CS 397. Naïve Bayes

Today’s goal

- Background of Classifiers
- Probability Basics
- Probabilistic Classification
- Naïve Bayes
- Example: Play Tennis
- Relevant Issues
- Conclusions

Background

- There are three methods to establish a classifier

  a) Model a classification rule directly

  Examples: k-NN, decision trees, perceptron, SVM

  b) Model the probability of class memberships given input data

  Example: multi-layered perceptron with the cross-entropy cost

  c) Make a probabilistic model of data within each class

  Examples: naive Bayes, model based classifiers

- *a*) and *b*) are examples of discriminative classification
- *c*) is an example of generative classification
- *b*) and *c*) are both examples of probabilistic classification

Probability Basics

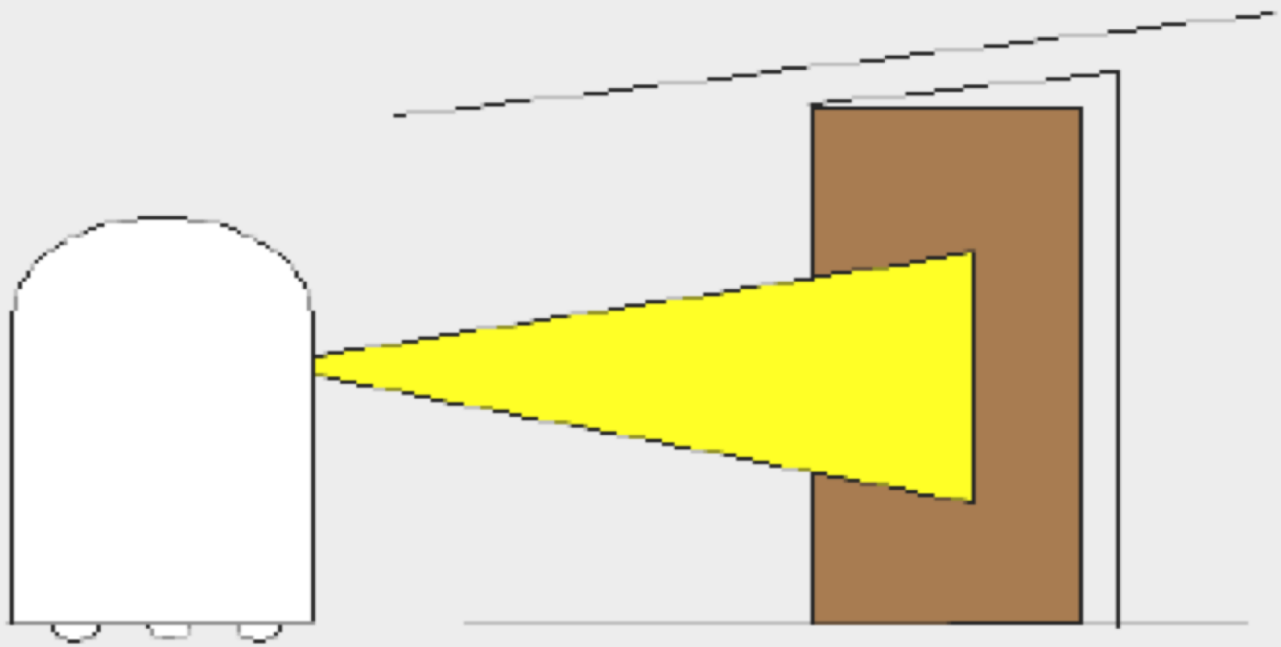
- Prior, conditional and joint probability
  - Prior probability: $P(X)$
  - Conditional probability: $P(X_1 \mid X_2), P(X_2 \mid X_1)$
  - Joint probability: $\mathbf{X} = (X_1, X_2), P(\mathbf{X}) = P(X_1, X_2)$
  - Relationship: $P(X_1, X_2) = P(X_2 \mid X_1)P(X_1) = P(X_1 \mid X_2)P(X_2)$
  - Independence: $P(X_2 \mid X_1) = P(X_2), P(X_1 \mid X_2) = P(X_1), P(X_1, X_2) = P(X_1)P(X_2)$
- Bayesian Rule

$$P(C \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C)P(C)}{P(\mathbf{X})}$$

$$Posterior = \frac{Likelihood \times Prior}{Evidence}$$

Simple Example of State Estimation

- Suppose a robot obtains measurement z
- What is *P(doorOpen|z)*?



Causal vs. Diagnostic Reasoning

- *P(open|z)* is diagnostic.
- *P(z|open)* is causal.
- Often causal knowledge is easier to obtain.

count frequencies!

- Bayes rule allows us to use causal knowledge:

$$P(open \mid z) = \frac{P(z \mid open)P(open)}{P(z)}$$

Example

- $P(z|open) = 0.6 \quad P(z|\neg open) = 0.3$
- $P(open) = P(\neg open) = 0.5$

$$P(open \mid z) = \frac{P(z \mid open)P(open)}{P(z \mid open)p(open) + P(z \mid \neg open)p(\neg open)}$$

$$P(open \mid z) = \frac{0.6 \cdot 0.5}{0.6 \cdot 0.5 + 0.3 \cdot 0.5} = \frac{2}{3} = 0.67$$

- z raises the probability that the door is open.

Probabilistic Classification

- Establishing a probabilistic model for classification
  - Discriminative model

$$P(C \mid \mathbf{X}) \quad C = c_1, \cdots, c_L, \mathbf{X} = (X_1, \cdots, X_n)$$

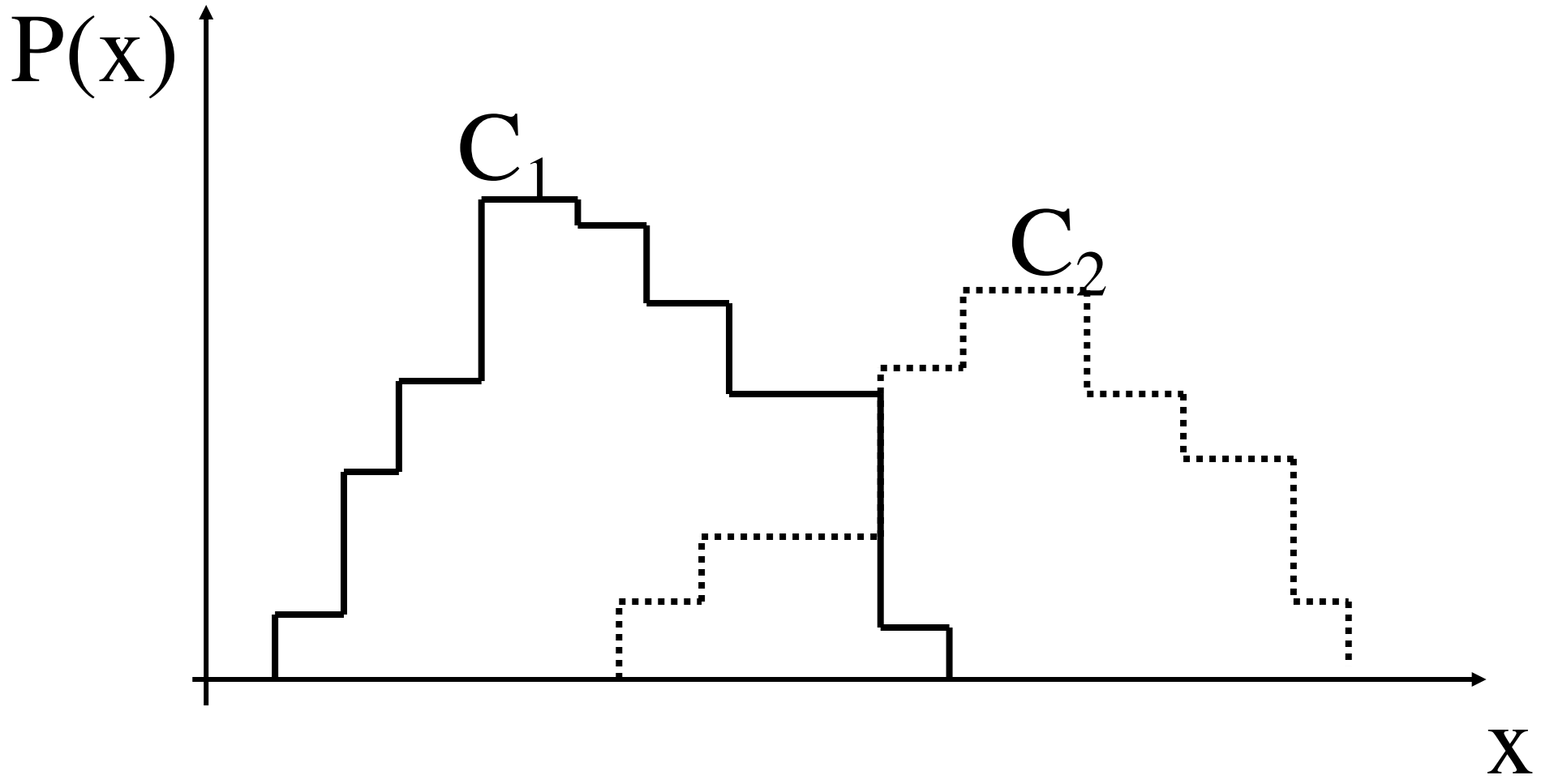
  - Generative model

$$P(\mathbf{X} \mid C) \quad C = c_1, \cdots, c_L, \mathbf{X} = (X_1, \cdots, X_n)$$

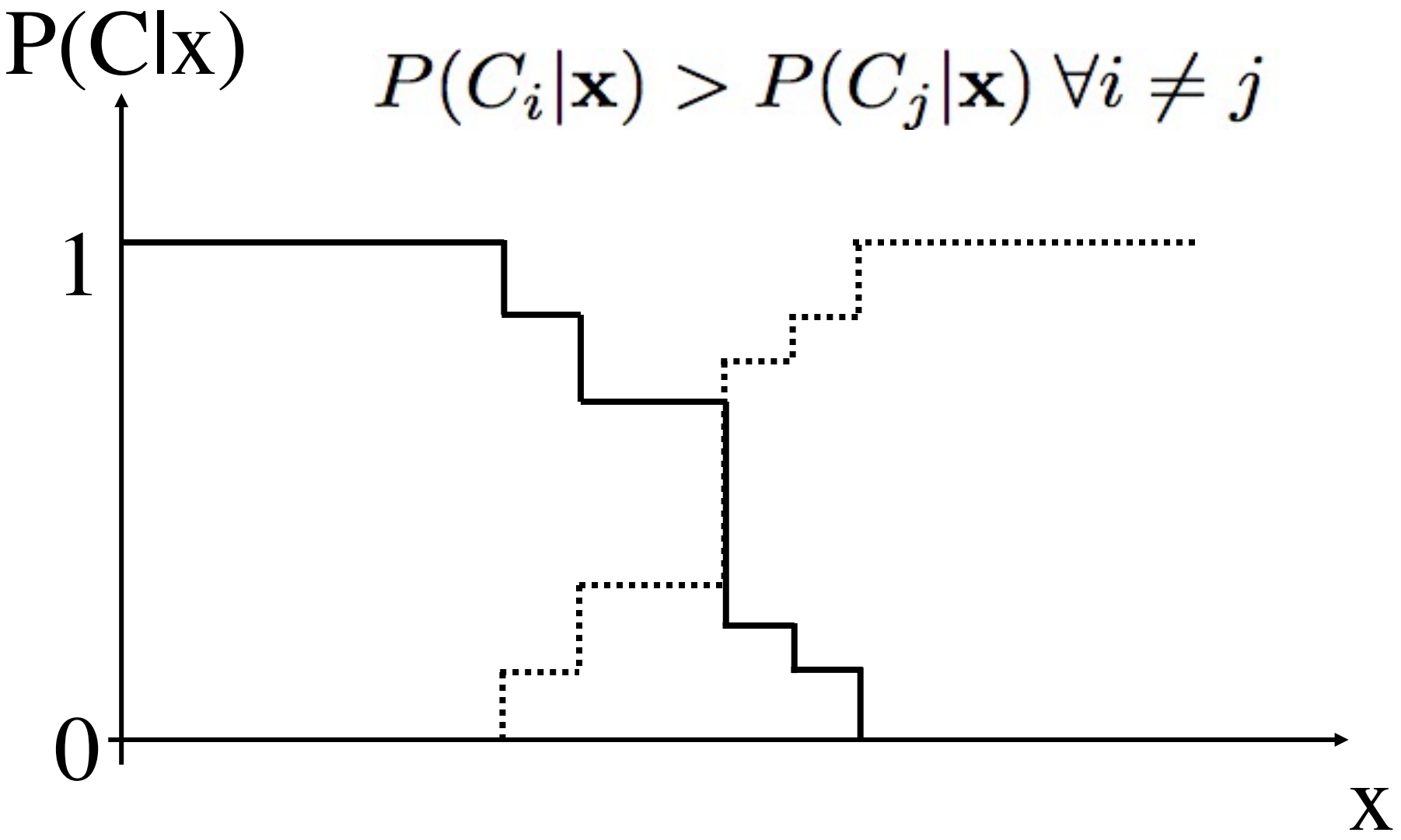
- MAP classification rule
  - MAP: Maximum A Posterior
  - Assign x to c^* if $P(C = c^* \mid \mathbf{X} = \mathbf{x}) > P(C = c \mid \mathbf{X} = \mathbf{x}) \quad c \neq c^*, \; c = c_1, \cdots, c_L$
- Generative classification with the MAP rule
  - Apply Bayesian rule to convert:

$$P(C \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C)P(C)}{P(\mathbf{X})} \propto P(\mathbf{X} \mid C)P(C)$$

Feature Histograms



Posterior Probability



Naïve Bayes

- Bayes classification

$$P(C \mid \mathbf{X}) \propto P(\mathbf{X} \mid C)P(C) = P(X_1, \cdots, X_n \mid C)P(C)$$

Difficulty: learning the joint probability $P(X_1, \cdots, X_n \mid C)$

- Naïve Bayes classification
  - Making the assumption that all input attributes are independent

$$\begin{aligned} P(X_1, X_2, \cdots, X_n \mid C) &= P(X_1 \mid X_2, \cdots, X_n; C)P(X_2, \cdots, X_n \mid C) \\ &= P(X_1 \mid C)P(X_2, \cdots, X_n \mid C) \\ &= P(X_1 \mid C)P(X_2 \mid C) \cdots P(X_n \mid C) \end{aligned}$$

  - MAP classification rule

$$[P(x_1 \mid c^*) \cdots P(x_n \mid c^*)]P(c^*) > [P(x_1 \mid c) \cdots P(x_n \mid c)]P(c), \quad c \neq c^*, c = c_1, \cdots, c_L$$

Naïve Bayes

- Naïve Bayes Algorithm (for discrete input attributes)
  - Learning Phase: Given a training set **S**,

For each target value of c_i $(c_i = c_1, \cdots, c_L)$

$\hat{P}(C = c_i) \leftarrow$ estimate $P(C = c_i)$ with examples in **S**;

For every attribute value a_{jk} of each attribute x_j $(j = 1, \cdots, n; k = 1, \cdots, N_j)$

$\hat{P}(X_j = a_{jk} \mid C = c_i) \leftarrow$ estimate $P(X_j = a_{jk} \mid C = c_i)$ with examples in **S**;

Output: conditional probability tables; for x_j, $N_j \times L$ elements

  - Test Phase: Given an unknown instance $\mathbf{X}' = (a_1', \cdots, a_n')$,

Look up tables to assign the label c^* to **X'** if

$$[\hat{P}(a_1' \mid c^*) \cdots \hat{P}(a_n' \mid c^*)]\hat{P}(c^*) > [\hat{P}(a_1' \mid c) \cdots \hat{P}(a_n' \mid c)]\hat{P}(c), \quad c \neq c^*, c = c_1, \cdots, c_L$$

Example

- Example: Play Tennis

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example

- Learning Phase

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=*No*
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

$P(\text{Play}=Yes) = 9/14$ $P(\text{Play}=No) = 5/14$

Example

- Test Phase
  - Given a new instance,

    $\mathbf{x}'$=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)
  - Look up tables

    P(Outlook=*Sunny*|Play=*Yes*) = 2/9

    P(Temperature=*Cool*|Play=*Yes*) = 3/9

    P(Huminity=*High*|Play=*Yes*) = 3/9

    P(Wind=*Strong*|Play=*Yes*) = 3/9

    P(Play=*Yes*) = 9/14

    P(Outlook=S*unny*|Play=*No*) = 3/5

    P(Temperature=*Cool*|Play==*No*) = 1/5

    P(Huminity=*High*|Play=*No*) = 4/5

    P(Wind=*Strong*|Play=*No*) = 3/5

    P(Play=*No*) = 5/14
  - MAP rule

    P(*Yes*|$\mathbf{x}'$): [P(*Sunny*|Yes)P(*Cool*|*Yes*)P(*High*|Yes)P(*Strong*|*Yes*)]P(Play=*Yes*) = 0.0053

    P(*No*|$\mathbf{x}'$): [P(*Sunny*|No) P(*Cool*|No)P(*High*|*No*)P(*Strong*|*No*)]P(Play=*No*) = 0.0206

    Given the fact P(*Yes*|$\mathbf{x}'$) < P(*No*|$\mathbf{x}'$), we label $\mathbf{x}'$ to be "*No*".

Relevant Issues

- Violation of Independence Assumption
  - For many real world tasks, $P(X_1,\cdots,X_n \mid C) \neq P(X_1 \mid C)\cdots P(X_n \mid C)$
  - Nevertheless, naïve Bayes works surprisingly well anyway!
- Zero conditional probability Problem
  - If no example contains the attribute value $X_j = a_{jk}$, $\hat{P}(X_j = a_{jk} \mid C = c_i) = 0$
  - In this circumstance, $\hat{P}(x_1 \mid c_i)\cdots\hat{P}(a_{jk} \mid c_i)\cdots\hat{P}(x_n \mid c_i) = 0$ during test
  - For a remedy, conditional probabilities estimated with

$$\hat{P}(X_j = a_{jk} \mid C = c_i) = \frac{n_c + mp}{n + m}$$

n_c : number of training examples for which $X_j = a_{jk}$ and $C = c_i$

n : number of training examples for which $C = c_i$

p : prior estimate (usually, $p = 1/t$ for t possible values of X_j)

m : weight to prior (number of "virtual" examples, $m \geq 1$)

Relevant Issues

- Continuous-valued Input Attributes
  - Numberless values for an attribute
  - Conditional probability modeled with the normal distribution

$$\hat{P}(X_j \mid C = c_i) = \frac{1}{\sqrt{2\pi}\sigma_{ji}} \exp\left(-\frac{(X_j - \mu_{ji})^2}{2\sigma_{ji}^2}\right)$$

μ_{ji} : mean (avearage) of attribute values X_j of examples for which $C = c_i$

σ_{ji} : standard deviation of attribute values X_j of examples for which $C = c_i$

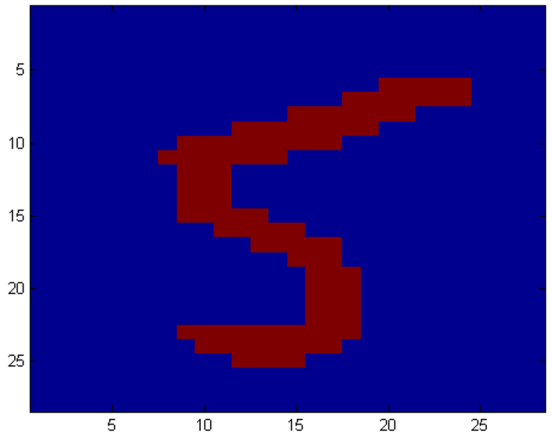
  - Learning Phase: for $\mathbf{X} = (X_1, \cdots, X_n)$, $C = c_1, \cdots, c_L$
    Output: $n \times L$ normal distributions and $P(C = c_i)$ $i = 1, \cdots, L$
  - Test Phase: for $\mathbf{X}' = (X_1', \cdots, X_n')$
    - Calculate conditional probabilities with all the normal distributions
    - Apply the MAP rule to make a decision

Conclusions

- Naïve Bayes based on the independence assumption
  - Training is very easy and fast; just requiring considering each attribute in each class separately
  - Test is straightforward; just looking up tables or calculating conditional probabilities with normal distributions
- A popular generative model
  - Performance competitive to most of state-of-the-art classifiers even in presence of violating independence assumption
  - Many successful applications, e.g., spam mail filtering
  - Apart from classification, naïve Bayes can do more…

Applications

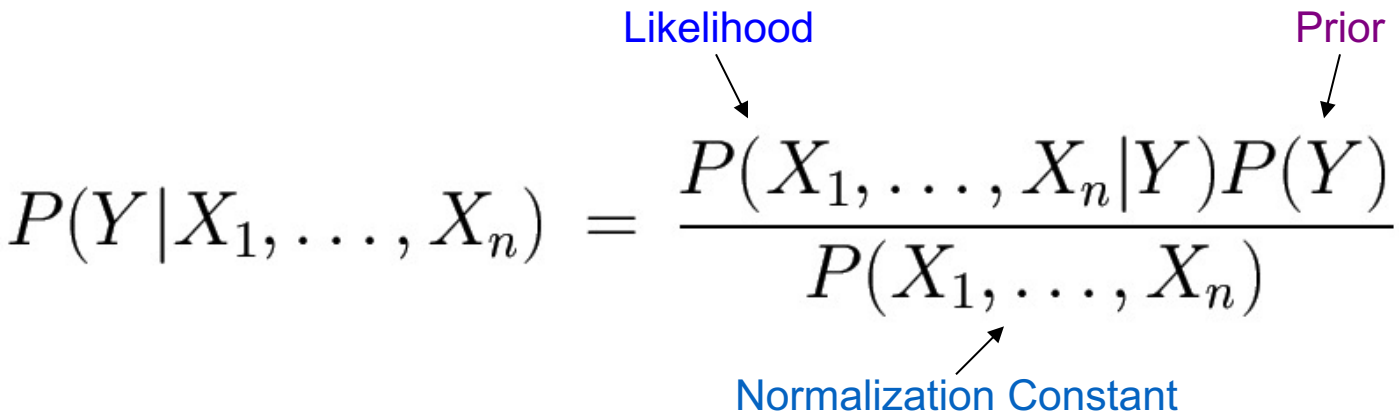
- **Digit Recognition**



5

- **$X_1,\ldots,X_n \in \{0,1\}$ (Black vs. White pixels)**
- **$Y \in \{5,6\}$ (predict whether a digit is a 5 or a 6)**

The Bayes Classifier



- Why did this help? Well, we think that we might be able to specify how features are "generated" by the class label

The Bayes Classifier

- Let's expand this for our digit recognition task:

$$P(Y=5|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y=5)P(Y=5)}{P(X_1,\ldots,X_n|Y=5)P(Y=5)+P(X_1,\ldots,X_n|Y=6)P(Y=6)}$$

$$P(Y=6|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y=6)P(Y=6)}{P(X_1,\ldots,X_n|Y=5)P(Y=5)+P(X_1,\ldots,X_n|Y=6)P(Y=6)}$$

- To classify, we'll simply compute these two probabilities and predict based on which one is greater

Model Parameters

- For the Bayes classifier, we need to “learn” two functions, the likelihood and the prior

- How many parameters are required to specify the prior for our digit recognition example? (Supposing that each image is 30x30 pixels)

Model Parameters

- The problem with explicitly modeling $P(X_1,\ldots,X_n|Y)$ is that there are usually way too many parameters:
  - We'll run out of space
  - We'll run out of time
  - And we'll need tons of training data (which is usually not available)

The Naïve Bayes Model

- The *Naïve Bayes Assumption*: Assume that all features are *independent* **given the class label Y**
- Equationally speaking:

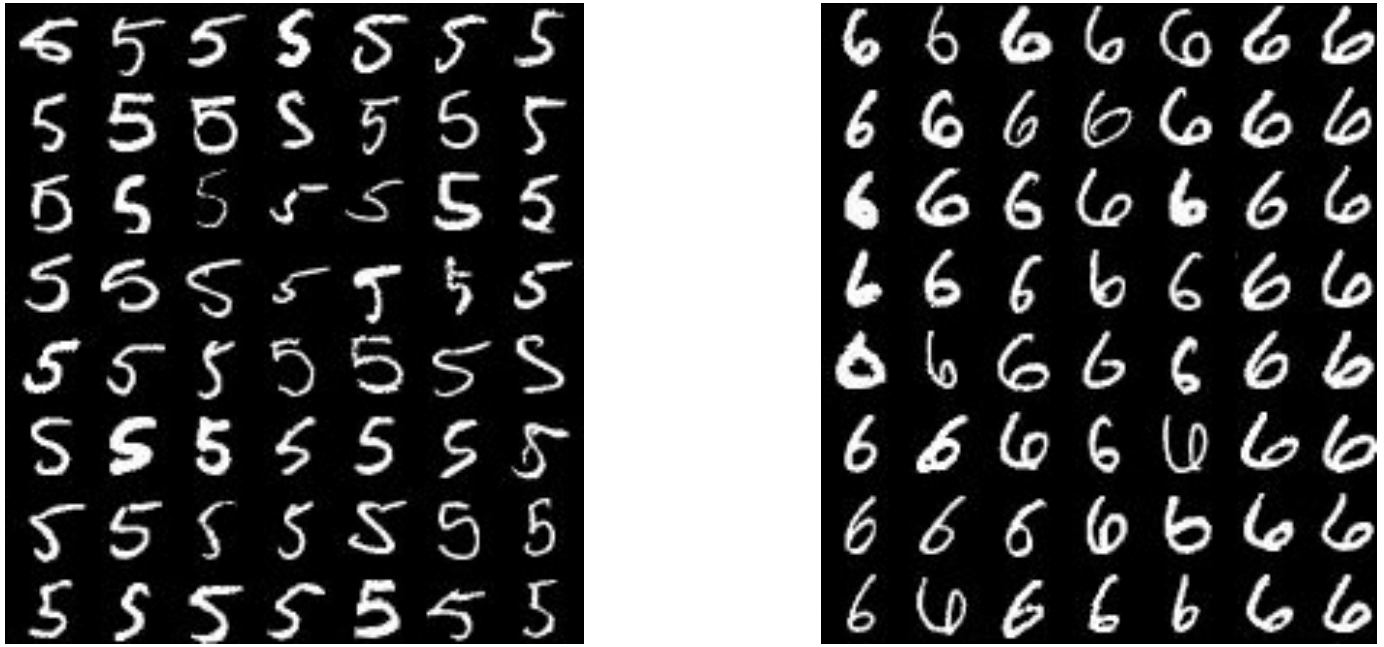
$$P(X_1, \ldots, X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$$

Why is this useful?

- # of parameters for modeling $P(X_1,\ldots,X_n|Y)$:
  - $2(2^n-1)$
- # of parameters for modeling $P(X_1|Y),\ldots,P(X_n|Y)$
  - $2n$

Naïve Bayes Training

- Now that we've decided to use a Naïve Bayes classifier, we need to train it with some data:



MNIST Training Data

Naïve Bayes Training

- Training in Naïve Bayes is **easy**:
  - Estimate P(Y=v) as the fraction of records with Y=v

$$P(Y = v) = \frac{Count(Y = v)}{\#\ records}$$

  - Estimate $P(X_i=u|Y=v)$ as the fraction of records with Y=v for which $X_i=u$

$$P(X_i = u|Y = v) = \frac{Count(X_i = u \wedge Y = v)}{Count(Y = v)}$$

- (This corresponds to Maximum Likelihood estimation of model parameters)

Naïve Bayes Training

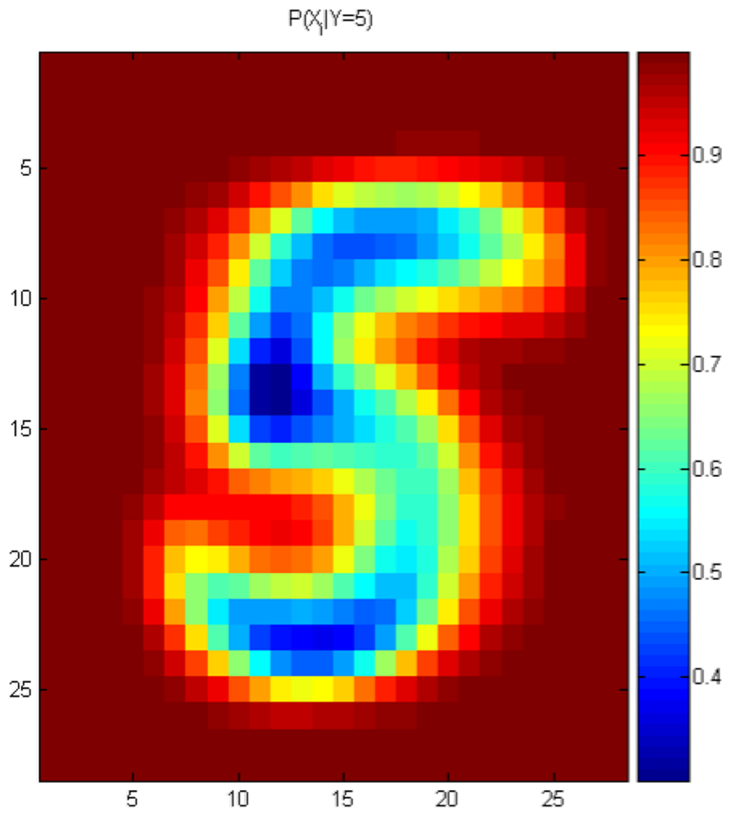
- In practice, some of these counts can be zero
- Fix this by adding “virtual” counts:

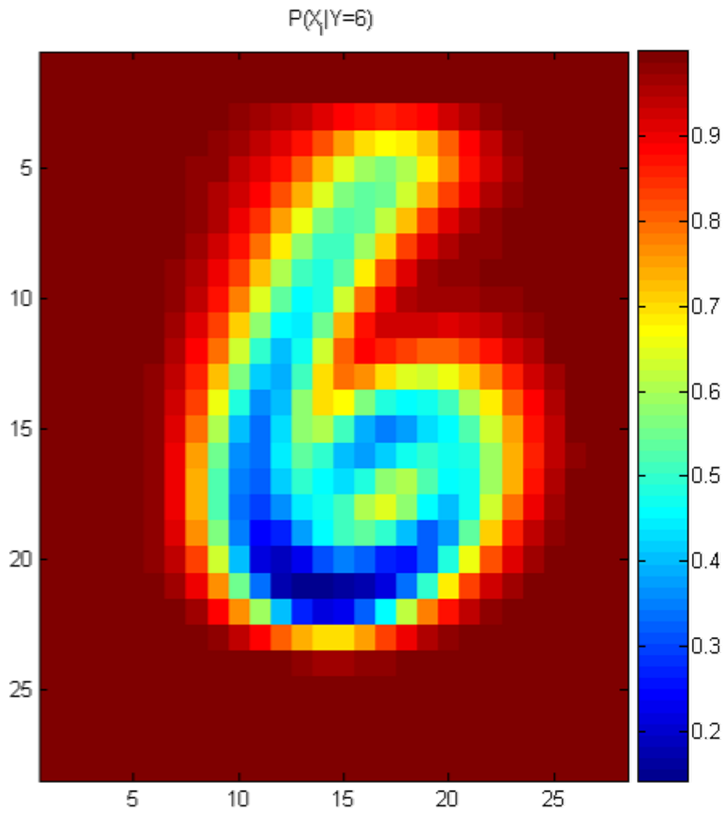
$$P(X_i = u | Y = v) = \frac{Count(X_i = u \wedge Y = v) + 1}{Count(Y = v) + 2}$$

  - (This is like putting a prior on parameters and doing MAP estimation instead of MLE)
  - This is called *Smoothing*

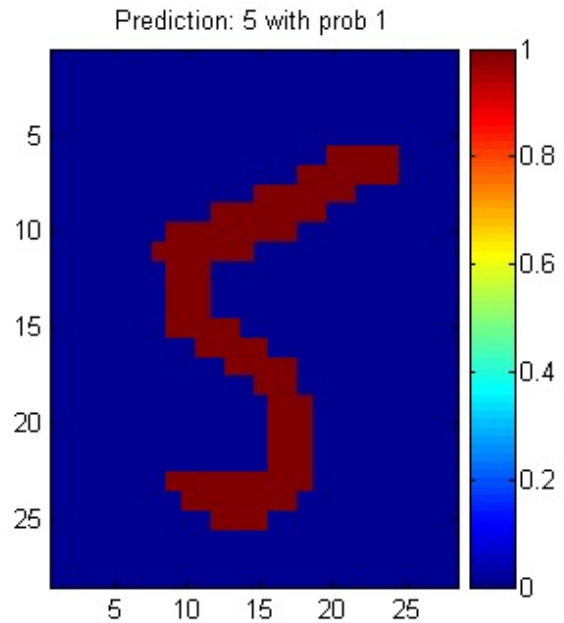
Naïve Bayes Training

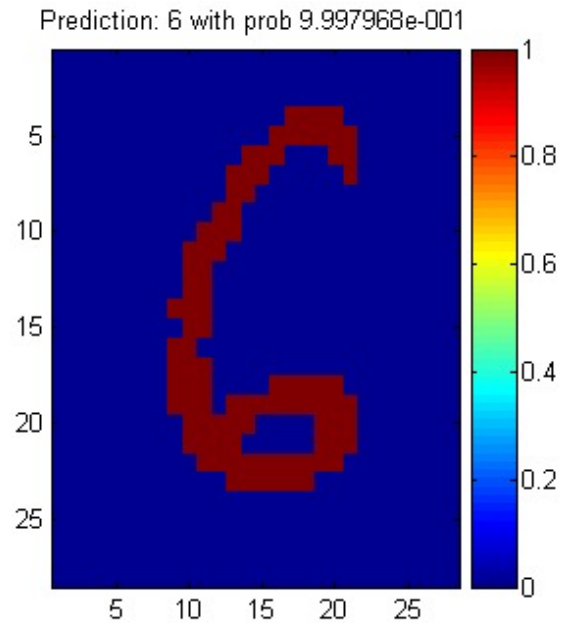
- For binary digits, training amounts to averaging all of the training fives together and all of the training sixes together.

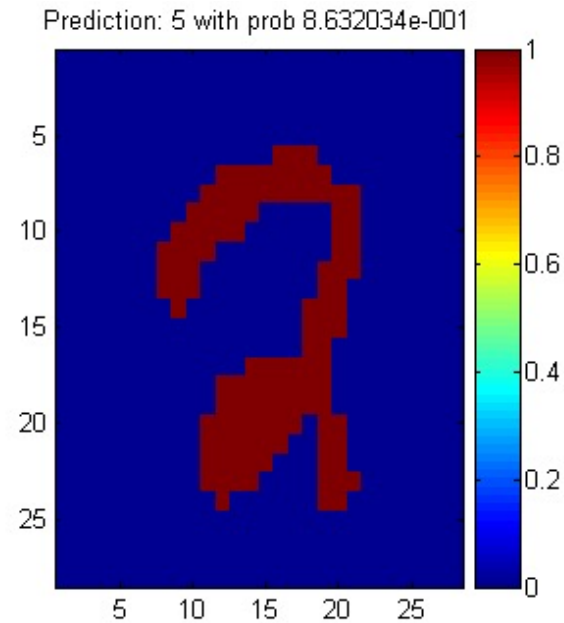




Naïve Bayes Classification







Evaluating classification algorithms

I tell you that it achieved 95% accuracy on my data.

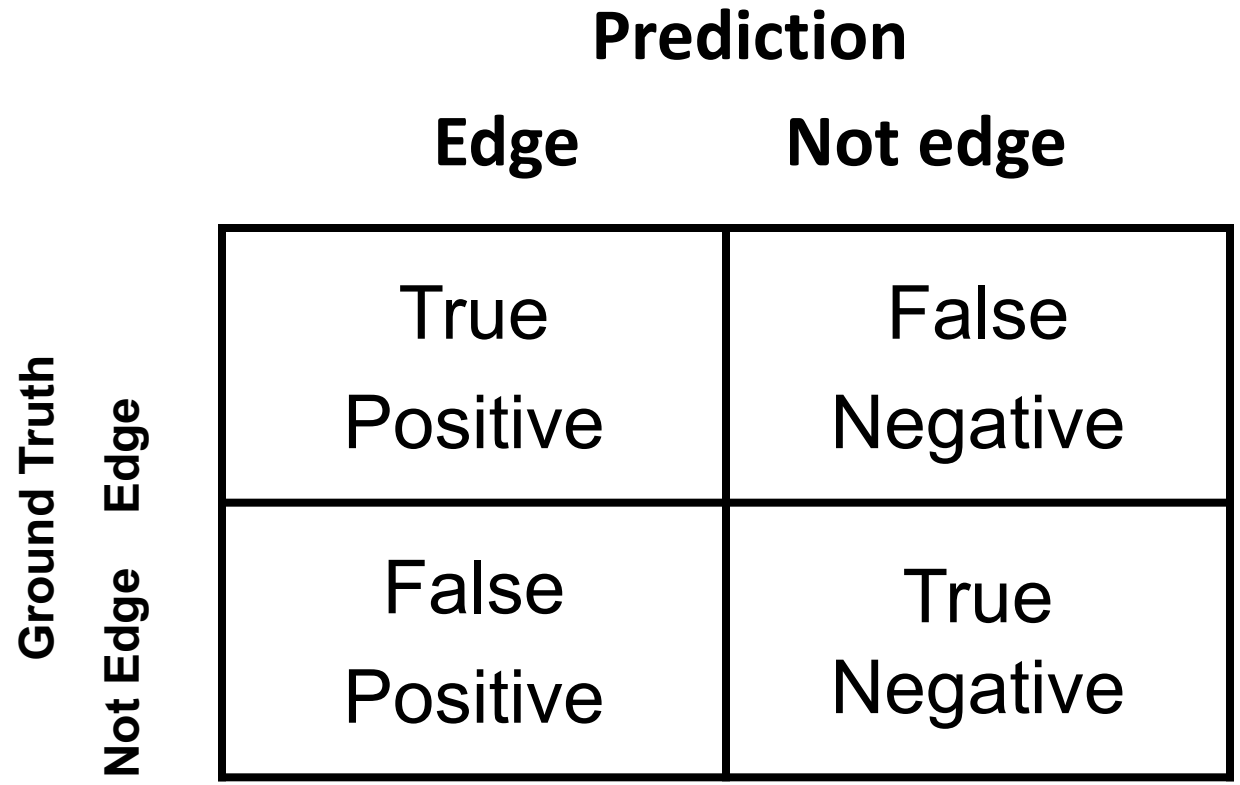
Is your technique a success?

Types of errors

- But suppose that
  - The 95% is the correctly classified pixels
  - Only 5% of the pixels are actually edges
  - It misses all the edge pixels

- How do we count the effect of different types of error?

Types of errors



Two parts to each: whether you got it correct or not, and what you guessed. For example for a particular pixel, our guess might be labelled…

True Positive

Did we get it correct?
True, we did get it correct.

What did we say?
We said ‘positive’, i.e. edge.

or maybe it was labelled as one of the others, maybe…

False Negative

Did we get it correct?
False, we did not get it correct.

What did we say?
We said ‘negative, i.e. not edge.

Sensitivity and Specificity

Count up the total number of each label (TP, FP, TN, FN) over a large dataset. In ROC analysis, we use two statistics:

$$\textbf{Sensitivity} = \frac{TP}{TP+FN}$$

Can be thought of as the likelihood of spotting a positive case when presented with one.

Or… the proportion of edges we find.

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

Can be thought of as the likelihood of spotting a negative case when presented with one.

Or… the proportion of non-edges that we find

$$\text{Sensitivity} = \frac{TP}{TP+FN} = ? \qquad \text{Specificity} = \frac{TN}{TN+FP} = ?$$

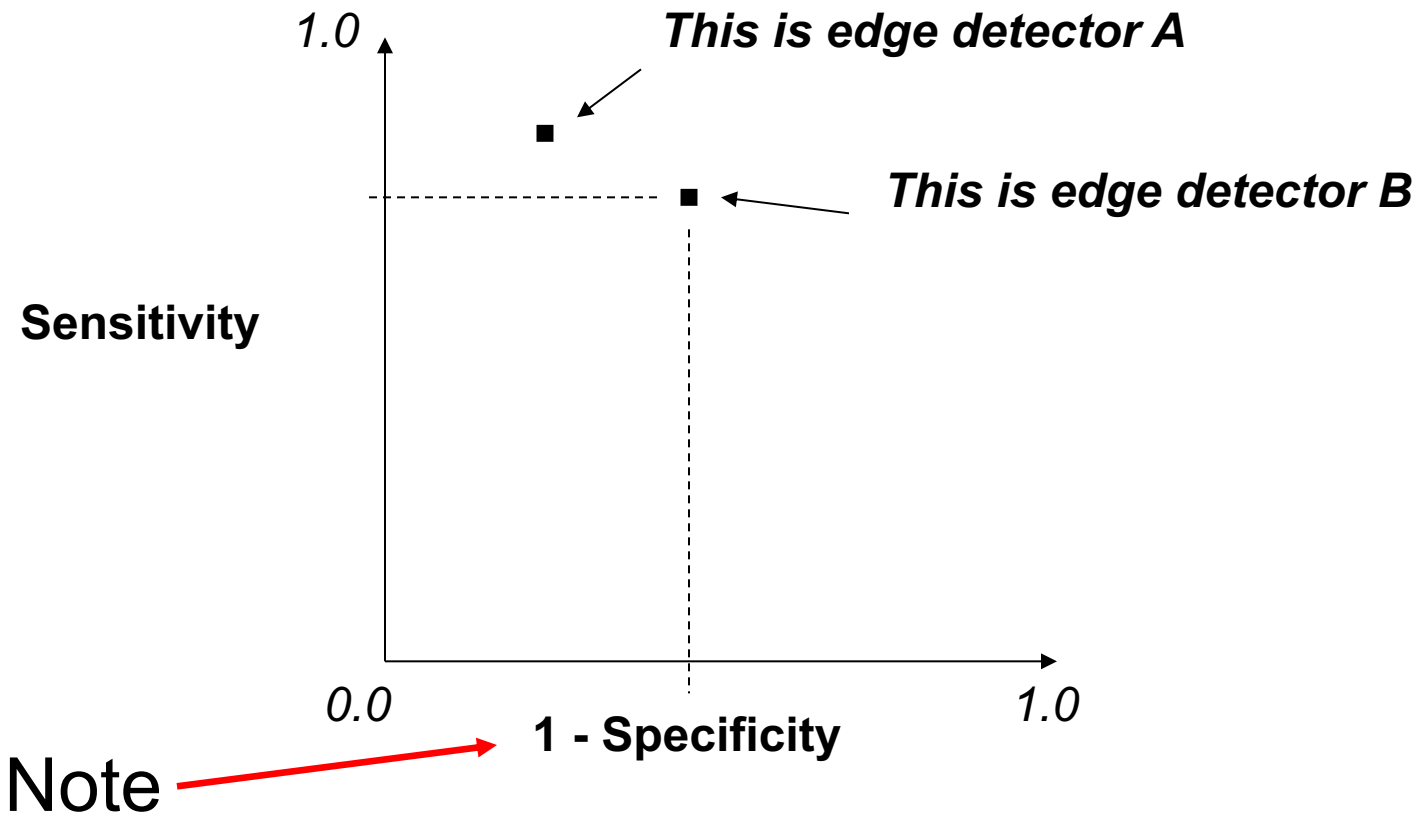
		Prediction 1	**Prediction** 0
Ground Truth	**1**	60	30
	0	80	20

60+30 = 90 cases in the dataset were class 1 (edge)

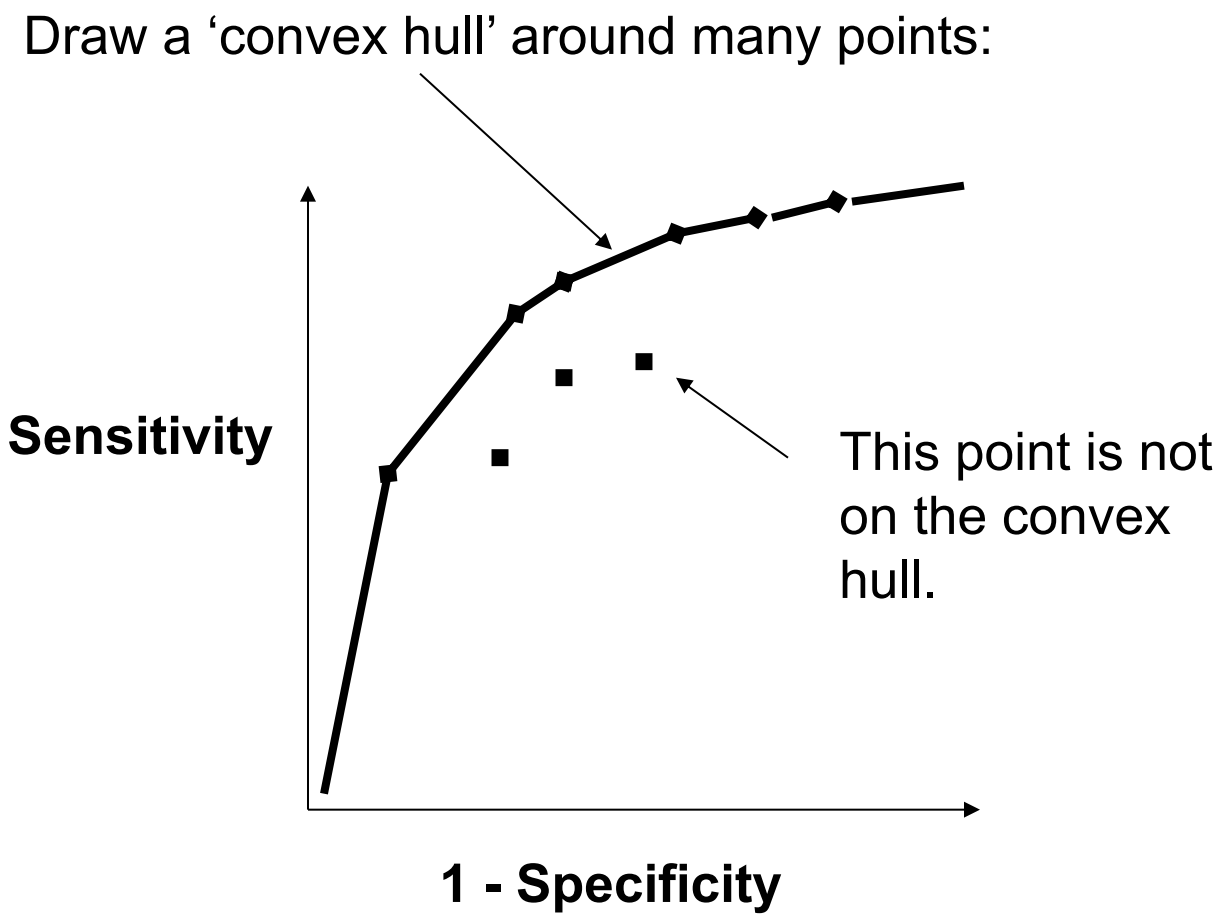
80+20 = 100 cases in the dataset were class 0 (non-edge)

90+100 = 190 examples (pixels) in the data overall

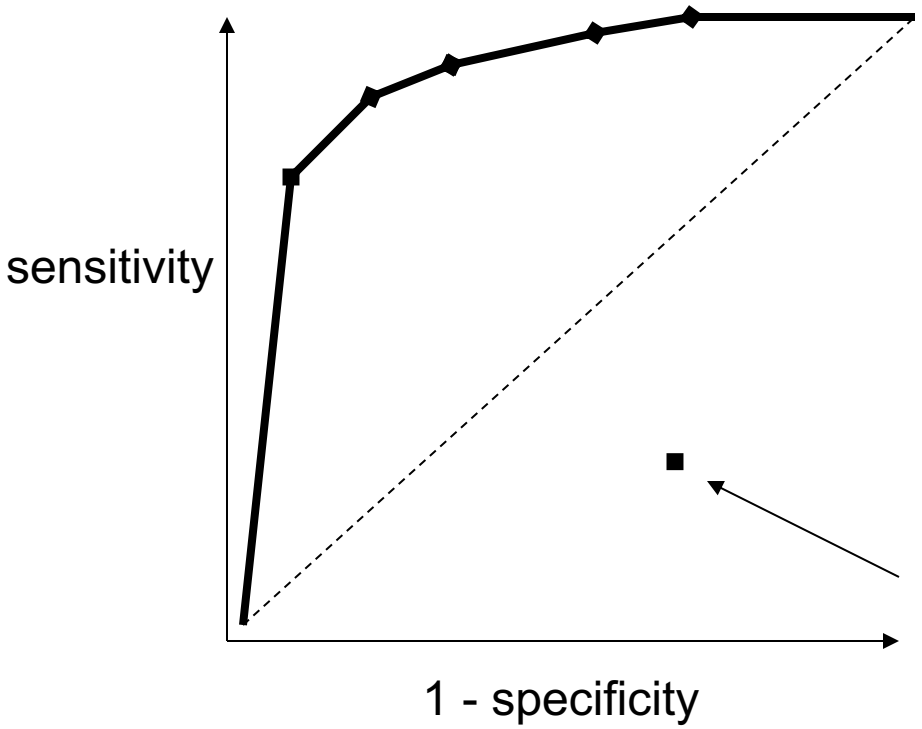
The ROC space



The ROC Curve



ROC Analysis



All the optimal detectors lie on the convex hull.

Which of these is best depends on the ratio of edges to non-edges, and the different cost of misclassification

Any detector on this side can lead to a better detector by flipping its output.

Take-home point : You should always quote sensitivity and specificity for your algorithm, if possible plotting an ROC graph. Remember also though, any statistic you quote should be an average over a suitable range of tests for your algorithm.

Holdout estimation

- What to do if the amount of data is limited?
- The *holdout* method reserves a certain amount for testing and uses the remainder for training

➔Usually: one third for testing, the rest for training

Holdout estimation

- Problem: the samples might not be representative
  - Example: class might be missing in the test data

- Advanced version uses *stratification*
  - Ensures that each class is represented with approximately equal proportions in both subsets

Repeated holdout method

- Repeat process with different subsamples

➔ more reliable

  - In each iteration, a certain proportion is randomly selected for training (possibly with stratificiation)
  - The error rates on the different iterations are averaged to yield an overall error rate

Repeated holdout method

- Still not optimum: the different test sets overlap
  - Can we prevent overlapping?
  - Of course!

Cross-validation

- *Cross-validation* avoids overlapping test sets
  - First step: split data into *k* subsets of equal size
  - Second step: use each subset in turn for testing, the remainder for training

- Called *k-fold cross-validation*

Cross-validation

- Often the subsets are stratified before the cross-validation is performed
- The error estimates are averaged to yield an overall error estimate

More on cross-validation

- Standard method for evaluation: stratified ten-fold cross-validation
- Why ten?
  - Empirical evidence supports this as a good choice to get an accurate estimate
  - There is also some theoretical evidence for this
- Stratification reduces the estimate's variance
- Even better: repeated stratified cross-validation
  - E.g. ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance)

Leave-One-Out cross-validation

- Leave-One-Out:
  a particular form of cross-validation:
  - Set number of folds to number of training instances
  - I.e., for *n* training instances, build classifier *n* times
- Makes best use of the data
- Involves no random subsampling
- Very computationally expensive
  - (exception: NN)

Leave-One-Out-CV and stratification

- Disadvantage of Leave-One-Out-CV: stratification is not possible
  - It *guarantees* a non-stratified sample because there is only one instance in the test set!

Recap

- We defined a *Bayes classifier* but saw that it's intractable to compute $P(X_1,\ldots,X_n|Y)$
- We then used the *Naïve Bayes assumption* – that everything is independent given the class label Y
- A natural question: is there some happy compromise where we only assume that *some* features are conditionally independent?
  - Stay Tuned…
- Naïve Bayes is:
  - Really easy to implement and often works well
  - Often a good first thing to try
  - Commonly used as a "punching bag" for smarter algorithms
- Evaluation of machine learning models