MACHINE LEARNING

UNIT-1

Binary Classification
and
Related Tasks

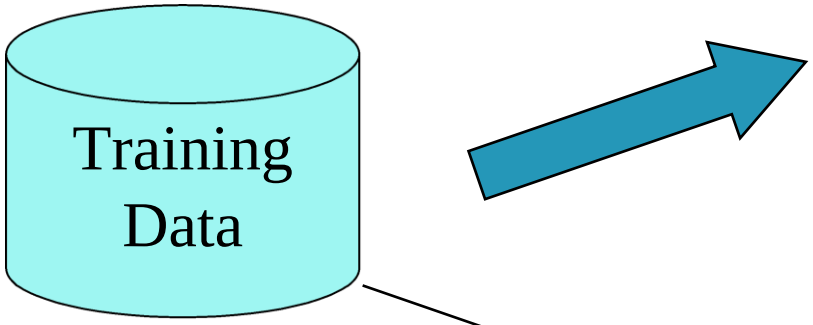
Topics

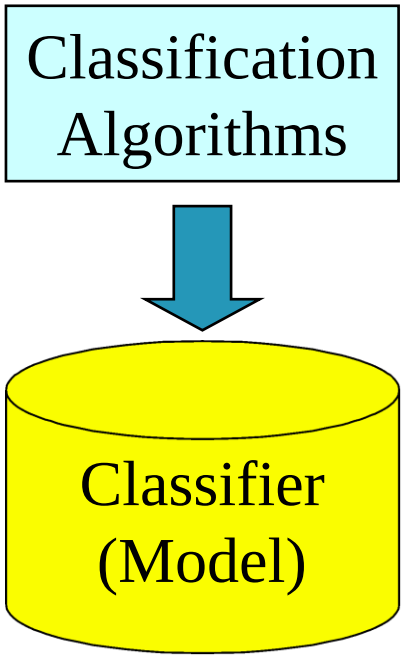
- **Classification**
- **Scoring and Ranking**
- **Class Probability Estimation**

Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to classify new data
- Note: If *the test set* is used to select models, it is called validation (test) set

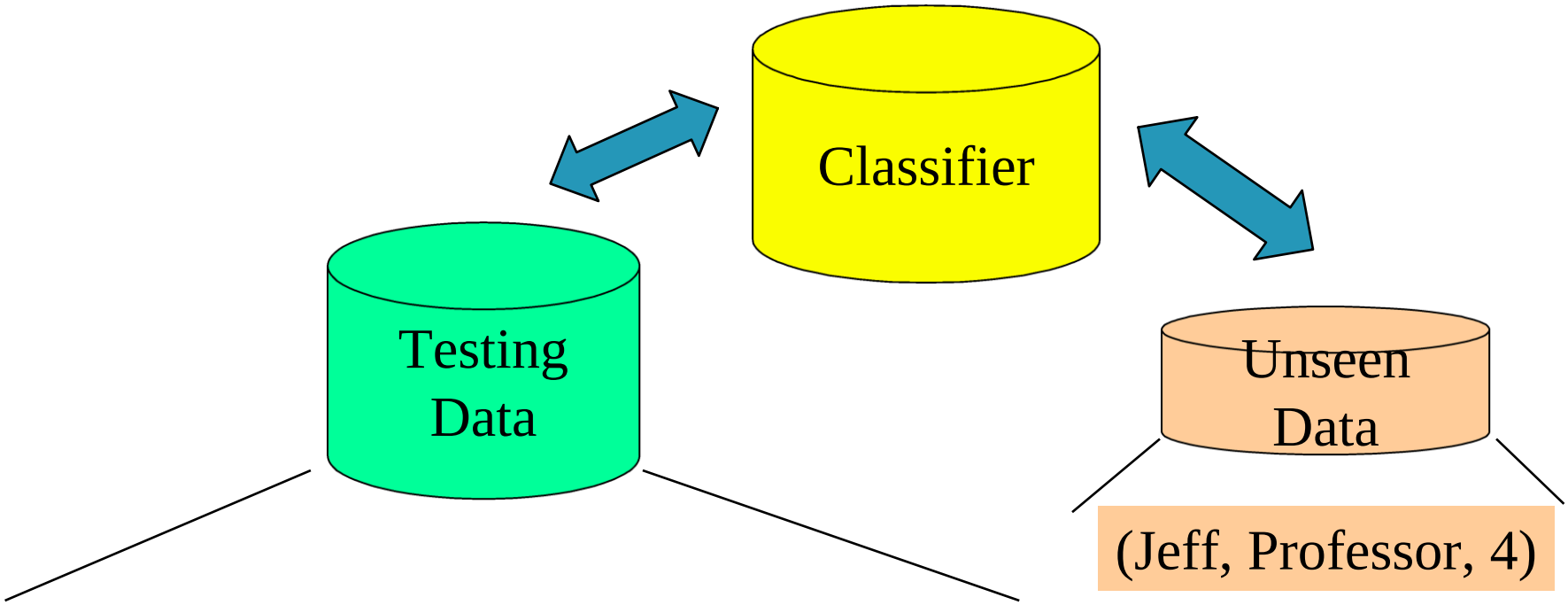
Process (1): Model Construction





NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Process (2): Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Classification

- Classification is the most common task in machine learning.
- Mapping from Instance space“$\mathscr{X}$“ to a class label of different classes.

$$\hat{c}: \mathscr{X} \rightarrow \mathscr{C}, \text{where } \mathscr{C} = \{C_1, C_2, \ldots, C_k\}$$

- If only two classes are existed then it is called as binary classification.
  - Positive + or Negative –
  - +1 or -1
- Binary classification is also called as concept learning. Positive class is called as concept.

Assessing Classification Performance

- Performance of classifier can be summarized using a table called contingency table or confusion matrix.
- Row-Actual and Column-Predicted

	Predicted positive	predicted negative	
Actual Positive	True Positive Model correctly Predicts the +ve class	False Negative Model in-correctly Predicts the +ve class	A
Actual Negative	False Positive Model in-correctly Predicts the -ve class	True Negative Model correctly Predicts the -ve class	B
	X	Y	

- Row and column sums are called marginals.

	Predicted ⊕	*Predicted* ⊖	
Actual ⊕	30	20	50
Actual ⊖	10	40	50
	40	60	100

Metrics

- *Accuracy = TP+TN/(TP+FP+FN+TN)*
- *Error rate=1-Accuracy*
- *Precision(PPR) = TP/(TP+FP)*
- *Recall (TPR)= TP/(TP+FN)*
- *F1-Score = 2*(Recall * Precision) / (Recall + Precision)*
- *Specificity(TNR)=TN/(TN+FP)*
- *FPR=1-Specificity*

Metrics: Accuracy and Error rate

- ***Accuracy = TP+TN/(TP+FP+FN+TN)***

$$acc = \frac{1}{|Te|} \sum_{x \in Te} I[\hat{c}(x) = c(x)]$$

TP	FN
FP	TN

- Here, the function *I* [·] denotes the *indicator function*, which is 1 if its argument evaluates to true, and 0 otherwise.

- Alternatively, we can calculate the *error rate* as the proportion of incorrectly classified instances.
- Clearly, accuracy and error rate sum to 1.

Metrics: Sensitivity and Specificity

- The **True Positive Rate (TPR)** is the proportion of positives correctly classified.

TP	FN
FP	TN

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

Can be defined mathematically as

$$tpr = \frac{\sum_{x \in Te} I[\hat{c}(x) = c(x) = \oplus]}{\sum_{x \in Te} I[c(x) = \oplus]}$$

$$P_{\mathcal{X}}(\hat{c}(x) = \oplus | c(x) = \oplus).$$

- **True Negative Rate (TNR):** is the proportion of negatives correctly classified.

$$\text{TNR} = \frac{TN}{FP+TN}$$

TNR = ?

Metrics: Sensitivity and Specificity

- The **False Positive Rate (FPR)** is the proportion of negatives incorrectly classified.

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

TP	FN
FP	TN

- Can be defined mathematically as

- **False Negative Rate (FNR):** is the proportion of negatives correctly classified.

$$\text{TNR} = \frac{FN}{TP+FN}$$

All Measures

Measure	*Definition*	*Equal to*	*Estimates*
number of positives	$Pos = \sum_{x \in Te} I[c(x) = \oplus]$		
number of negatives	$Neg = \sum_{x \in Te} I[c(x) = \ominus]$	$\lvert Te \rvert - Pos$	
number of true positives	$TP = \sum_{x \in Te} I[\hat{c}(x) = c(x) = \oplus]$		
number of true negatives	$TN = \sum_{x \in Te} I[\hat{c}(x) = c(x) = \ominus]$		
number of false positives	$FP = \sum_{x \in Te} I[\hat{c}(x) = \oplus, c(x) = \ominus]$	$Neg - TN$	
number of false negatives	$FN = \sum_{x \in Te} I[\hat{c}(x) = \ominus, c(x) = \oplus]$	$Pos - TP$	
proportion of positives	$pos = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[c(x) = \oplus]$	$Pos / \lvert Te \rvert$	$P(c(x) = \oplus)$
proportion of negatives	$neg = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[c(x) = \ominus]$	$1 - pos$	$P(c(x) = \ominus)$
class ratio	$clr = pos / neg$	Pos / Neg	

TP	FN
FP	TN

All Measures

Measure	*Definition*	*Equal to*	*Estimates*
(*) accuracy	$acc = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[\hat{c}(x) = c(x)]$		$P(\hat{c}(x) = c(x))$
(*) error rate	$err = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[\hat{c}(x) \neq c(x)]$	$1 - acc$	$P(\hat{c}(x) \neq c(x))$
true positive rate, sensitivity, recall	$tpr = \frac{\sum_{x \in Te} I[\hat{c}(x)=c(x)=\oplus]}{\sum_{x \in Te} I[c(x)=\oplus]}$	TP/Pos	$P(\hat{c}(x) = \oplus \mid c(x) = \oplus)$
true negative rate, specificity, negative recall	$tnr = \frac{\sum_{x \in Te} I[\hat{c}(x)=c(x)=\ominus]}{\sum_{x \in Te} I[c(x)=\ominus]}$	TN/Neg	$P(\hat{c}(x) = \ominus \mid c(x) = \ominus)$
false positive rate, false alarm rate	$fpr = \frac{\sum_{x \in Te} I[\hat{c}(x)=\oplus, c(x)=\ominus]}{\sum_{x \in Te} I[c(x)=\ominus]}$	$FP/Neg = 1 - tnr$	$P(\hat{c}(x) = \oplus \mid c(x) = \ominus)$
false negative rate	$fnr = \frac{\sum_{x \in Te} I[\hat{c}(x)=\ominus, c(x)=\oplus]}{\sum_{x \in Te} I[c(x)=\oplus]}$	$FN/Pos = 1 - tpr$	$P(\hat{c}(x) = \ominus \mid c(x) = \oplus)$
precision, confidence	$prec = \frac{\sum_{x \in Te} I[\hat{c}(x)=c(x)=\oplus]}{\sum_{x \in Te} I[\hat{c}(x)=\oplus]}$	$TP/(TP+FP)$	$P(c(x) = \oplus \mid \hat{c}(x) = \oplus)$

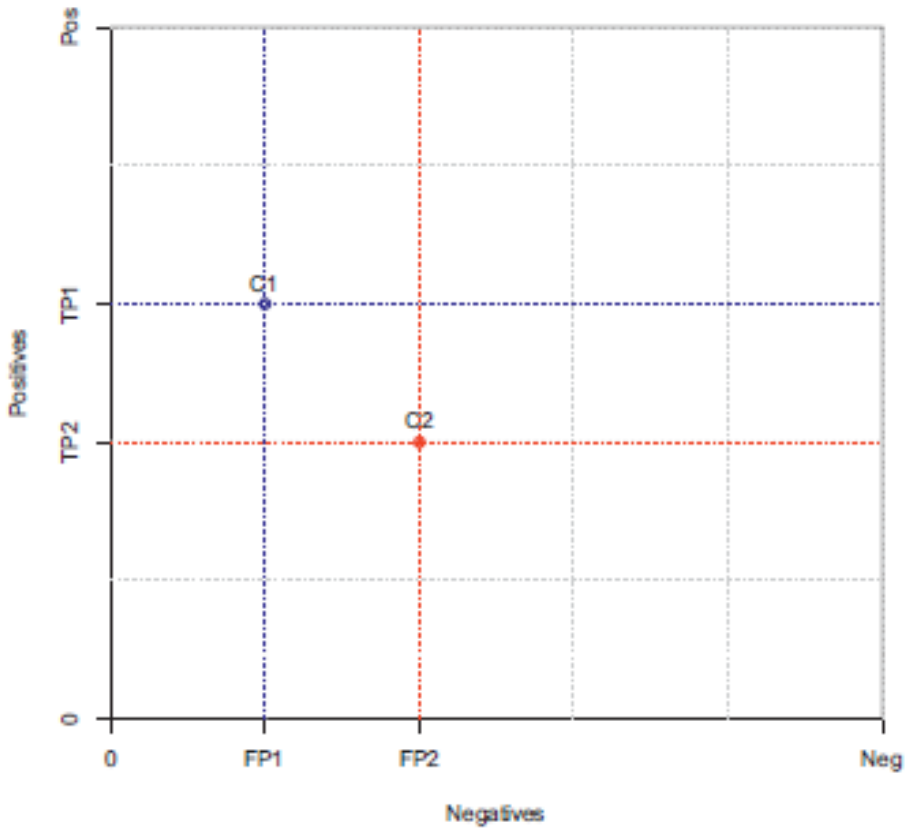
Coverage Plot

- A coverage plot visualizes the
  - number of positives *Pos*,
  - number of negatives *Neg*,
  - number of true positives *TP* and
  - number of false positives *FP*.
- by means of a rectangular coordinate system and a point.

Visualize classification Performance: Coverage Plot

	Predicted ⊕	Predicted ⊖	
Actual ⊕	30	20	50
Actual ⊖	10	40	50
	40	60	100

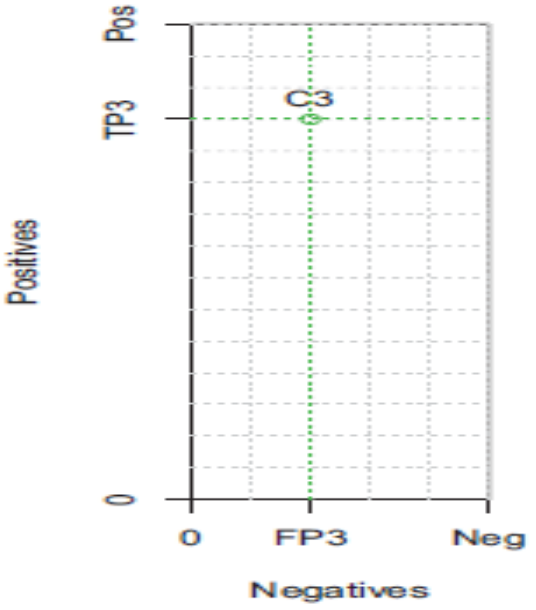
	⊕	⊖	
⊕	20	30	50
⊖	20	30	50
	40	60	100



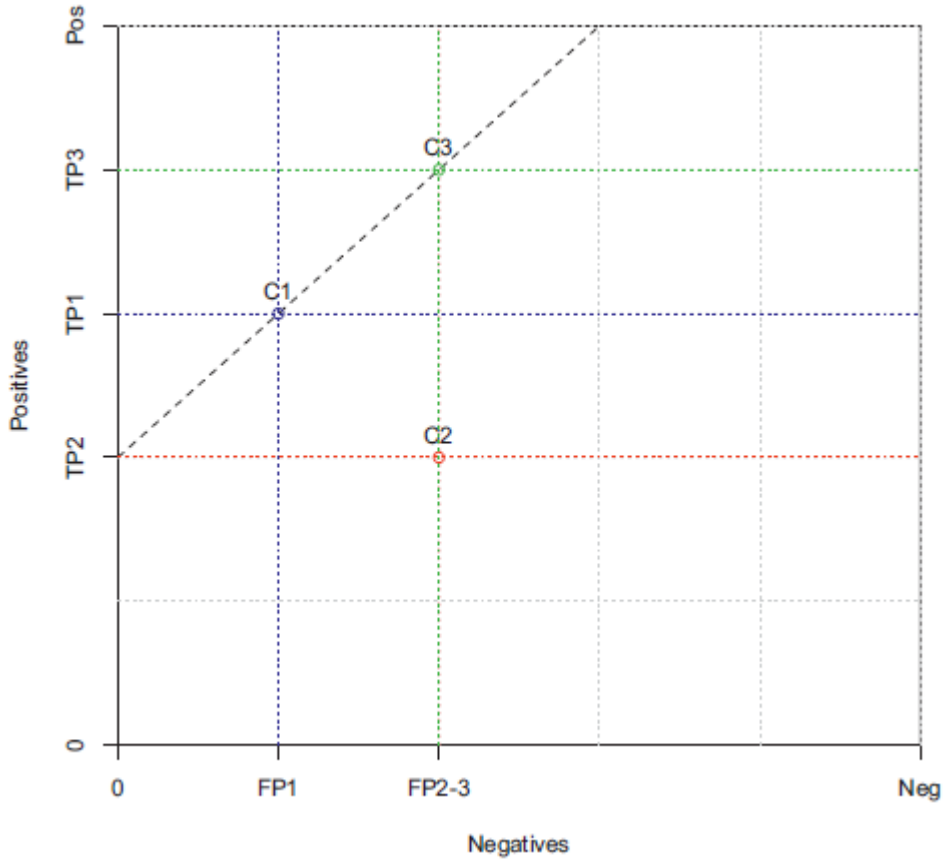
C1 is better than C2

C1 has More TP and Fewer FP as Compared with C2

	Predicted ⊕	Predicted ⊖	
Actual ⊕	60	15	75
Actual ⊖	10	15	25
	70	30	100



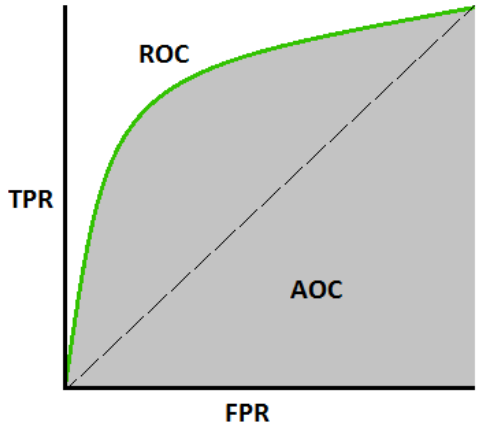
Coverage Plot



- *In a coverage plot, classifiers with the same accuracy are connected by line segments with slope 1.*
- *Here C1 and C3 classifiers have same accuracy.*
- Choice between C1 and C3 is arbitrary; if true positives are more important we should choose C3, if false positives are more important we prefer C1.

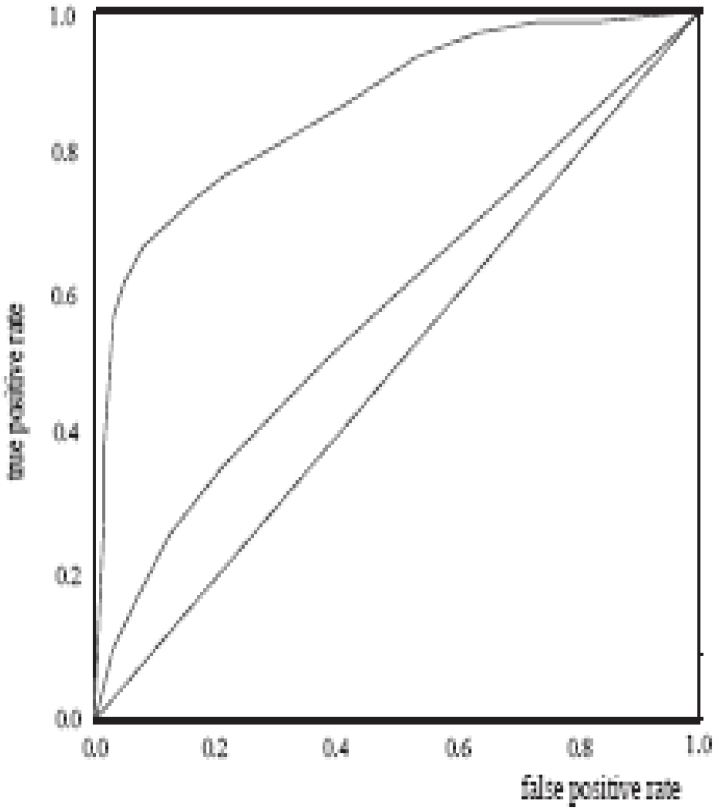
ROC Plot

- ROC stands for *receiver operating characteristic*
- *AOC stands for Area Under the Curve*
- ROC plots are much more common than coverage plots, but both have their specific uses.
- coverage plot is used if you are working with a single data set.
- An ROC plot is useful if you want to combine results from different data sets with different class distributions.



ROC PLOT

- **ROC** (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Shows the **trade-off between the true positive rate and the false positive rate**
- The area under the ROC curve is a measure of the accuracy of the model
- Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model



- A model with perfect accuracy will have an area of 1.0

ROC PLOT Example

Tuple #	Class	Prob.
1	*P*	0.90
2	*P*	0.80
3	*N*	0.70
4	*P*	0.60
5	*P*	0.55
6	*N*	0.54
7	*N*	0.53
8	*N*	0.51
9	*P*	0.50
10	*N*	0.40

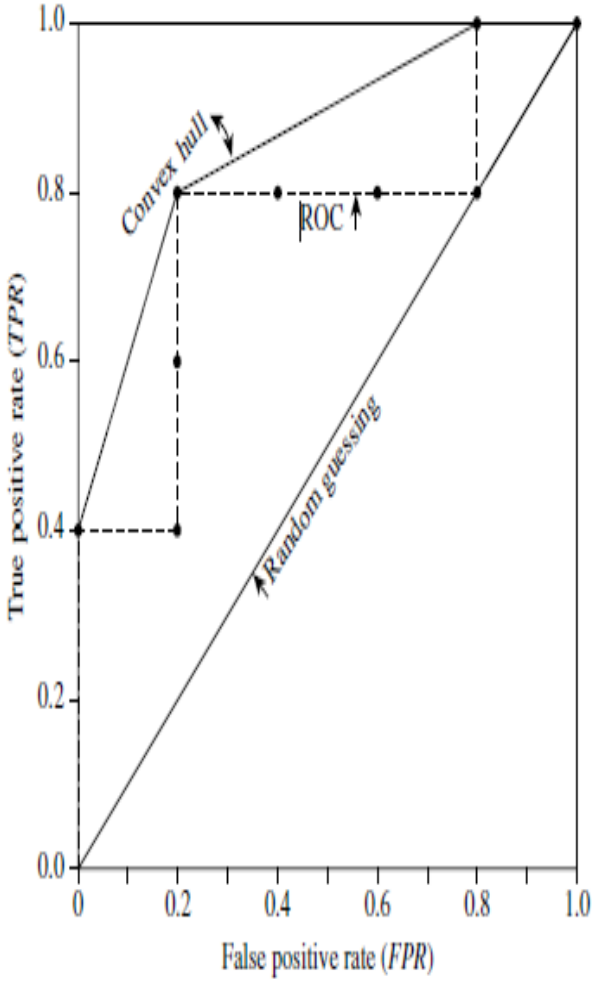
TP	FP	TN	FN	TPR	FPR
1	0	5	4		
2	0	5	3		
2	1	4	3		

Actual Positive tuple P=5
Actual Negative tuples N=5

ROC PLOT Example

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	*P*	0.90	1	0	5	4	0.2	0
2	*P*	0.80	2	0	5	3	0.4	0
3	*N*	0.70	2	1	4	3	0.4	0.2
4	*P*	0.60	3	1	4	2	0.6	0.2
5	*P*	0.55	4	1	4	1	0.8	0.2
6	*N*	0.54	4	2	3	1	0.8	0.4
7	*N*	0.53	4	3	2	1	0.8	0.6
8	*N*	0.51	4	4	1	1	0.8	0.8
9	*P*	0.50	5	4	0	1	1.0	0.8
10	*N*	0.40	5	5	0	0	1.0	1.0

Actual Positive tuple P=5
Actual Negative tuples N=5



Topics

- **Classification**
- **Scoring and Ranking**
- **Class Probability Estimation**

Scoring Classifier

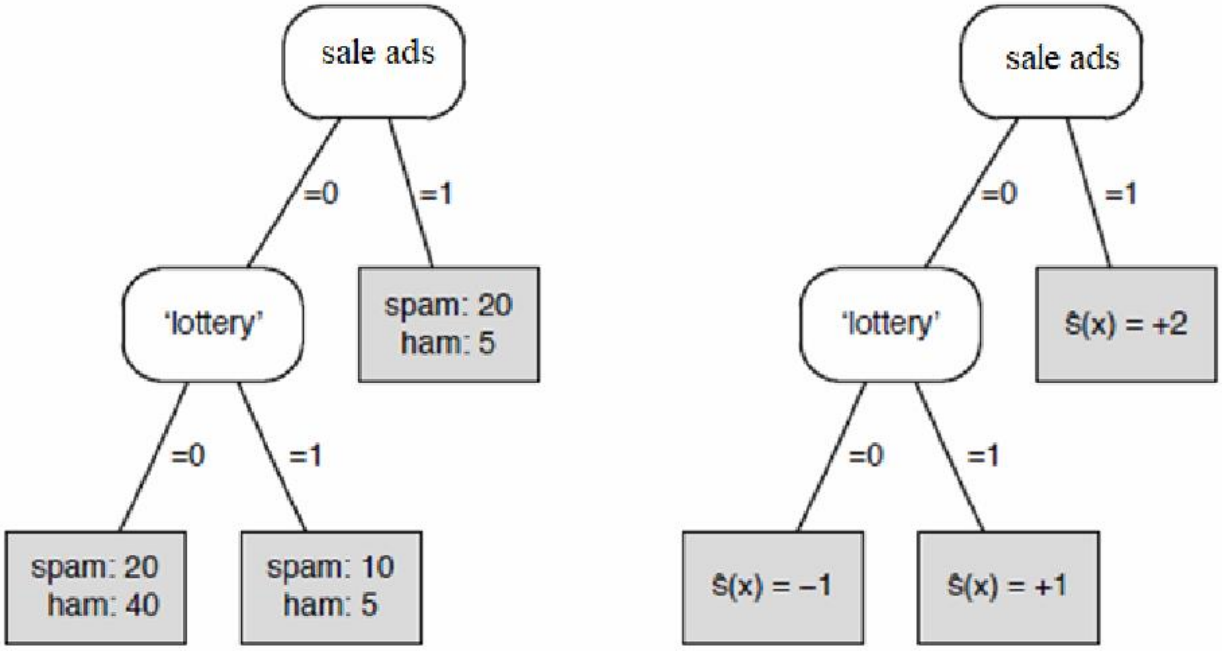
FOR BINARY CLASSIFIER: $\hat{c} : \mathcal{X} \rightarrow \mathcal{C}$, where $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$

A *scoring classifier* is a mapping $\hat{\mathbf{s}} : \mathcal{X} \rightarrow \mathbb{R}^k$, i.e., a mapping from the instance space to a k-vector of real numbers.

The boldface notation indicates that a scoring classifier outputs a vector $\hat{\mathbf{s}}(x) = (\hat{s}_1(x), \ldots, \hat{s}_k(x))$ rather than a single number; $\hat{s}_i(x)$ is the score assigned to class C_i for instance x.

This score indicates how likely it is that class label C_i applies.

Scoring Classifier



(left) A feature tree with training set class distribution in the leaves. **(right)** A scoring tree using the logarithm of the class ratio as scores; spam is taken as the positive class.

Margins and Loss Function

- If we take the true class $c(x)$ as +1 for positive examples and -1 for negative examples, then the quantity $z(x) = c(x) * \hat{s}(x)$ is positive for correct predictions and negative for incorrect predictions.

- The quantity $z(x)$ is called the margin assigned by the scoring classifier.

We would like to reward large positive margins, and penalise large negative values. This is achieved by means of a so-called *loss function* $L : \mathbb{R} \mapsto [0, \infty)$ which maps each example's margin $z(x)$ to an associated loss $L(z(x))$.

Margins and Loss Function

We will assume that $L(0) = 1$, which is the loss incurred by having an example on the decision boundary. We furthermore have $L(z) \geq 1$ for $z < 0$, and usually also $0 \leq L(z) < 1$ for $z > 0$.

The average loss over a test set Te is $\frac{1}{|Te|} \sum_{x \in Te} L(z(x))$.

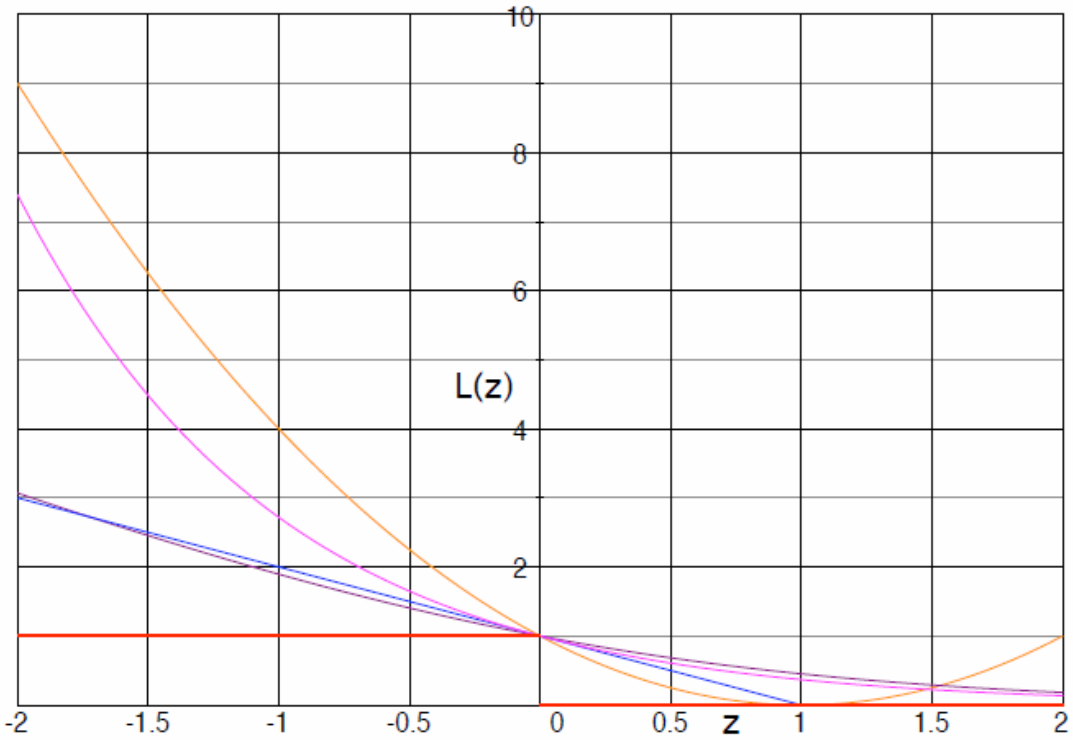
Margins and Loss Function

The simplest loss function is *0–1 loss*, which is defined as $L_{01}(z) = 1$ if $z \leq 0$ and $L(z) = 0$ if $z > 0$. The average 0–1 loss is simply the proportion of misclassified test examples:

$$\frac{1}{|Te|}\sum_{x \in Te} L_{01}(z(x)) = \frac{1}{|Te|}\sum_{x \in Te} I[c(x)\hat{s}(x) \leq 0] = \frac{1}{|Te|}\sum_{x \in Te} I[c(x) \neq \hat{c}(x)] = err$$

where $\hat{c}(x) = +1$ if $\hat{s}(x) > 0$, $\hat{c}(x) = 0$ if $\hat{s}(x) = 0$, and $\hat{c}(x) = -1$ if $\hat{s}(x) < 0$.

Different Loss Functions



From bottom-left (*i*) 0–1 loss $L_{01}(z) = 1$ if $z \leq 0$, and $L_{01}(z) = 0$ if $z > 0$;
(*ii*) hinge loss $L_{\mathrm{h}}(z) = (1 - z)$ if $z \leq 1$, and $L_{\mathrm{h}}(z) = 0$ if $z > 1$;
(*iii*) logistic loss $L_{\log}(z) = \log_2(1 + \exp(-z))$;
(*iv*) exponential loss $L_{\exp}(z) = \exp(-z)$;
(*v*) squared loss $L_{\mathrm{sq}}(z) = (1 - z)^2$ (this can be set to 0 for $z > 1$, just like hinge loss).

Ranking Error and Ranking Accuracy

- Suppose x and x' are two instances such that x receives a lower score:

$$\hat{s}(x) < \hat{s}(x').$$

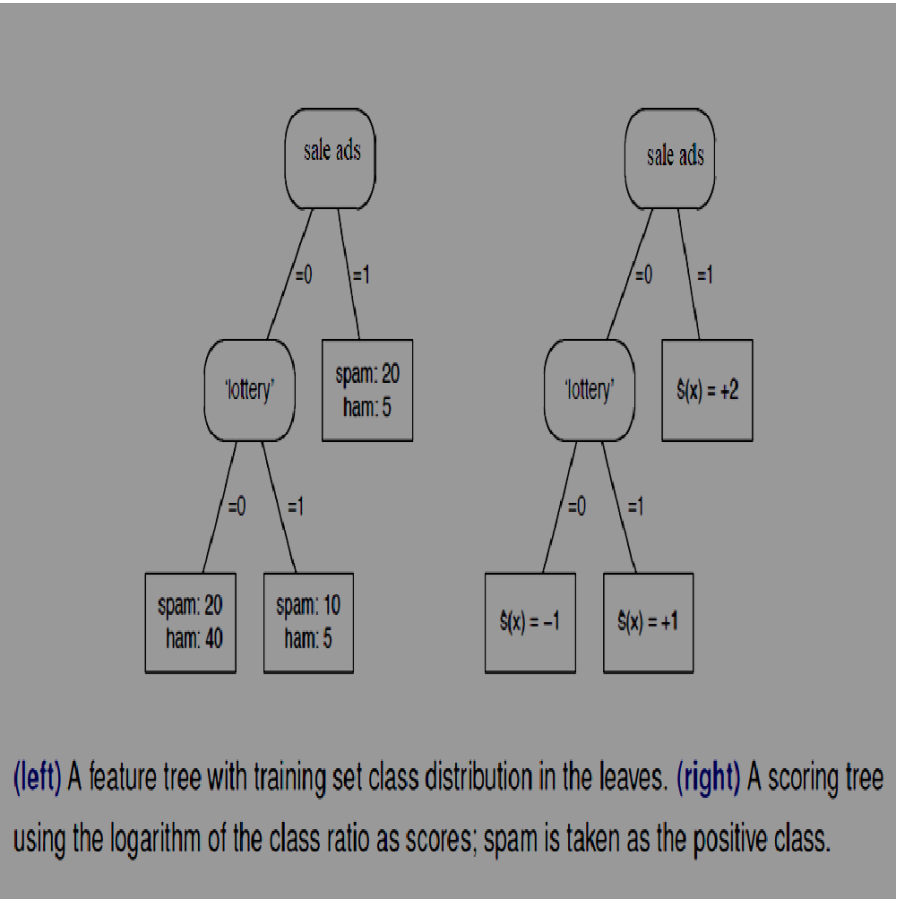
- Higher scores express a stronger belief that the instance in question is positive.
- For the case if x is an actual positive and x' is an actual negative, the expression is wrong.
- This is called as **Ranking Error**

The *ranking error rate* is defined as

$$rank\text{-}err = \frac{\sum_{x \in Te^{\oplus}, x' \in Te^{\ominus}} I[\hat{s}(x) < \hat{s}(x')] + \frac{1}{2} I[\hat{s}(x) = \hat{s}(x')]}{Pos \cdot Neg}$$

$$rank\text{-}acc = \frac{\sum_{x \in Te^{\oplus}, x' \in Te^{\ominus}} I[\hat{s}(x) > \hat{s}(x')] + \frac{1}{2} I[\hat{s}(x) = \hat{s}(x')]}{Pos \cdot Neg} = 1 - rank\text{-}err$$

Example for ranking error



(left) A feature tree with training set class distribution in the leaves. (right) A scoring tree using the logarithm of the class ratio as scores; spam is taken as the positive class.

The *ranking error rate* is defined as

$$\text{rank-err} = \frac{\sum_{x \in Te^{\oplus}, x' \in Te^{\ominus}} I[\hat{s}(x) < \hat{s}(x')] + \frac{1}{2} I[\hat{s}(x) = \hat{s}(x')]}{Pos \cdot Neg}$$

- The 5 negatives in the right leaf are scored higher than the 10 positives in the middle leaf and the 20 positives in the left leaf, resulting in 50+100 = 150 ranking errors.
- The 5 negatives in the middle leaf are scored higher than the 20 positives in the left leaf, giving a further 100 ranking errors.
- In addition, the left leaf makes 800 half ranking errors (because 20 positives and 40 negatives get the same score), the middle leaf 50 and the right leaf 100.
- In total we have 725 ranking errors out of a possible 50 * 50 = 2500, corresponding to a ranking error rate of 29% or a ranking accuracy of 71%.

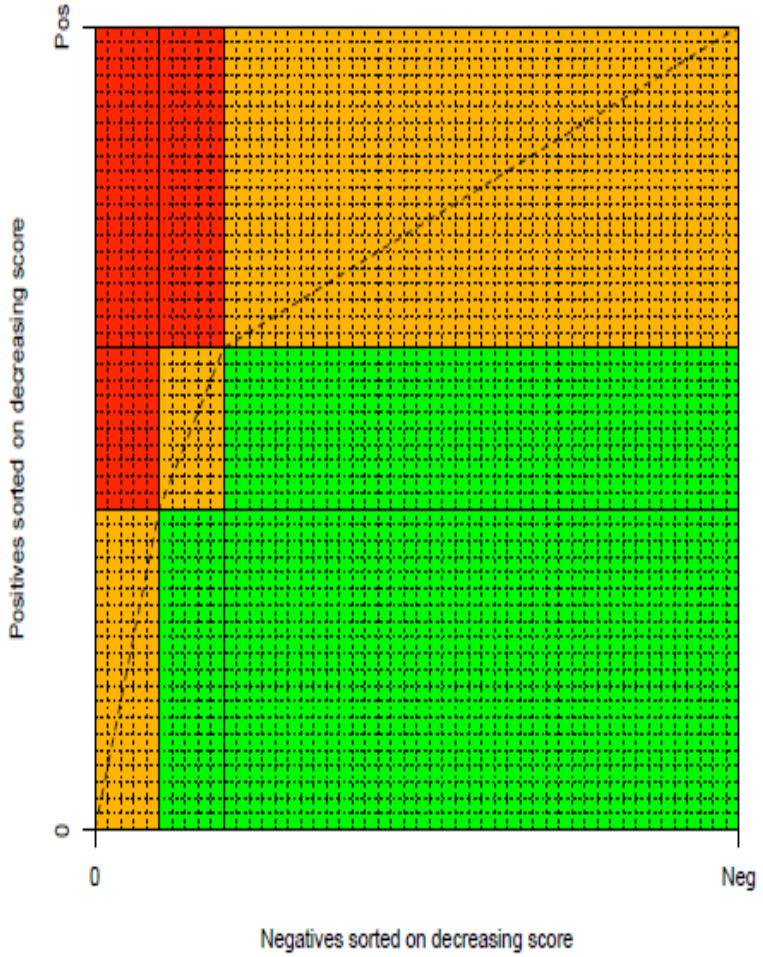
Coverage Plot and ROC Plot

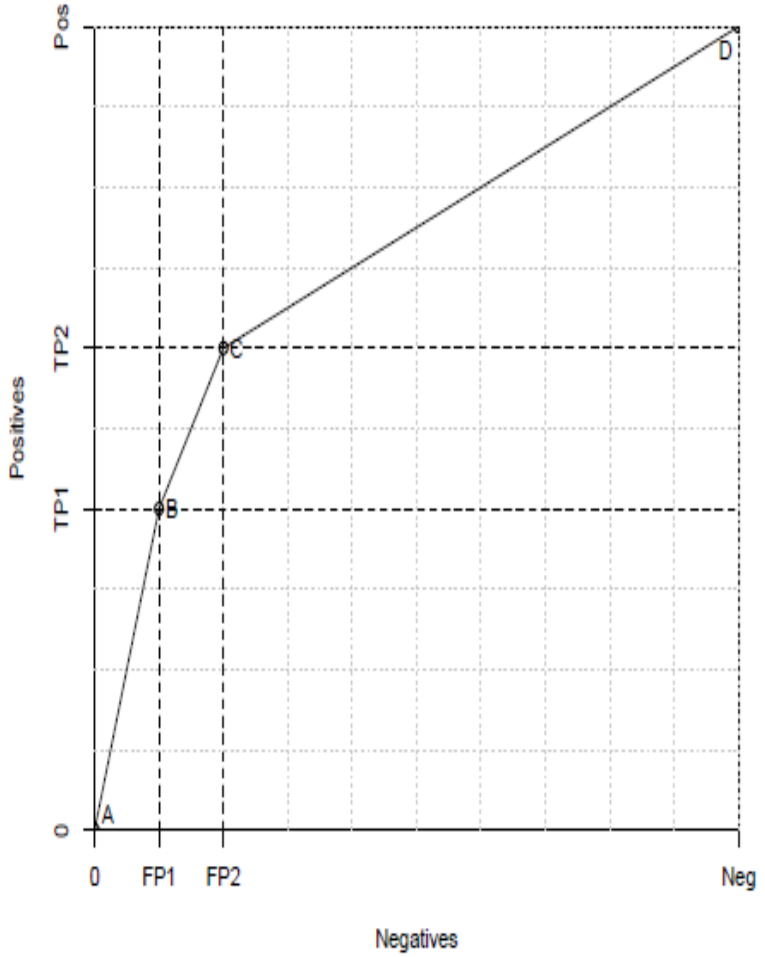
- The coverage plots and ROC plots which are used for visualizing classifier performance also used for visualizing ranking performance too.

- If Pos positives and Neg negatives are plotted on the vertical and horizontal axes, respectively, then each positive–negative pair occupies a unique 'cell' in this plot.

- If we order the positives and negatives on decreasing score, i.e., examples with higher scores are closer to the origin, then we can clearly distinguish the correctly ranked pairs at the bottom right, the ranking errors at the top left, and the ties in between.

Coverage Curve and ROC Curve

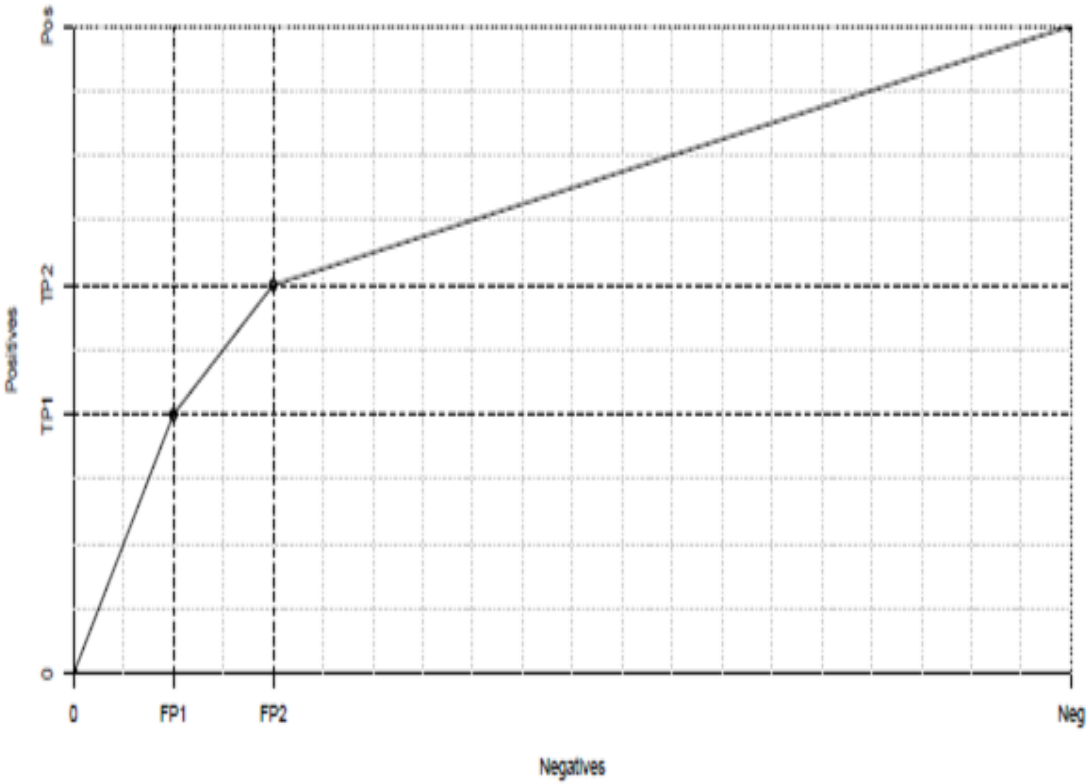


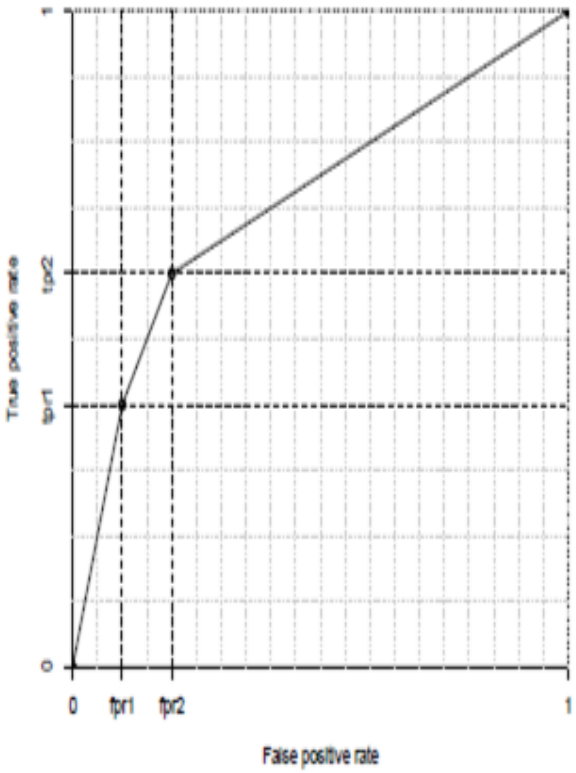


Class imbalance

- Suppose we feed the scoring with an additional batch of 50 negatives.
- The added negatives happen to be identical to the original ones, so the net effect is that the number of negatives in each leaf doubles.
- As a result the coverage curve changes (because the class ratio changes).
- but the ROC curve stays the same.
- Note that the ranking accuracy stays the same as well: while the classifier makes twice as many ranking errors, there are also twice as many positive–negative pairs, so the ranking error rate doesn't change.

Class Imbalance





- (left) A coverage curve obtained from a test set with class ratio clr =1/2.
- (right) The corresponding (axis-normalised) ROC curve is the same as the one corresponding to the coverage curve.
- The ranking accuracy is the Area Under the ROC Curve (AUC).

Topics

- **Classification**
- **Scoring and Ranking**
- **Class Probability Estimation**

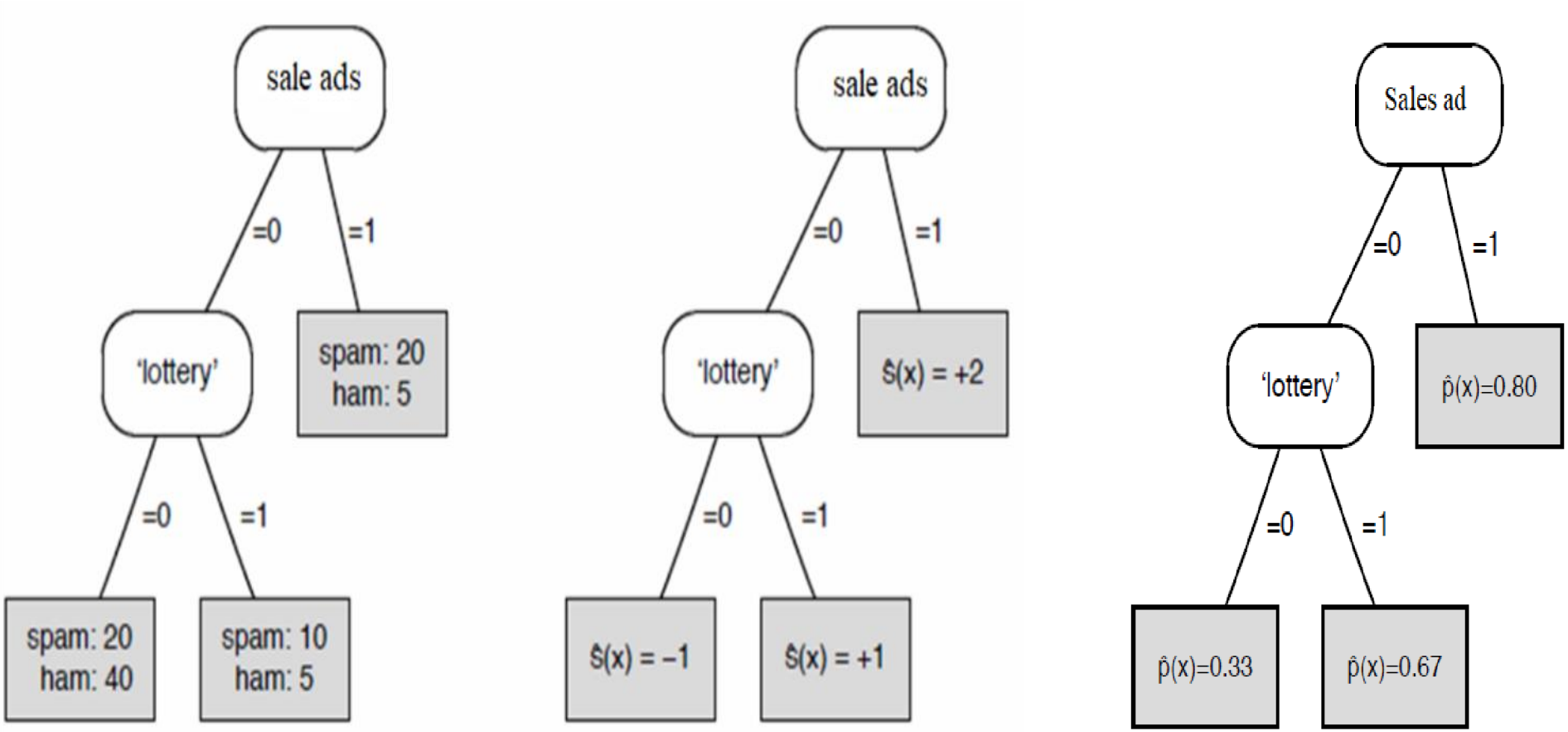
Class Probability Estimation

A *class probability estimator* – or probability estimator in short – is a scoring classifier that outputs probability vectors over classes, i.e., a mapping $\hat{\mathbf{p}} : \mathcal{X} \to [0,1]^k$. We write $\hat{\mathbf{p}}(x) = \big(\hat{p}_1(x), \ldots, \hat{p}_k(x)\big)$, where $\hat{p}_i(x)$ is the probability assigned to class C_i for instance x, and $\sum_{i=1}^{k} \hat{p}_i(x) = 1$.

If we have only two classes, the probability associated with one class is 1 minus the probability of the other class; in that case, we use $\hat{p}(x)$ to denote the estimated probability of the positive class for instance x.

As with scoring classifiers, we usually do not have direct access to the true probabilities $p_i(x)$.

Example


sale ads
=0
=1
'lottery'
spam: 20
ham: 5
=0
=1
spam: 20
ham: 40
spam: 10
ham: 5
sale ads
=0
=1
'lottery'
Ŝ(x) = +2
=0
=1
Ŝ(x) = −1
Ŝ(x) = +1
Sales ad
=0
=1
'lottery'
p̂(x)=0.80
=0
=1
p̂(x)=0.33
p̂(x)=0.67

Assessing Class Probability Estimates

We can define the *squared error* (*SE*) of the predicted probability vector $\hat{\mathbf{p}}(x) = \left(\hat{p}_1(x), \ldots, \hat{p}_k(x)\right)$ as

$$\mathrm{SE}(x) = \frac{1}{2}\sum_{i=1}^{k}(\hat{p}_i(x) - I[c(x) = C_i])^2$$

and the *mean squared error* (*MSE*) as the average squared error over all instances in the test set:

$$\mathrm{MSE}(Te) = \frac{1}{|Te|}\sum_{x \in Te}\mathrm{SE}(x)$$

Assessing Class Probability Estimates

Suppose one model predicts $(0.70, 0.10, 0.20)$ for a particular example x in a three-class task, while another appears much more certain by predicting $(0.99, 0, 0.01)$.

- If the first class is the actual class, the second prediction is clearly better than the first: the SE of the first prediction is $((0.70-1)^2 + (0.10-0)^2 + (0.20-0)^2)/2 = 0.07$, while for the second prediction it is $((0.99-1)^2 + (0-0)^2 + (0.01-0)^2)/2 = 0.0001$. The first model gets punished more because, although mostly right, it isn't quite sure of it.
- However, if the third class is the actual class, the situation is reversed: now the SE of the first prediction is $((0.70-0)^2 + (0.10-0)^2 + (0.20-1)^2)/2 = 0.57$, and of the second $((0.99-0)^2 + (0-0)^2 + (0.01-1)^2)/2 = 0.98$. The second model gets punished more for not just being wrong, but being presumptuous.

Example for Squared Error

Returning to the probability estimation tree in Figure 2.12, we calculate the squared error per leaf as follows (left to right):

$$SE_1 = 20(0.33-1)^2 + 40(0.33-0)^2 = 13.33$$

$$SE_2 = 10(0.67-1)^2 + 5(0.67-0)^2 = 3.33$$

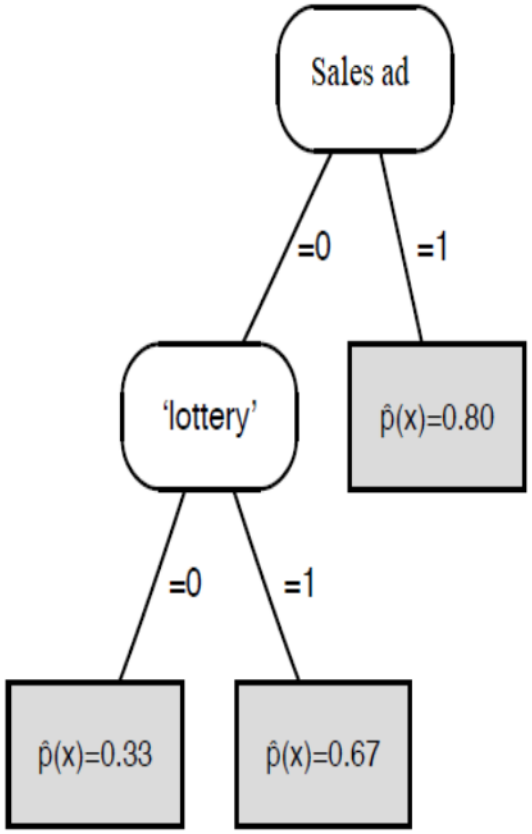
$$SE_3 = 20(0.80-1)^2 + 5(0.80-0)^2 = 4.00$$

which leads to a mean squared error of $MSE = \frac{1}{100}(SE_1 + SE_2 + SE_3) = 0.21$. Changing the predicted probabilities in the left-most leaf to 0.40 for spam and 0.60 for ham, or 0.20 for spam and 0.80 for ham, results in a higher squared error:

$$SE_1' = 20(0.40-1)^2 + 40(0.40-0)^2 = 13.6$$

$$SE_1'' = 20(0.20-1)^2 + 40(0.20-0)^2 = 14.4$$

Predicting probabilities obtained from the class distributions in each leaf is optimal in the sense of lowest MSE.



Why predicting empirical probabilities is optimal

The reason for this becomes obvious if we rewrite the expression for two-class squared error of a leaf as follows, using the notation $n^{\oplus}$ and $n^{\ominus}$ for the numbers of positive and negative examples in the leaf:

$$n^{\oplus}(\hat{p}-1)^2 + n^{\ominus}\hat{p}^2 = (n^{\oplus}+n^{\ominus})\hat{p}^2 - 2n^{\oplus}\hat{p} + n^{\oplus} = (n^{\oplus}+n^{\ominus})\left[\hat{p}^2 - 2\dot{p}\hat{p} + \dot{p}\right]$$
$$= (n^{\oplus}+n^{\ominus})\left[(\hat{p}-\dot{p})^2 + \dot{p}(1-\dot{p})\right]$$

where $\dot{p} = n^{\oplus}/(n^{\oplus}+n^{\ominus})$ is the relative frequency of the positive class among the examples covered by the leaf, also called the *empirical probability*. As the term $\dot{p}(1-\dot{p})$ does not depend on the predicted probability $\hat{p}$, we see immediately that we achieve lowest squared error in the leaf if we assign $\hat{p} = \dot{p}$.

Smoothing empirical probabilities

Empirical probabilities are important as they allow us to obtain or finetune probability estimates from classifiers or rankers. If we have a set S of labelled examples, and the number of examples in S of class C_i is denoted n_i, then the empirical probability vector associated with S is $\dot{\mathbf{p}}(S) = (n_1/|S|, \ldots, n_k/|S|)$. In practice, it is almost always a good idea to *smooth* these relative frequencies to avoid issues with extreme values (0 or 1). The most common way to do this is to set

$$\dot{p}_i(S) = \frac{n_i + 1}{|S| + k}$$

This is called the *Laplace correction*

Smoothing empirical probabilities

We can also apply non-uniform smoothing by setting

$$\dot{p}_i(S) = \frac{n_i + m \cdot \pi_i}{|S| + m}$$

This smoothing technique, known as the *m-estimate*, allows the choice of the number of pseudo-counts m as well as the prior probabilities π_i. The Laplace correction is a special case of the m-estimate with $m = k$ and $\pi_i = 1/k$.

Calibration Loss and Refinement Loss

- If all elements of S receive the same predicted probability vector ˆp(S) – which happens if S is a segment of a grouping model – then a similar derivation to the one above allows us to write the total incurred squared error over *S* in terms of estimated and empirical probabilities as

$$\mathrm{SE}(S) = \sum_{x \in S} \mathrm{SE}(x) = \sum_{x \in S} \frac{1}{2} \sum_{i=1}^{k} (\hat{p}_i(x) - I[c(x) = C_i])^2$$

$$= \frac{1}{2}|S| \sum_{i=1}^{k} (\hat{p}_i(S) - \dot{p}_i(S))^2 + \frac{1}{2}|S| \sum_{i=1}^{k} (\dot{p}_i(S)(1 - \dot{p}_i(S))$$

Calibration Loss and Refinement Loss

$$\mathrm{SE}(S) = \sum_{x \in S} \mathrm{SE}(x) = \sum_{x \in S} \frac{1}{2} \sum_{i=1}^{k} (\hat{p}_i(x) - I[c(x) = C_i])^2$$

$$= \frac{1}{2}|S| \sum_{i=1}^{k} (\hat{p}_i(S) - \dot{p}_i(S))^2 + \frac{1}{2}|S| \sum_{i=1}^{k} (\dot{p}_i(S)(1 - \dot{p}_i(S))$$

- The first term of the final expression is called the calibration loss, and measures squared error with respect to the empirical probabilities.
- Models with low calibration loss are said to be well calibrated.
- The second term is called the refinement loss; this depends only on the empirical probabilities, and is smaller if they are less uniform.

Topics

- **Classification**
- **Scoring and Ranking**
- **Class Probability Estimation**