf{N}}$ (predicted)
P (actual)	$TP = 3$	$FN = 0$
N (actual)	$FP = 1$	$TN = 2$

Evaluation metrics for binary classification

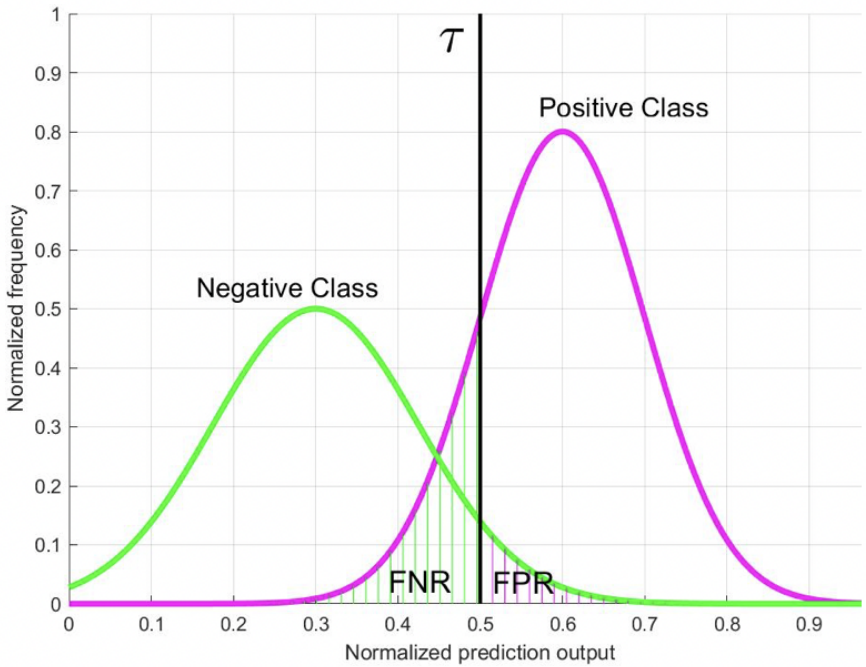


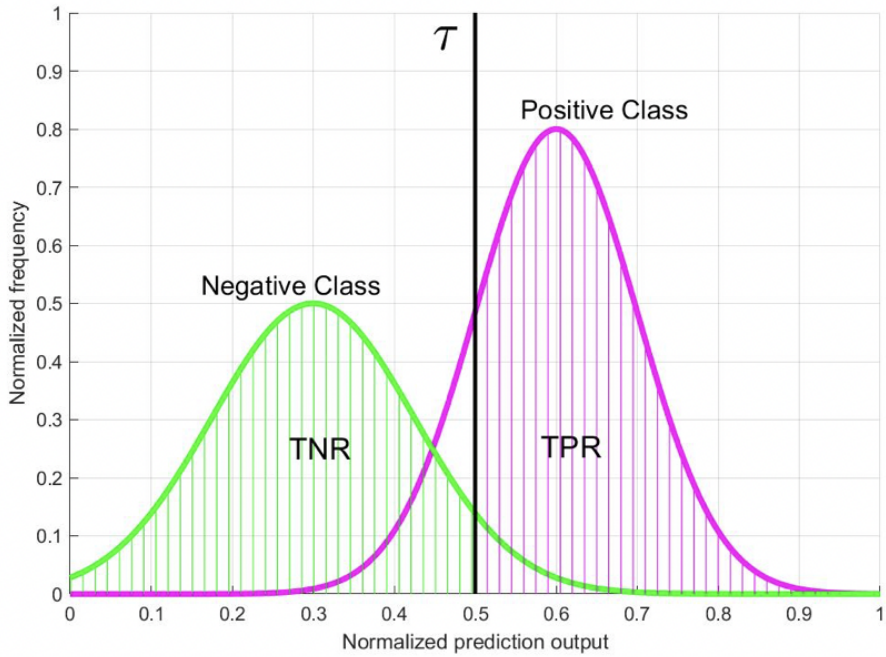


The predicted output y values can also be normalized and plotted as distribution curves for the two classes of data.
Here we see different decision threshold (τ) gives a different (FNR+FPR) or (TPR+TNR), which are indicators of classification accuracy.

Evaluation metrics for binary classification

The decision threshold can be varied, and we get another set of FPR and FNR. Each setting of decision threshold is called an operating point (which has different FNR and FPR values).

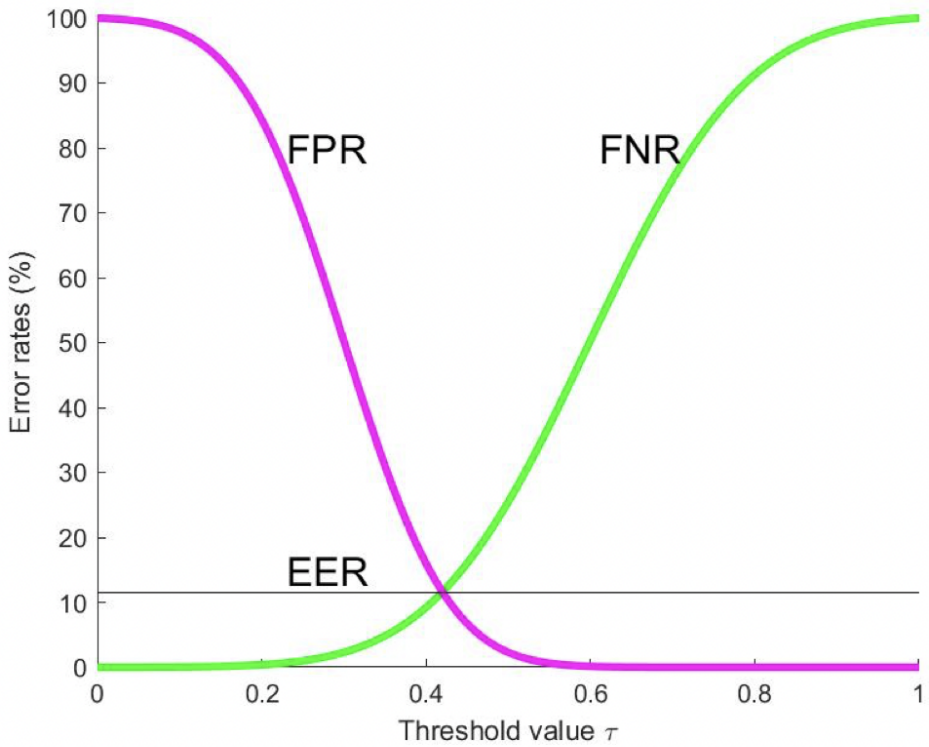




Evaluation metrics for binary classification

If we plot for all the operating points by changing the threshold (i.e., τ) from 0 to 1, we observe:

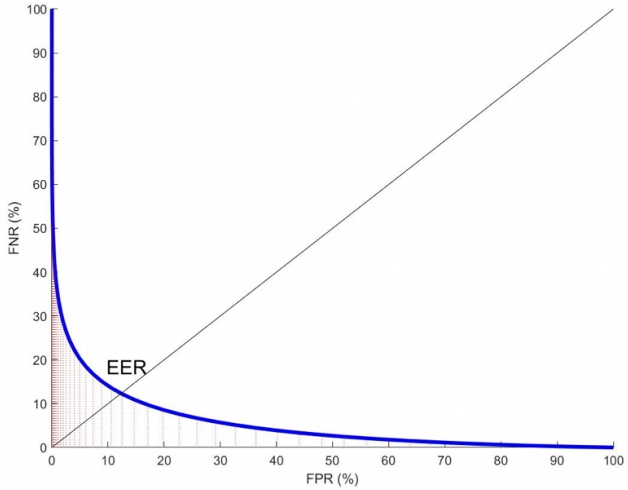
- FPR shows a decreasing trend
- FNR shows an increasing trend
- The intersection point of these two curves is when the two error rates are equal (i.e., FPR=FNR). This point is call the Equal Error Rate (EER).

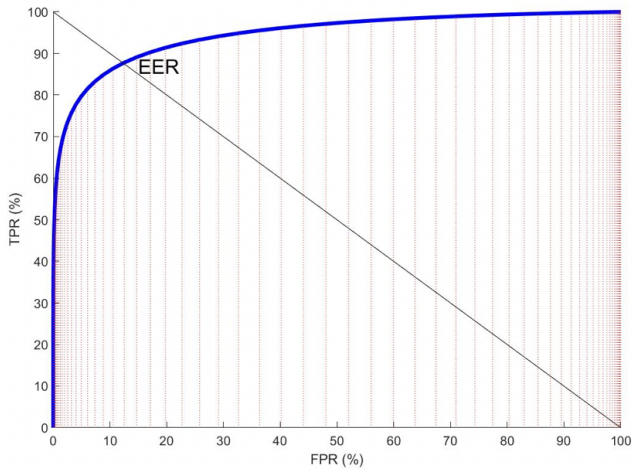


1. Motivation: performance evaluation of a learning algorithm
2. Cross-validation
   - Hyper-parameter tuning
   - Cross-validation
3. Evaluation metrics for regression
4. Evaluation metrics for classification
   - Confusion matrix
   - ROC curve and AUC
5. Computational and maintainability (optional)

Evaluation metrics for binary classification

- Left figure: When the FNR-curve is plotted as y-axis and the FPR-curve is plotted as x-axis, the plot is called the DET curve (Detection Error Trade-off curve). The lower the DET (bending to the bottom-left corner), the better is the classification accuracy.
- Right figure: When the TPR-curve (=1-FNR) is plotted as y-axis and the FPR-curve is plotted as x-axis, the plot is called the ROC curve (Receiver Operating Characteristic Curve). The higher the ROC (bending to the top-left corner), the better is the classification accuracy.
- The Area Under the ROC (AUC) can also be used to gauge the classifier's accuracy (in overall sense, *i.e.*, under all operating points).

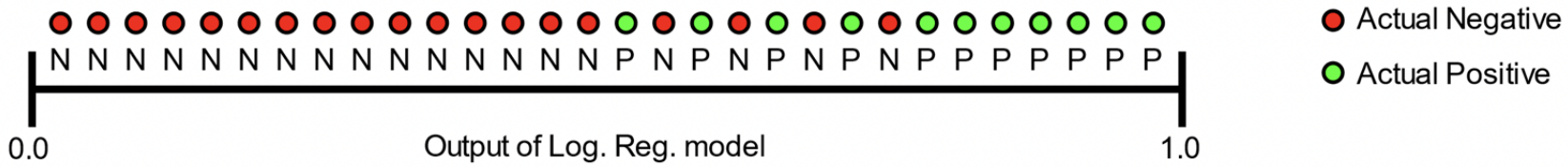




Evaluation metrics for binary classification

Area Under the Curve (ROC curve)

- AUC provides an aggregate measure of performance across all possible classification thresholds. It can be seen as the probability that the model ranks a random positive example more highly than a random negative example.



- AUC ranges in value from 0 (100% wrong prediction) to 1 (100% correct prediction). It has the following properties:
  - Scale-invariant: changing the range of the output will not change the AUC score;
  - Classification-threshold-invariant: Given any threshold, AUC score will not be changed.

Evaluation metrics for binary classification

Area Under the ROC Curve (AUC)

- Because the DET or ROC is an entire curve of operating points, and not a single value to quantify the accuracy, we need an interpretation which carries a single numerical value.
- Consider a set of binary samples indexed by $i = 1, ..., m^+$ for positive class $j = 1, ..., m^-$ and for negative class.
- Let $g(\mathbf{x})$ be a predictor, and $e_{ij} = g\left(\mathbf{x}_i^+\right) - g\left(\mathbf{x}_j^-\right)$ is the difference in the value of the predictor between a positive-class sample $\mathbf{x}_i^+$ and a negative-class sample $\mathbf{x}_j^-$.

Evaluation metrics for binary classification

Area Under the ROC Curve (AUC)

- Consider also a Heaviside step function given by

$$u(e) = \begin{cases} 1, & \text{if } e > 0 \\ 0.5 & \text{if } e = 0 \\ 0 & \text{if } e < 0 \end{cases}$$

- The Area Under the ROC Curve (AUC) can be expressed as

$$\text{AUC} = \frac{1}{m^+ m^-} \sum_{i=1}^{m^+} \sum_{j=1}^{m-} u\left(e_{ij}\right)$$

  - This AUC quantifies the OVERALL accuracy performance of a classifier over the entire operating range of thresholds.
  - When AUC=1, it is a perfect classification.
  - When AUC=0.5, the classifier is no better than a random guess.

Evaluation metrics for binary classification

Example of computing AUC on a limited data set

- Given a set of 4 points, including 2 positive and 2 negative points, denoted as $\{(\mathbf{x}_1^-, -), (\mathbf{x}_2^-, -), (\mathbf{x}_1^+, +), (\mathbf{x}_2^+, +)\}$
- The prediction score g of these 4 points are $0.1, 0.4, 0.35, 0.8$, respectively.
- As $m^+ = m^- = 2$, we need to compute 4 e_{ij}

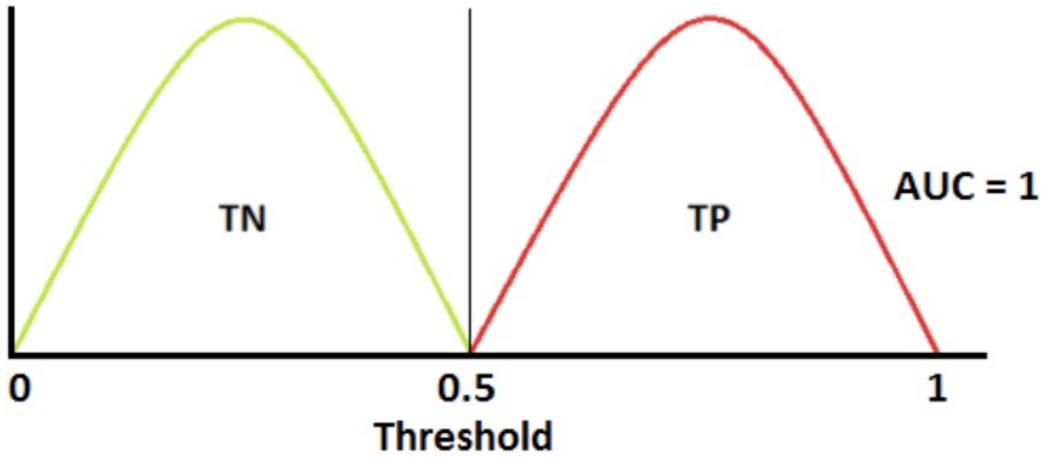
$$\begin{aligned} e_{11} &= g(\mathbf{x}_1^+) - g(\mathbf{x}_1^-) = 0.25, \\ e_{12} &= g(\mathbf{x}_1^+) - g(\mathbf{x}_2^-) = -0.05, \\ e_{21} &= g(\mathbf{x}_2^+) - g(\mathbf{x}_1^-) = 0.7, \\ e_{22} &= g(\mathbf{x}_2^+) - g(\mathbf{x}_2^-) = 0.4. \end{aligned}$$

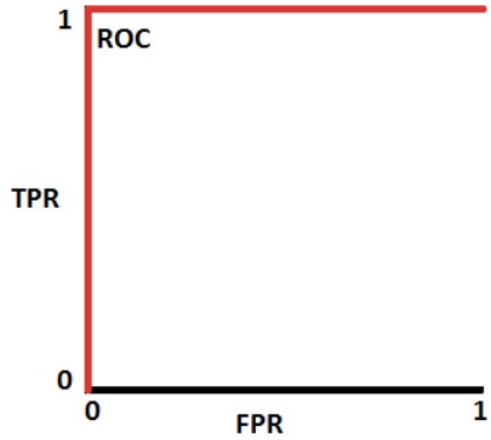
- Then, we have

$$AUC = \frac{1}{m^+ m^-} \sum_{i=1}^{m^+} \sum_{j=1}^{m-} u(e_{ij}) = \frac{1+0+1+1}{4} = 0.75.$$

Evaluation metrics for binary classification

Area Under the ROC Curve (AUC)



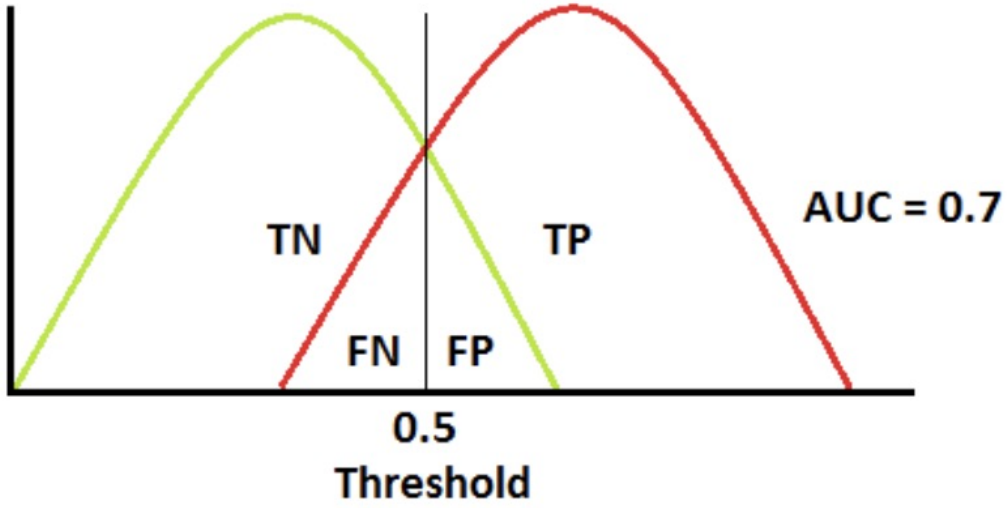


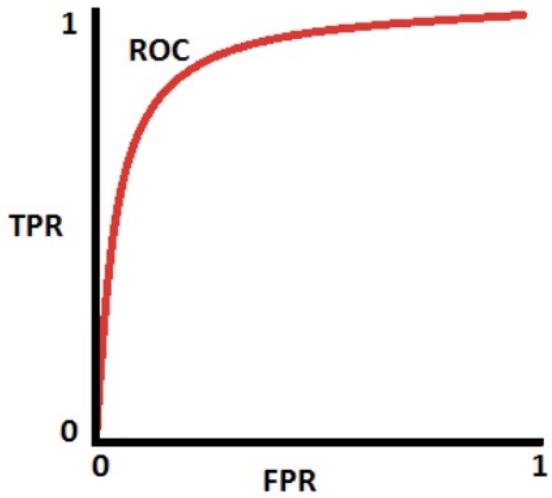
In the above case:

- All positive outputs are larger than all negative outputs
- Given the threshold 0.5, we have $\text{TPR} = 1$, $\text{TNR} = 1$, $\text{FPR} = 0$, $\text{FNR} = 0$
- When the threshold is lower than 0.5, we have $\text{TPR} = 1$, $\text{TNR} < 1$, $0 < \text{FPR} < 1$, $\text{FNR} = 0$
- When the threshold is larger than 0.5, we have $0 < \text{TPR} < 1$, $\text{TNR} = 1$, $\text{FPR} = 0$, $\text{FNR} > 0$
- Thus, the ROC curve is plotted as the top-right. It is easy to compute that $\text{AUC} = 1$.

Evaluation metrics for binary classification

Area Under the ROC Curve (AUC)



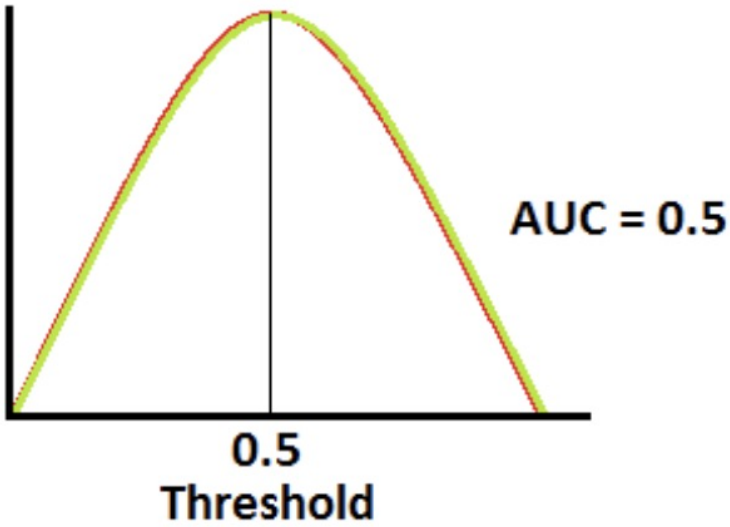


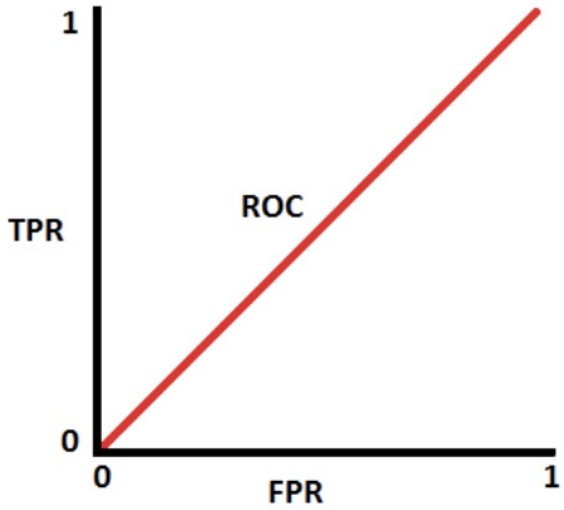
In the above case:

- We observe that given any threshold, we always have TPR $>$ FPR, except for threshold $= 0$, where TPR $=$ FPR $= 1$
- Thus, the ROC curve is plotted as the top-right. It is easy to know AUC > 0.5.

Evaluation metrics for binary classification

Area Under the ROC Curve (AUC)



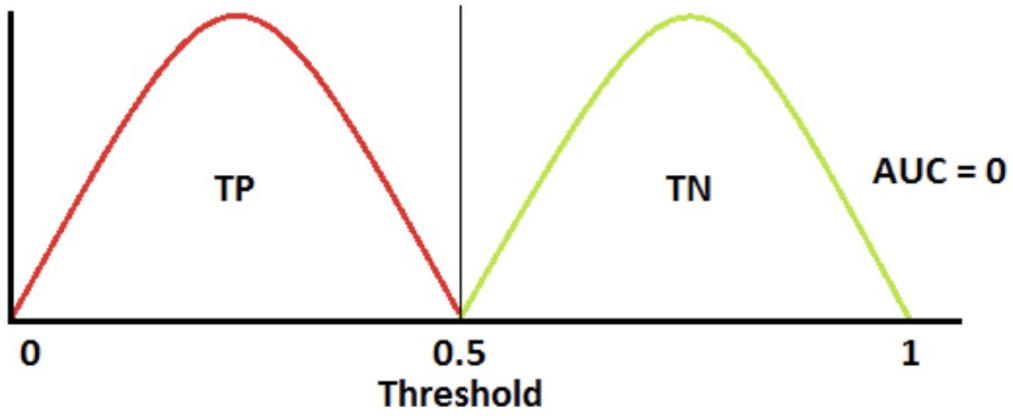


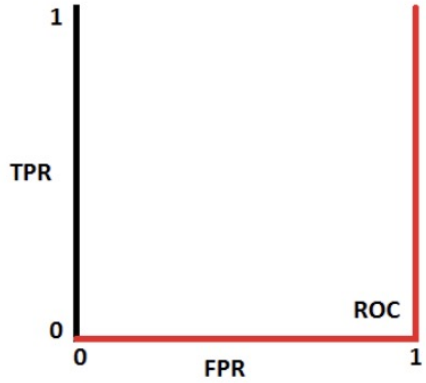
In the above case:

- Given any threshold, FPR is always equivalent to TPR
- Thus, the ROC curve is a line $y = x$, and AUC $= 0.5$.

Evaluation metrics for binary classification

Area Under the ROC Curve (AUC)





In the above case:

- All positive outputs are smaller than all negative outputs
- Given the threshold 0.5, we have $\text{TPR} = 0$, $\text{TNR} = 0$, $\text{FPR} = 1$, $\text{FNR} = 1$
- When the threshold is lower than 0.5, we have $\text{TPR} > 0$, $\text{TNR} = 0$, $\text{FPR} = 1$, $\text{FNR} < 1$
- When the threshold is larger than 0.5, we have $\text{TPR} = 0$, $\text{TNR} > 0$, $0 < \text{FPR} < 1$, $\text{FNR} > 0$
- Thus, the ROC curve is plotted as the top-right. It is easy to compute that $\text{AUC} = 0$.

Evaluation metrics for binary classification

Further readings for AUC:
Blog: https://blog.csdn.net/u013385925/article/details/80385873
Demo with code: https://blog.csdn.net/qq_22238533/article/details/78666436
Code: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html

Evaluation metrics for binary classification

Summary of evaluation metrics for binary classification:

- Confusion matrix: TP, TN, FN, FP
- TPR, TNR, FNR, FPR; EER
- DET curve, ROC curve
- Area under the ROC curve (AUC)

Evaluation metrics for multicategory classification

Confusion Matrix for Multicategory Classification

	$P_{\hat{1}}$ (predicted)	$P_{\hat{2}}$ (predicted)		$P_{\hat{C}}$ (predicted)
P_1 (actual)	$P_{1,\hat{1}}$	$P_{1,\hat{2}}$	...	$P_{1,\hat{C}}$
P_2 (actual)	$P_{2,\hat{1}}$	$P_{2,\hat{2}}$	...	$P_{2,\hat{C}}$
⋮	⋮	⋮	⋱	⋮
P_C (actual)	$P_{C,\hat{1}}$	$P_{C,\hat{2}}$		$P_{C,\hat{C}}$

Finally, for multicategory classification, the confusion matrix can be calculated as shown in the table.
Unfortunately, the ROC for multiclass is very complicated and has no consensus definition in the research community.

1. Motivation: performance evaluation of a learning algorithm

2. Cross-validation
   - Hyper-parameter tuning
   - Cross-validation

3. Evaluation metrics for regression

4. Evaluation metrics for classification
   - Confusion matrix
   - ROC curve and AUC

5. Computational and maintainability (optional)

Computational and Maintainability (optional)

Other Important Performances

Computational speed and efficiency

A Cautious Note

- A high computational performance (and efficiency) does not coincide very well with software quality attributes such as flexibility, extensibility, usability, modularity or maintainability.
- Where low-level programming is required to exploit parallel hardware to the best extent, common programming paradigms such as object-oriented programming cannot be applied any more.
- To achieve as much software quality as possible, the trade-off between computational efficiency and maintainability and portability has to be chosen carefully.

Reference:
https://dl.acm.org/doi/10.5555/3019106.3019109