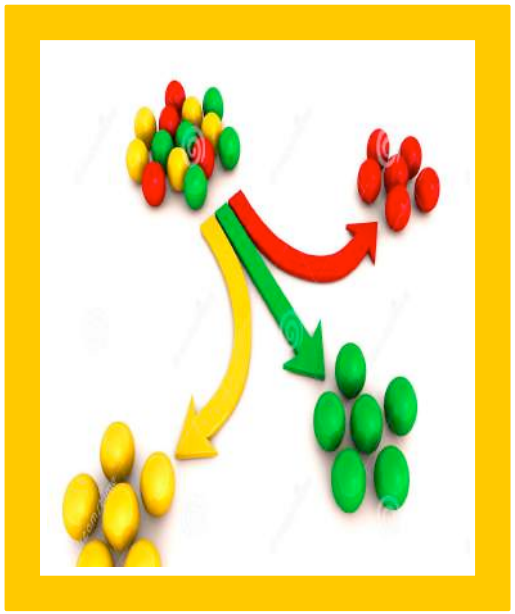


Classification

Introduction to classifiers, classification and performance analysis

The machine learning framework

- Apply a prediction function to a feature representation of the “sample” to get the desired output:

f() = “apple”

f() = “tomato”

f() = “cow”

The machine learning framework

$$y = f(\mathbf{x})$$

output — prediction function — feature or representation

- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error.
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Slide credit: L. Lazebnik

Steps

Training

Training Labels

Training Data

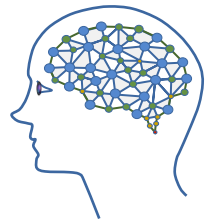
Features → Training → Learned model

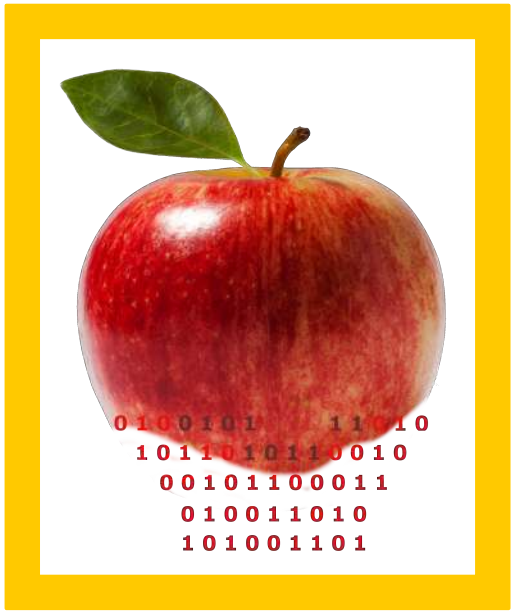
Testing

Test sample

Features → Learned model → Prediction

Slide credit: D. Hoiem and L. Lazebnik



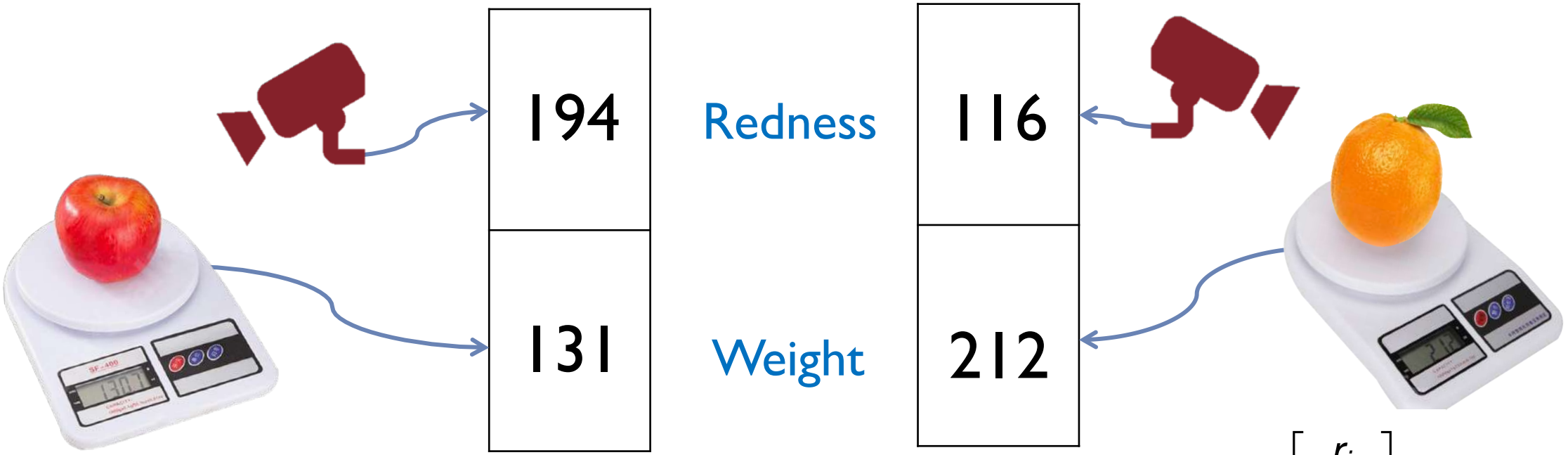


Data Representation

How Real World Objects Become Vectors or Points

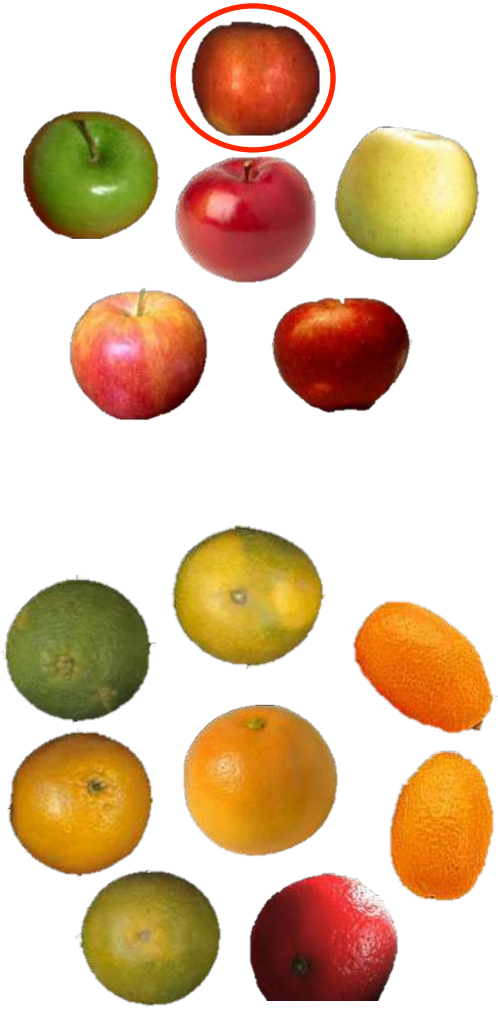
Characterising Apples and Oranges

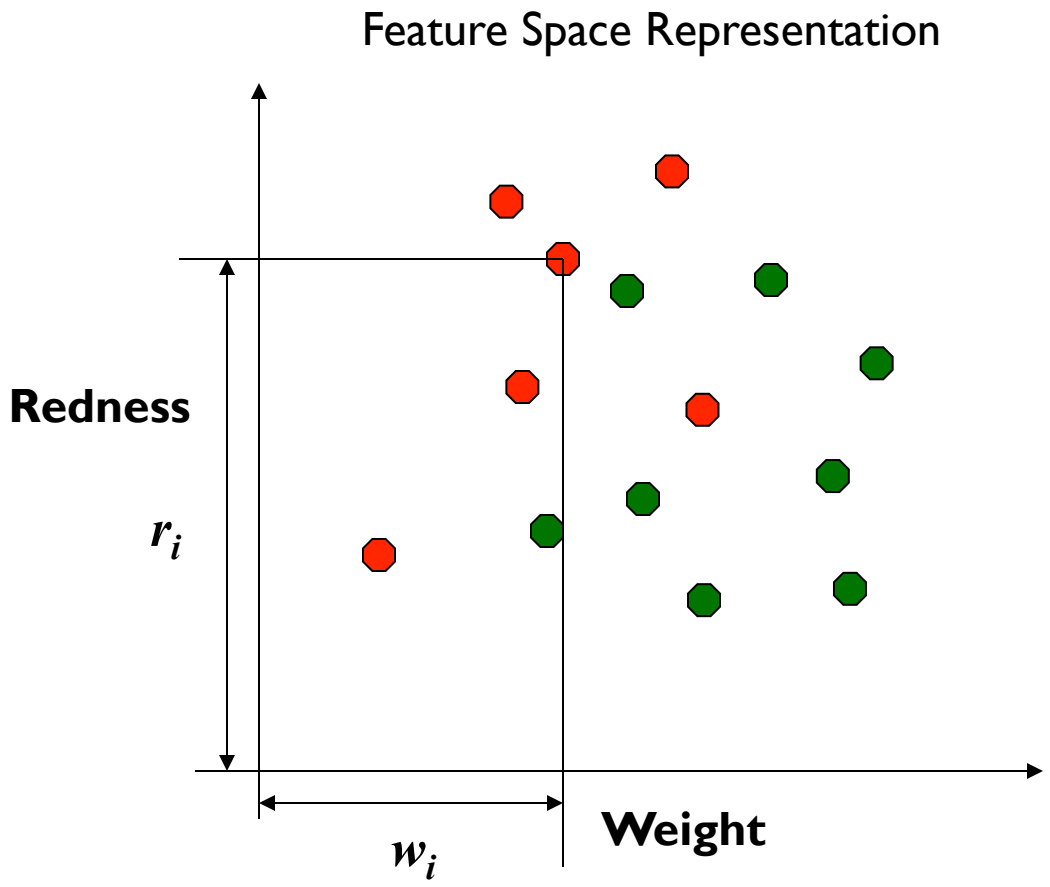
- What makes Apples and Oranges different?



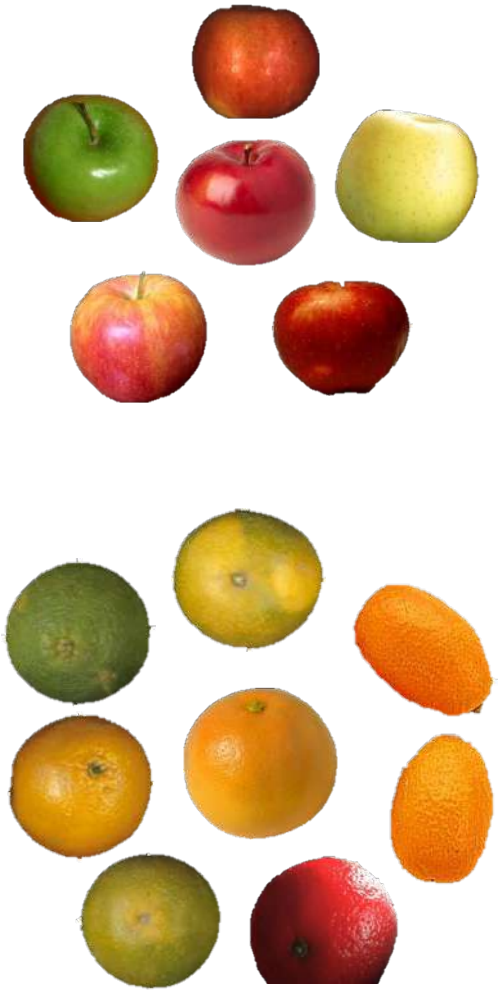
- Now each fruit is represented as a 2D vector: $\mathbf{x}_i = \begin{bmatrix} r_i \\ w_i \end{bmatrix}$
- The components r_i and w_i are called features of i th fruit.

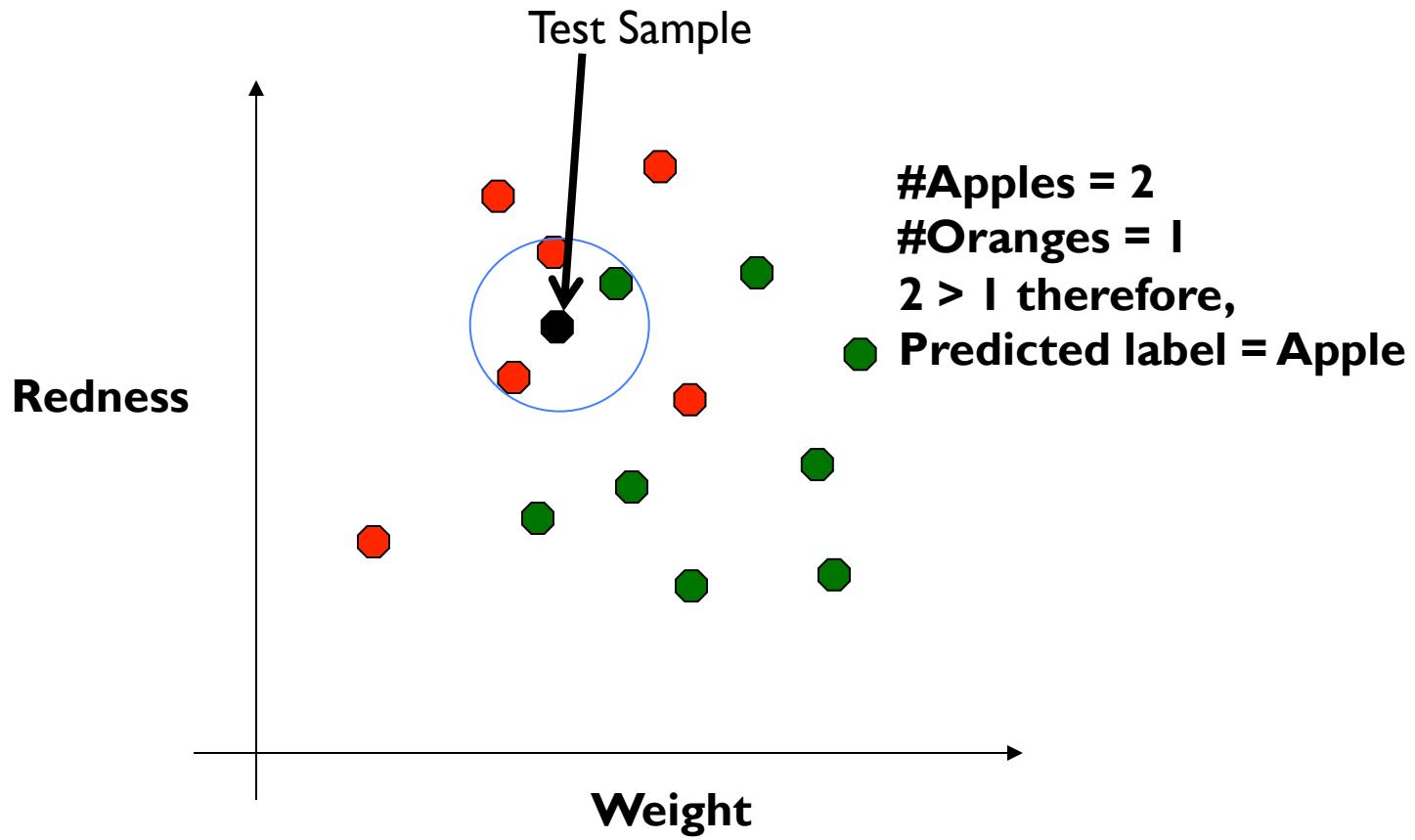
Feature Space (*Here: 2D*)





KNN Classifier (K=3)





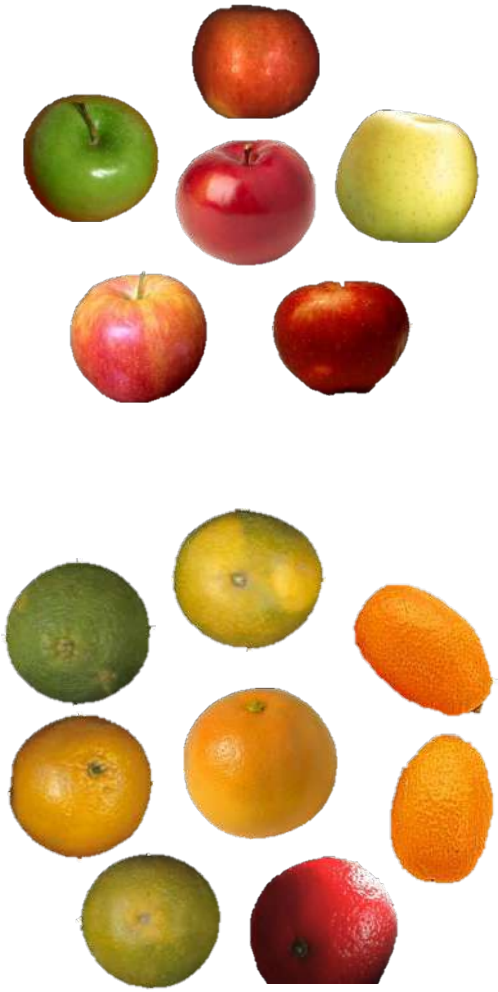
Summary-1

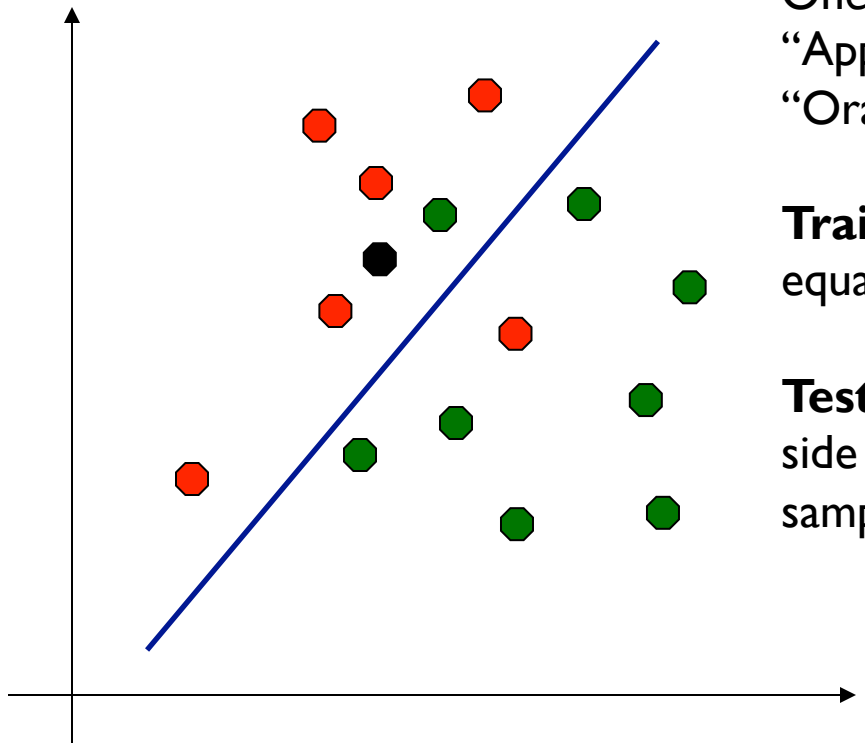
- **Training** Samples:
  - Examples **(Sample, label)** that we had access before testing/deploying
- **Test** Samples:
  - The **samples** where we want to know the label. (i.e., predict the label. Label is here either "apple" or "orange")
  - We may assume "1" imply apple and "o" imply orange.
- When there are only two classes, this is called **binary classification**. More than two, problem is **multiclass classification**.

Summary-1

- The procedure does not change even if we had another class/label here. (i.e., mango)
  - Multiclass classification is natural to KNN ☺
- KNN: Observations
  - We need to carry the "training samples" whenever we want to test. Lazy learning.
- KNN: Concerns:
  - How to fix "K", "Distance Functions", "Scaling of Features"?

Linear Classification (*In 2D*)





One side of the line is "Apple". Other side is "Orange"

Training: Finding the equation of the line.

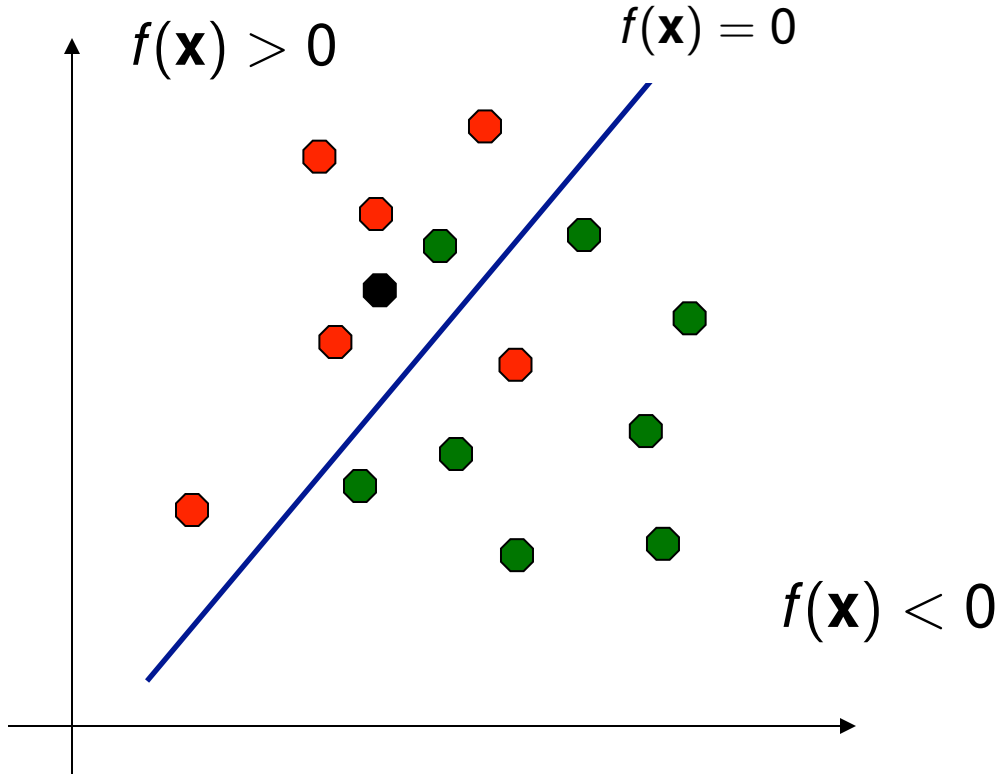
Testing: Find which side of the line the sample lies.

Linear Classification

$$f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_d x_d$$

Feature Vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$

Parameters to be learned $\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$



Compactly: $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ or $\mathbf{w} \cdot \mathbf{x}$

A Simple Case

Let there are two features x_1 and x_2 and Equation of the line be:

$$x_1 = mx_2 + c$$

or

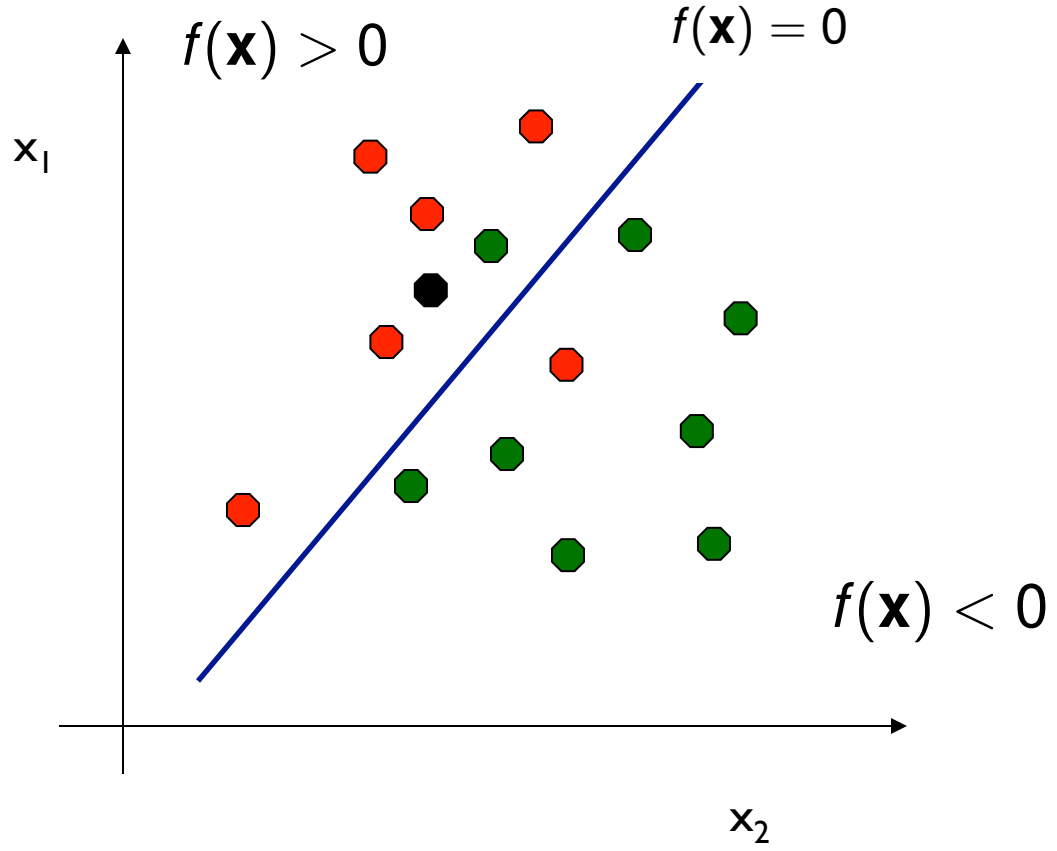
$$x_1 - mx_2 - c = 0$$

We can rewrite as:

$$f(x) = x_1 - mx_2 - c = 0$$

$$f(\mathbf{x} = [1, -m, -c]^T [x_1, x_2, 1]$$

Note: We added an extra "1" into the feature vector. Without this, we can not represent "all" lines. (lines that do not pass through origin). Parameter –c corresponding to the extra "1" is also called bias.



Summary-2

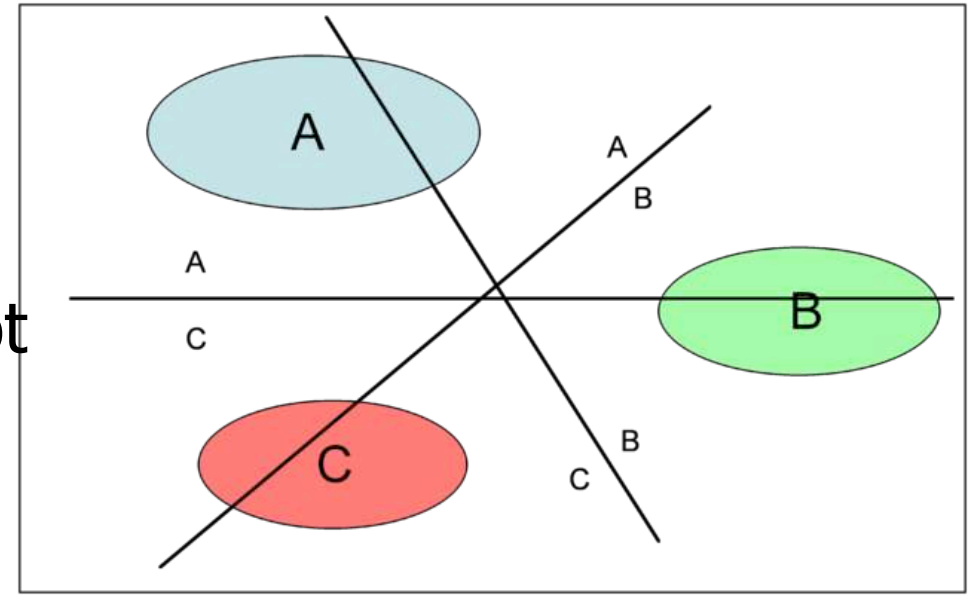
- Classifier is represented as $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$
- There are d features (really d-1, since x has an augmented "1"). We have many such samples.
- There are d parameter in **w** also. We call **w** as the model. Finding **w** is often called learning.

Summary-2

- Linear Classifiers are efficient at "test" time (unlike KNN).
  - No need to carry samples. Just a dot product.
- We do not know how to find the **w**
  - Finding **w** is learning. Please wait for sometime.
- We know how to test, given **w**: Evaluate $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$
  - Classify as "1" if $f(\mathbf{x})>0$ else classify as "0".
- (In fact f(x) can also be nonlinear. But training is harder.)

Extending to Multiclass Classification

- What if there are M classes? (Say m=3)
  - Linear multi-class classification is not trivial.
- A simple solution:
  - For every test sample,
  - Compute all pairwise classifiers ($_mC_2$ =3)
  - Find the majority label being predicted.
    - Say, if at least 2 of 3 says, "A", label as "A".



Summary-3

- Linear Classifier:
  - Popular. Useful in many cases.
  - Many Algorithms to train (Please wait).
- Easy to visualize as:
  - Line in 2D, Plane in 3D, and Hyperplane in nD
  - General form $\mathbf{w}^T\mathbf{x}$
- Multiclass extensions exists.
  - But primarily binary.

Are All Features Equally Important?

No!! Never.

Selecting Features as Matrix Multiplication

Selecting first and third feature

$$\begin{bmatrix} x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Selecting first and fourth feature

$$\begin{bmatrix} x_1 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$\mathbf{x}' = \mathbf{A}\mathbf{x}$$

A "new" set of features are selected/extracted from the original one by a matrix multiplication.

Rows of **A** decide what the new features are. (They need not be 0 and 1.)

Often #rows of A is smaller than #columns of A. This is also called **dimensionality reduction**.

Feature Selection and Extraction

- Selection:
  - Select some features out of a pool. (Simple A with 0/1.)
- Extraction:
  - Extract a set of new features. (elements of A need not be 0/1)
- Extraction is often required:
  - To visualize in 2D/3D.
  - To remove some “useless” or “less useful” features.
  - Make computations efficient. (Note: original data could be 1000s of dimension!!)

Feature Extraction with PCA

- Principal Component Analysis
  - More in a later lecture.
- When rows of **A** are principal "eigen vectors" of a covariance matrix, we get a set of small "useful" features.
  - Very popular.

Summary-4

- Project the original features to a lower dimensional space.
  - Model this as "Matrix Multiplication"
- Many feature extraction schemes that give "useful" features.
  - Eg. PCA
- New extracted features may have lesser interpretation (semantics). But could be useful for the classification.

How do we measure the performance?

Everybody has their own requirements!!

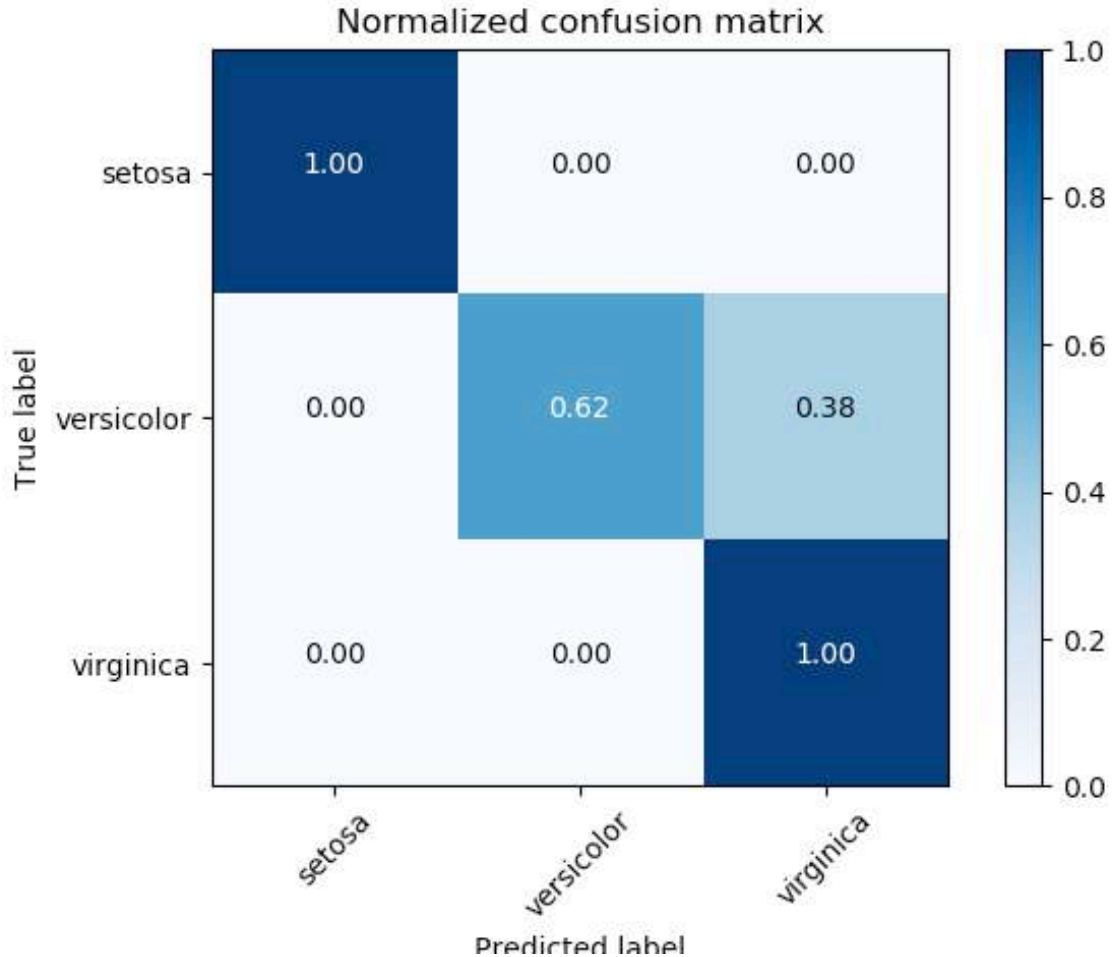
Accuracy

- Simple intuitive measure:

$$\frac{\text{NumCorrectlyClassifiedTestSamples}}{\text{NumTotalTestSamples}}$$

- Scalar. A positive quantity less than or 1.0
  - Often expressed as percentage (multiplied by 100).

Confusion Matrix



A specific case (Binary)

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Accuracy = (100+50)/165 = 0.91
Misclassification = (10+5)/165 = 0.09

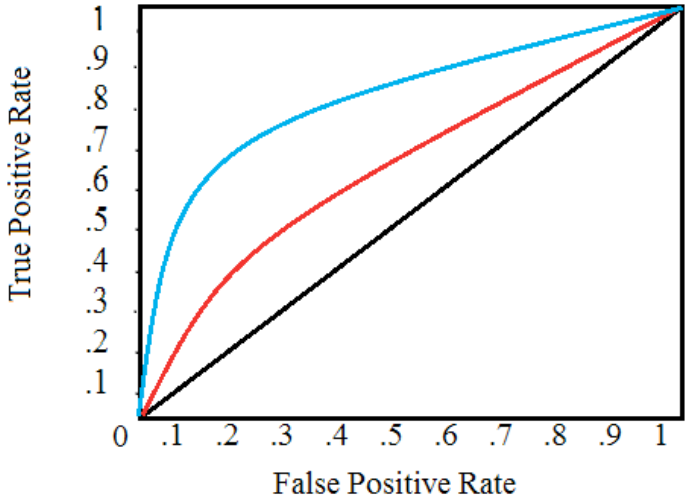
True Positive Rate (TP): 100/105 = 0.95
False Positive Rate (FP): 10/60 = 0.17

True Negative Rate (TN): 50/60 = 0.833
False Negative Rate (FN): 5/105 = 0.048

When you do a cancer screening, what do you care?
High TP
When you classify between "apple" and "orange?
High Accuracy or High TP and High TN
Automatic firing on detecting a violation.
Very Low FP

Trade Off

- TRUE if $\mathbf{w^T x} > \theta$ or $\mathbf{w^T x} - \theta > 0$; Else FALSE
- If θ is very low, everything is TRUE.
- If θ is very high, everything is FALSE.



If you need to compare two screening tests, we should look at the "ROC" (Receiver Operating Characteristics). The higher the curve, the better.

Problem of Retrieval

- You give a query **q**
- You get a ranked list of documents (say 10)
  - d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}
- The document d_i could be "relevant" (+) or "irrelavent"(-)
- Let us assume the relevances are:
  - +, +, -, -, +, -, +, -, -, +
- How do we evaluate the "quality"/performance?

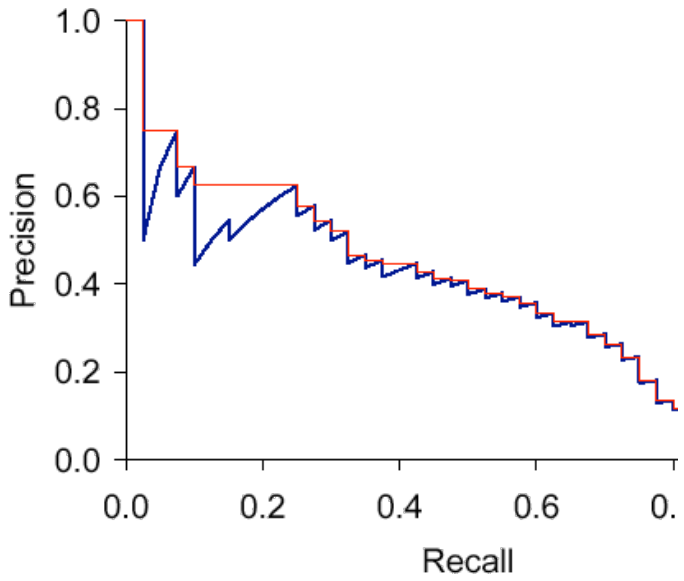
Precision and Recall

- Precision = Percentage of the retrieved documents that are relevant.
- Recall = Percentage of the relevant documents that we are able to retrieve.
- Precision = TP/(TP+FP)
- Recall = TP/(TP+FN)
- (We do not know FN often in “google”s).

Precision and Recall

- Assume there were 10 "True"/"Relevant" documents (often we do not know this)

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
	+	+	-	-	+	-	+	-	-	+
P@K	1.0	1.0	0.66	0.50	0.60	0.50	0.57	0.50	0.44	0.50
R@K	0.1	0.2	0.2	0.2	0.3	0.3	0.4	0.4	0.4	0.5



Area under the Precision-Recall Curve is a very popular measure called "Average Precision" (AP)
And mean over multiple queries mAP.

Summary-5

- Many metrics:
  - Accuracy, TP, FP, AUC, Precision, Recall, AP/mAP
  - Also terms also. Read wikipedia pages.
- Many problems demand many measures.
  - Choice of right measure is very important.
- Confusion Matrix: Important to analyze and refine soln.
- Curves provide "Trade off" and help to choose operating point.

Part-II: Probabilistic View of Classification

A Toy Problem

- You are captured by the Sentinelese tribe while on your excursion to the islands. You are brought to the chieftian for prosecution. You are blindfolded and the chief selects a fruits from a basket containing 85 green mangoes, 5 yellow mangoes, 2 green pears and 8 yellow pears.
- *If you guess the fruit correctly, you are set free. If not .. :-(*
- What is your guess?

What is your guess?

- Probability of it being mango
  - P(mango)= 90/100 = 0.90
- Probability of being pear
  - P(pear) = 10/100 = 0.10
- Take the safe bet:
  - It is "**mango**"

An Evidence

- You get a glimpse through the blindfold and you see a slight yellow colour in the chief’s hand.
  - What is your guess?

Help from Bayes

$$P(mango|yellow) = \frac{p(yellow|mango) \cdot P(mango)}{P(yellow)}$$

$$= \frac{\frac{5}{90} \cdot 0.90}{\frac{13}{100}} = 0.385$$

$$P(pear|yellow) = \frac{p(yellow|pear) \cdot P(pear)}{P(yellow)}$$

$$= \frac{\frac{8}{10} \cdot 0.10}{\frac{13}{100}} = 0.615$$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

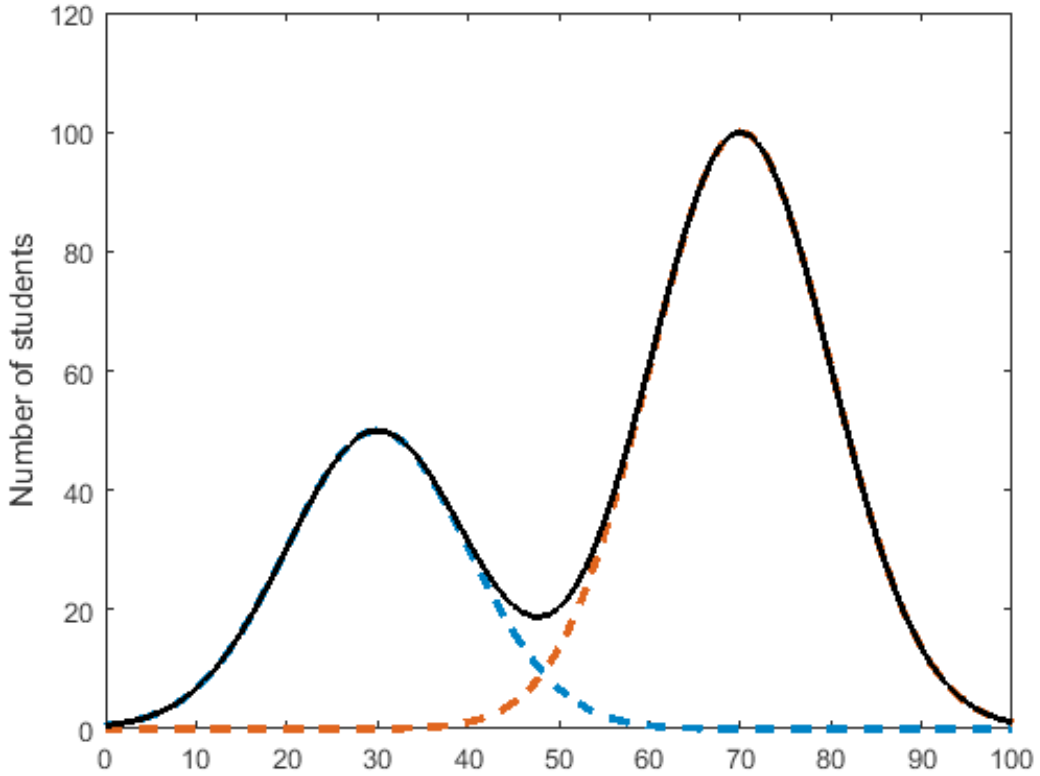
What is your guess now?

Summary-6

- Bayes theorem allows to convert
  - Belief (prior probability) to posterior probability (truth) with the help of evidence.
- Maximum A Posteriori (MAP) Classification is fundamental.
- If we assume that prior probabilities are equal, we get Maximum Likelihood (MLE) Classifier.
- Note: In the previous example, denominator did not contribute to the decision.

Optimal Bayesian Classifier

Adults and kids form a normal distribution with mean 70 and 30 respectively.



In a group of adults and kids, given the height x(inch), find whether the person is adult of kid?

Decide adult if P(adult|x) > P(kid|x) else kid

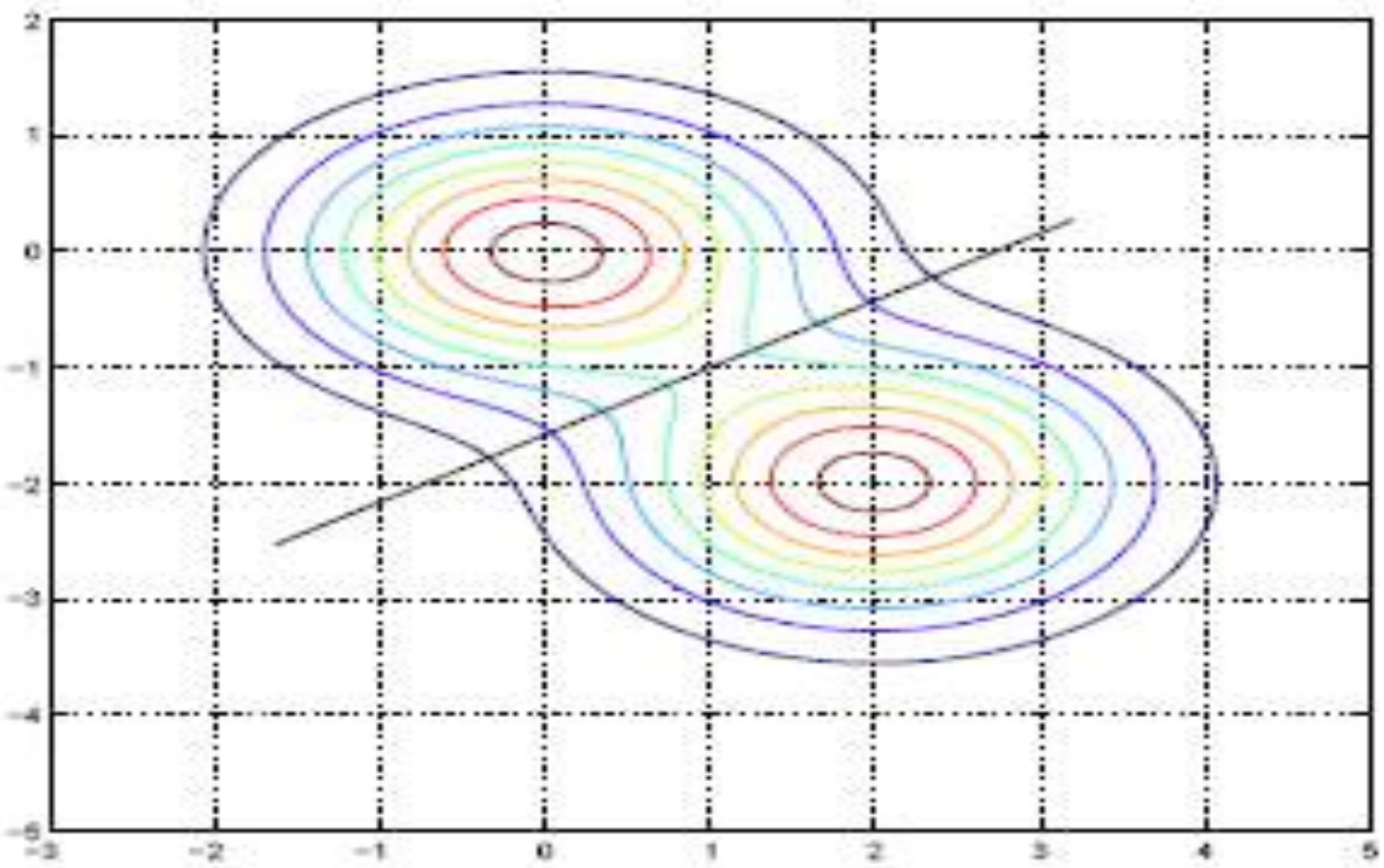
P(adult|x) = P(kid|x) when x = 50

This is the best that one can achieve.

However,

Estimating these probabilities is difficult. Also one assumes Gaussian distribution.

Optimal Bayesian Classifier in 2D is Linear*



*Conditions applly.

Naïve Bayes Classifier

- *Naive Bayes* classifier is a classification algorithms based on Bayes' Theorem.
- Naive Bayes makes the assumption that each feature is conditionally independent of the others.
- For a given target value (class), the distribution of each feature is independent of the other predictors.

Naïve Bayes Assumption

$$P(\mathbf{x}|class = 1) =$$

$$P(x_1|class = 1) \times P(x_2|class = 1) \ldots \times \ldots \times P(x_d|class = 1)$$

Example: Classify “a very close game”

Text	Category
“A great game”	Sports
“The Election was over”	Not Sports
“Very Clean Match”	Sports
“A clean but forgettable game”	Sports
“It was close election”	Not Sports

P(Sports) = 3/5 = 0.6
P(Not Sports) = 2/5 = 0.4

Naïve Assumption: Words are independent.

There are 11 words in “sports” and 9 words in “Not Sports” and all together 14 words.

P(game|Sports) = 2/11

P(a very close game|sports) =
P(a|sports)XP(very|sports)X
P(close|Sports)XP(game|Sports)

Example (Cont.)

word	P(word|Sport)	P(word| NotSport)
A	(2+1)/(11+14)	(1+1)/(9+14)
Very	(1+1)/(11+14)	(0+1)/(9+14)
Close	(0+1)/(11+14)	(1+1)/(9+14)
Game	(2+1)/(11+14)	(0+1)/(9+14)

Smoothing (Laplace Smoothing):

Some times, count can be zero leading to zero probabilities. Product of probabilities will also become zero.

To avoid this:

- Add 1 to the numerator always (so it is never zero)
- Add total words to the denominator also.

Example (Cont.)

P(a|Sports) X P(Very|Sports) X P(Close |Sports) X P(game|Sports) X P(Sports)
= 2.76 X 10^{-5}

P(a | Not Sports) X P(Very| Not Sports) X P(Close |Not Sports) X P(game|Not Sports)
X P(Not Sports) **= 0.572 X 10^{-5}**

"A Very Close Game" is classified as "Sports" !!

Summary-7

- Bayes theorem provides an optimal classifier.
  - Theoretically useful. Practically hard to implement.
- Naïve Bayes classifier provides an efficient and scalable classifier.
- Linear Classifiers are optimal under certain conditions.
  - (Eg. Gaussian; Equal covariance's.)
- We know now how to “build” our second classifier. (first KNN)!!

Thanks!!. Questions?

Data Capture

- Need to classify an object as:
  - Dangerous vs. Harmless
- Capture should contain required information
- Will also contain useless information
- Options:
  1. Capture only relevant information
  2. Extract relevant information from raw data