CS 484
Data Mining

Classification 5

Some slides are from Professor Eamonn Keogh at UC Riverside

Handling Missing Attribute Values

- Missing values affect decision tree construction in three different ways:
  - Affects how impurity measures are computed
  - Affects how to distribute instance with missing value to child nodes
  - Affects how a test instance with missing value is classified

Distribute Instances

Tid	Home Owner	Marital Status	Annual Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No

Home Owner

Yes

Class=Yes	0
Class=No	3

No

Class=Yes	2
Class=No	4

Tid	Home Owner	Marital Status	Annual Income	Class
10	?	Single	90K	Yes

Home Owner

Yes

Class=Yes	0 + 3/9
Class=No	3

No

Class=Yes	2 + 6/9
Class=No	4

Probability that Home_Owner=Yes is 3/9

Probability that Home_Owner=No is 6/9

Assign record to the left child with weight = 3/9 and to the right child with weight = 6/9

Other Issues

- Data Fragmentation
- Search Strategy
- Expressiveness
- Tree Replication

Data Fragmentation

- Number of instances gets smaller as you traverse down the tree

- Number of instances at the leaf nodes could be too small to make any statistically significant decision

Search Strategy

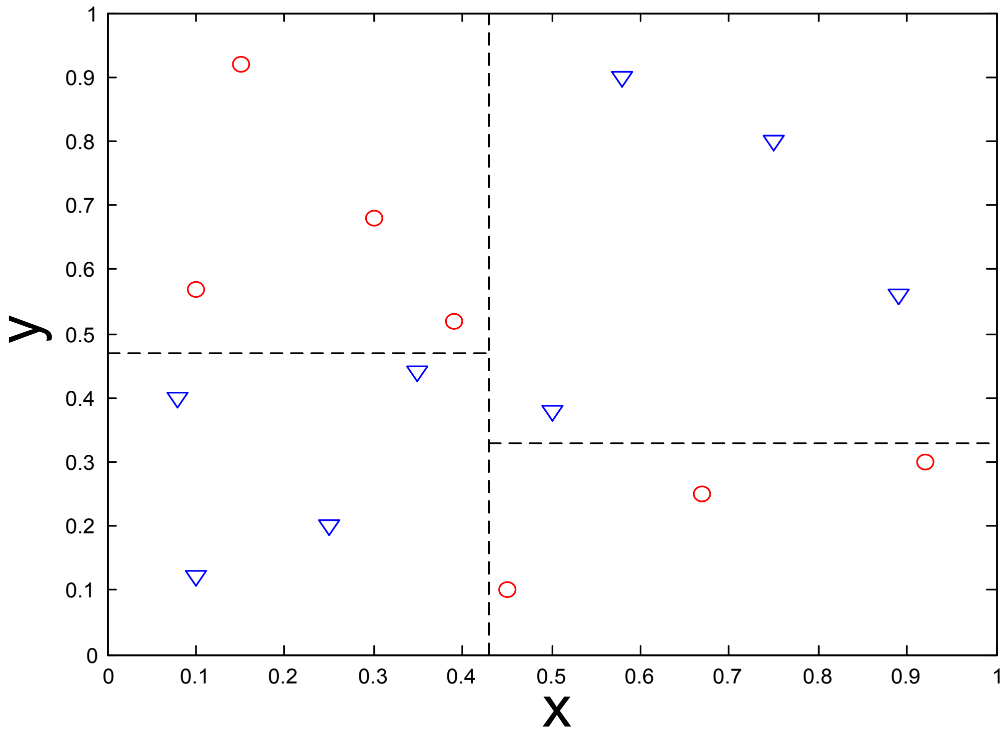
- Finding an optimal decision tree is NP-hard
- The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution
- Other strategies?
  - Bottom-up
  - Bi-directional

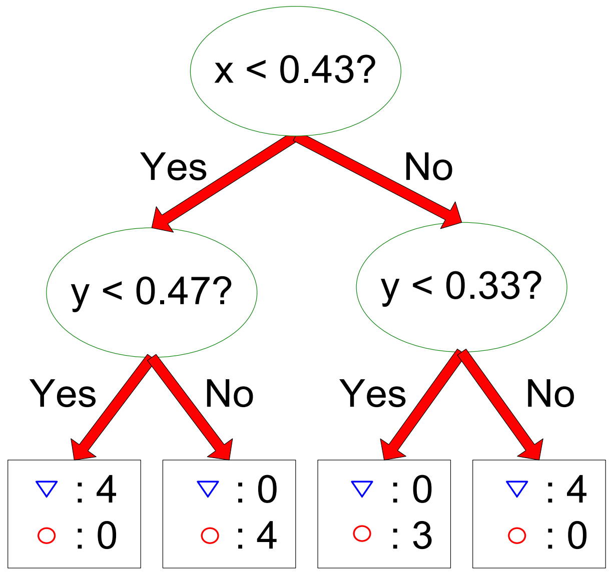
Expressiveness

- Decision tree provides expressive representation for learning discrete-valued function
  - But they do not generalize well to certain types of Boolean functions
    - Example: parity function:
      - Class = 1 if there is an even number of Boolean attributes with truth value = True
      - Class = 0 if there is an odd number of Boolean attributes with truth value = True
    - For accurate modeling, must have a complete tree

- Not expressive enough for modeling continuous variables
  - Particularly when test condition involves only a single attribute at-a-time

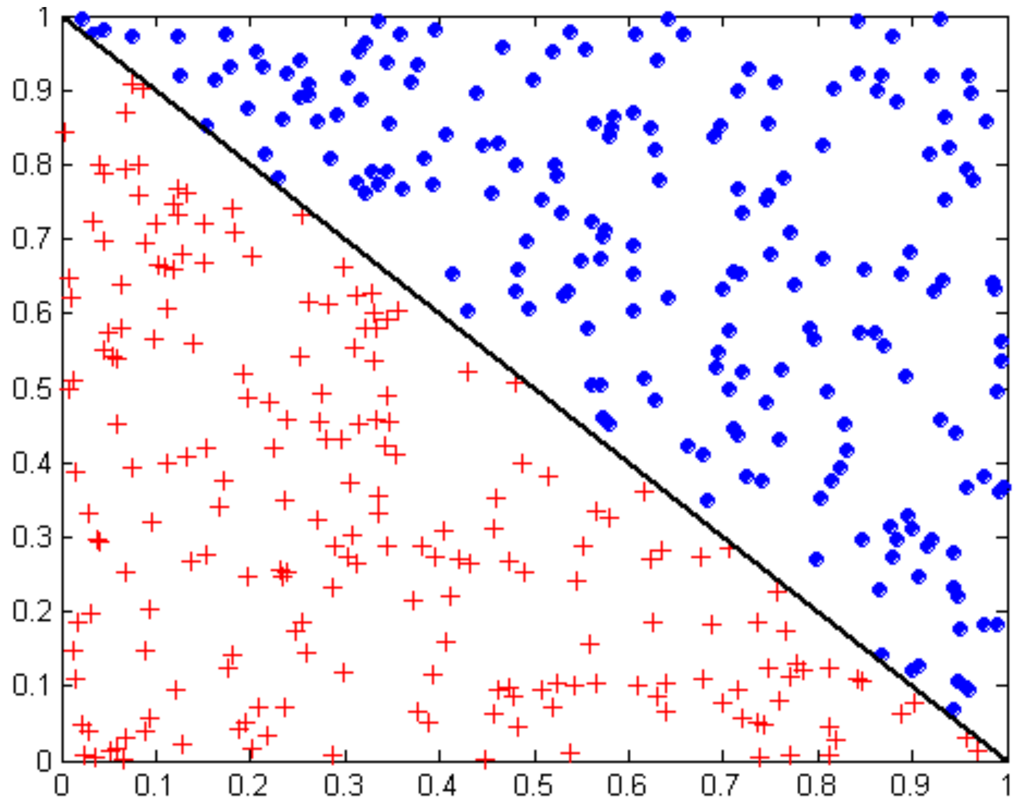
Decision Boundary

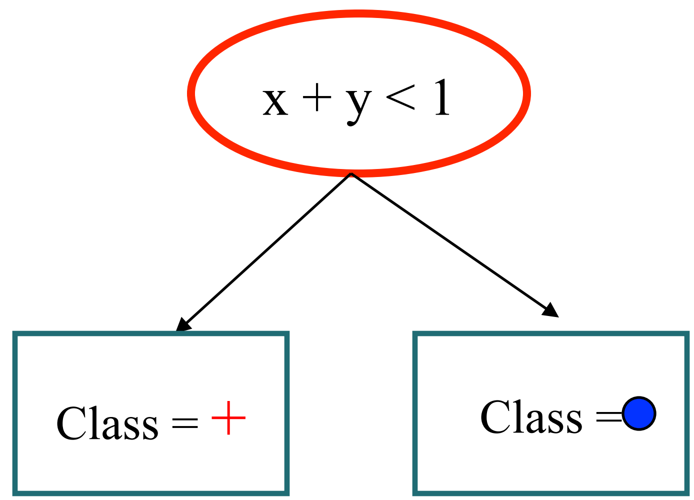




- **Border line between two neighboring regions of different classes is known as decision boundary**
- **Decision boundary is parallel to axes because test condition involves a single attribute at-a-time**

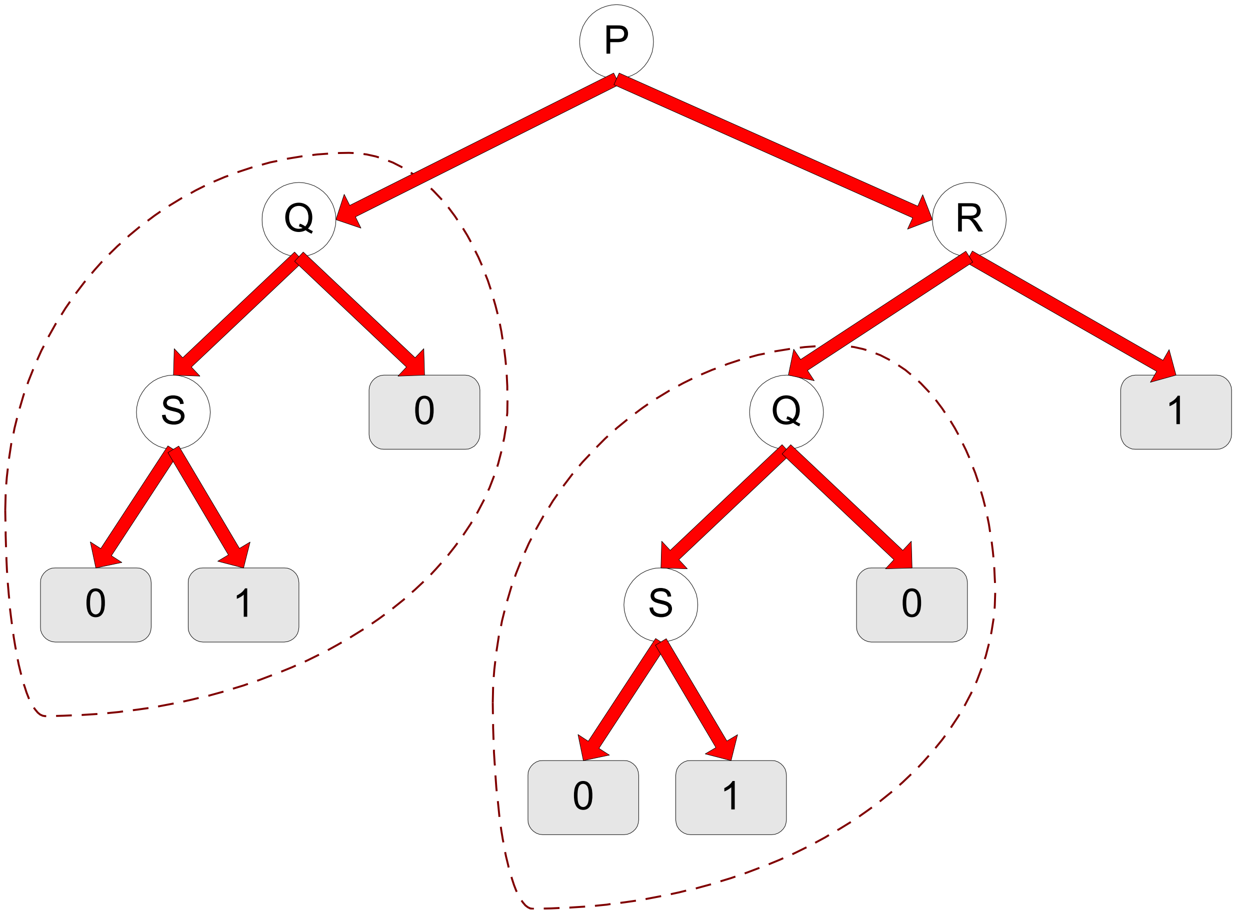
Oblique Decision Trees





- **Test condition may involve multiple attributes**
- **More expressive representation**
- **Finding optimal test condition is computationally expensive**

Tree Replication



- **Same subtree appears in multiple branches**

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Metrics for Performance Evaluation

Focus on the predictive capability of a model, rather than how fast it takes to classify or build models, scalability, etc.

$$\text{Accuracy} = \frac{\text{Number of correct classifications}}{\text{Number of instances in our database}}$$

Accuracy is a single number, we may be better off looking at a **confusion matrix**. This gives us additional useful information…

Classified as...

True label is…

	Cat	Dog	Pig
Cat	100	0	0
Dog	9	90	1
Pig	45	45	10

Metrics for Performance Evaluation

- Confusion Matrix:

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a	b
	Class=No	c	d

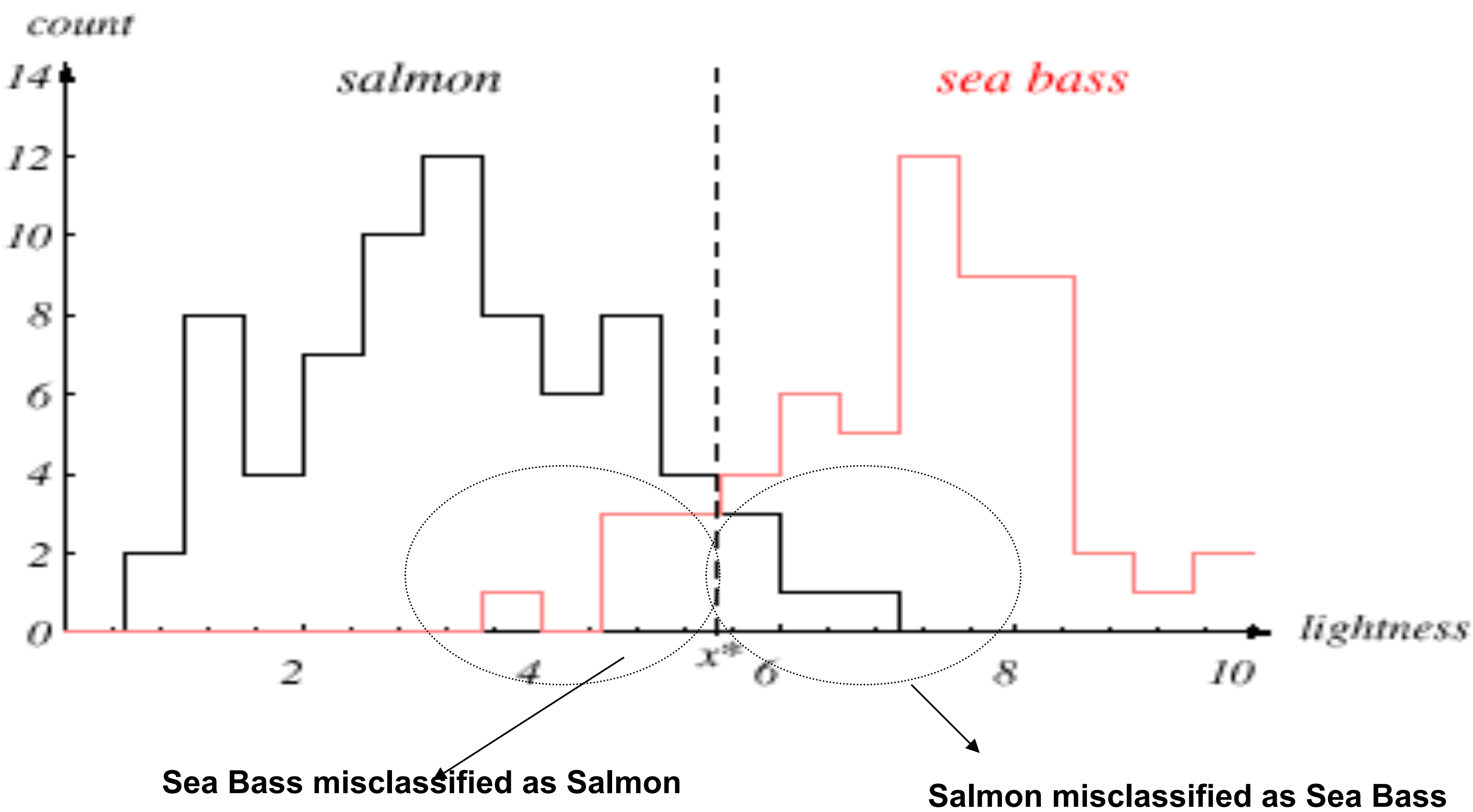
a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Remember this Example?



Metrics for Performance Evaluation

- Confusion Matrix:

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Predicted as...

True label is…

	Salmon	Sea Bass
Salmon		
Sea Bass		

Metrics for Performance Evaluation…

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

(% of correctly classified items)

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Methods for Performance Evaluation

- How to obtain a reliable estimate of performance?
- Performance of a model may depend on other factors besides the learning algorithm:
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets

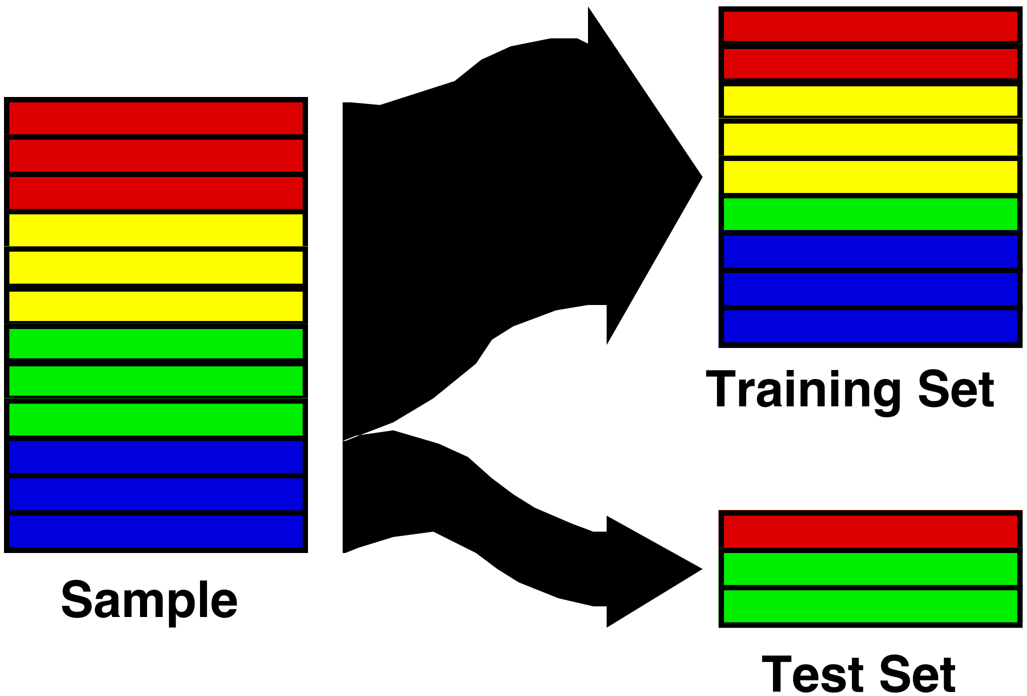
Methods of Estimation

- Holdout
  - Reserve 2/3 for training and 1/3 for testing
- Random subsampling
  - Repeated holdout
- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n
- Stratified sampling
  - oversampling vs undersampling
- Bootstrap
  - Sampling with replacement

Hold-out validation: simple holdout set

Partition data into training set and test set

In some domains it makes sense to partition temporally (training set before t, test set after t)



challenges: 1) what if by accident you selected a particularly easy/hard test set? 2) do you have an idea of the variation in model accuracy due to training?

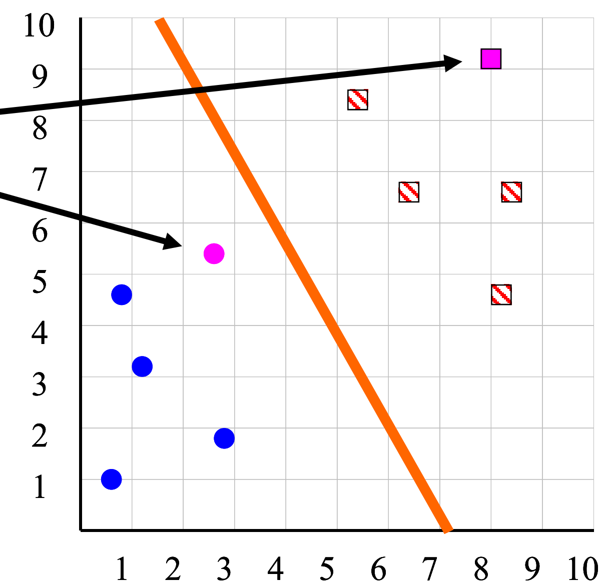
K-Fold Cross Validation

We divide the dataset into *K* equal sized sections. The algorithm is tested *K* times, each time leaving out one of the *K* section from building the classifier, but using it to *test* the classifier instead

$$\text{Accuracy} = \frac{\text{Number of correct classifications}}{\text{Number of instances in our database}}$$

$K = 5$

Insect ID	Abdomen Length	Antennae Length	Insect Class
1	2.7	5.5	Grasshopper
2	8.0	9.1	Katydid
3	0.9	4.7	Grasshopper
4	1.1	3.1	Grasshopper
5	5.4	8.5	Katydid
6	2.9	1.9	Grasshopper
7	6.1	6.6	Katydid
8	0.5	1.0	Grasshopper
9	8.3	6.6	Katydid
10	8.1	4.7	Katydids



Limitation of Accuracy

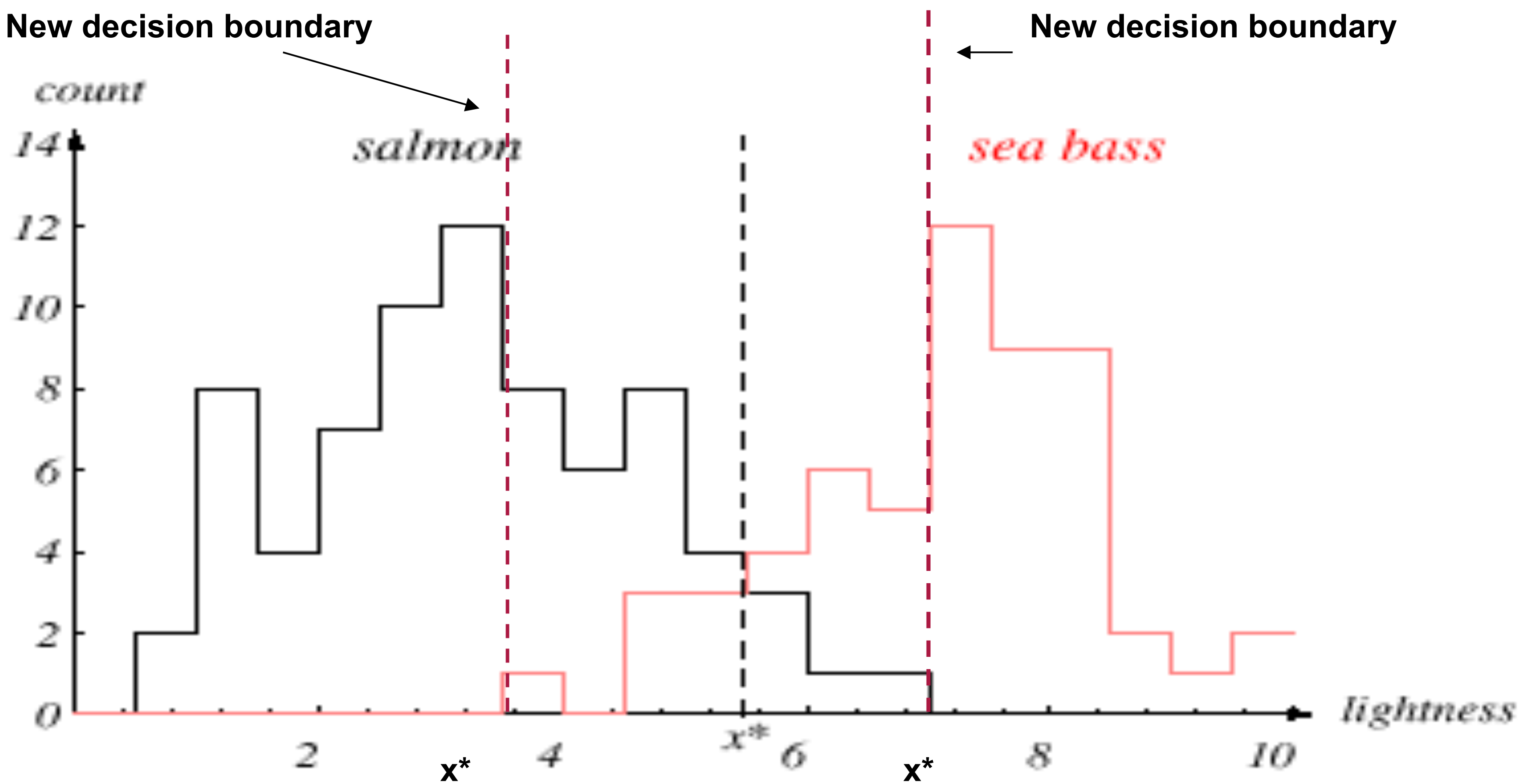
- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10

- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example

What if?

Salmon is more expensive than Bass?

Bass is more expensive than Salmon?



Cost sensitive classification

- Penalize misclassifications of one class more than the other
- Changes decision boundaries

Cost Matrix

	PREDICTED CLASS	
ACTUAL CLASS C(i, j)	**Class=Yes**	**Class=No**
Class=Yes	C(Yes, Yes)	C(Yes, No)
Class=No	C(No, Yes)	C(No, No)

C(i, j): Cost of misclassifying class i example as class j

Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
ACTUAL CLASS	C(i, j)	+	-
	+	-1	100
	-	1	0

False negative error cost

False positive error cost

Model M_1	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	150	40
	-	60	250

Accuracy = 80%

Cost = 3910

Model M_2	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	250	45
	-	5	200

Accuracy = 90%

Cost = 4255

Cost vs Accuracy

Count	PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

Cost	PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	p	q
	Class=No	q	p

Accuracy is proportional to cost if
1. C(Yes, No)=C(No, Yes) = q
2. C(Yes, Yes)=C(No, No) = p

$N = a + b + c + d$

$\text{Accuracy} = (a + d)/N$

$$\begin{aligned}\text{Cost} &= p\,(a + d) + q\,(b + c)\\ &= p\,(a + d) + q\,(N - a - d)\\ &= q\,N - (q - p)(a + d)\\ &= N\,[q - (q - p) \times \text{Accuracy}]\end{aligned}$$

Cost-Sensitive Measures

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

$$\text{F1-measure (F1)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

Count	PREDICTED CLASS	
	Class=Yes	Class=No
ACTUAL CLASS Class=Yes	a	b
ACTUAL CLASS Class=No	c	d

- Precision is biased towards C(Yes, Yes) & C(No, Yes)
- Recall is biased towards C(Yes, Yes) & C(Yes, No)
- F1-measure is biased towards all except C(No, No)

$$\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}$$

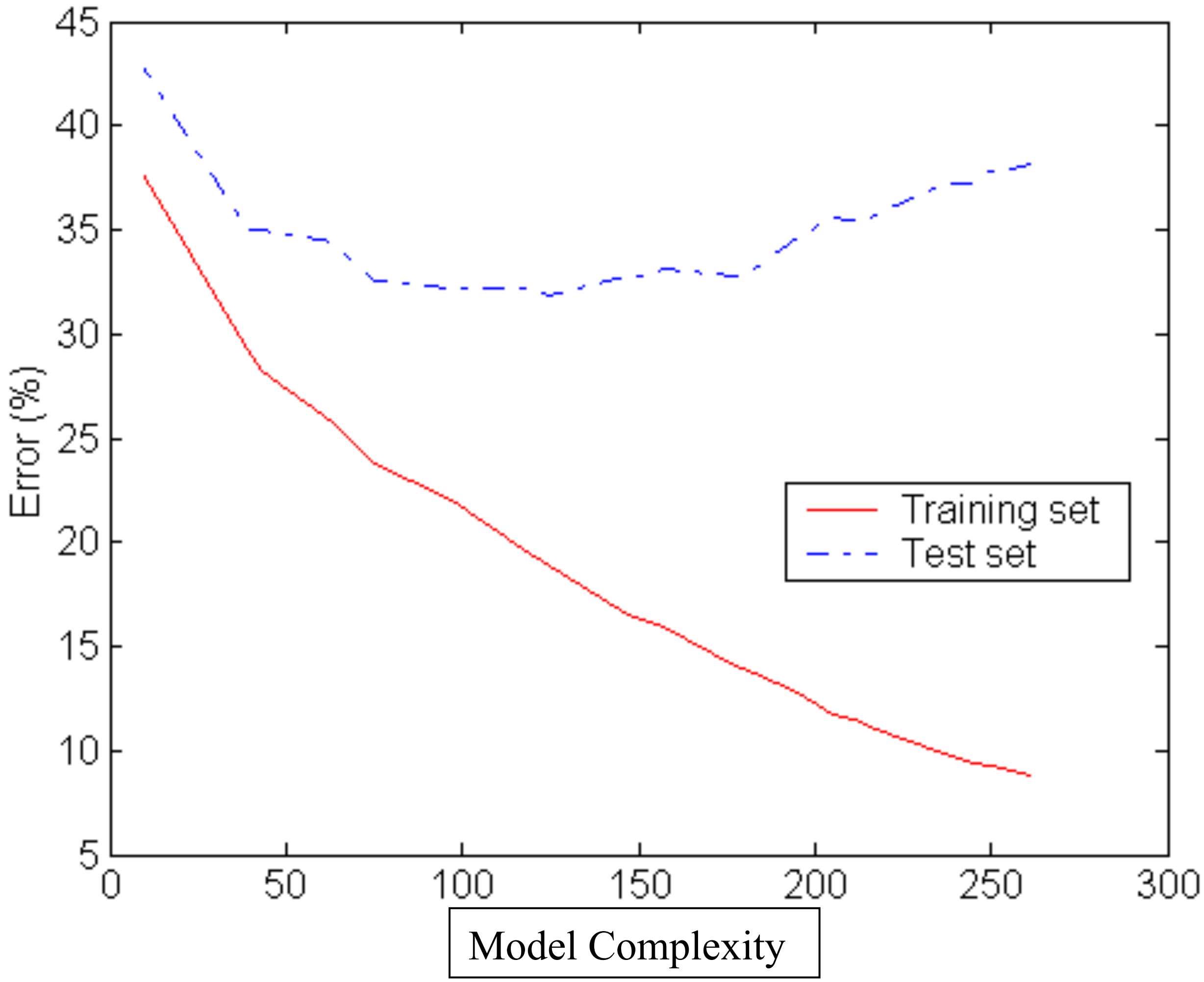
Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

revisit

Tool for model performance analytics: The fitting curve

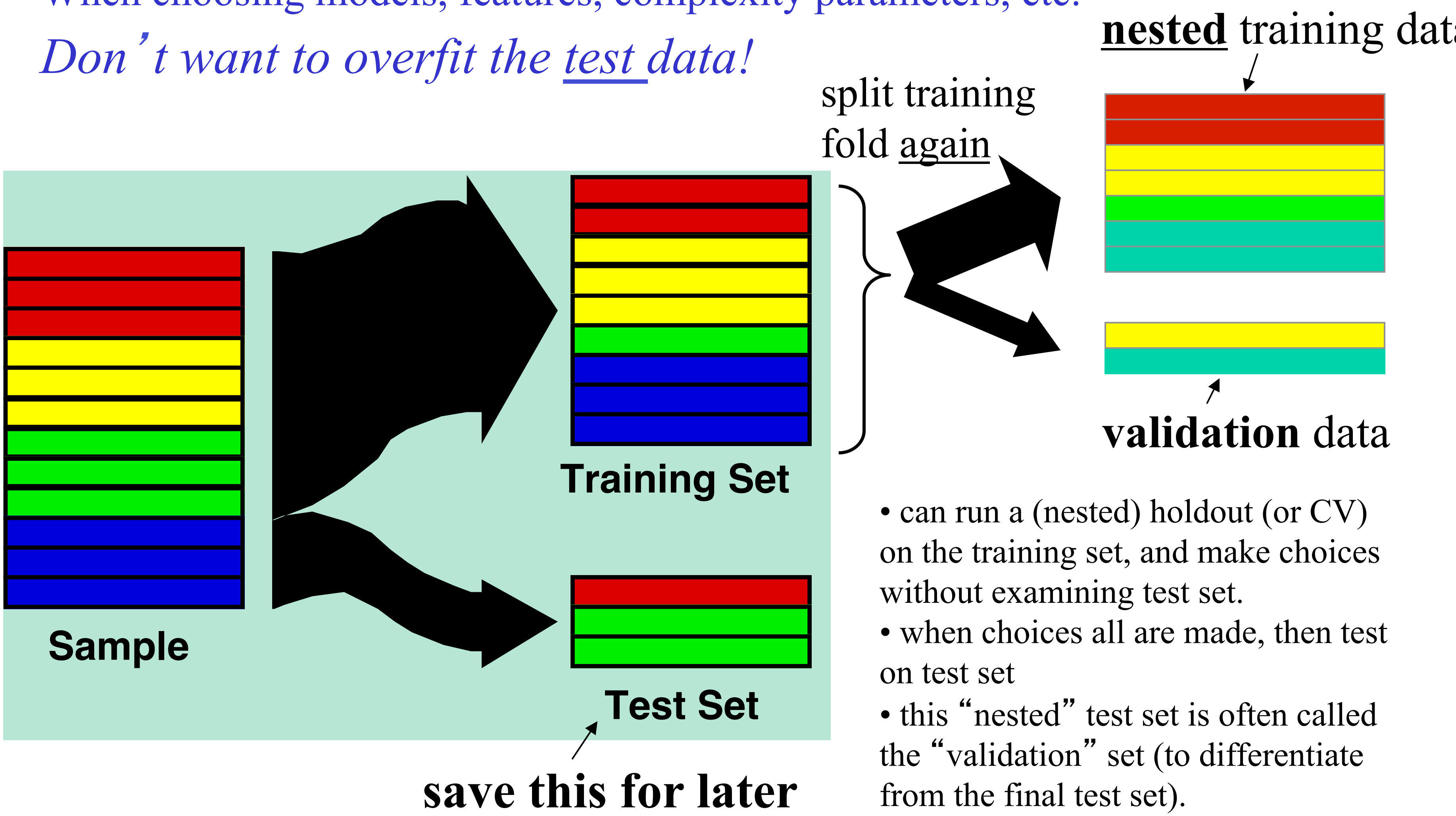
But can we find the right complexity?



Nested holdout for complexity control

Need to be careful making data mining decisions based on testing data (or CV)

- When choosing models, features, complexity parameters, etc.
- *Don't want to overfit the <u>test data</u>!*



- can run a (nested) holdout (or CV) on the training set, and make choices without examining test set.
- when choices all are made, then test on test set
- this "nested" test set is often called the "validation" set (to differentiate from the final test set).

ROC (Receiver Operating Characteristic)

- Developed in 1950s for signal detection theory to analyze noisy signals
  - Characterize the trade-off between positive hits and false alarms
- ROC curve plots TPR (on the y-axis) against FPR (on the x-axis)
- Performance of each classifier represented as a point on the ROC curve
  - changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point

ROC (Receiver Operating Characteristic)

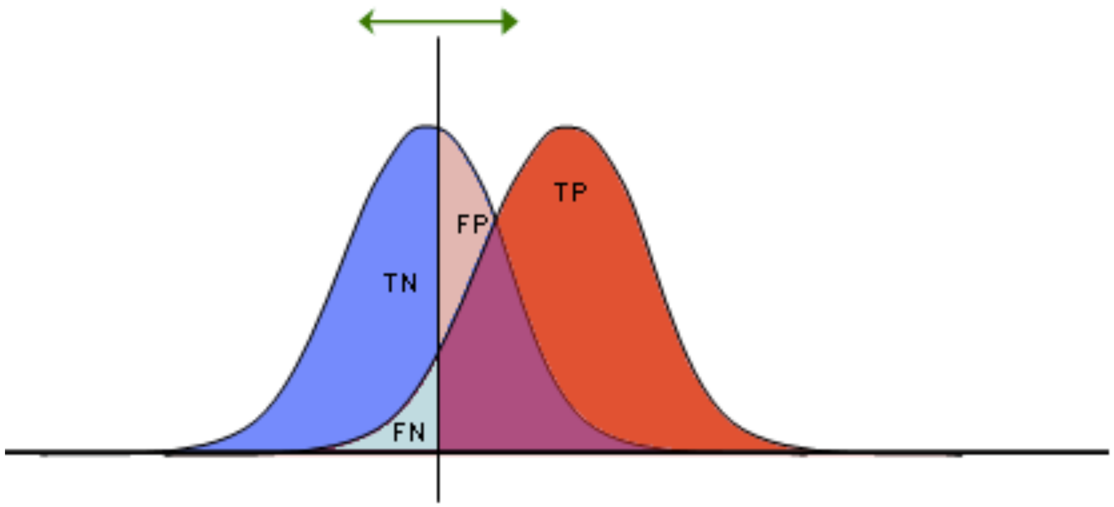
- Developed in 1950s for signal detection theory to analyze noisy signals
  - Characterize the trade-off between positive hits and false alarms
- **ROC** curve plots **TPR** (on the **y**-axis) against **FPR** (on the **x**-axis)

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

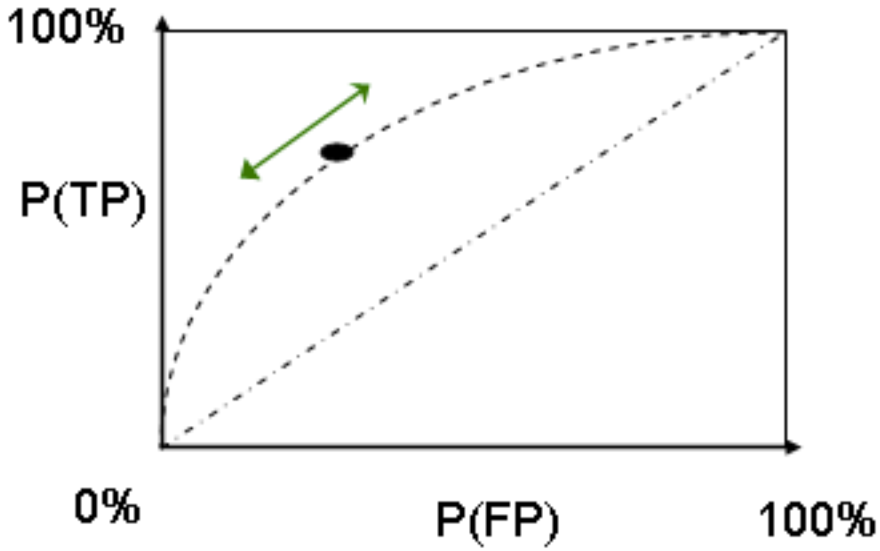
$$FPR = \frac{FP}{FP + TN}$$

	PREDICTED CLASS		
		Yes	No
Actual	Yes	a (TP)	b (FN)
	No	c (FP)	d (TN)

http://en.wikipedia.org/wiki/Receiver_operating_characteristic

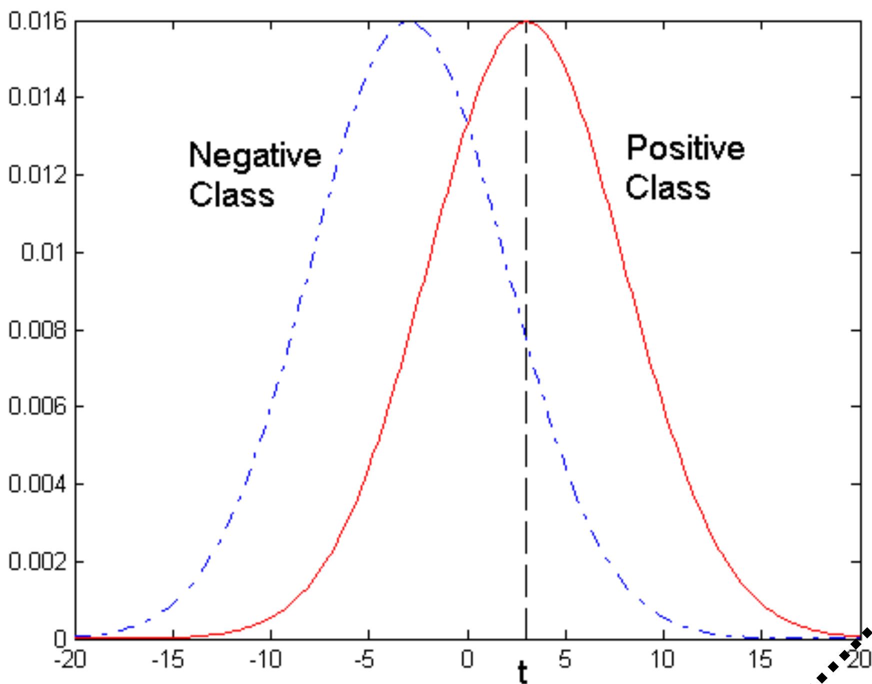


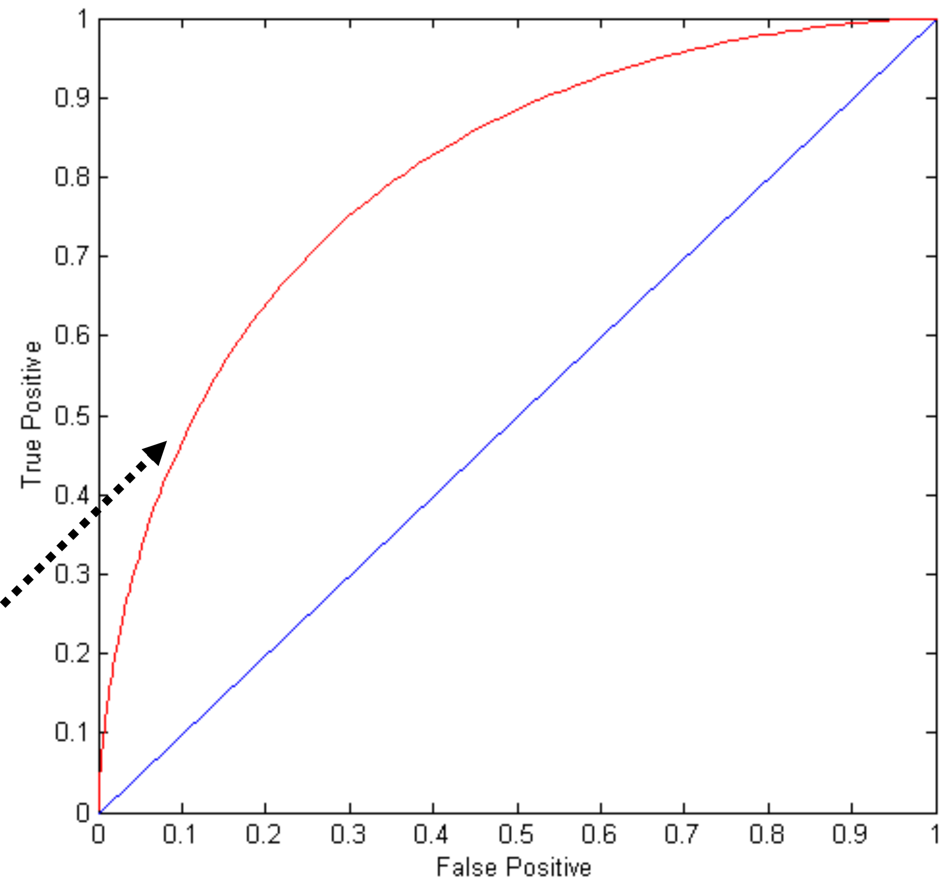
TP	FP
FN	TN
1	1



ROC Curve

- 1-dimensional data set containing 2 classes (positive and negative)
- any points located at x > t is classified as positive





At threshold t:

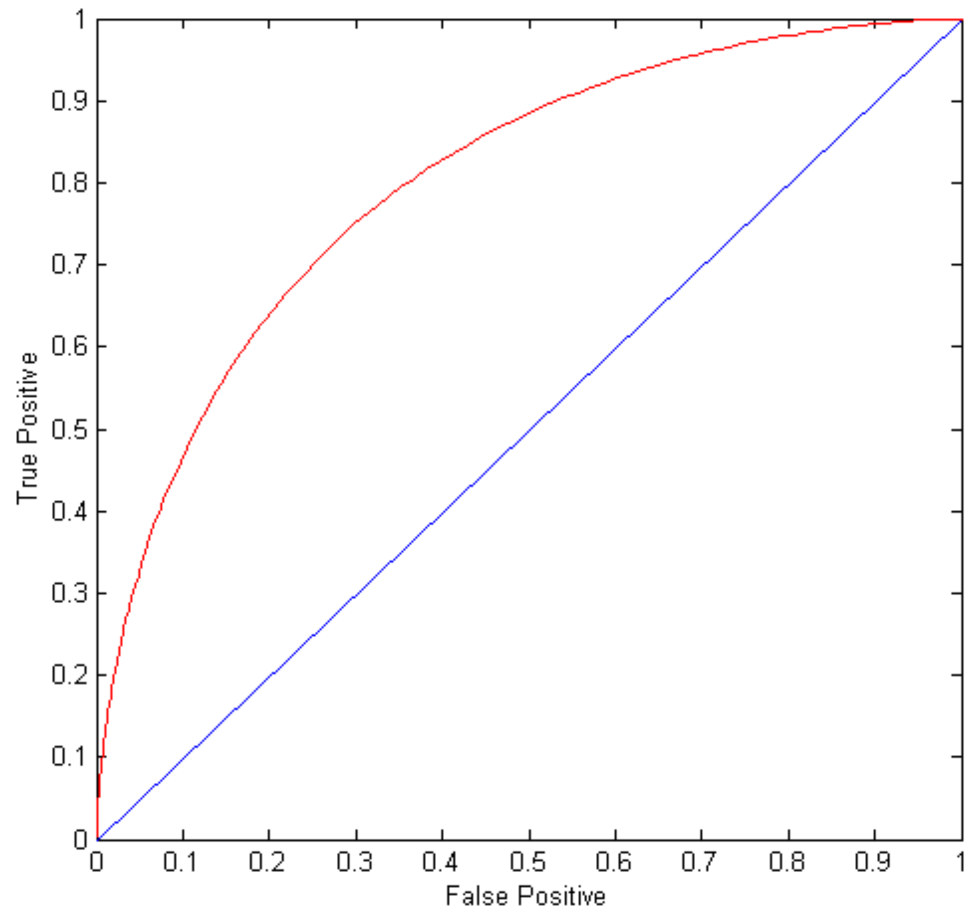
TP=0.5, FN=0.5, FP=0.12, TN=0.88

ROC Curve

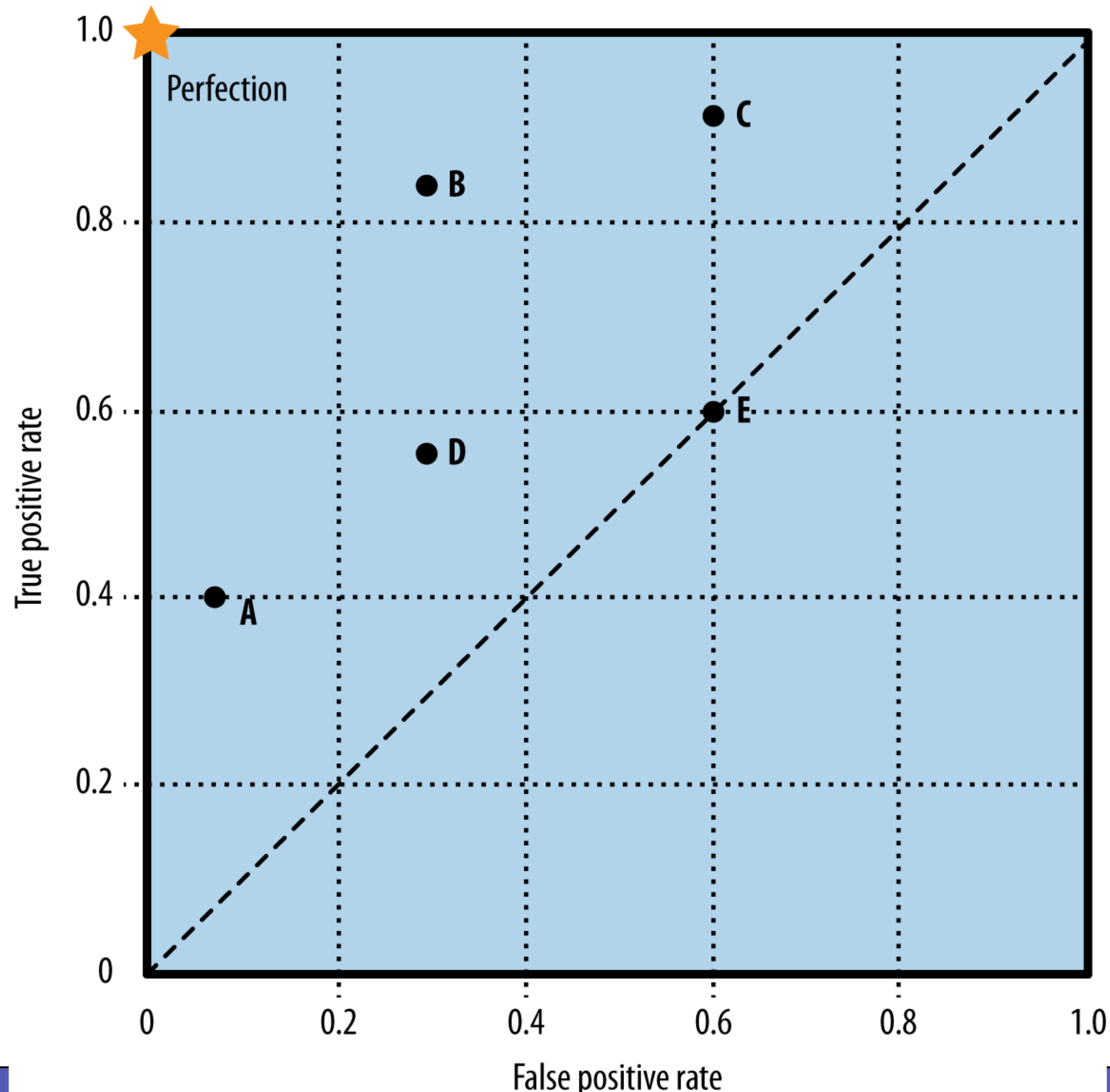
(TP,FP):

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal

- Diagonal line:
  - Random guessing
  - Below diagonal line:
    - prediction is opposite of tl true class

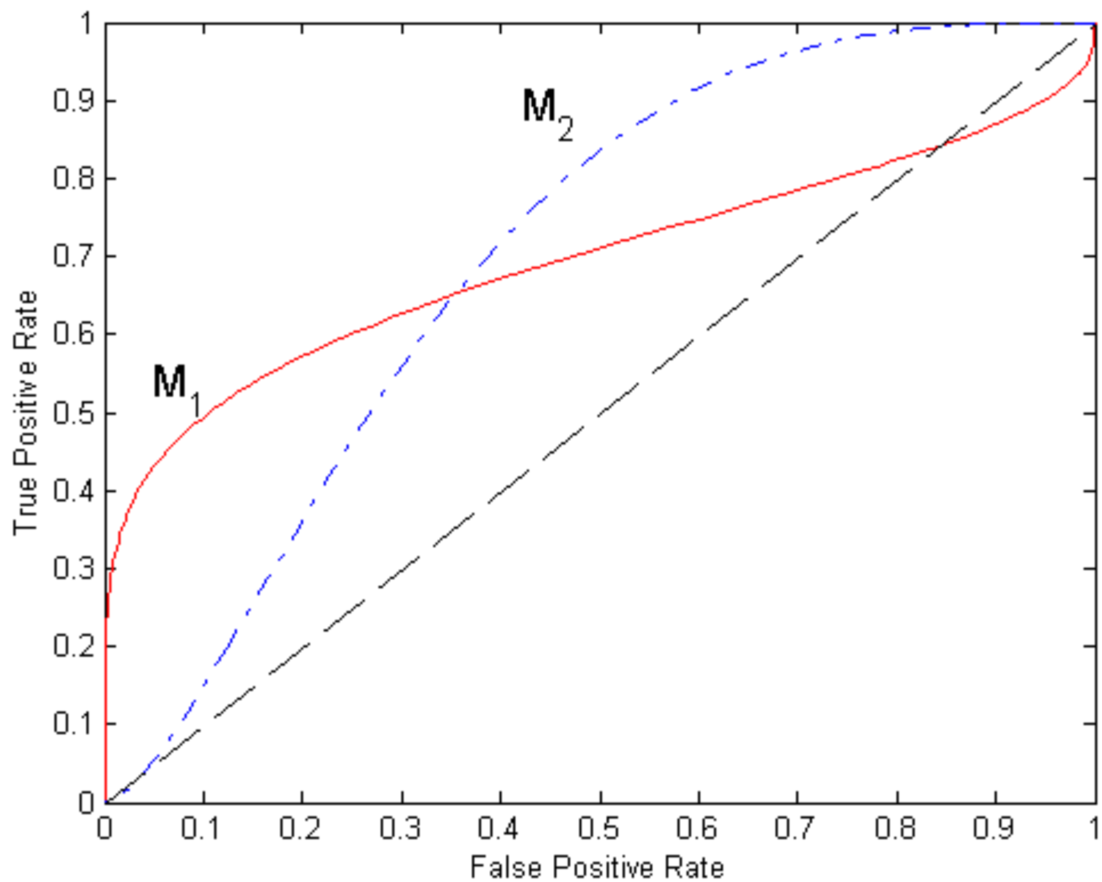


ROC space



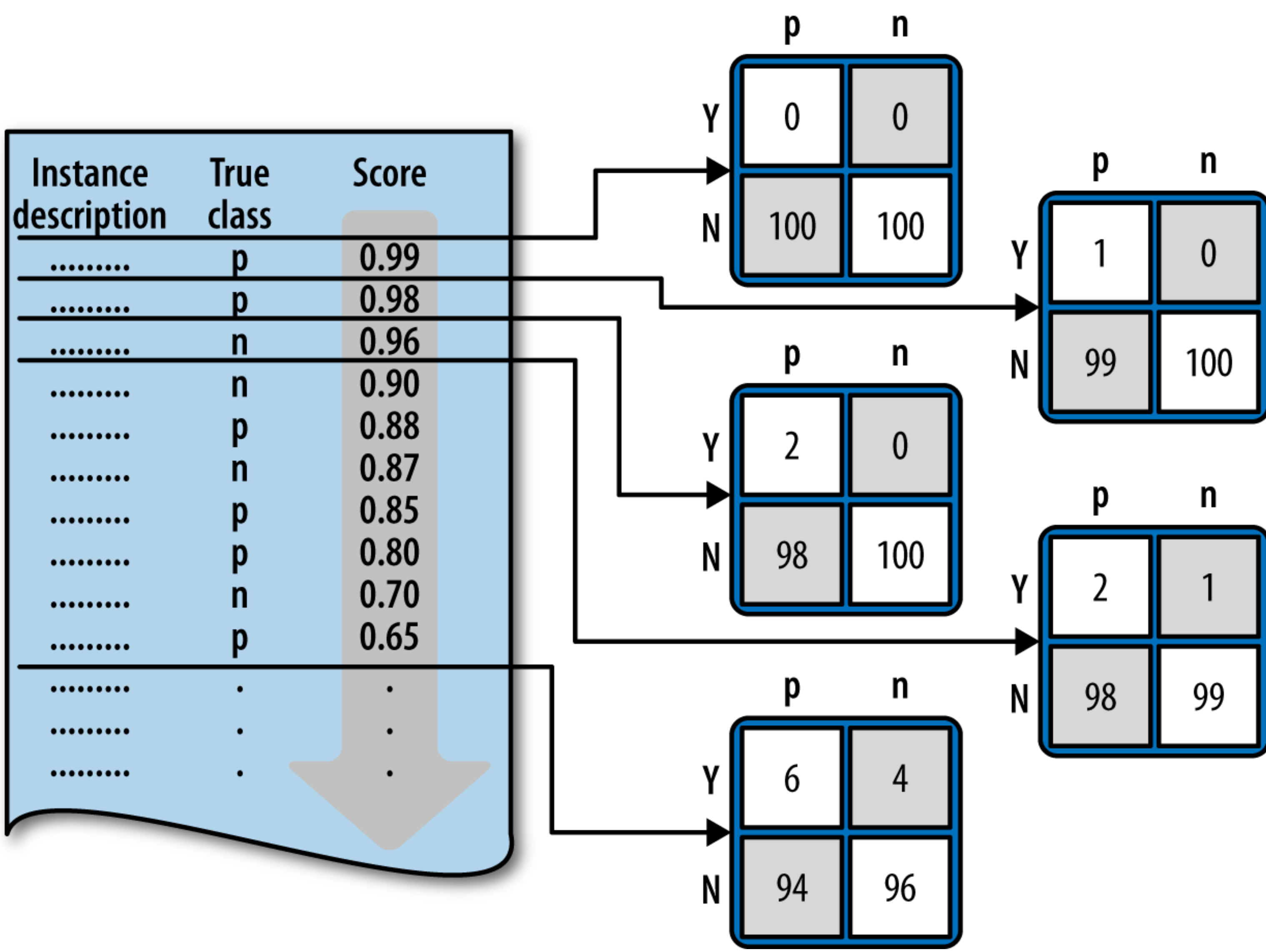
1.0
0.8
0.6
0.4
0.2
0
Perfection
C
B
E
D
A
0
0.2
0.4
0.6
0.8
1.0
True positive rate
False positive rate

Using ROC for Model Comparison

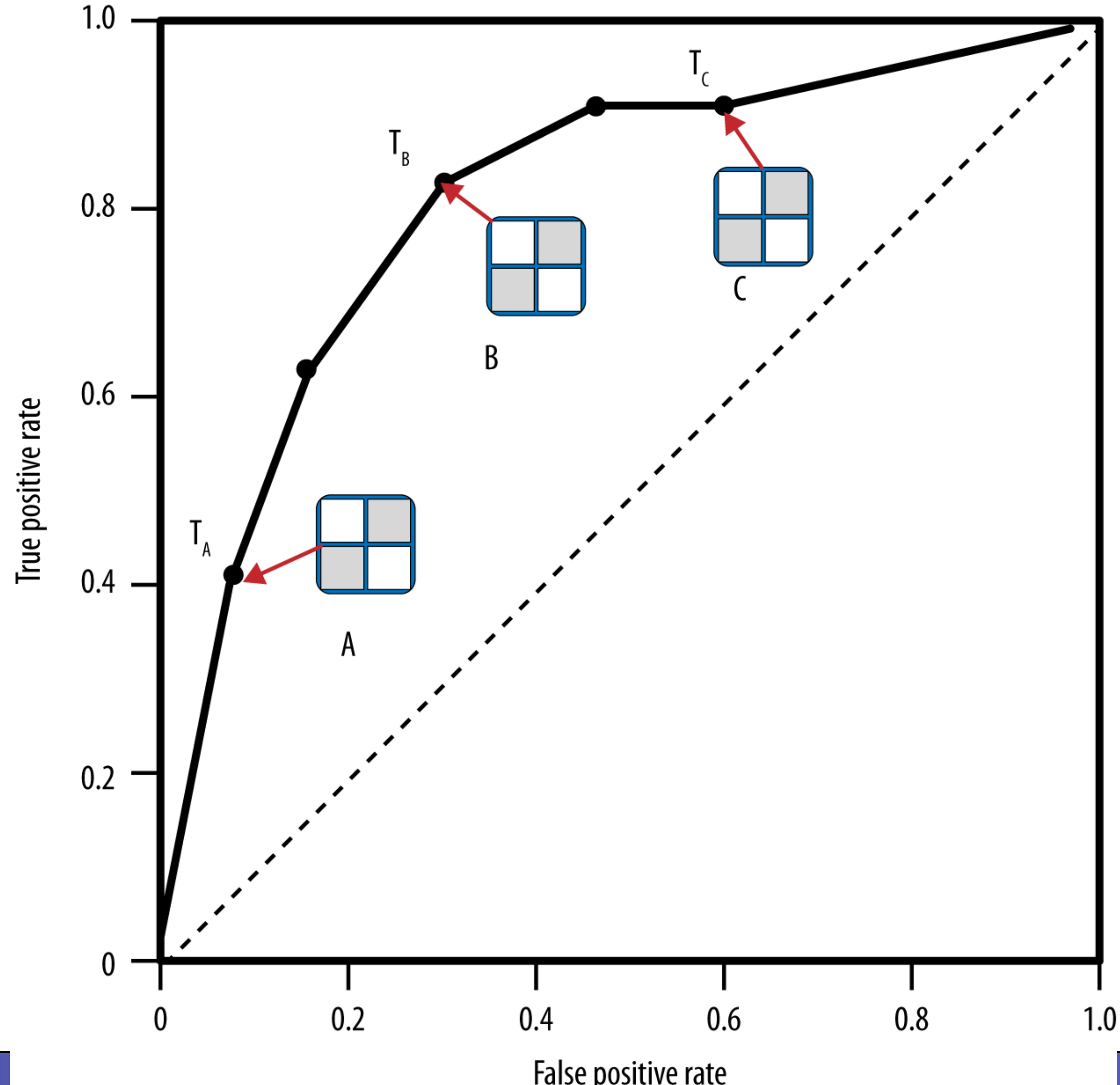


- No model consistently outperform the other
  - M_1 is better for small FPR
  - M_2 is better for large FPR
- Area Under the ROC curve
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5

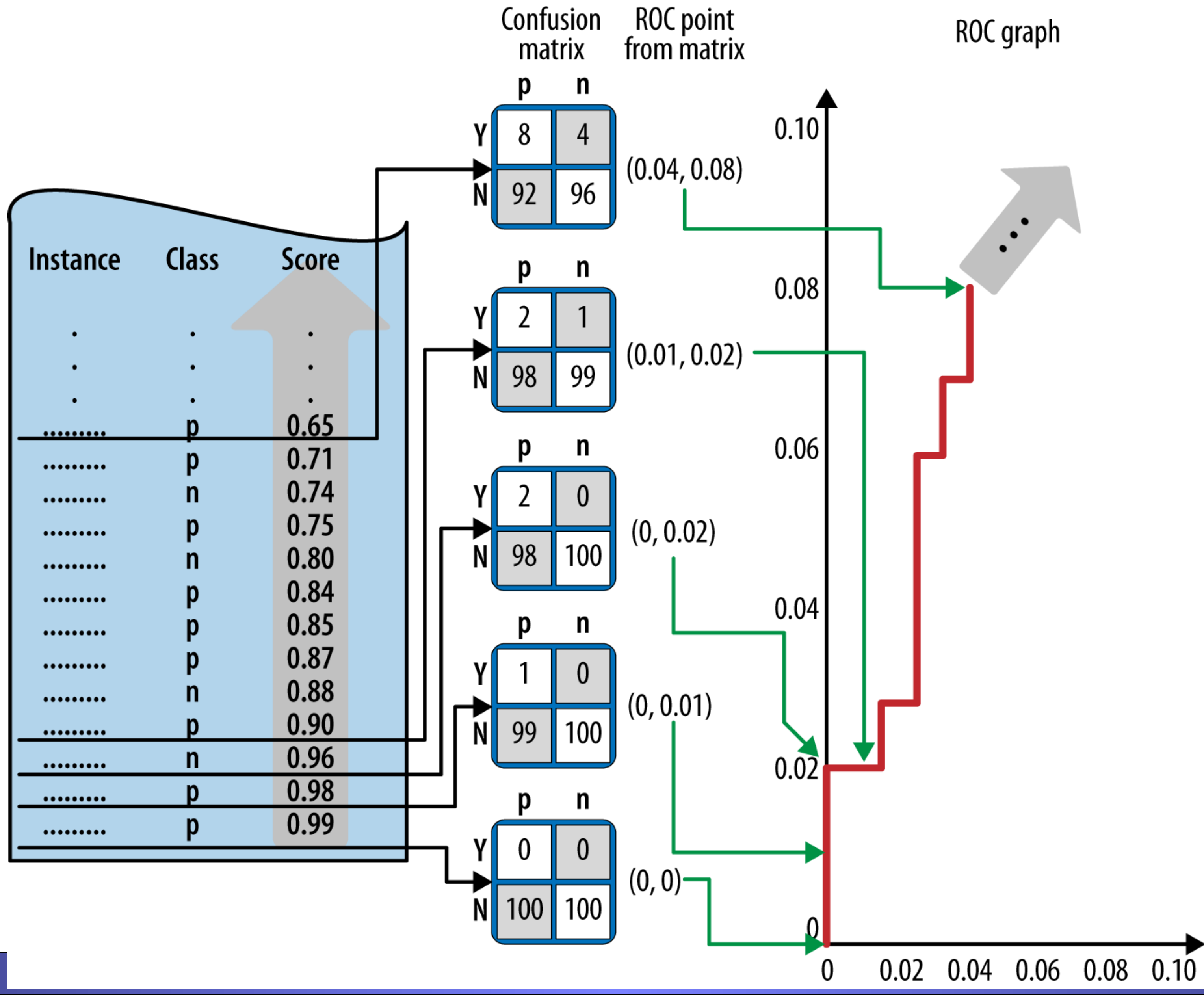
How To Construct an ROC Curve?



How To Construct an ROC Curve?



How To Construct an ROC Curve?



How to Construct an ROC curve

Instance	P(+|A)	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

- Use classifier that produces posterior probability for each test instance P(+|A)
- Sort the instances according to P(+|A) in decreasing order
- Apply threshold at each unique value of P(+|A)
- Count the number of TP, FP, TN, FN at each threshold
- TP rate, TPR = TP/(TP+FN)
- FP rate, FPR = FP/(FP + TN)

How to construct an ROC curve

Class	+	-	+	-	-	-	+	-	+	+	
Threshold >=	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

ROC Curve: