CSCE 421: Machine Learning

Lecture 2: Measures of Model Performance

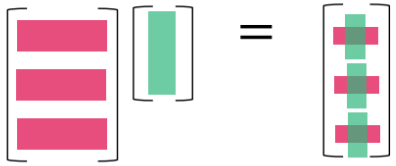
Texas A&M University
Section 201/501
Bobak Mortazavi
Ryan King
Zhale Nowroozilarki

Goals

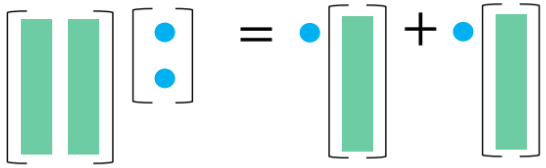
- Understand measures of supervised learning performance
- Be able to calculate class-specific metrics
- Be able to create receiver operating characteristic curves
- kNN – a simple supervised learning model

Matrix times Vector -2 Ways

Mv1



Mv2



The row vectors of A are multiplied by a vector $\boldsymbol{x}$ and become the three dot-product elements of $A\boldsymbol{x}$.

The product $A\boldsymbol{x}$ is a linear combination of the column vectors of A.

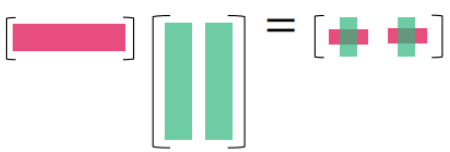
$$A\boldsymbol{x} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} (x_1+2x_2) \\ (3x_1+4x_2) \\ (5x_1+6x_2) \end{bmatrix}$$

$$A\boldsymbol{x} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} + x_2 \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$$

At first, you learn (Mv1). But when you get used to viewing it as (Mv2), you can understand Ax as a linear combination of the columns of A. Those products fill the column space of A denoted as C(A). The solution space of Ax=0 is the nullspace of A denoted as N(A)

Vector times Matrix – 2 Ways

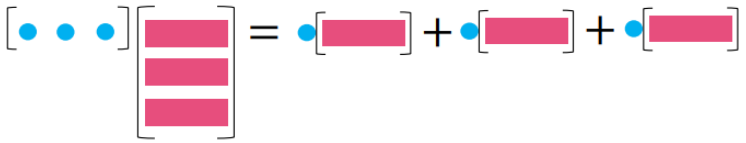
vM1



$$\boldsymbol{y}A = [y_1 \quad y_2 \quad y_3]\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix} = [(y_1+3y_2+5y_3) \quad (2y_1+4y_2+6y_3)]$$

A row vector $\boldsymbol{y}$ is multiplied by the two column vectors of A and become the two dot-product elements of $\boldsymbol{y}A$.

vM2

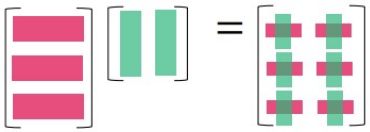


$$\boldsymbol{y}A = [y_1 \quad y_2 \quad y_3]\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix} = y_1[1 \quad 2] + y_2[3 \quad 4] + y_3[5 \quad 6]$$

The product $\boldsymbol{y}A$ is a linear combination of the row vectors of A.

Matrix times Matrix – 4 Ways

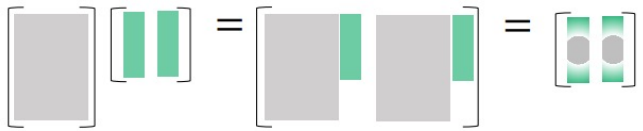
MM 1



Every element becomes a dot product of row vector and column vector.

$$\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}\begin{bmatrix}x_1 & y_1\\ x_2 & y_2\end{bmatrix}=\begin{bmatrix}(x_1+2x_2) & (y_1+2y_2)\\ (3x_1+4x_2) & (3y_1+4y_2)\\ (5x_1+6x_2) & (5y_1+6y_2)\end{bmatrix}$$

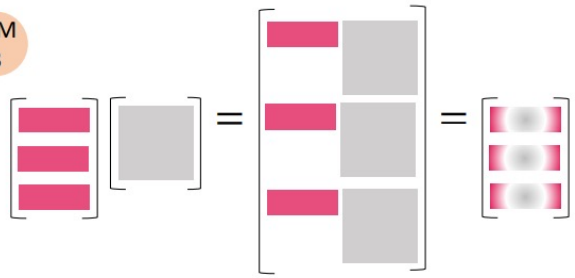
MM 2



$A\boldsymbol{x}$ and $A\boldsymbol{y}$ are linear combinations of columns of A.

$$\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}\begin{bmatrix}x_1 & y_1\\ x_2 & y_2\end{bmatrix}=A[\boldsymbol{x}\quad \boldsymbol{y}]=[A\boldsymbol{x}\quad A\boldsymbol{y}]$$

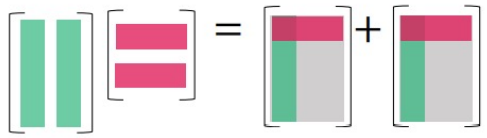
MM 3



The produced rows are linear combinations of rows.

$$\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}\begin{bmatrix}x_1 & y_1\\ x_2 & y_2\end{bmatrix}=\begin{bmatrix}\boldsymbol{a}_1^*\\ \boldsymbol{a}_2^*\\ \boldsymbol{a}_3^*\end{bmatrix}X=\begin{bmatrix}\boldsymbol{a}_1^*X\\ \boldsymbol{a}_2^*X\\ \boldsymbol{a}_3^*X\end{bmatrix}$$

MM 4



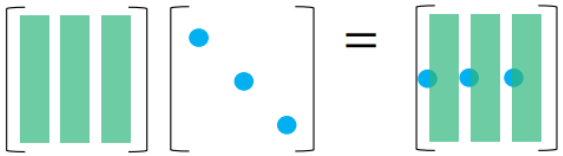
Multiplication AB is broken down to a sum of rank 1 matrices.

$$\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}\begin{bmatrix}b_{11} & b_{12}\\ b_{21} & b_{22}\end{bmatrix}=[\boldsymbol{a}_1\quad \boldsymbol{a}_2]\begin{bmatrix}\boldsymbol{b}_1^*\\ \boldsymbol{b}_2^*\end{bmatrix}=\boldsymbol{a}_1\boldsymbol{b}_1^*+\boldsymbol{a}_2\boldsymbol{b}_2^*$$

$$=\begin{bmatrix}1\\ 3\\ 5\end{bmatrix}[b_{11}\quad b_{12}]+\begin{bmatrix}2\\ 4\\ 6\end{bmatrix}[b_{21}\quad b_{22}]=\begin{bmatrix}b_{11} & b_{12}\\ 3b_{11} & 3b_{12}\\ 5b_{11} & 5b_{12}\end{bmatrix}+\begin{bmatrix}2b_{21} & 2b_{22}\\ 4b_{21} & 4b_{22}\\ 6b_{21} & 6b_{22}\end{bmatrix}$$

Practical Patterns (1/2)

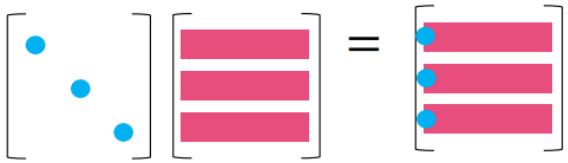
P1'



Applying a diagonal matrix from the right scales each column.

$$AD = [\boldsymbol{a_1} \quad \boldsymbol{a_2} \quad \boldsymbol{a_3}] \begin{bmatrix} d_1 & & \\ & d_2 & \\ & & d_3 \end{bmatrix} = [d_1\boldsymbol{a_1} \quad d_2\boldsymbol{a_2} \quad d_3\boldsymbol{a_3}]$$

P2'

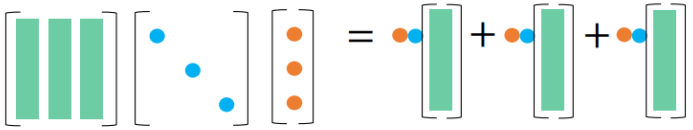


Applying a diagonal matrix from the left scales each row.

$$DB = \begin{bmatrix} d_1 & & \\ & d_2 & \\ & & d_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_1^* \\ \boldsymbol{b}_2^* \\ \boldsymbol{b}_3^* \end{bmatrix} = \begin{bmatrix} d_1\boldsymbol{b}_1^* \\ d_2\boldsymbol{b}_2^* \\ d_3\boldsymbol{b}_3^* \end{bmatrix}$$

Practical Patterns (2/2)

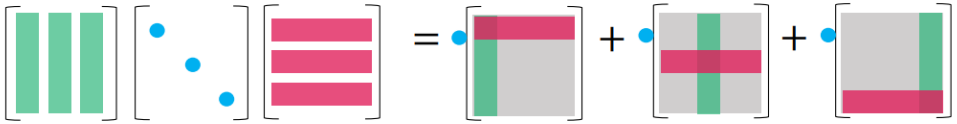
P3



This pattern makes another combination of columns.
You will encounter this in differential/recurrence equations.

$$XDc = [x_1 \quad x_2 \quad x_3]\begin{bmatrix} d_1 & & \\ & d_2 & \\ & & d_3 \end{bmatrix}\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = c_1 d_1 x_1 + c_2 d_2 x_2 + c_3 d_3 x_3$$

P4



A matrix is broken down to a sum of rank 1 matrices,
as in singular value/eigenvalue decomposition.

$$U\Sigma V^{\mathrm{T}} = [u_1 \quad u_2 \quad u_3]\begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{bmatrix}\begin{bmatrix} v_1^{\mathrm{T}} \\ v_2^{\mathrm{T}} \\ v_3^{\mathrm{T}} \end{bmatrix} = \sigma_1 u_1 v_1^{\mathrm{T}} + \sigma_2 u_2 v_2^{\mathrm{T}} + \sigma_3 u_3 v_3^{\mathrm{T}}$$

Making prediction using a trained model - Regression

- Linear Regression model:

$$model(\boldsymbol{x}, \boldsymbol{w}^*) = \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{w}^* = \boldsymbol{w}_0^* + \boldsymbol{x}_1\boldsymbol{w}_1^* + \cdots + \boldsymbol{x}_N\boldsymbol{w}_N^*$$

$\boldsymbol{w}^*$: the optimal set of weights found by minimizing a regression cost function.

Prediction: Given input $\boldsymbol{x}$, the output $\hat{y}$ is predicted by

$$model(\boldsymbol{x}, \boldsymbol{w}^*) = \hat{y}$$

Judging the quality of a trained regression model

- Least Squares: Mean Square Error (MSE)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(model(\boldsymbol{X}_i, \boldsymbol{w}^*) - \boldsymbol{y}_i)^2$$

- Least Absolute Deviation: Mean Absolute Deviation (MAD):

$$MAD = \frac{1}{N}\sum_{i=1}^{N}|model(\boldsymbol{X}_i, \boldsymbol{w}^*) - \boldsymbol{y}_i|$$

Judging the quality of a trained regression model

- Least Squares: Mean Square Error (MSE)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(model(\boldsymbol{x}_i, \boldsymbol{w}^*) - \boldsymbol{y}_i)^2$$

- Least Absolute Deviation: Mean Absolute Deviation (MAD):

$$MAD = \frac{1}{N}\sum_{i=1}^{N}|model(\boldsymbol{x}_i, \boldsymbol{w}^*) - \boldsymbol{y}_i|$$

- Comparison
  - MSE is sensitive to outliers
  - MAD is better measure of error than MSE if forecast error does not have asymmetric distribution

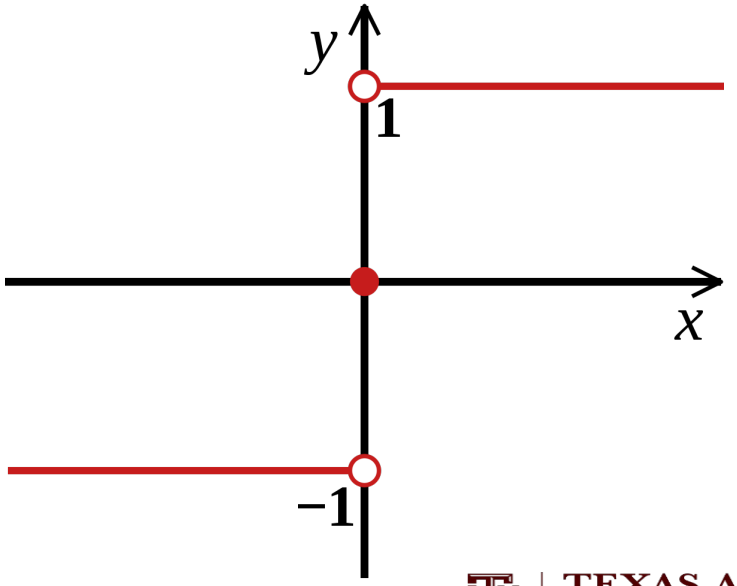
Making predictions using a trained model - Classification

- Classification model (PLA):

$$model(\boldsymbol{x}, \boldsymbol{w}^*) = sign(\boldsymbol{x}^T \boldsymbol{w}^*) = sign(\boldsymbol{w}_0^* + \boldsymbol{x}_1 \boldsymbol{w}_1^* + \cdots + \boldsymbol{x}_N \boldsymbol{w}_N^*)$$

- Labels: $y_p = \{-1, +1\}$

- $sign(x) = \begin{cases} +1 \; if \; x \geq 0 \\ -1 \; if \; x < 0 \end{cases}$



Making predictions using a trained model - Classification

- Classification model (PLA):

$$model(\boldsymbol{x}, \boldsymbol{w}^*) = sign(\boldsymbol{x}^T \boldsymbol{w}^*) = sign(w_0^* + x_1 w_1^* + \cdots + x_N w_N^*)$$

Prediction: Given input $\boldsymbol{x}$, the output y is predicted by

$$model(\boldsymbol{x}, \boldsymbol{w}^*) = \hat{y}$$

Confidence Scoring

- Confidence of the decision boundary: The distance between points and the boundary
  - Zero confidence: points lie along the boundary. The boundary cannot tell accurately which class such points belong to.
  - Little Confidence: Near the boundary. When the boundary has slight change, the points close to the original boundary can end up on the opposite side of the new boundary
  - High Confidence: The prediction labels of points far from the decision boundary. The predicted labels will not change if we make a small change to the location of the decision boundary.

Confidence scoring

- Distance: $d = \frac{b^* + \mathbf{x}_p^T \boldsymbol{\omega}^*}{||\boldsymbol{\omega}^*||_2}$
  - Bias: $b* = w_0^*$
  - Weights: $\boldsymbol{\omega}^* = \begin{bmatrix} w_1^* \\ w_2^* \\ \vdots \\ w_N^* \end{bmatrix}$
  - Confidence in the predicted label of a point $x = \sigma(d)$
    Sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$

Judging the quality of a trained model using accuracy

- The prediction of $\boldsymbol{x}_i$ is $\hat{\boldsymbol{y}}_i = model(\boldsymbol{x}_i, \boldsymbol{w}^*)$, and the true label is $\boldsymbol{y}_i$.
- Indicator function: returns 1 where condition is true

$$I_{[\hat{\boldsymbol{y}}_i = \boldsymbol{y}_i]} = \begin{cases} 0 \; if \hat{\boldsymbol{y}}_i \neq \boldsymbol{y}_i \\ 1 \; if \hat{\boldsymbol{y}}_i = \boldsymbol{y}_i \end{cases}$$

- Number of correctly labeled samples: $\sum_{i=1}^{N} I_{[\hat{y}_i = y_i]}$
- Accuracy: $\mathrm{A} = \frac{1}{\mathrm{N}} \sum_{i=1}^{N} I_{[\hat{\boldsymbol{y}}_i = \boldsymbol{y}_i]}$
  - The accuracy ranges form 0 (no points are classified correctly) to 1 (all points are classified correctly)

Judging the quality of a trained model using balanced accuracy

- Classification accuracy is easy and excellent, but it can paint an incomplete picture of how well we have really solved a classification problem

Judging the quality of a trained model using balanced accuracy

- Classification accuracy is easy and excellent, but it can paint an incomplete picture of how well we have really solved a classification problem
  - A dataset consists of highly imbalanced classes: if one class makes us 95% of all data points, a naïve classifier that blindly assigns the label of the majority class to every training point achieves an accuracy of 95% but here misclassifying 5% amounts to completely misclassifying an entire class of data.

Judging the quality of a trained model using balanced accuracy

- Balanced Accuracy
  - Denote the indices of samples with labels $y_i = +1$ and $y_i = -1$ as Ω_{+1} and Ω_{-1}
  - Number of correctly classified samples on $+1$ class: $\sum_{i \in \Omega_{+1}} I_{[\hat{y}_i = y_i]}$
  - Number of correctly classified samples on -1 class: $\sum_{i \in \Omega_{-1}} I_{[\hat{y}_i = y_i]}$
  - Accuracy:
    - $\mathrm{A}_{+1} = \frac{1}{|\Omega_{+1}|} \sum_{i \in \Omega_{+1}} I_{[\hat{y}_i = y_i]}$
    - $\mathrm{A}_{-1} = \frac{1}{|\Omega_{-1}|} \sum_{i \in \Omega_{-1}} I_{[\hat{y}_i = y_i]}$
    - $A_{balanced} = \frac{\mathrm{A}_{+1} + \mathrm{A}_{-1}}{2}$

Confusion Matrix and Accuracy

		Predicted	
	Total Population	Positive	Negative
Actual	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

- Accuracy: $A = \frac{TP+TN}{TP+TN+FP+FN}$
- Accuracy of each class: $A_+ = \frac{TP}{TP+FN} = TPR, A_- = \frac{TN}{TN+FP} = TNR$
- Balanced Accuracy: $\frac{1}{2}A_+ + \frac{1}{2}A_- = \frac{1}{2}TPR + \frac{1}{2}TNR = \frac{1}{2}\frac{TP}{TP+FN} + \frac{1}{2}\frac{TN}{TN+FP}$

Confusion Matrix and Accuracy

		Predicted	
	Total Population	Positive	Negative
Actual	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

- Precision $= \frac{TP}{TP+FP}$
- Recall $= \frac{TP}{TP+FN}$
- F_1: $\frac{2TP}{2TP+FP+FN} = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$

Confusion Matrix and Accuracy

		Predicted	
	Total Population	Positive	Negative
Actual	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

- Precision $= \frac{TP}{TP+FP}$
- Recall $= \frac{TP}{TP+FN}$
- $F_1 = \frac{2TP}{2TP+FP+FN} = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$
- Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$
- Balanced Accuracy $= \frac{1}{2}\frac{TP}{TP+FN} + \frac{1}{2}\frac{TN}{TN+FP}$

Example

		Predicted	
	Total Population	Positive	Negative
Actual	Positive	TP=25	FN=5
	Negative	FP=5	TN=25

- Precision $= \frac{TP}{TP+FP}$
- Recall $= \frac{TP}{TP+FN}$
- $F_1 = \frac{2TP}{2TP+FP+FN} = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$
- Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$
- Balanced Accuracy $= \frac{1}{2}\frac{TP}{TP+FN} + \frac{1}{2}\frac{TN}{TN+FP}$

Example

		Predicted	
	Total Population	Positive	Negative
Actual	Positive	TP=25	FN=5
	Negative	FP=5	TN=25

- Precision $= \frac{TP}{TP+FP} = \frac{25}{25+5} = 0.833$
- Recall $= \frac{TP}{TP+FN} = \frac{25}{25+5} = 0.833$
- $F_1 = \frac{2TP}{2TP+FP+FN} = \frac{TP}{TP+\frac{1}{2}(FP+FN)} = \frac{25}{25+\frac{1}{2}(5+5)} = 0.833$
- Accuracy $= \frac{TP+TN}{TP+TN+FP+FN} = \frac{25+25}{25+25+5+5} = 0.833$
- Balanced Accuracy $= \frac{1}{2}\frac{TP}{TP+FN} + \frac{1}{2}\frac{TN}{TN+FP} = \frac{1}{2}\frac{25}{25+5} + \frac{1}{2}\frac{5}{25+5} = 0.5$

Practice 1

		Predicted	
	Total Population	Positive	Negative
Actual	Positive	TP=2	FN=3
	Negative	FP=85	TN=201

- Calculate the accuracy based on the given confusion matrix

Practice 2

		Predicted	
	Total Population	Positive	Negative
Actual	Positive	TP=2	FN=3
	Negative	FP=85	TN=201

- Calculate the precision, recall, and F_1 score based on the given confusion matrix

F-measure

Why F-measure?

		Predicted	
	Total Population	Positive	Negative
Actual	Positive	TP=2	FN=3
	Negative	FP=85	TN=201

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{2+201}{2+85+3+201} = 0.6975945017$
- Not bad if only looking at accuracy.
- If only looking at class +1, this is a poor classifier. Most samples are classified to class -1.

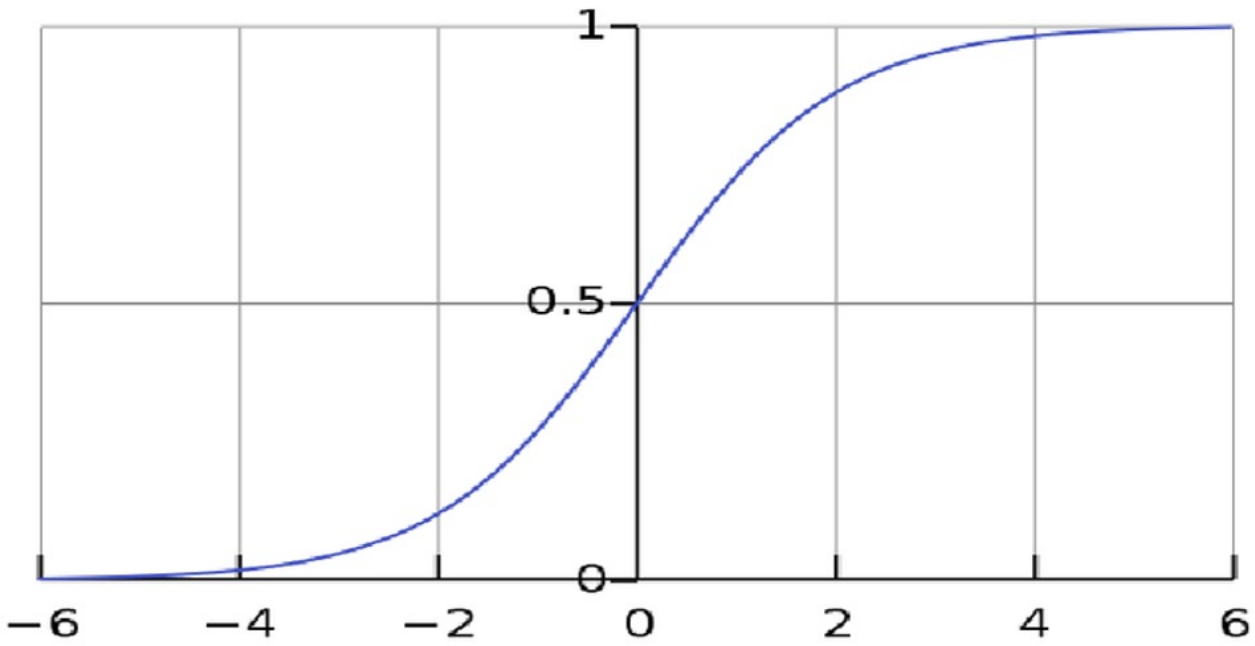
F-measure

Why F-measure?

		Predicted	
	Total Population	Positive	Negative
Actual	Positive	TP=2	FN=3
	Negative	FP=85	TN=201

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{2+201}{2+85+3+201} = 0.6975945017$
- $Precision = \frac{TP}{TP+FP} = \frac{2}{2+85} = 0.0229885057$
- $Recall = \frac{TP}{TP+FN} = \frac{2}{2+3} = 0.4$
- $F1 = \frac{TP}{TP+\frac{1}{2}(FP+FN)} = \frac{2}{2+\frac{1}{2}(85+3)} = 0.0434782609$

Decision Threshold



Decision Threshold

- If we pick a decision threshold of $p(x) > 0.5$ what happens?

Predicted	Ground Truth
2%	0
3%	0
5%	1
1%	0
15%	0
25%	1
24%	1
13%	0
8%	0
12%	1

Decision Threshold

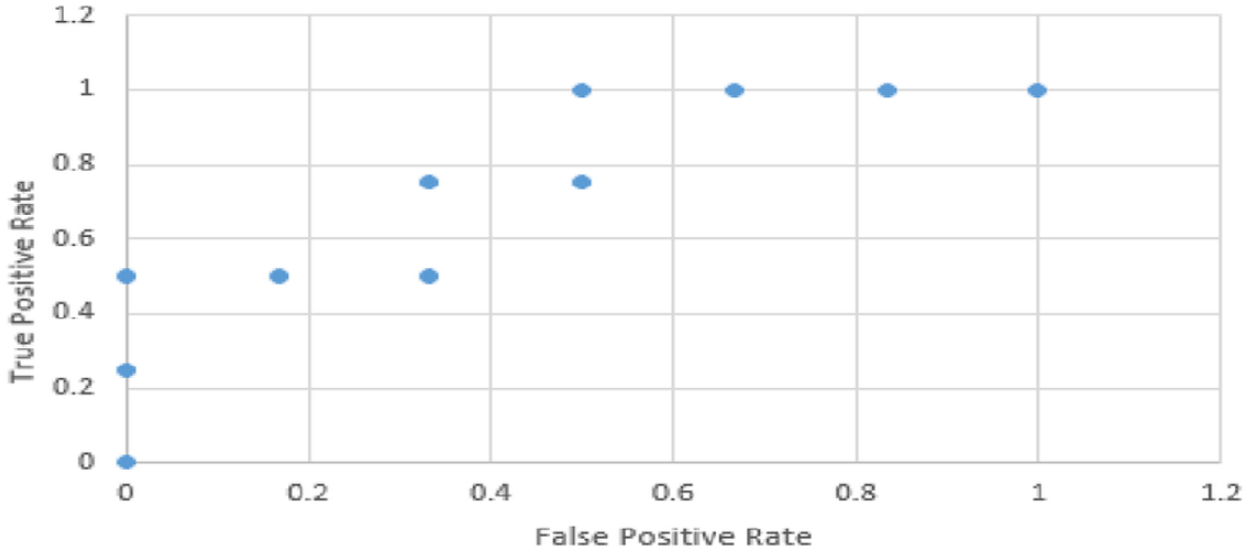
Predicted	TRUE	50	25	24	15	13	12	8	5	3	2	1
1	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	0	0	0	1	1	1
5	1	0	0	0	0	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0	1	1	1	1	1
12	1	0	0	0	0	0	1	1	1	1	1	1
13	0	0	0	0	0	1	1	1	1	1	1	1
15	0	0	0	0	1	1	1	1	1	1	1	1
24	1	0	0	1	1	1	1	1	1	1	1	1
25	1	0	1	1	1	1	1	1	1	1	1	1

Decision Threshold

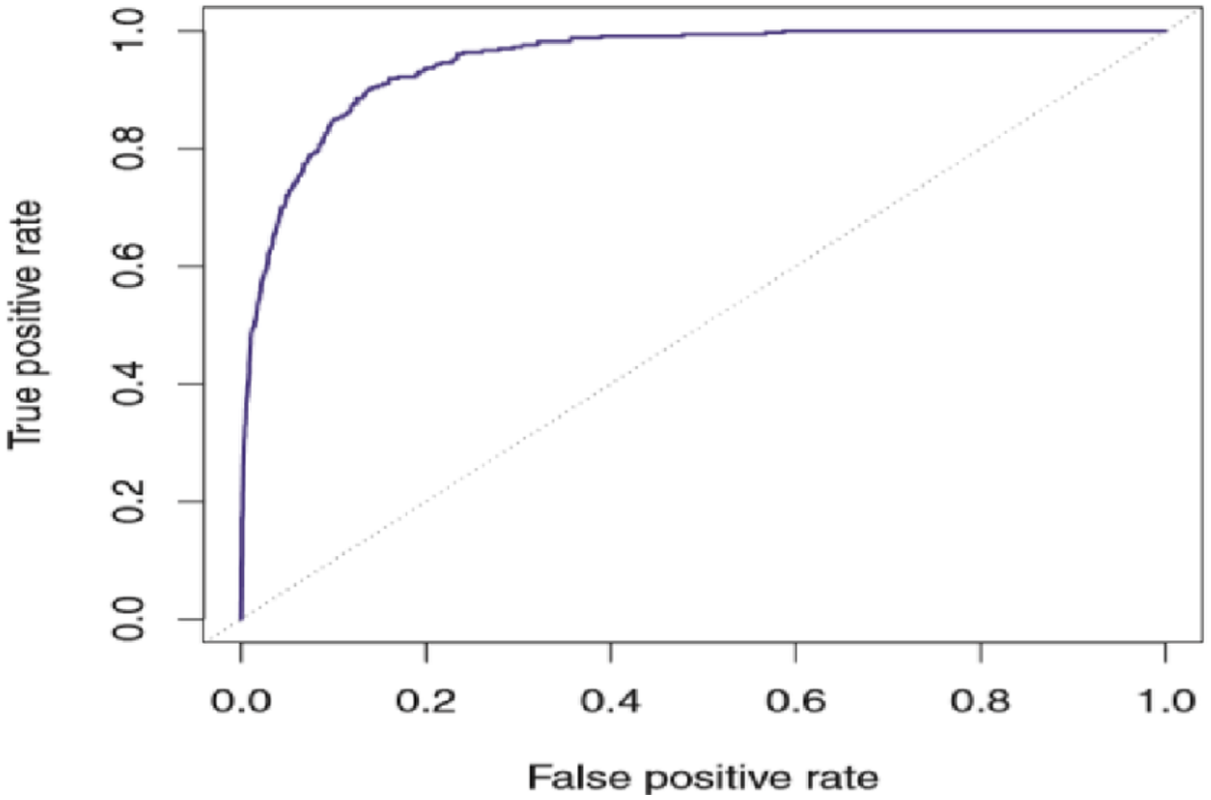
	50	25	24	15	13	12	8	5	3	2	1
TP	0	1	2	2	2	3	3	4	4	4	4
FP	0	0	0	1	2	2	3	3	4	5	6
TN	6	6	6	5	4	4	3	3	2	1	0
FN	4	3	2	2	2	1	1	0	0	0	0
TPR	0	0.25	0.5	0.5	0.5	0.75	0.75	1	1	1	1
FPR	0	0	0	0.166667	0.333333	0.333333	0.5	0.5	0.666667	0.833333	1

Decision Threshold

	50	25	24	15	13	12	8	5	3	2	1
TP	0	1	2	2	2	3	3	4	4	4	4
FP	0	0	0	1	2	2	3	3	4	5	6
TN	6	6	6	5	4	4	3	3	2	1	0
FN	4	3	2	2	2	1	1	0	0	0	0
TPR	0	0.25	0.5	0.5	0.5	0.75	0.75	1	1	1	1
FPR	0	0	0	0.166667	0.333333	0.333333	0.5	0.5	0.666667	0.833333	1



Receiver Operating Characteristic Curve



AUC-ROC Curve

What is AUC-ROC curve

- Area Under the Curve (AUC): the measure of separability
- Receiver Characteristic Operator (ROC): a probability curve
- AUC-ROV curve is a performance measurement for classification problem at various threshold settings
- It tells how much a model is capable of distinguishing between classes
- By predicting probability instead of label, can generate ROC Curve
- Can choose threshold on ROC curve that optimizes some threshold-specific measurement.
- Can also plot Precision-Recall Curve

AUC-ROC Curve

- True Positive Rate (TPR) / Recall / Sensitivity:

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

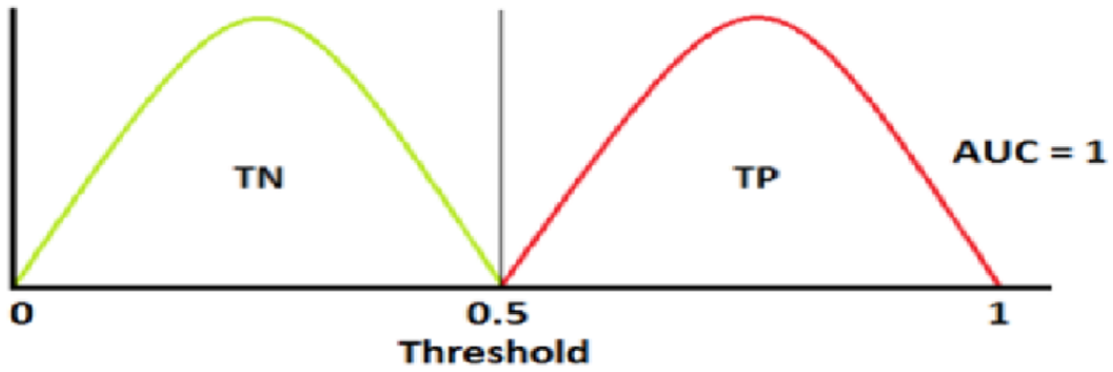
- Specificity:

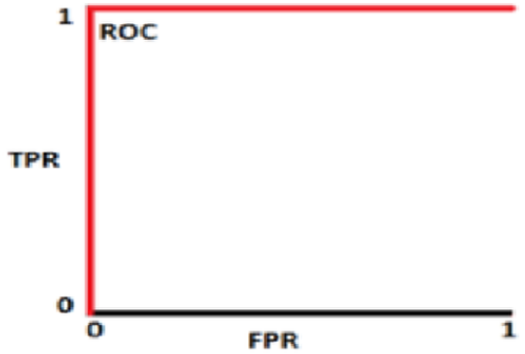
$$Specificity = \frac{TN}{TN + FP}$$

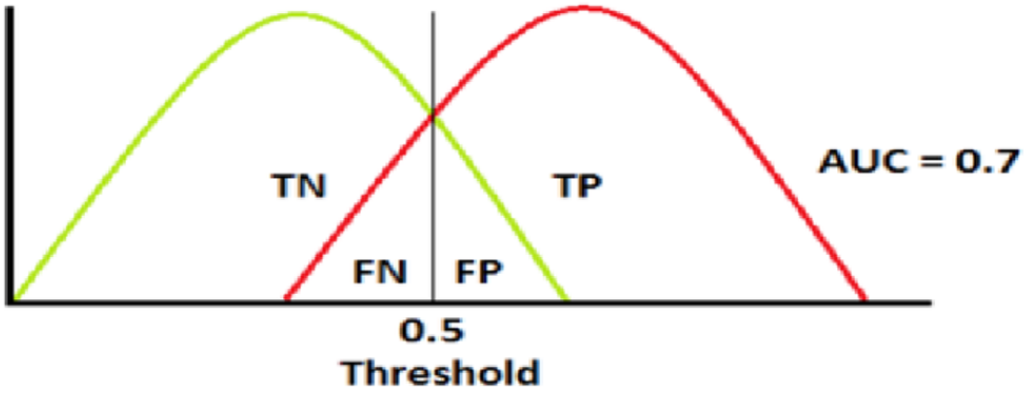
- False Positive Rate (FPR):

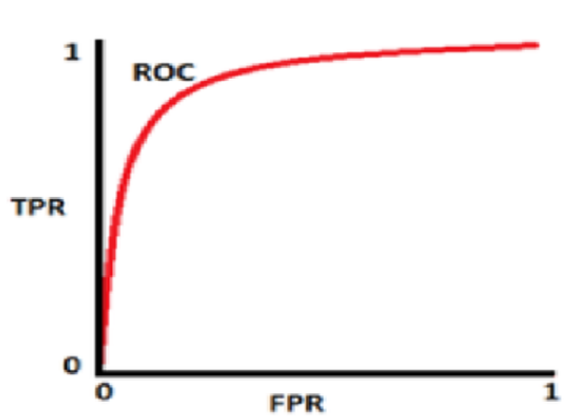
$$FPR = 1 - Specificity = \frac{FP}{TN + FP}$$

AUC-ROC Curve

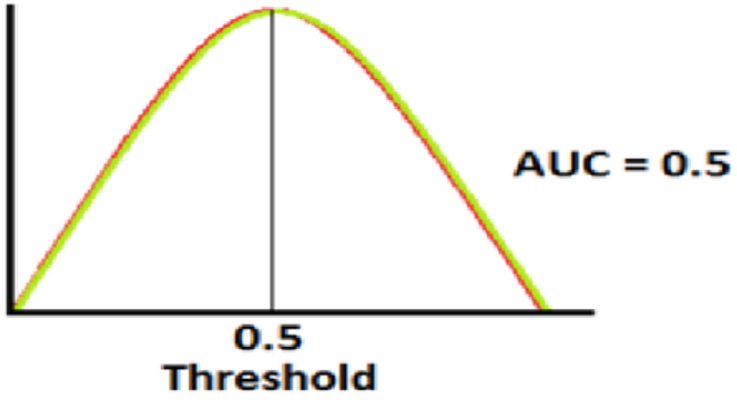


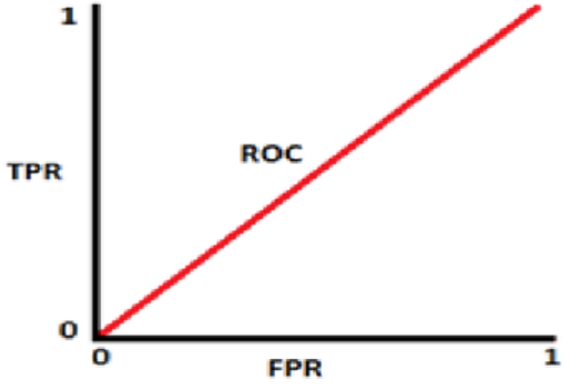


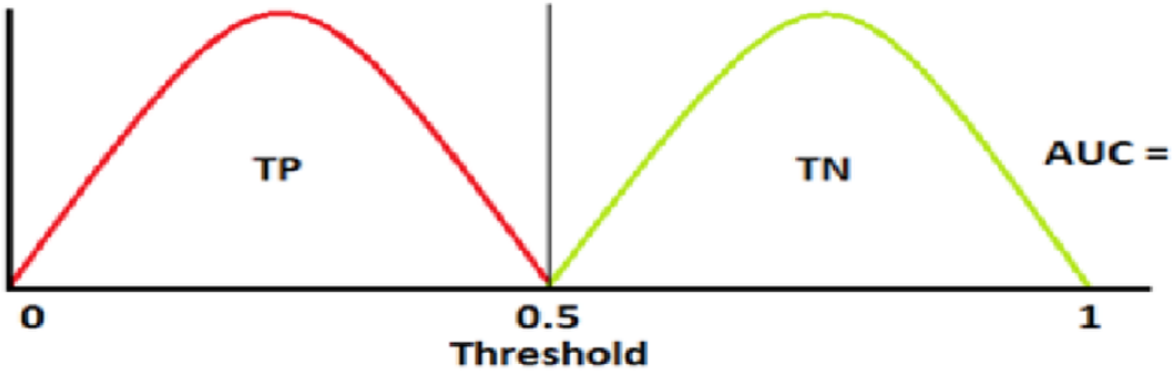


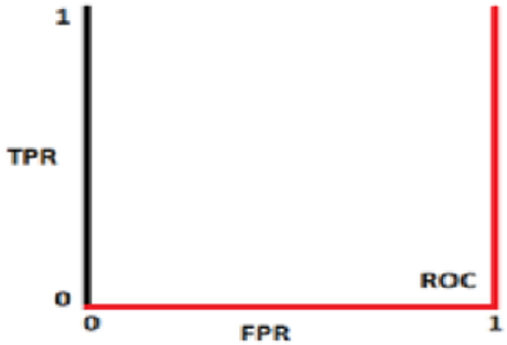


AUC-ROC Curve









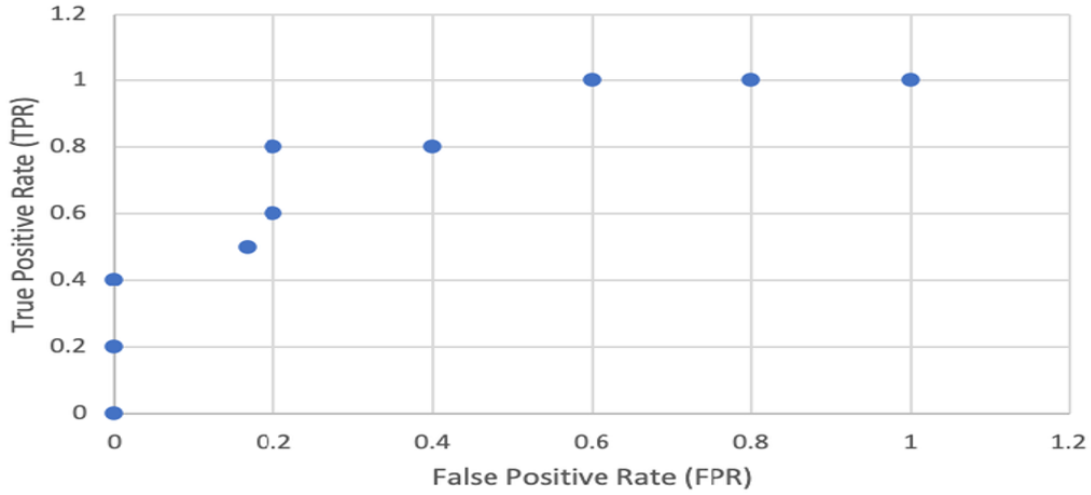
AUC-ROC Curve

- Practice 3: create AUC-ROC curve from the given samples below:

Predicted	Ground Truth
13%	1
8%	0
4%	0
45%	1
75%	1
64%	1
23%	0
18%	0
32%	1
55%	0

AUC-ROC Curve

	80	70	60	50	40	30	20	10	5
TP	0	1	2	2	3	4	4	5	5
FP	0	0	0	1	1	1	2	3	4
TN	5	5	5	5	4	4	3	2	1
FN	5	4	3	2	2	1	1	0	0
TPR	0	0.2	0.4	0.5	0.6	0.8	0.8	1	1
FPR	0	0	0	0.1666667	0.2	0.2	0.4	0.6	0.8



K-Nearest Neighbor: Example

Recognizing types of Iris flowers (by R. Fisher)

setosa

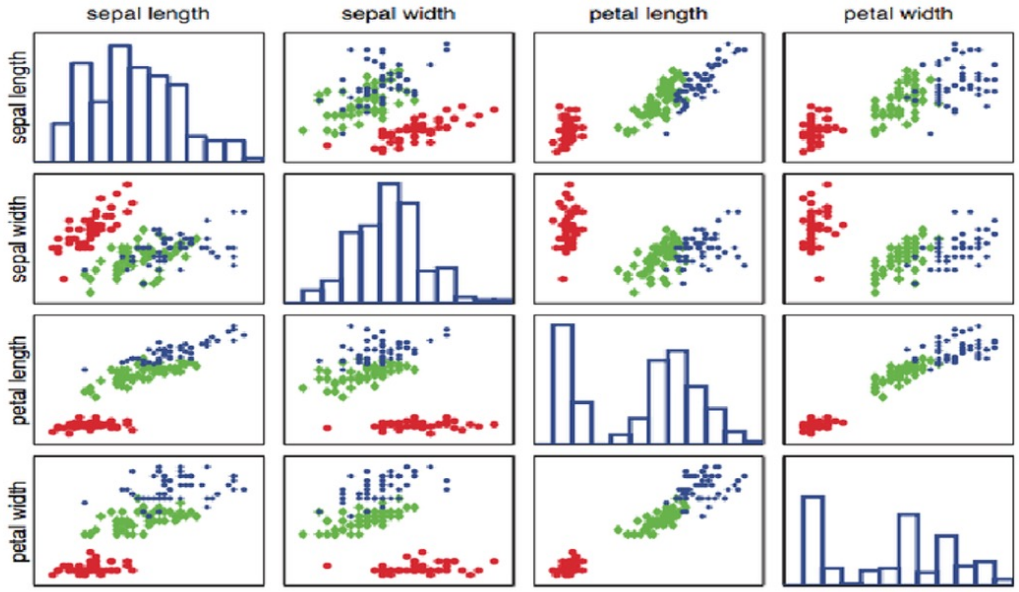
versicolor

virginica

Features: the widths and lengths of sepal and petal

K-Nearest Neighbor: Examples

Visualizing features to get better intuition about our data

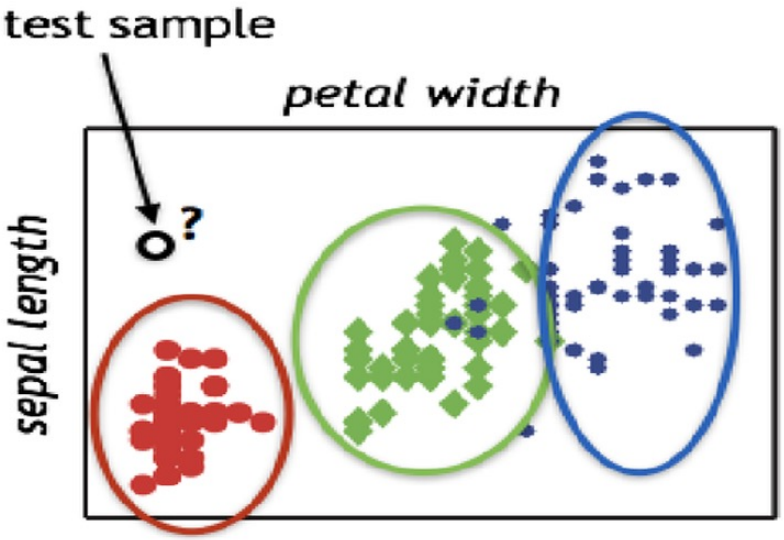


Each colored datapoint is one sample

setosa, versicolor, virginica

K-Nearest Neighbor: Examples

Using two features: sepal length & petal width



setosa, versicolor, virginica

Test sample is closer to red cluster → label it as setosa

K-Nearest Neighbor: Examples

Recognizing types of Iris flowers (by R. Fisher)

Often data is organized in a table

Each row is one sample with 4 features and 1 label

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
```

```
Attribute Information:
1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
   -- Iris Setosa
   -- Iris Versicolour
   -- Iris Virginica
```

[Source: https://archive.ics.uci.edu/ml/datasets/iris]

K-Nearest Neighbor: Representation

- Training Data
  - N samples/datapoints/instances: $S^{train} = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\}$
  - Used for learning representation: $f: \boldsymbol{x} \rightarrow y$
- Testing Data
  - M samples/datapoints/instances: $S^{test} = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_{\boldsymbol{M}}, y_M)\}$
  - Used to assess how well $f(\cdot)$ will do in predicting an unseen sample
- Train and test data should not overlap: $S^{train} \cap S^{test} = \emptyset$

K-Nearest Neighbor: Representation

- Classify data into one out of multiple classes
  - Input: $\boldsymbol{x} \in \mathbb{R}^D$
  - Output: y $\in \{1,2,\dots,C\}$, C is the number of classes
  - Model: $f: \boldsymbol{x} \rightarrow y$

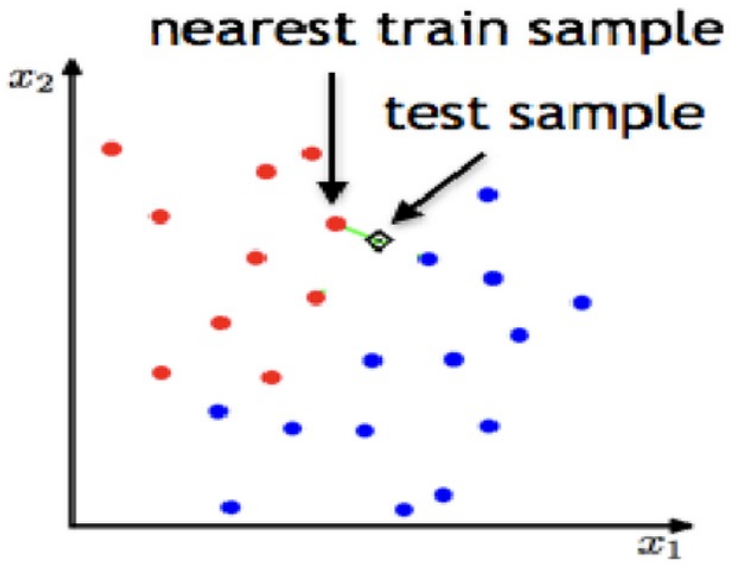
- Special case: binary classification (C=2)
  - Output: y $\in \{1,2\}$ or $\{0,1\}$ or $\{-1,1\}$

K-Nearest Neighbor: Representation

- Nearest Neighbor (or 1-Nearest Neighbor, 1-NN)
  - Assigns test sample $\boldsymbol{x}'$ to the closest training sample
  - Model:
    - $\hat{y} = f(\boldsymbol{x}) = \boldsymbol{y}_{\boldsymbol{nn}(\boldsymbol{x})}$
    - $nn(\boldsymbol{x}) = arg\min_{i=1,\dots,M} \left|\left|\boldsymbol{x}' - \boldsymbol{x}_i\right|\right|^2 = arg\min_{i=1,\dots,M} \sum_{j=1}^{D} \left(\boldsymbol{x}'_j - \boldsymbol{x}_{ij}\right)^2$

K-Nearest Neighbor: Representation

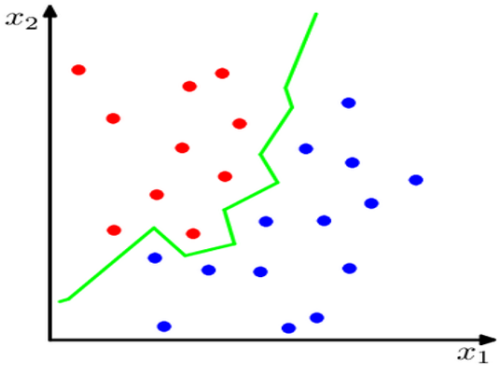
- Nearest Neighbor (or 1-Nearest Neighbor, 1-NN): Example



- The nearest point to test sample $\boldsymbol{x}'$ is a red training instance, therefore $\boldsymbol{x}'$ will be labeled as red

K-Nearest Neighbor: Representation

- Nearest Neighbor (or 1-Nearest Neighbor, 1-NN): Example
  - Decision boundary: For every point in the space, we can determine its label using the nearest neighbor rule. This gives us a decision boundary that partitions the space into different regions.



- The above decision boundary is very sensitive to noise. What would be the solution for this?

K-Nearest Neighbor: Representation

Increase number of nearest neighbors to use

- 1-nearest neighbor: $nn_1(\boldsymbol{x}') = arg \min_{i \epsilon \{1,\dots,M\}} \|\boldsymbol{x}' - \boldsymbol{x}_i\|_2^2$
- 2-nearest neighbor: $nn_2(\boldsymbol{x}') = arg \min_{i \epsilon \{1,\dots,M\} \setminus nn_1(\boldsymbol{x}\prime)} \|\boldsymbol{x}' - \boldsymbol{x}_i\|_2^2$
- 3-nearest neighbor: $nn_3(\boldsymbol{x}') = arg \min_{i \epsilon \{1,\dots,M\} \{nn_1(\boldsymbol{x}\prime), nn_2(\boldsymbol{x}\prime)\}} \|\boldsymbol{x}' - \boldsymbol{x}_i\|_2^2$

The set of K-nearest neighbors is

$$knn(\boldsymbol{x}) = '\{nn_1(\boldsymbol{x}'), \dots, nn_K(\boldsymbol{x}')\}$$

Neighbors $nn_1, \dots, nn_K$ in order of increasing distance from sample $\boldsymbol{x}'$

K-Nearest Neighbor: Representation

- K-NN Model
  - Each neighbor in $knn(\boldsymbol{x}) = \{nn_1(\boldsymbol{x}), \dots, nn_K(\boldsymbol{x})\}$ votes one class
  - Count the number of neighbors that have voted each class (C)

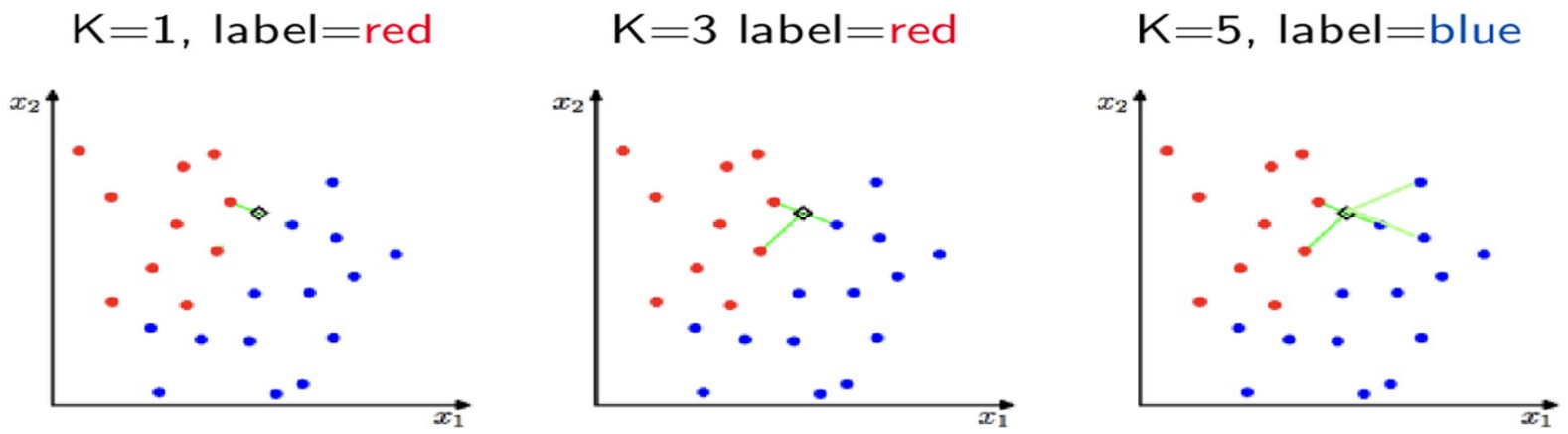
$$v_c = \sum_{k \in knn(\boldsymbol{x})} I_{[y_k = c]}, c = 1, \dots, C$$

  - Assign test sample $\boldsymbol{x}$ to the majority class membership of the K neighbors

$$y = f(\boldsymbol{x}) = arg \max_{c=1,\dots,C} v_c$$

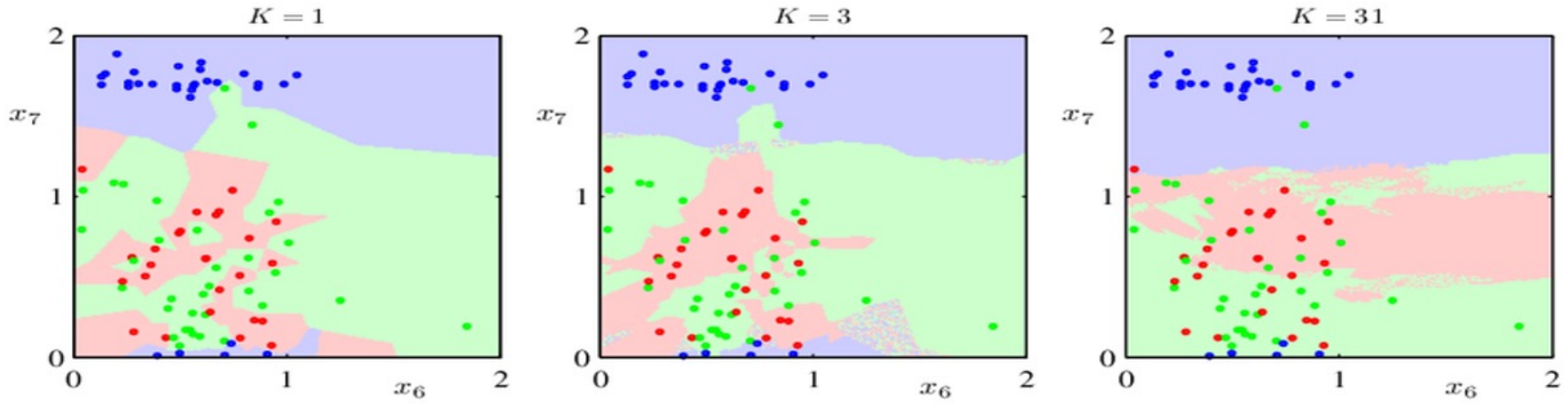
K-Nearest Neighbor: Representation

- K-NN Example



K-Nearest Neighbor: Representation

K-NN Decision Boundary



Number of neighbors K controls the degree of smoothing

- $K \downarrow$: many small regions of each class
- $K \uparrow$: fewer larger regions of each class