Lecture 3 - Module 1.2: Model evaluation (part 1)

COMP 551 Applied machine learning

Yue Li
Assistant Professor
School of Computer Science
McGill University

January 14 & 16, 2025

Outline

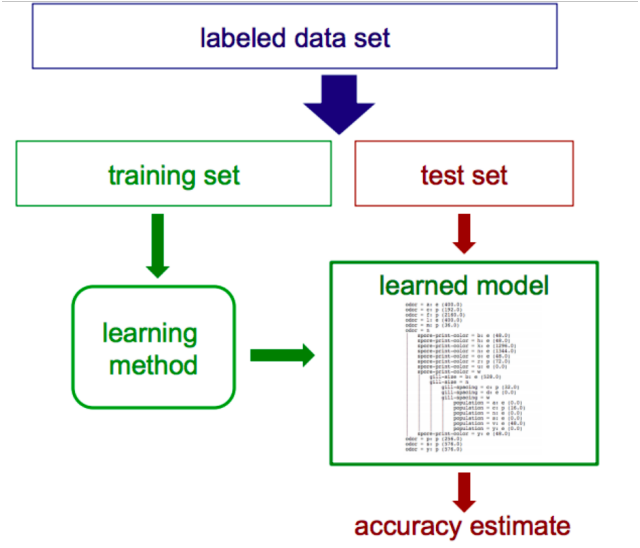
Learning objectives

Understanding the following concepts

- Evaluating generalization performance
- Confusion table
- Receiver operating characteristics (ROC)

Outline

A typical workflow to evaluate a classification model



Generalization error (or generalization accuracy)

- What we really care about is the performance of the model on *new data*.
- In other words, we want to see how our model *generalizes* to unseen data.
- An assumption that justifies deployment of our model on unseen data is the fact that our training and unseen data come from the **same** distribution.
- In fact very often we assume that there exists some distribution $p(x, y) = p(y|x)p(x)$ over our features and labels, such that our training data is composed of independent samples from the same distribution – that is $x^{(n)} \sim p_x$ and $y^{(n)} \sim p_{y|x}$ for all $x^{(n)}, y^{(n)} \in \mathcal{D}$ (i.e., the data are *i.i.d.*).
- We assume that unseen data are also samples from *the same* distribution.

Generalization error is the **expected error** of our model $f : x \mapsto y$ under this distribution:

$$Err(f) = \mathbb{E}_{x,y\sim p}[\ell(f(x), y)].$$

Here ℓ is some *loss function* such as classification error $\ell(y, \hat{y}) = \mathbb{I}(y \neq \hat{y})$ or squared loss $\ell(y, \hat{y}) = (y - \hat{y})^2$ that we often use in regression.

Test set

- Unfortunately, we don't have access to the true data distribution, we only have *samples* from the distribution.
- We can estimate the generalization error by setting aside a portion of our dataset that **we do not use in any way in learning or selecting the model**.
- This part of the dataset is called the **test set**. Let's use $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ to this partitioning of our original dataset $\mathcal{D}$.

The **test error** is

$$\widehat{Err}(f) = \mathbb{E}_{x,y\sim\mathcal{D}_{\text{test}}}[\ell(f(x),y)] = \frac{1}{|\mathcal{D}_{\text{test}}|}\sum_{x,y\in\mathcal{D}_{\text{test}}}\ell(f(x),y).$$

where $|\mathcal{D}_{\text{test}}|$ is the cardinality of the test set $\mathcal{D}_{\text{test}}$ (i.e., the number of test samples).

Prostate cancer prediction problem

Suppose you want to learn to predict if a person has a prostate cancer based on two easily-measured variables obtained from blood sample: Complete Blood Count (CBC) and Prostate-specific antigen (PSA). We have collected data from patients known to have or not have prostate cancer:

CBC	PSA	Status
142	67	Normal
132	58	Normal
178	69	Cancer
188	46	Normal
183	68	Cancer
...		

Goal: Train classifier to predict the class of new patients, from their CBC and PSA.

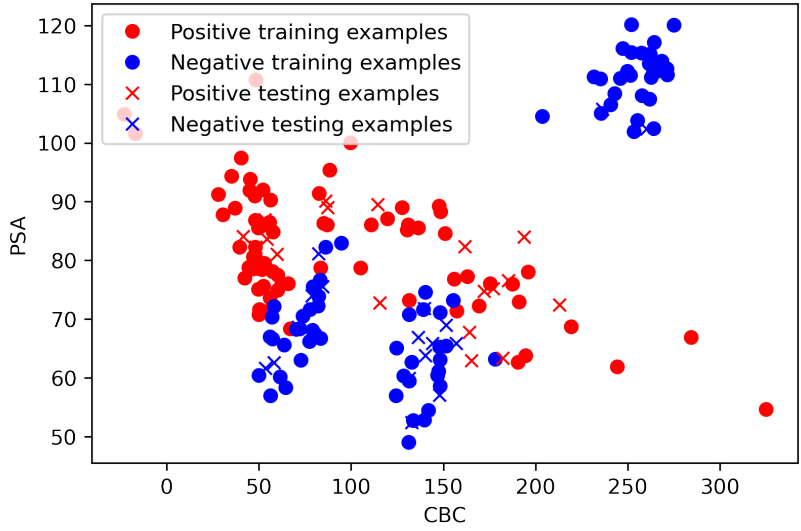
Split the dataset into training and testing datasets

We split the data into 80% training and 20% testing. In Scikit-learn, this is easy to do:

```
1 from sklearn import model_selection
2
3 X_train, X_test, y_train, y_test =
  ↪  model_selection.train_test_split(X, y, test_size = 0.2,
  ↪  random_state=1, shuffle=True)
```

- `random_state` set to a fixed number for reproducibility
- `shuffle` by default is `True` to randomly permute the orders of the rows to avoid splitting examples of the same class into training or testing set. For example, if rows are ordered by classes.

Prostate cancer data



Model prediction

Model training:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier() # n_neighbors=5 (default)
fit = knn.fit(X_train, y_train)
```

Model prediction on the test data:

```
y_test_pred = fit.predict(X_test)
```

- Our prediction is binary 0 (healthy) or 1 (cancer) based on whether the predicted probabilities are greater than 0.5.

Classification Accuracy

We then evaluate the prediction accuracy:

$$\textit{Accuracy} = \frac{\text{Correct predictions}}{\text{Total number of predictions}}$$

```
1 acc_test = np.sum(y_test_pred==y_test)/len(y_test) % 0.975
```

Outline

True/false positives and negatives

True positive (TP)
Positive example that is predicted to be positive

- A person who is predicted to have cancer and in fact has cancer

False positive (FP)
Negative example that is predicted to be positive

- A person who is predicted to have cancer but in fact is healthy

True negative (TN)
Negative example that is predicted to be negative

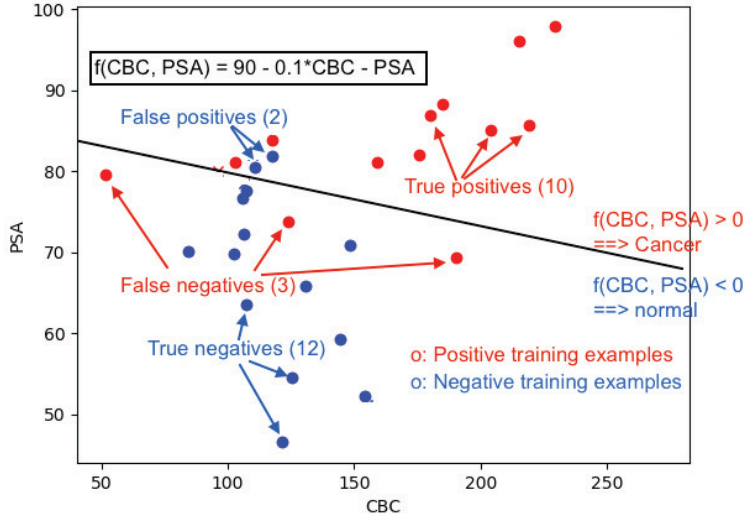
- A person who is predicted to be healthy and in fact is healthy

False negative (FN)
Positive example that is predicted to be negative

- A person who is predicted to be healthy but in fact has cancer

Classification errors

Here: TP = 10, TN = 12, FP = 2, FN = 3.



Confusion matrix

Confusion matrix: A table of counts for TPs, FPs, TNs, and FNs

	Predicted negative	Predicted positive
Actual negative	TN = 12	FP = 2
Actual positive	FN = 3	TP = 10

In scikit-learn, we can get the confusion matrix for the KNN by:

```
from sklearn.metrics import confusion_matrix

X_train, X_test, y_train, y_test = \
train_test_split(X, y, test_size=0.5, random_state=100)

knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)
y_test_pred = knn.predict(X_test)

cm = confusion_matrix(y_test, y_test_pred)
```

True and false positive rates

At a specific threshold, we can calculate TPR and FPR:

True positive rate (TPR) (aka sensitivity)

The proportion of positive examples that are predicted to be positive

- e.g., fraction of cancer patients who are predicted to have cancer.

$$TPR = \frac{TP}{TP + FN}$$

False positive rate (FPR) (aka 1 - specificity)

The proportion of negative examples that are predicted to be positive

- e.g., fraction of non-cancer patients who are predicted to have cancer.

$$FPR = \frac{FP}{FP + TN}$$

Computing TPR and FPR from the confusion table

	Predicted negative	Predicted positive
Actual negative	TN = 12	FP = 2
Actual positive	FN = 3	TP = 10

$$TPR = \frac{TP}{TP + FN} = \frac{10}{10 + 3} = 77\%$$

$$FPR = \frac{FP}{FP + TN} = \frac{2}{2 + 12} = 14\%$$

True/false positive rates (pop quiz)

	predicted negative	predicted positive
actual negative	1	2
actual positive	3	7

$$TPR = \frac{TP}{TP + FN} = \frac{?}{? + ?} = ?$$

$$FPR = \frac{FP}{FP + TN} = \frac{?}{? + ?} = ?$$

Outline

A classification model often produces probabilities instead of hard decision

We can express our *uncertainty* via *the class probabilities*:

$$p(y^{(*)} = c|\mathbf{x}^{(*)}) = \frac{1}{K} \sum_{n \in \mathcal{N}_K(\mathbf{x}^{(*)}, \mathcal{D})} \mathbb{I}(y^{(n)} = c)$$

In binary classification (i.e., cancer or normal), we can reduce the formula to predicting only the positive class:

$$p(y^{(*)} = 1|\mathbf{x}^{(*)}) = \frac{1}{K} \sum_{n \in \mathcal{N}_K(\mathbf{x}^{(*)}, \mathcal{D})} \mathbb{I}(y^{(n)} = 1)$$

Classification threshold: how to decide who are cancer patients?

- By default, `fit.predict(X_test)` uses 0.5 as threshold.

```
if pred_prob > 0.5:
    cancer = 1
else:
    cancer = 0
```

- This results in the following predictions:

patient index	$p(y = 1\|x)$	pred_label	true_label
0	0.1	0	0
1	0.4	0	0
2	0.5	0	1
3	0.8	1	1

- What accuracy do we get with a different threshold say 0.6?
- Can we evaluate the model using *all* thresholds?

True/false positive rates (pop quiz)

What will be the TPR and FPR if **the threshold is 1**?

patient index	$p(y = 1\|x)$	pred_label	true_label
0	0.1		0
1	0.4		0
2	0.5		1
3	0.8		1

	PN	PP
AN		
AP		

True positive rate (TPR):

$$TPR = \frac{TP}{TP + FN} =$$

False positive rate (FPR):

$$FPR = \frac{FP}{FP + TN} =$$

True/false positive rates (pop quiz)

What will be the TPR and FPR if **the threshold is 0.6**?

patient index	$p(y = 1\|x)$	pred_label	true_label
0	0.1		0
1	0.4		0
2	0.5		1
3	0.8		1

	PN	PP
AN		
AP		

True positive rate (TPR):

$$TPR = \frac{TP}{TP + FN} =$$

False positive rate (FPR):

$$FPR = \frac{FP}{FP + TN} =$$

True/false positive rates (pop quiz)

What will be the TPR and FPR if **the threshold is 0.3**?

patient index	$p(y = 1\|x)$	pred_label	true_label
0	0.1		0
1	0.4		0
2	0.5		1
3	0.8		1

	PN	PP
AN		
AP		

True positive rate (TPR):

$$TPR = \frac{TP}{TP + FN} =$$

False positive rate (FPR):

$$FPR = \frac{FP}{FP + TN} =$$

True/false positive rates (pop quiz)

What will be the TPR and FPR if **the threshold is -1**?

patient index	$p(y = 1\|x)$	pred_label	true_label
0	0.1		0
1	0.4		0
2	0.5		1
3	0.8		1

	PN	PP
AN		
AP		

True positive rate (TPR):

$$TPR = \frac{TP}{TP + FN} =$$

False positive rate (FPR):

$$FPR = \frac{FP}{FP + TN} =$$

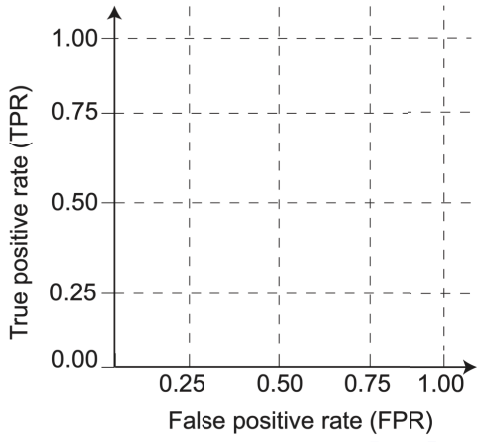
Receiver Operating Characteristic (ROC) curve

- We can create a table for TPR and FPR at each Threshold.
- ROC curve plots TPR (y-axis) versus FPR (x-axis)
- *Area under the curve (AUC)* is a metric common used to evaluate the model.

Threshold	TPR	FPR
1		
0.6		
0.3		
-1		

Consider three extreme cases:

1. What does ROC look like for a dummy model that predicts everything 0.5?
2. What does ROC look like for a perfect model?
3. What does ROC look like for a model opposite to perfect?



Dummy model for thresholds -1, 0.3, 0.6, 1

patient index	$p(y=1\|x)$	pred_label				true_label
0	0.5					0
1	0.5					0
2	0.5					1
3	0.5					1

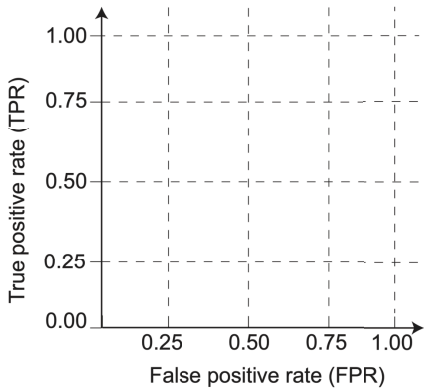
	PN	PP
AN		
AP		

True positive rate (TPR):

$$TPR = \frac{TP}{TP + FN} =$$

False positive rate (FPR):

$$FPR = \frac{FP}{FP + TN} =$$



Perfect model for thresholds -1, 0.3, 0.6, 1

patient index	$p(y=1\|x)$	pred_label				true_label
0	0					0
1	0					0
2	1					1
3	1					1

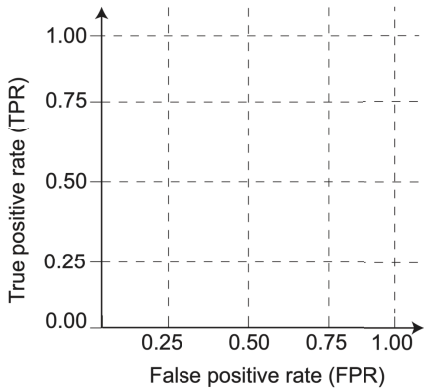
	PN	PP
AN		
AP		

True positive rate (TPR):

$$TPR = \frac{TP}{TP + FN} =$$

False positive rate (FPR):

$$FPR = \frac{FP}{FP + TN} =$$



Opposite to perfect model for thresholds -1, 0.3, 0.6, 1

patient index	$p(y=1\|x)$	pred_label				true_label
0	1					0
1	1					0
2	0					1
3	0					1

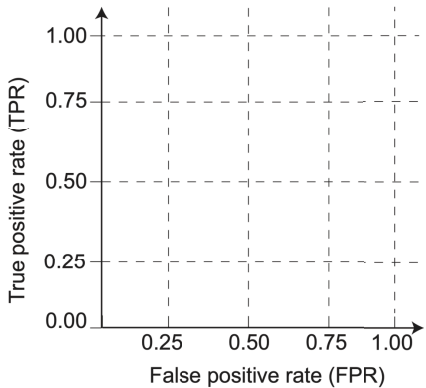
	PN	PP
AN		
AP		

True positive rate (TPR):

$$TPR = \frac{TP}{TP + FN} =$$

False positive rate (FPR):

$$FPR = \frac{FP}{FP + TN} =$$



Computing ROC and AUROC using Scikit-learn function (Colab)

```
from sklearn.metrics import roc_curve, roc_auc_score

knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_test_prob = knn.predict_proba(X_test)[:,1] # column 0 is healthy,
↪  column 1 is cancer
fpr, tpr, thresholds = roc_curve(y_test, y_test_prob)
roc_auc = roc_auc_score(y_test, y_test_prob)
plt.clf()
plt.plot(fpr, tpr, "b-", lw=2, label="AUROC = %0.2f"%roc_auc)
plt.axline((0, 0), (1, 1), linestyle="--", lw=1, color='gray')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC in predicting cancer')
plt.legend(loc="best")
```

The resulting ROC