Supervised Learning
Classification

Alexander Trofimov
PhD, professor, NRNU MEPhI

lab@neuroinfo.ru
http://datalearning.ru

Course “Machine Learning”

October 2021

Classification. Problem Statement

Given:

$\mathscr{D} = \{(x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})\}$ — available data sample

$\mathscr{Y} = \{1, ..., K\}$ — class labels

$f_1(x), ..., f_M(x)$ — features of object $x \in \mathscr{X}$

To build learning algorithm we must specify:

Class of hypotheses $\mathscr{H} = \{h : h(x), h(x) \in \mathscr{Y}\}$

Loss function $L(h, (x, y)) \in \mathbb{R}^+, \quad h \in \mathscr{H}, \quad x \in \mathscr{X}, \quad y \in \mathscr{Y}$

Find out:

Construct hypothesis $h \in \mathscr{H}$ that minimizes empirical risk over train sample: $R^*(h) \to \min\limits_{h \in \mathscr{H}}$

$K = 2 \Rightarrow$ binary classification problem

$K > 2 \Rightarrow$ multiclass classification problem

Binary Classification Problem

Let's denote class labels as –1 (negative class) and +1 (positive class): $\mathscr{Y} = \{-1, 1\}$

Suppose that:
Class of hypotheses $\mathscr{H} = \{h : h(x) = \operatorname{sign} \varphi(x, w)\}$,

where $\varphi(x, w) \in \mathbb{R}$ — classification score for object $x \in \mathscr{X}$

$w \in \mathbb{R}^L$ — vector of parameters

Definition

Classification margin $m((x, y), w)$ at $x \in \mathscr{X}$ is defined as product

$$m((x, y), w) = y\varphi(x, w)$$

Positive values of $m((x, y), w)$ indicate correct classification
Negative values — incorrect classification

Classification Loss Functions

Definition

Classification loss function $L(h, (x, y)) \in \mathbb{R}^+$ at $x \in \mathscr{X}$ is some measure of predictive inaccuracy of classification model $h \in \mathscr{H}$ at $x \in \mathscr{X}$

When comparing the same type of loss among many classification models, lower loss indicates a better classification model

Let's define loss as a function of margin $L(m)$:

$m > 0 \Rightarrow$ correct classification, do not have to contribute much to the loss

$m < 0 \Rightarrow$ incorrect classification, have to contribute to the loss

Types of Loss Functions

- 0-1 loss

$$L(m) = \begin{cases} 1, & m \leq 0, \\ 0, & m > 0. \end{cases}$$

  Robust to outliers but hard to optimize

- Hinge loss

$$L(m) = \max(0, 1 - m)$$

  Equals to 0-1 loss when $m \geq 1$, wrong classified points penalized in a linear fashion, not smooth and cannot be used with gradient descent methods

- Logistic loss

$$L(m) = \ln\left(1 + e^{-m}\right)$$

  Does not assign zero penalty to any points, sensitive to outliers in the data, smooth, gradient descent methods can be utilized

Types of Loss Functions

- Exponential loss

$$L(m) = e^{-m}$$

Does not assign zero penalty to any points, strong penalty for wrong classified points with high score

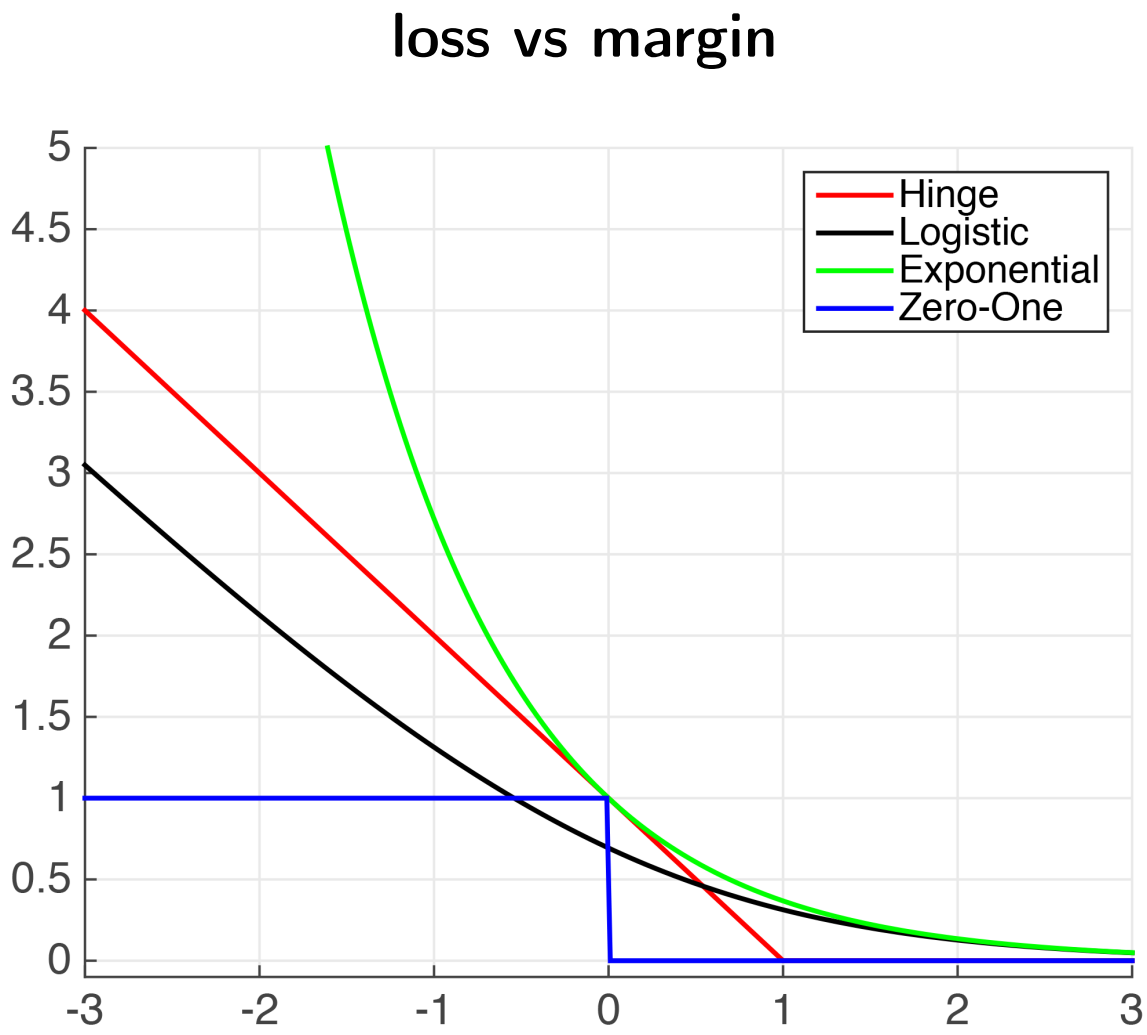
- Quadratic loss

$$L(m) = (1 - m)^2$$

Equals to 0-1 loss when $m = 0$ and $m = 1$, simple, smooth, tends to penalize outliers excessively, appropriate if $\varphi(x)$ doesn't yield high values, correctly classified points with high score will be penalized

Truncated quadratic loss: $L(m) = \begin{cases} (1 - m)^2, & m \leq 1, \\ 0, & otherwise. \end{cases}$

Loss Functions. Plots



Statistical View

$\mathscr{D} = \{(x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})\}$ — sample drawn from joint probability distribution $f_{XY}(x, y)$

Suppose that $f_{XY}(x, y)$ is parametrized by vector $w \in \mathbb{R}^L$: $f_{XY}(x, y|w)$

To estimate w given $\mathscr{D}$ we use method of maximum likelihood

Likelihood $\mathscr{L}(\mathscr{D}, w) = \prod_{i=1}^{n} f_{XY}\left(x^{(i)}, y^{(i)}|w\right)$

Maximum likelihood estimator (MLE) of w is a solution of optimization problem:

$$\mathscr{L}(\mathscr{D}, w) \to \max_w$$

Log-likelihood:

$$\ln \mathscr{L}(\mathscr{D}, w) = \sum_{i=1}^{n} \ln f_{XY}\left(x^{(i)}, y^{(i)}|w\right) \to \max_w$$

MLE and ERM Principle

ERM principle: $R^*(h) = \sum_{i=1}^{n} L\left(m\left((x^{(i)}, y^{(i)}), w\right)\right) \to \min\limits_{w}$

MLE: $-\ln \mathscr{L}(\mathscr{D}, w) = -\sum_{i=1}^{n} \ln f_{XY}\left(x^{(i)}, y^{(i)}|w\right) \to \min\limits_{w}$

These problems are equivalent if

$$-\ln f_{XY}\left(x^{(i)}, y^{(i)}|w\right) = L\left(m\left((x^{(i)}, y^{(i)}), w\right)\right)$$

The loss function is related to probabilistic model of the data:

we define $L(m) \Rightarrow$ we impose some probabilistic model in $\mathscr{X} \times \mathscr{Y}$

we define $f_{XY}(x, y|w) \Rightarrow$ we impose loss function $L(m)$

Bayes' theorem

Suppose that vector $w \in \mathbb{R}^L$ is a random vector drawn from distribution $f_W(w)$

Then joint probability distribution of random vector (X, Y, W):

$$f_{XYW}(x, y, w) = f_{XY}(x, y|w) f_W(w)$$

Bayes' theorem:

$$f_W(w|x, y) = \frac{f_{XY}(x,y|w) f_W(w)}{f_{XY}(x,y)}$$

$f_W(w)$ — prior distribution of parameter W
$f_W(w|x, y)$ — posterior distribution of W given the data (x, y)
$f_{XY}(x, y|w)$ — likelihood (conditional distribution of (X, Y))
$f_{XY}(x, y)$ — evidence (marginal distribution of (X, Y))

Bayes' theorem: $posterior = \frac{likelihood \times prior}{evidence}$

Bayesian Estimator

$\mathscr{D} = \{(x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})\}$ — sample drawn from joint probability distribution $f_{XY}(x, y)$

Most probable bayesian estimator of parameters $w \in \mathbb{R}^L$ is such that maximizes $f_W(w|\mathscr{D})$:

$$f_W(w|\mathscr{D}) = \frac{\mathscr{L}(\mathscr{D}|w) f_W(w)}{\mathscr{L}(\mathscr{D})} \to \max_w$$

$\mathscr{L}(\mathscr{D})$ is a joint distribution of all samples given size n, doesn't depend on w, can be skipped from optimization:

$$\mathscr{L}(\mathscr{D}|w) f_W(w) \to \max_w$$

For uniform prior distribution $f_W(w) = const$:

$$\mathscr{L}(\mathscr{D}|w) \to \max_w$$

Bayesian estimator coincides with the maximum likelihood estimator for a uniform prior

Bayesian Estimator and ERM Principle

Assume the prior distribution $f_W(w)$ of parameters $w \in \mathbb{R}^L$ to be non-uniform

The optimization problem

$$f_W(w|\mathscr{D}) = \mathscr{L}(\mathscr{D}|w) f_W(w) \to \max_w$$

is equivalent to

$$-\ln f_W(w|\mathscr{D}) = -\ln \mathscr{L}(\mathscr{D}|w) - \ln f_W(w) \to \min_w$$

$$-\ln f_W(w|\mathscr{D}) = -\sum_{i=1}^n \ln f_{XY}\left(x^{(i)}, y^{(i)}|w\right) - \ln f_W(w) \to \min_w$$

ERM principle:

$$R^*(h) = \sum_{i=1}^n L\left(m\left((x^{(i)}, y^{(i)}), w\right)\right) \to \min_w$$

Regularized Empirical Risk

Regularized empirical risk:

$$R'(h) = R^*(h) + r(w)$$

$r(w)$ — regularizer (doesn't depend on data)

Assume

$$-\ln f_{XY}\left(x^{(i)}, y^{(i)}|w\right) = L\left(m\left((x^{(i)}, y^{(i)}), w\right)\right)$$

$$-\ln f_W(w) = r(w)$$

then the Bayesian estimation problem and the regularized empirical risk minimization problem are equivalent

we define $r(w) \Leftrightarrow$ we impose some prior distribution $f_W(w)$

Gaussian Prior

$W_1, ..., W_L$ — independent and identically distributed (i.i.d.) random variables, $W_i \sim N(0, \sigma), \quad i = 1, ..., L$:

$$f_{W_i}(w) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{w^2}{2\sigma^2}\right)$$

Joint probability distribution:

$$f_W(w) = \frac{1}{(\sigma\sqrt{2\pi})^L} \exp\left(-\frac{w_1^2+...+w_L^2}{2\sigma^2}\right)$$

L_2-regularizer:

$$r(w) = -\ln f_W(w) = \frac{1}{2\sigma^2} \sum_{i=1}^{L} w_i^2 + const$$

Advantage: smooth, simple
Disadvantage: tends to make all parameters close to zero

Laplacian Prior

$W_1, ..., W_L$ — i.i.d. random variables, $W_i \sim L(0, b), \quad i = 1, ..., L$

$$f_{W_i}(w) = \frac{1}{2b} \exp\left(-\frac{|w|}{b}\right)$$

Joint probability distribution:

$$f_W(w) = \frac{1}{(2b)^L} \exp\left(-\frac{|w_1|+...+|w_L|}{b}\right)$$

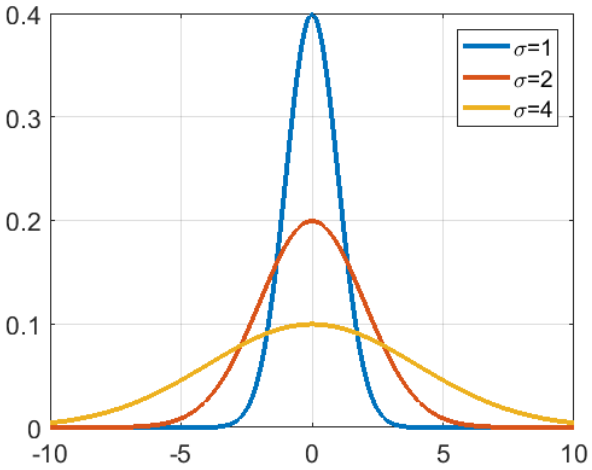
L_1-regularizer:

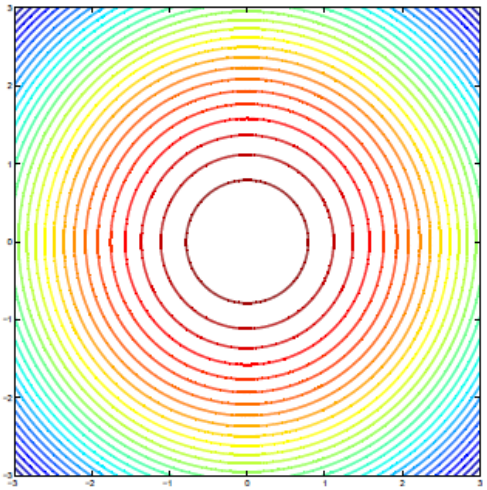
$$r(w) = -\ln f_W(w) = \frac{1}{2b} \sum_{i=1}^{L} |w_i| + const$$

Advantage: towering peak and heavy tails
Disadvantage: not smooth

Gaussian vs Laplacian Priors

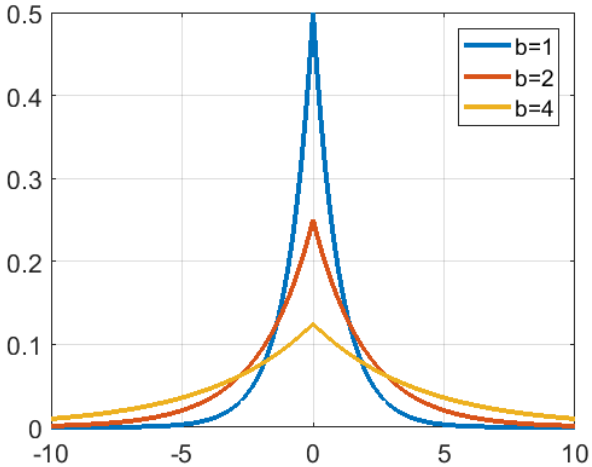
Gaussian

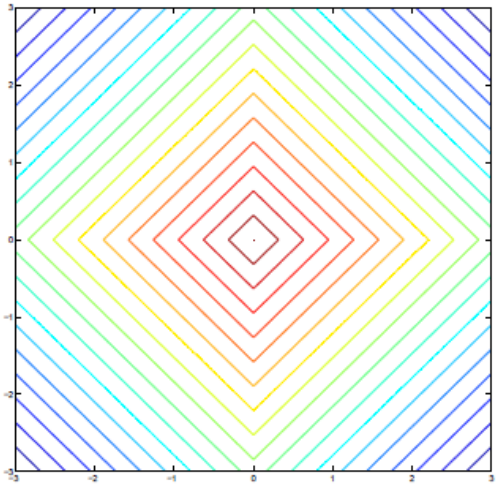




Euclidean metric

Laplace

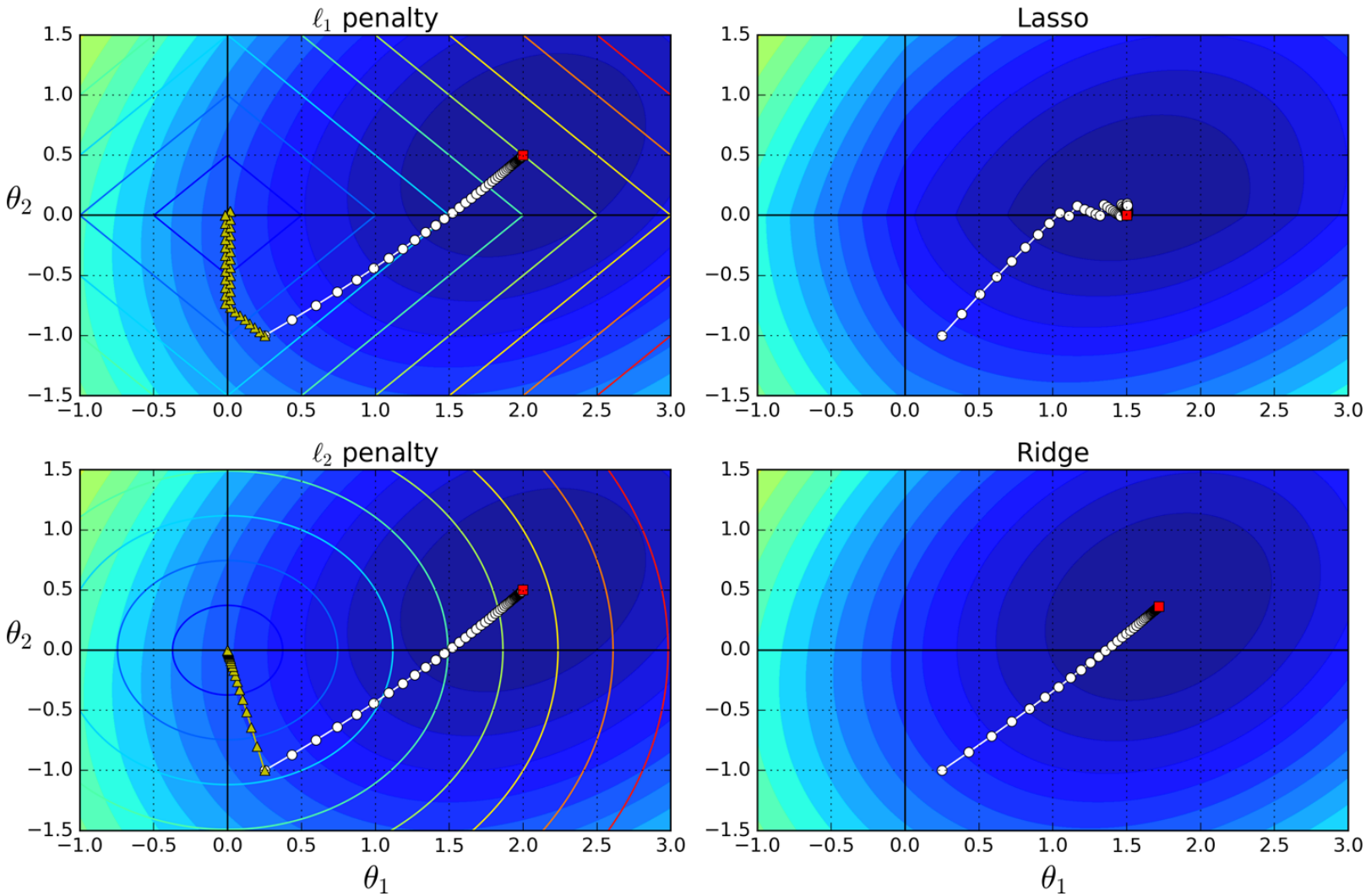




City-block metric

Regularized ERM. Illustration

Regularized ERM: $R'(h) = R^*(h) + r(w) \to \min_w$



Supervised Learning Methods

Regression Methods

Loss function	Regularizer	Method
Squared loss	—	Ordinary regression
Squared loss	L_2	Ridge regression
Squared loss	L_1	LASSO regression
Squared loss	$L_2 + L_1$	Elastic net regression
Hinge loss	L_2	SVM regression

Classification Methods

Loss function	Regularizer	Method
0-1 loss	—	Bayesian classifier
Hinge loss	L_2	Standard SVM
Hinge loss	L_1	LASSO SVM
Logistic loss	L_2	Logistic regression
Exponential loss	—	AdaBoost classifier

Estimation of Classification Error

$\mathscr{D} = (\mathscr{D}_T \cup \mathscr{D}_V) \cup \mathscr{D}_{Tst}$ — available data
$h \in \mathscr{H}$ — model trained on $\mathscr{D}_T$

How to estimate the error of the model h?

Empirical risk $R^*_{Tst}(h)$ over test set is not sufficient to estimate the error of the model h!

The reason is discrete character of responses

Example
Suppose that population consists of 90% positive cases and 10% negative cases

The model $h(x) \equiv 1$ is correct for 90% cases!
But... is $h(x)$ a good model?

Measures of Classification Performance

We need additional metrics to evaluate classification performance

- Confusion matrix based measures
  — Measure the performance of given classifier h
  — Deal with different types of binary classification outcomes
  — Derived from confusion matrix
- Model-wide measures
  — Measure the performance of parametrized set of classifiers $\{h_b, b \in \mathbb{R}\}$, not of given classifier h
  — Calculate multiple confusion matrix based measures for many $b \in \mathbb{R}$
  — Measure the separability of trained classification scores

Confusion Matrix

		Prediction	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

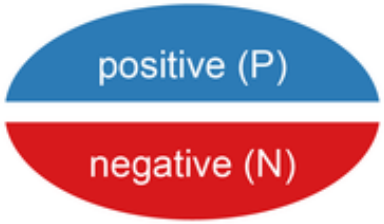
Four outcomes of classification:

- TP — True Positive
- FP — False Positive
- TN — True Negative
- FN — False Negative

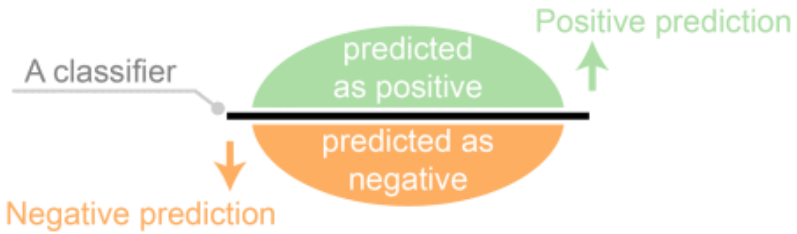
TP — actually positive, are correctly included in the positive class
FP — actually negative, are incorrectly included in the positive class
TN — actually negative, are correctly included in the negative class
FN — actually positive, are incorrectly included in the negative class

Confusion Matrix. Illustration

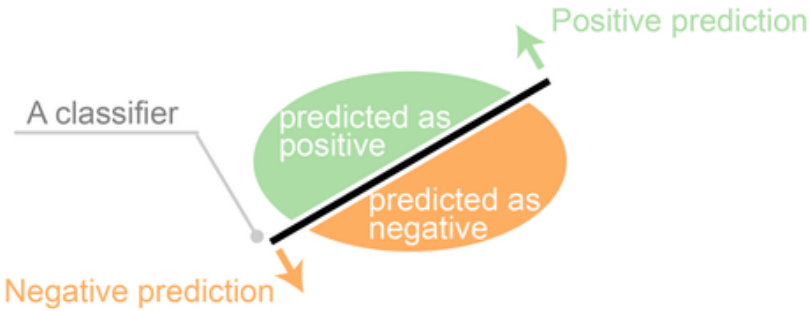
Two actual classes or observed labels



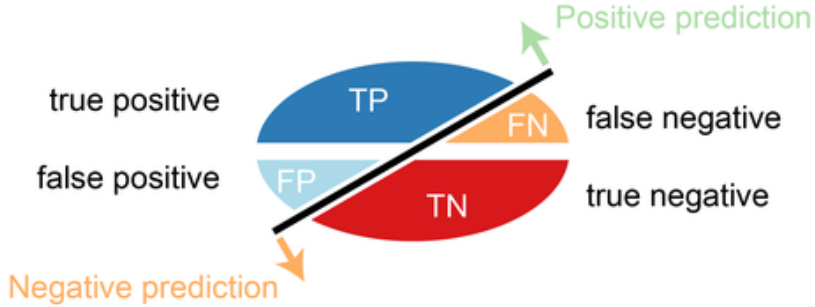
Predicted classes of a perfect classifier



Predicted classes of a classifier



Four outcomes of a classifier

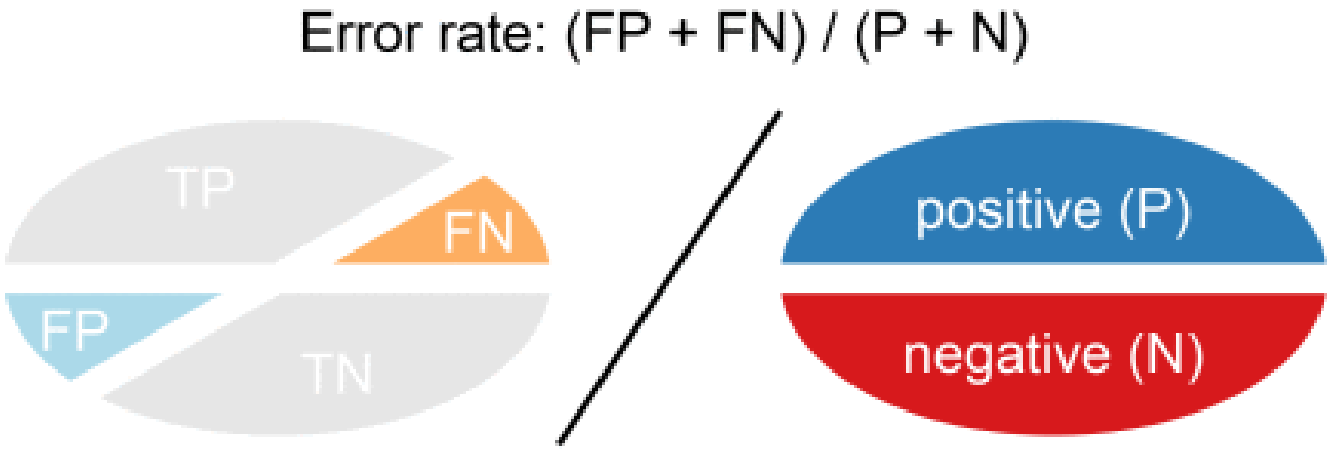


Error Rate

Error rate (ERR) is the number of all incorrect predictions divided by the total number of cases:

$$ERR = \frac{FP + FN}{TP + FN + FP + TN}$$

The best error rate is 0.0, the worst is 1.0

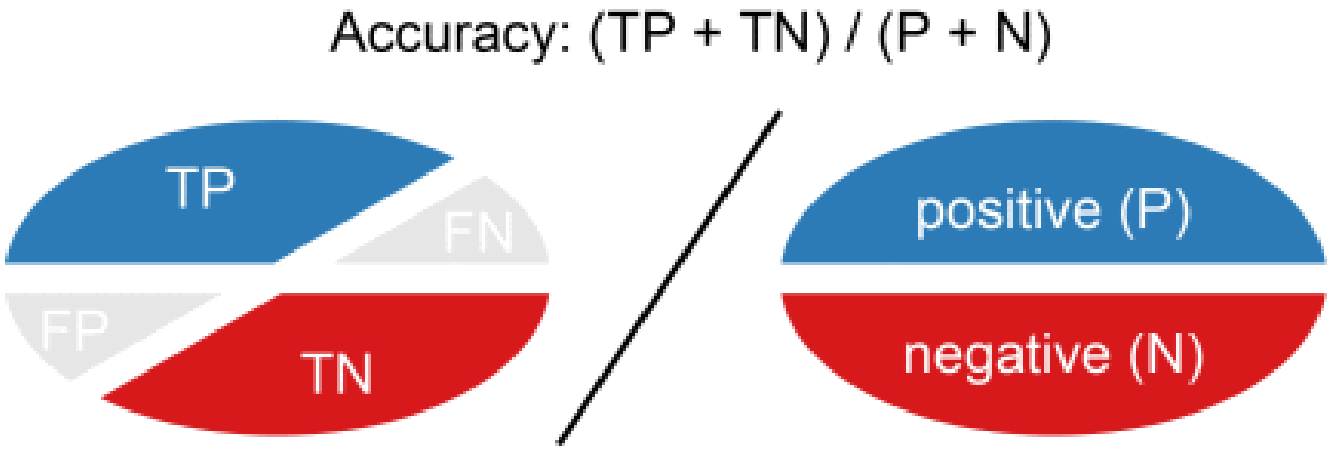


Accuracy

Accuracy (ACC) is the number of all correct predictions divided by the total number of cases:

$$ACC = \frac{TP + TN}{TP + FN + FP + TN} = 1 - ERR$$

The best accuracy is 1.0, the worst is 0.0



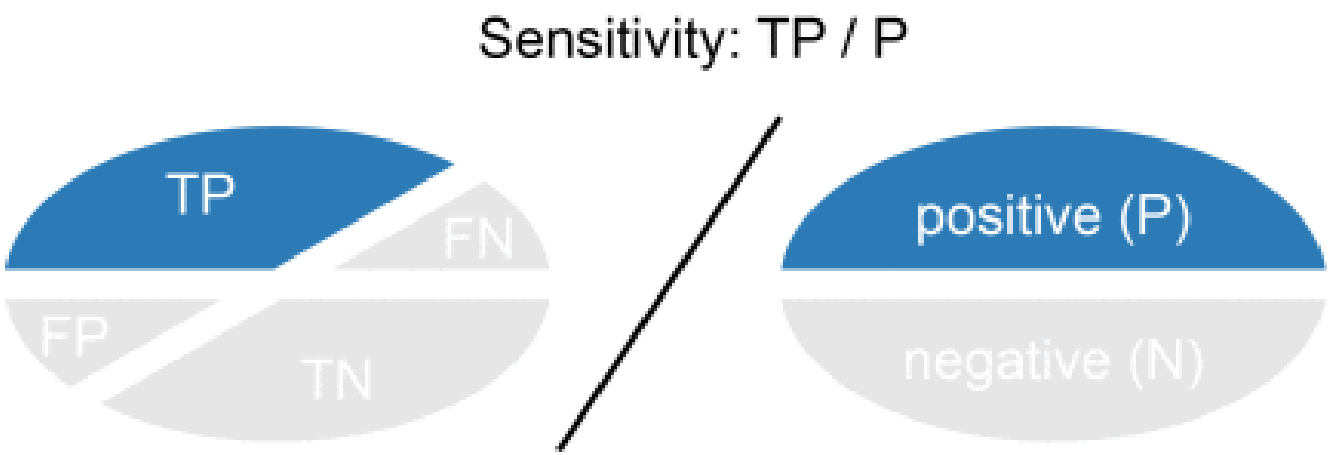
Sensitivity (Recall)

Sensitivity (SENS) is the number of correct positive predictions divided by the total number of positives:

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

It is also called recall (REC) or true positive rate (TPR)

The best sensitivity is 1.0, the worst is 0.0



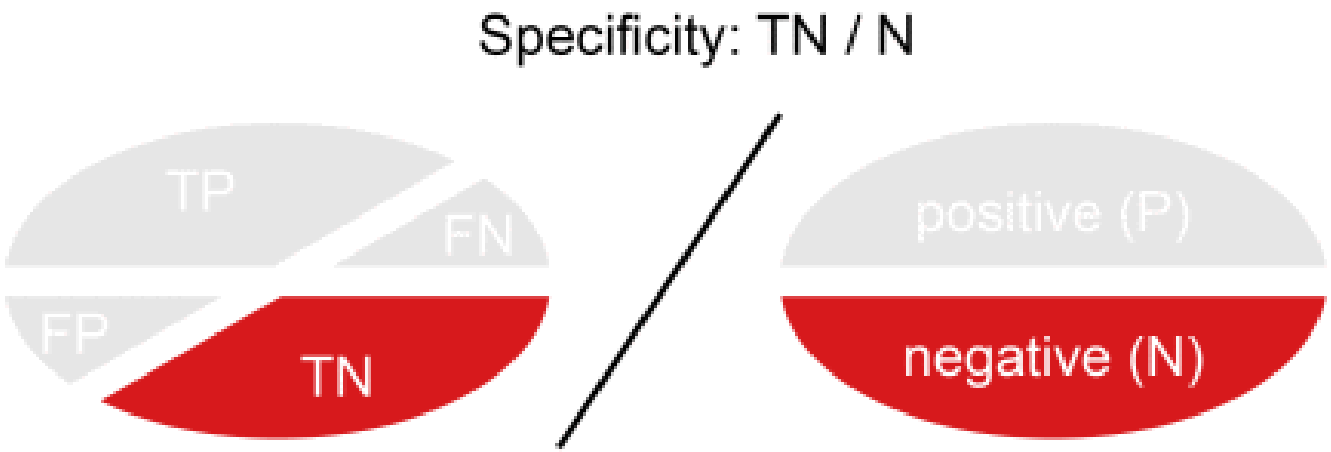
Specificity

Specificity (SPEC) is the number of correct negative predictions divided by the total number of negatives:

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

It is also called true negative rate (TNR)

The best specificity is 1.0, the worst is 0.0



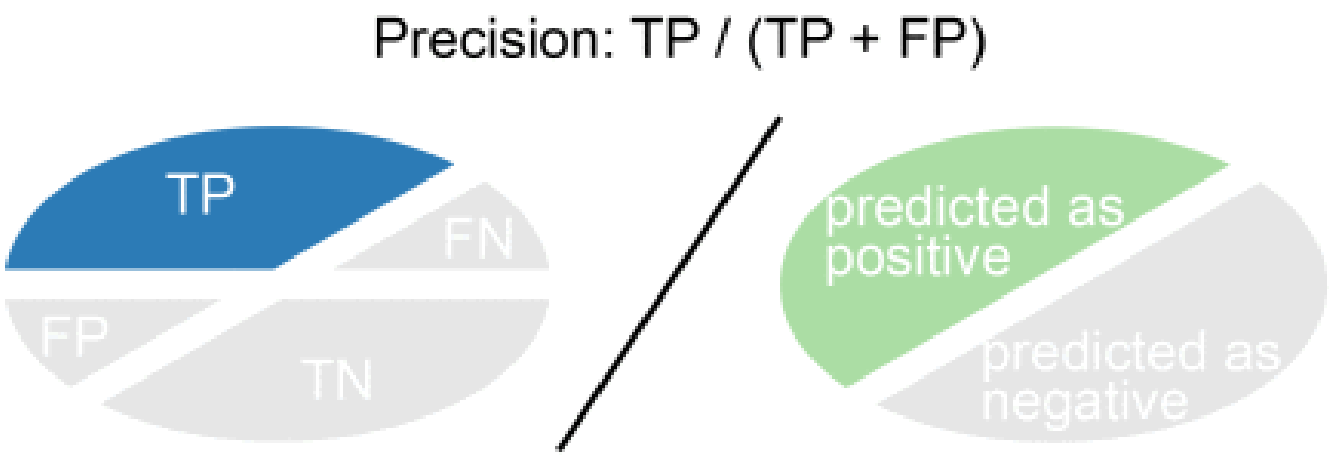
Precision

Precision (PREC) is the number of correct positive predictions divided by the total number of positive predictions:

$$PREC = PPV = \frac{TP}{TP + FP}$$

It is also called positive predictive value (PPV)

The best precision is 1.0, whereas the worst is 0.0



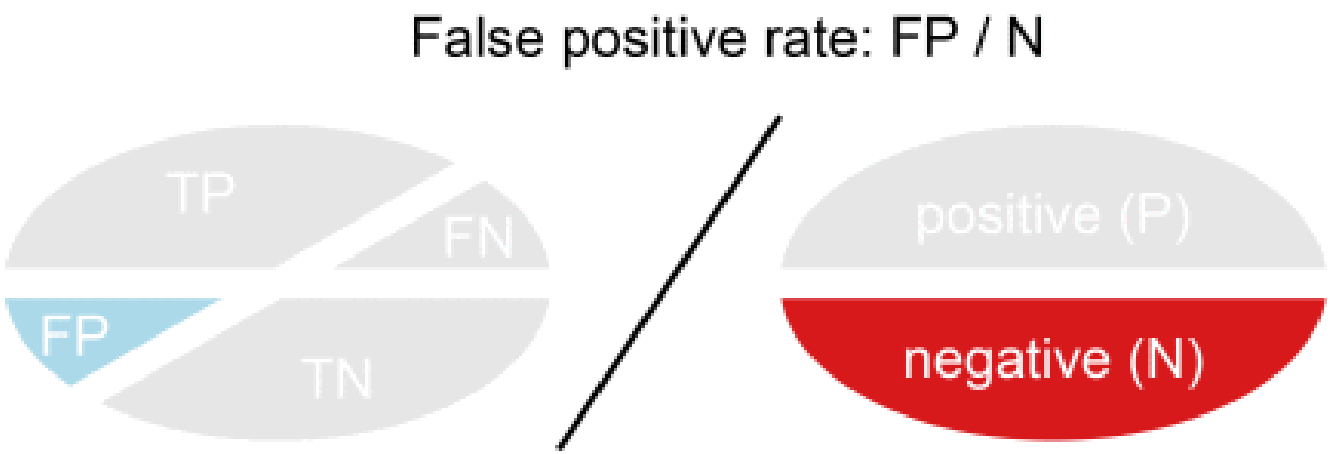
Fall-out

Fall-out (FALL) is the number of incorrect positive predictions divided by the total number of negatives:

$$FALL = \frac{FP}{TN + FP} = 1 - SPEC$$

It is also called false positive rate (FPR)

The best fall-out is 0.0, the worst is 1.0



F-score

F-score is a harmonic mean of precision and recall:

$$F_\beta = \frac{a+b}{\frac{a}{REC} + \frac{b}{PREC}} = \frac{(1+\beta^2)(PREC \cdot REC)}{\beta^2 \cdot PREC + REC}$$

where $\beta^2 = \frac{a}{b}$ — weight of precision in harmonic mean

Parameter β is commonly 0.5, 1, or 2

For example, $\beta = 1$:

$$F_1 = \frac{2 \cdot PREC \cdot REC}{PREC + REC}$$

The best F-score is 1.0, the worst is 0.0

Cohen's Kappa

Cohen's kappa is a performance measure of given classifier over a random classifier:

$$\kappa = \frac{ACC - ACC_0}{1 - ACC_0} = 1 - \frac{1 - ACC}{1 - ACC_0}$$

where ACC_0 is a probability of concordance between given classifier h and random classifier y_0:

$$ACC_0 = P(h = -1)P(y_0 = -1) + P(h = 1)P(y_0 = 1)$$

where

$P(h = -1) = \frac{TN+FN}{P+N}$ $\quad$ $P(h = 1) = \frac{TP+FP}{P+N}$

$P(y_0 = -1) = \frac{TN+FP}{P+N} = \frac{N}{P+N}$ $\quad$ $P(y_0 = 1) = \frac{TP+FN}{P+N} = \frac{P}{P+N}$

And ACC (accuracy) is a probability of concordance between given classifier h and perfect classifier y

The best Cohen's kappa is 1.0, the worst is 0.0 and below

Confusion Matrix Based Measures

- Error rate
- Accuracy
- Sensitivity (Recall, True positive rate)
- Specificity (True negative rate)
- Precision (Positive predictive value)
- Fall-out (False positive rate)
- False discovery rate (FDR): $FDR = 1 - PPV$
- False negative rate (FNR): $FNR = 1 - TPR$
- F-score
- Cohen's kappa
- ...

Threshold Classifiers

$\mathscr{D} = \{(x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})\}$ — available data sample

Class of hypotheses $\mathscr{H} = \{h : h(x) = \operatorname{sign} \varphi(x, w)\}$,

where $\varphi(x, w) \in \mathbb{R}$ — classification score for object x

$w \in \mathbb{R}^L$ — vector of parameters

Suppose that $h \in \mathscr{H}$ is some trained classifier, $\varphi(x^{(i)}, w)$ is the classification score for object $x^{(i)}$, $i = 1, ..., n$

Let's introduce a set of classifiers $\{h_b, b \in \mathbb{R}\}$:

$$h_b(x) = \operatorname{sign}(\varphi(x, w) - b)$$

where b is a threshold

The given classifier h is when threshold is equal to 0: $h = h_0$

ROC Curve

Each classifier $h_b, b \in \mathbb{R}$, has the specificity and sensitivity calculated over sample $\mathscr{D}$: $SENS(b), SPEC(b)$

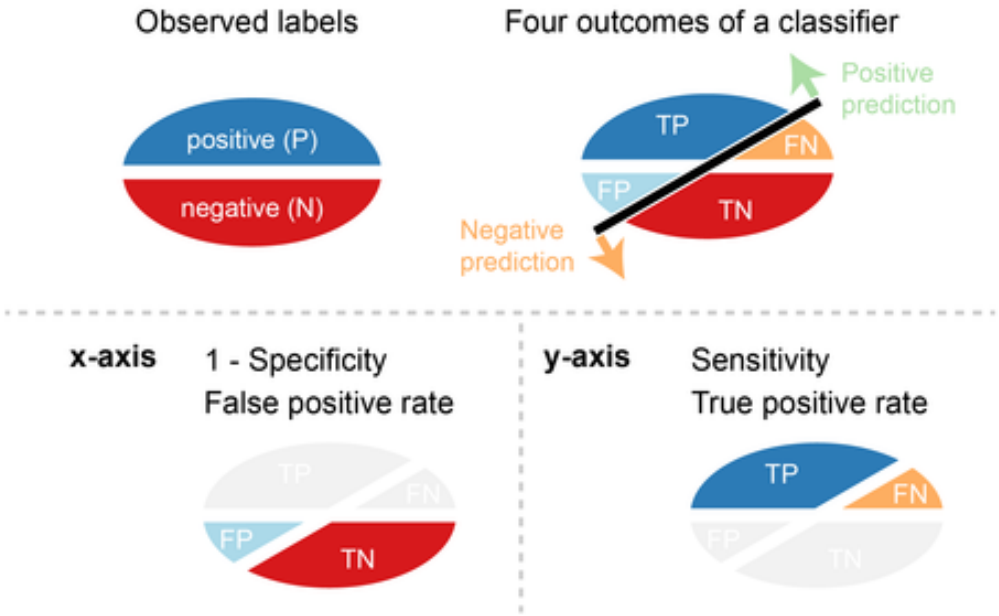
Definition

Receiver operating characteristic (ROC) curve of classifier h is a graphical plot of its sensitivity (true positive rate) against the 1-specificity (false positive rate) at various thresholds $b \in \mathbb{R}$

In practice, the specificity and sensitivity can be calculated only for the thresholds $b^{(i)}, i = 2, ..., n$:

$$b^{(i)} = \frac{1}{2}\left(\varphi(x^{(i-1)}, w) + \varphi(x^{(i)}, w)\right)$$

ROC Curve. Illustration



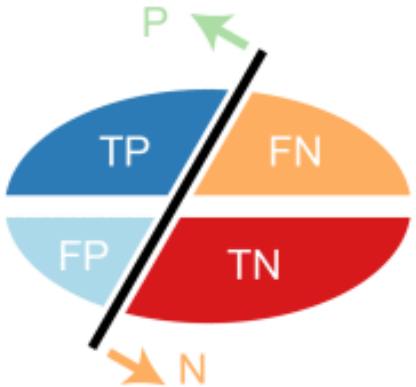
$$1 - SPEC = \frac{FP}{TN + FP}$$

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

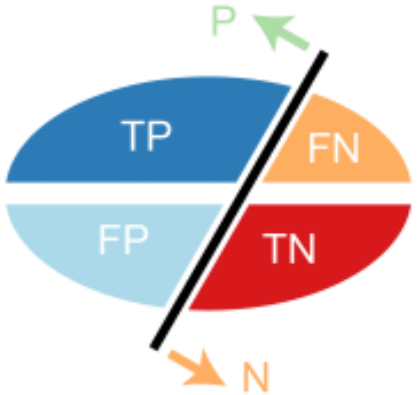
Perfect classifier:

$$1 - SPEC = 0$$
$$SENS = 1$$

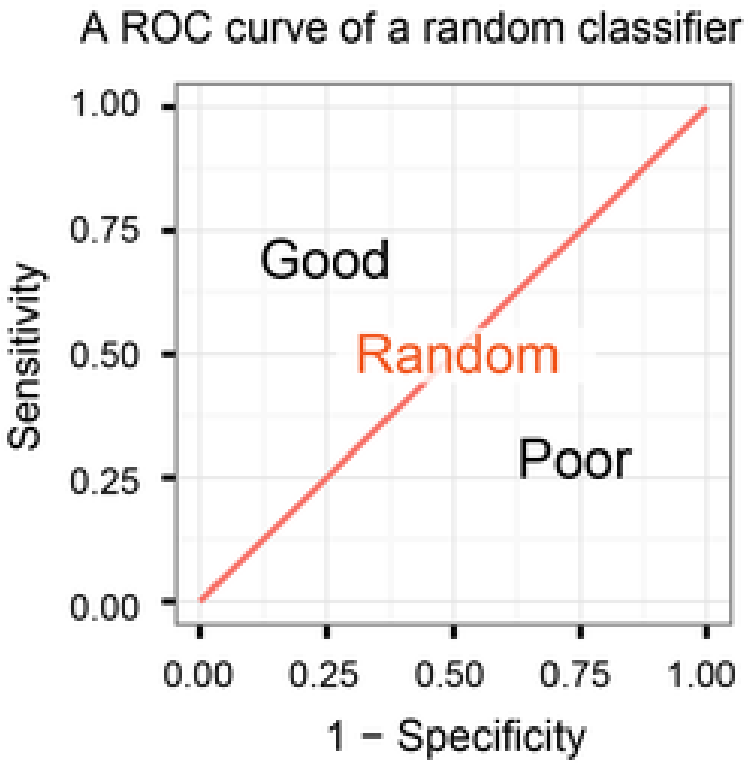
Threshold value: 2.0



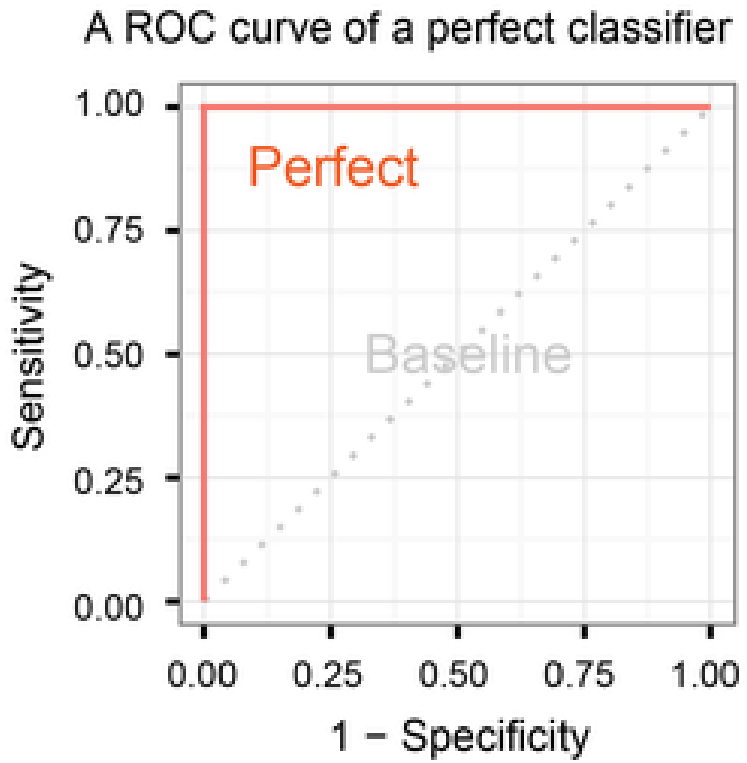
Threshold value: 1.0



ROC Curve. Random and Perfect Classification

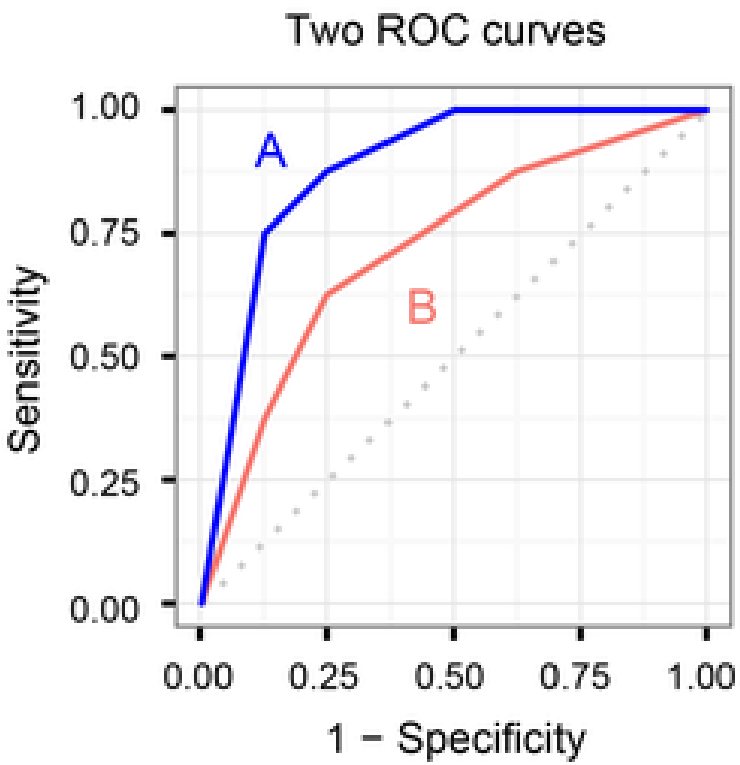


A ROC curve represents
non-separable classes

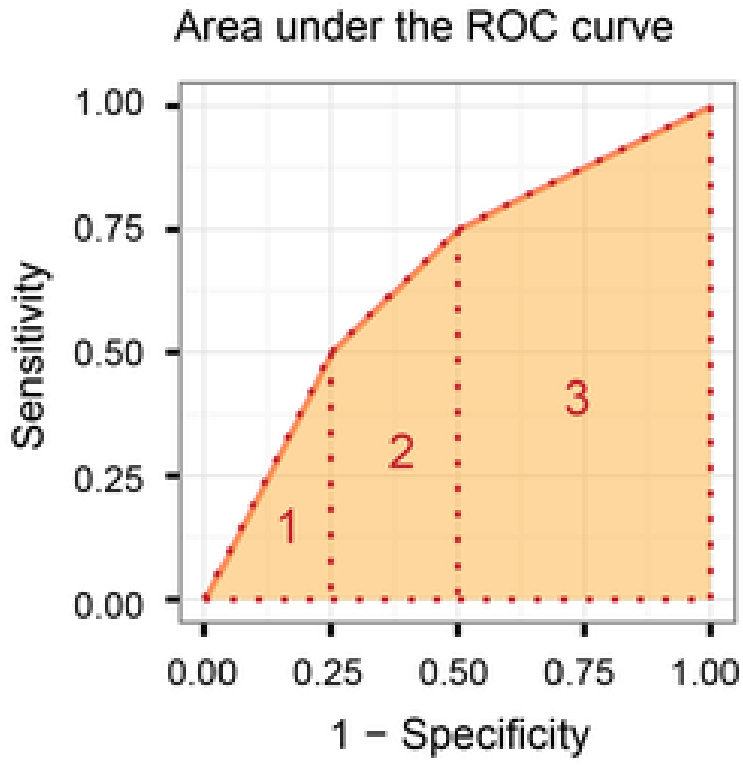


A ROC curve represents
threshold-separable classes

Area Under ROC Curve



Classifier A clearly outperforms classifier B



AUC can be calculated as sum of areas 1,2,3

AUC Calculation

AUC is not a measure of given classifier h but a measure of threshold separability of classification scores

$$\varphi(x^{(1)}, w), ..., \varphi(x^{(n)}, w),$$

i.e. is a measure of the map $\varphi(x, w)$ quality

AUC can be calculated using ROC curve or analytically*:

$$AUC = \frac{\sum_{i=1}^{N+P} \sum_{j=1}^{N+P} \left[\varphi(x^{(i)}) > \varphi(x^{(j)}) \& (y^{(i)} = 1) \& (y^{(j)} = -1)\right]}{NP}$$

*Mason, S. J., Graham, N. E. (2002). Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: Statistical significance and interpretation. Quarterly Journal of the Royal Meteorological Society, 128(584), 2145-2166

AUC and Mann-Whitney's Statistic

AUC is an estimate of the probability that the classifier h ranks a randomly chosen positive case higher than a randomly chosen negative case

Mann-Whitney's statistics:

$$U_P = R_P - \frac{P(P+1)}{2}, \qquad U_N = R_N - \frac{N(N+1)}{2}$$

$$R_P + R_N = 1 + ... + (N+P) = \frac{(N+P)(N+P+1)}{2}$$

$$U_P + U_N = NP$$

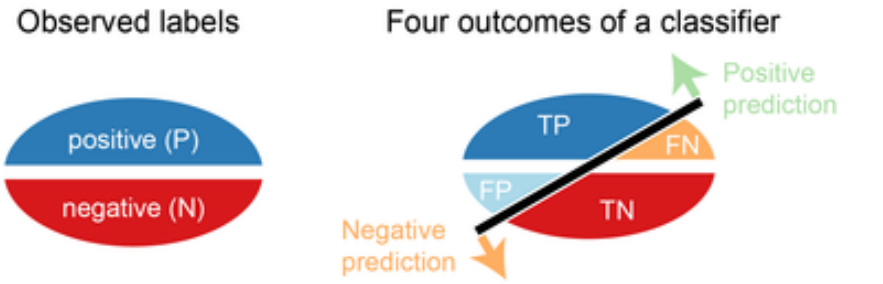
where R_P is the sum of the ranks related to positive class in sample $\varphi(x^{(1)}, w), ..., \varphi(x^{(n)}, w)$

It can be shown that

$$AUC = \frac{U_P}{NP}$$

So, AUC is related to well-known U-statistic. It allows us to calculate confidence intervals and perform statistical tests for AUC

Precision-Recall Curve. Illustration



x-axis Recall
Sensitivity

y-axis Precision
Positive predictive value

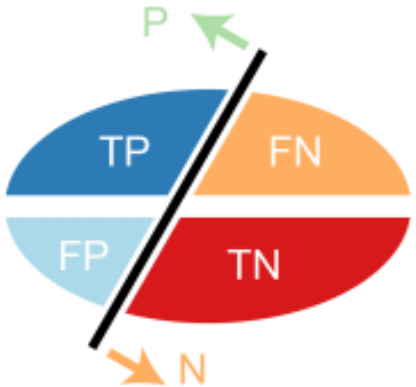
$$SENS = \frac{TP}{TP + FN}$$

$$PREC = \frac{TP}{TP + FP}$$

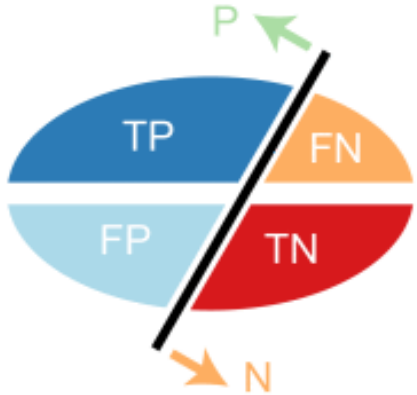
Perfect classifier:

$$SENS = 1$$
$$PREC = 1$$

Threshold value: 2.0



Threshold value: 1.0

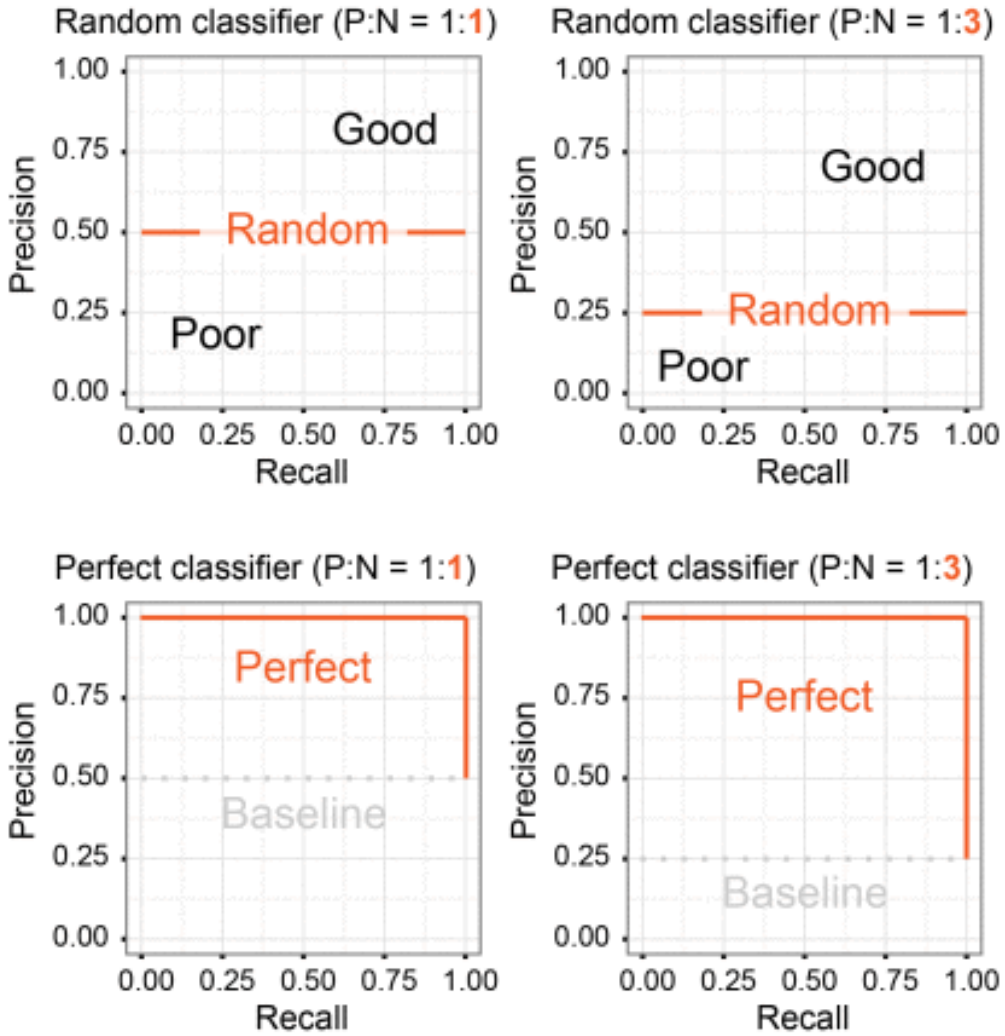


PR Curve. Random and Perfect Classification

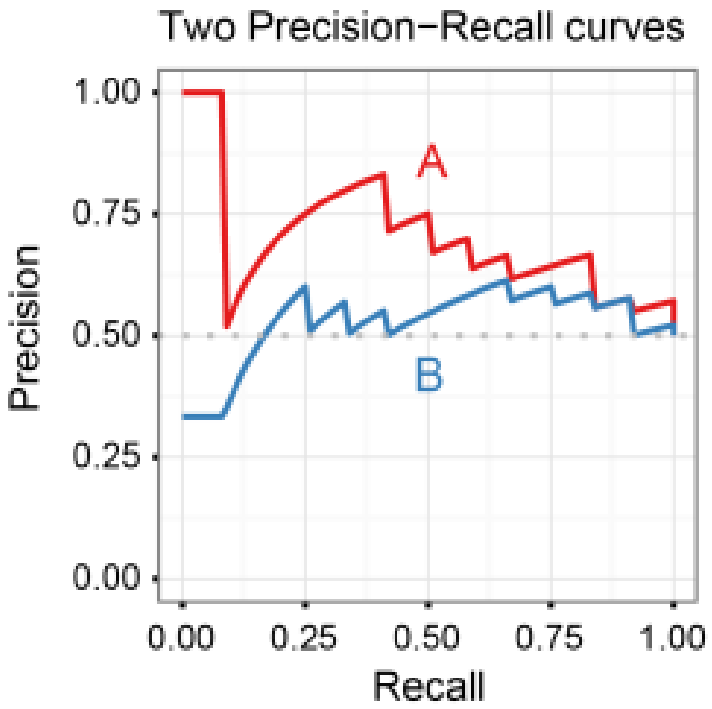
$$baseline = \frac{P}{P+N}$$

A PRC represents
non-separable classes

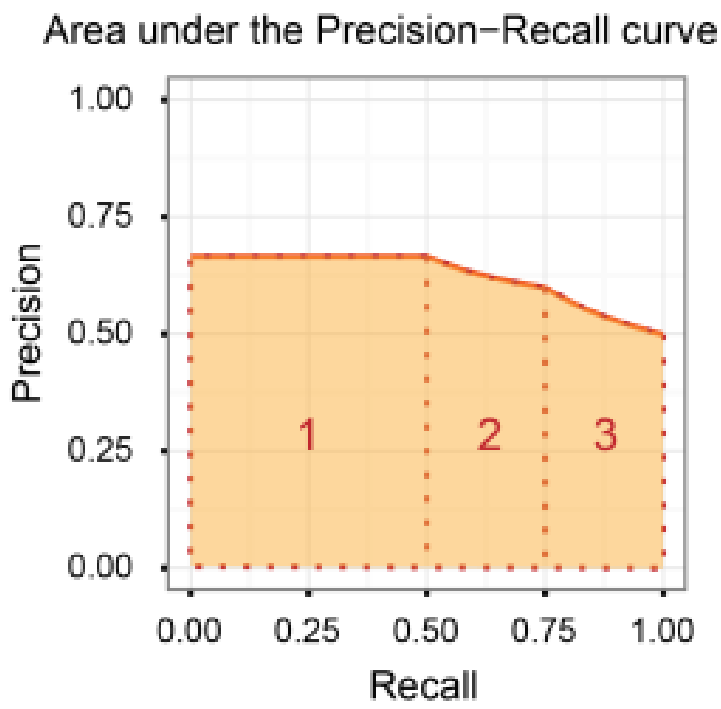
A PRC represents
threshold-separable classes



Area Under PR Curve



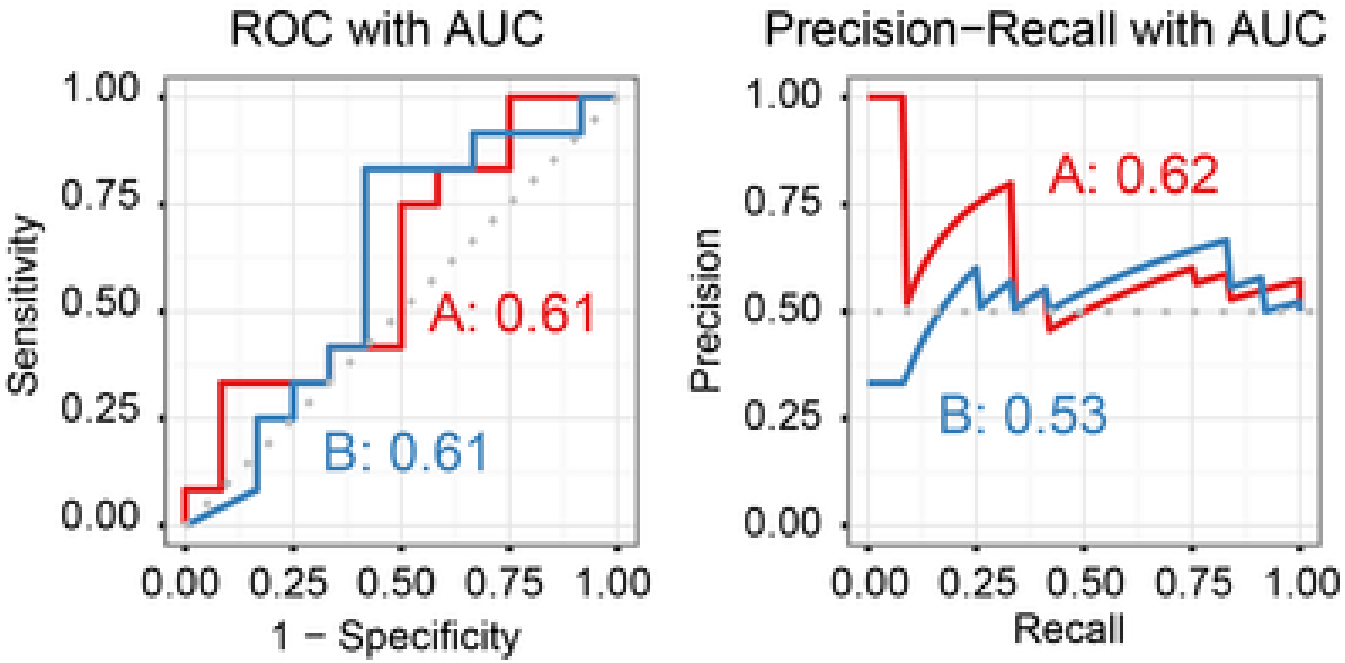
Classifier A clearly outperforms classifier B



AUC can be calculated as sum of areas 1,2,3

ROC AUC and PR AUC

Points in ROC space and PR space have a one-to-one relationship



ROC shows the same AUC for classifiers A (0.61) and B (0.61)
PRC shows different PR AUC for A (0.62) and B (0.53)

ROC vs PRC

$$Precision = \frac{\#correctly\ predicted\ as\ positive}{\#all\ predicted\ as\ positive}$$

$$Recall = Sens = \frac{\#correctly\ predicted\ as\ positive}{\#all\ positives\ in\ sample}$$

$$Spec = \frac{\#correctly\ predicted\ as\ negative}{\#all\ predicted\ as\ negative}$$

PRC is more informative when dealing with "needle-in-haystack" type problems or problems where the positive class is more important than the negative class

Other properties:

- Linear interpolation in ROC space leads to non-linear interpolation in PR space
- Algorithms that optimize ROC AUC are not guaranteed to optimize PR AUC
- A curve dominates in ROC space iff it dominates in PR space

ROC vs PRC. Example

Information retrieval problem:

We want to find a set of 100 relevant documents out of a list of 1 million possibilities based on some query. Let's say we've got two algorithms we want to compare with the following performance:
Algorithm 1: 100 retrieved documents, 90 relevant
Algorithm 2: 2000 retrieved documents, 90 relevant

In ROC space:

Algorithm 1: $TPR = 0.9, \qquad FPR = 0.00001$
Algorithm 2: $TPR = 0.9, \qquad FPR = 1910/10^6 = 0.00191$
$\Delta FPR = 0.00191 - 0.00001 = 0.0019 \approx 0$

In PR space:

Algorithm 1: $REC = TPR = 0.9, \qquad PREC = 0.9$
Algorithm 2: $REC = TPR = 0.9, \qquad PREC = 90/2000 = 0.045$
$\Delta PREC = 0.9 - 0.045 = 0.855$

Threshold Criteria

How to choose the threshold on ROC curve or PR curve?

Threshold criteria for ROC curve:

- $|SENS - SPEC| \to \min$
  Intersection of the ROC curve with line (0,1)—(1,0)
- $Y = (SENS + SPEC - 1) \to \max, \quad Y$ — Youden's index
  The point on the ROC curve whose tangent has a slope of one
- $(1 - SENS)^2 + (SPEC - 1)^2 \to \min$
  The closest point on the ROC curve to "perfect" point (0,1)
- $SENS \to \max$ at fixed $SPEC$
- $SPEC \to \max$ at fixed $SENS$
- $F_1 \to \max$
- $\kappa \to \max$
- ...

Model-Wide Measures

- ROC curve, ROC AUC
- PR curve, PR AUC
- CROC curve, CROC AUC
  plot TPR against transformed FPR
- CC-plot, CC AUC
  plot normalized expected cost against probability of correctly classified positives
- ...

Multiclass Classification Problem

$\mathscr{D} = \{(x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})\}$ — available data sample

$Y = \{1, ..., K\}$ — class labels

Approaches to multiclass classification problem:

- Reduction to binary
  Reduction of the multiclass classification problem to a set of binary classification problems
- Extension from binary
  Modifications of the existing binary classifiers to solve multi-class classification problems
- Hierarchical classification
  Decomposition of the multiclass classification problem to hierarchically organized classification problems

Error-Correcting Output Coding

Error-correcting output coding (ECOC) is a method for decomposing a multiclass classification problem into many binary classification tasks, and then combining the results of the subtasks into a hypothesized solution to the original problem

Stages of ECOC:

- Construct many binary classifiers (base learners)
- Apply a voting scheme to the outputs of each base learner

Coding design determines the classes that the base learners train on

Decoding scheme determines how the results (predictions) of the binary classifiers are aggregated

Coding Design

Coding design matrix is a $K \times L$ matrix that consists of –1,0 or 1:

$$M = \begin{pmatrix} m_{11} & \dots & m_{1L} \\ \dots & \dots & \dots \\ m_{K1} & \dots & m_{KL} \end{pmatrix}$$

K — number of classes

L — number of base learners

$m_{kl} = -1$ indicates that k-th class is to be included in negative class for l-th binary learner

$m_{kl} = 1$ indicates that k-th class is to be included in positive class for l-th binary learner

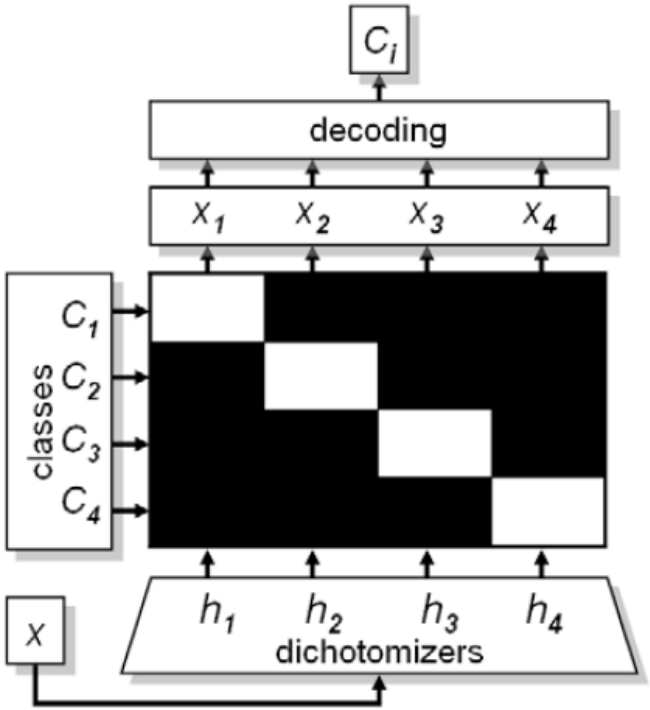
$m_{kl} = 0$ indicates that k-th class is to be ignored for l-th binary learner

One vs All (OVA)

For each binary learner, one class is positive and the rest are negative

Number of learners: $L = K$

Coding design matrix: square matrix, ones on principal diagonal, rest elements are –1



Advantages:

— Low computational complexity

Disadvantages:

— Each binary learner deals with unbalanced classes

— Need to re-train all learners if a new class added

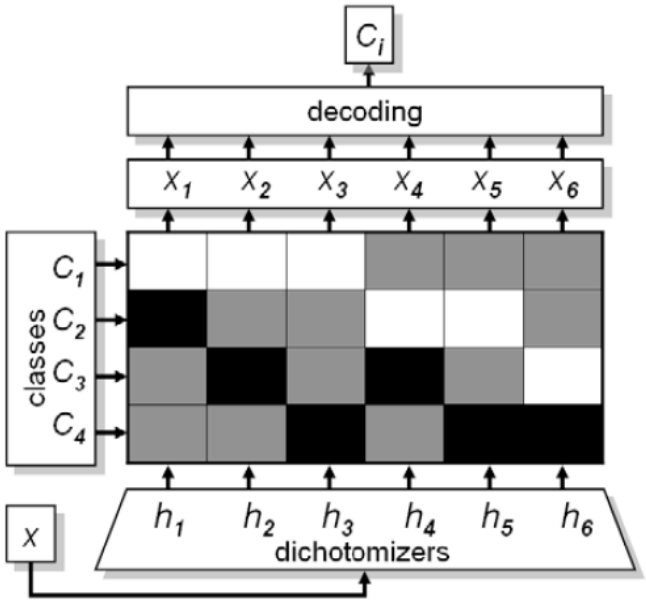
— Low robustness

One vs One (OVO)

For each binary learner, one class is positive, another is negative, and the rest are ignored

Number of learners: $L = \frac{K(K-1)}{2}$

Coding design matrix: each column contains one 1 and one –1, rest elements are 0

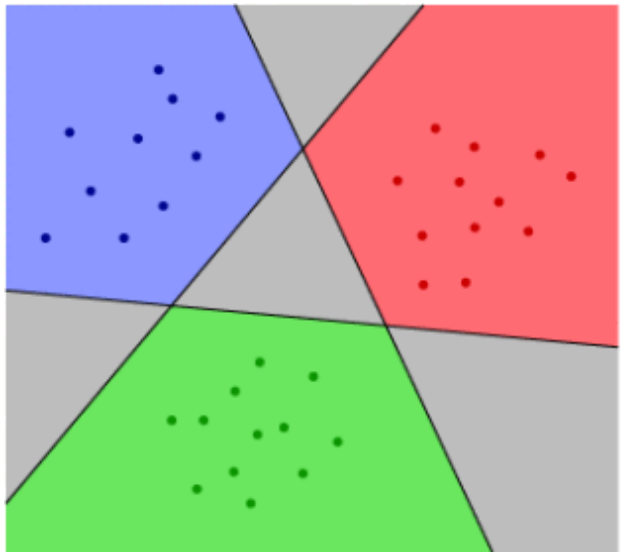


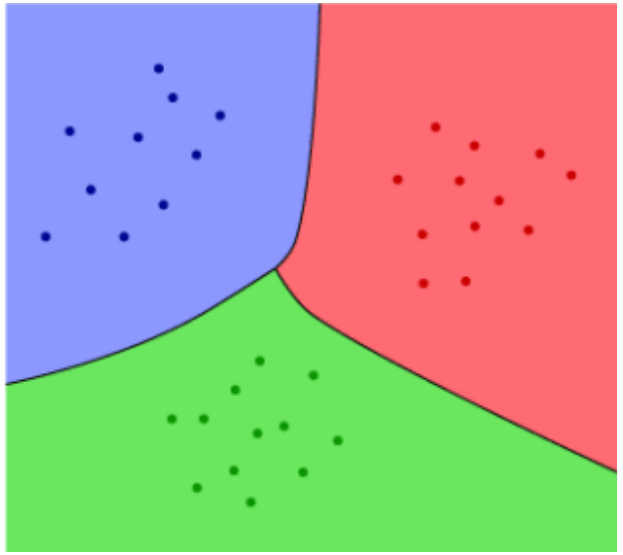
Advantages:
Don't need to re-train all learners if a new class added

Disadvantages:
High computational complexity: $L \sim K^2$

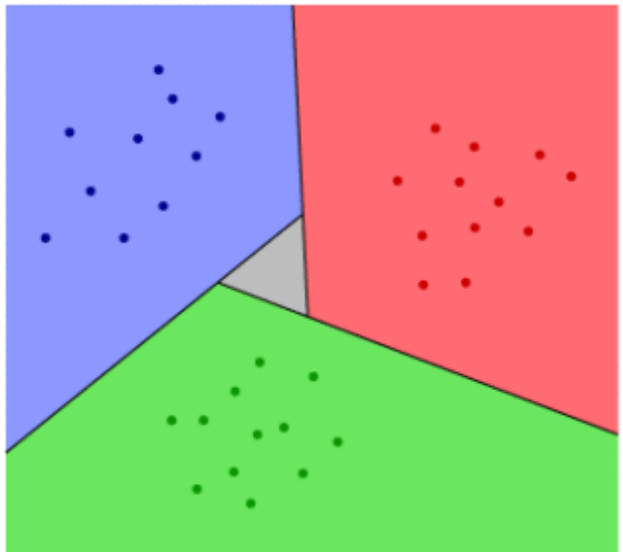
OVA vs OVO

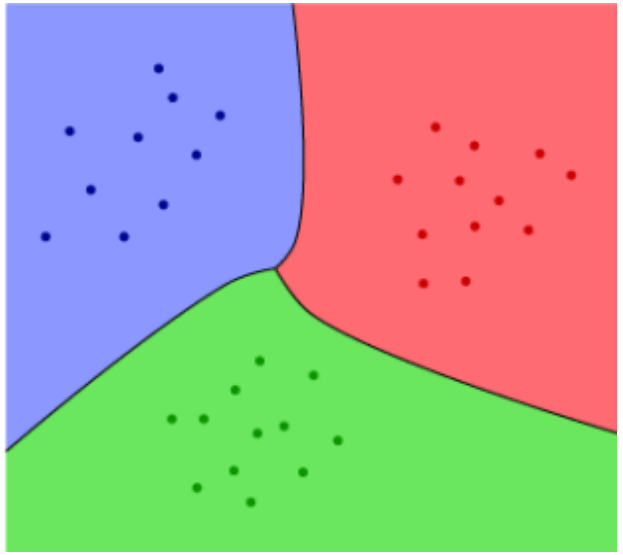
One vs All





One vs One





Binary Complete Coding

This design partitions the classes into all binary combinations, and does not ignore any classes

Number of learners: $L = 2^{K-1} - 1$
Coding design matrix: l-th column is a bipolar representation of number l, $l = 1, ..., 2^{K-1} - 1$

For $K = 4$:

$$M = \begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & +1 & +1 & +1 & +1 \\ -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 \end{pmatrix}$$

At this design even if we fail in a bit, we still are able to obtain the correct classification

Ternary Complete Coding

This design partitions the classes into all ternary combinations

Number of learners: $L = \frac{3^K - 2^{K+1} + 1}{2}$
Coding design matrix: each column contains al least one 1 and one –1

For $K = 3$:

$$M = \begin{pmatrix} -1 & -1 & -1 & -1 & -1 & 0 \\ -1 & +1 & +1 & +1 & 0 & -1 \\ +1 & -1 & +1 & 0 & +1 & +1 \end{pmatrix}$$

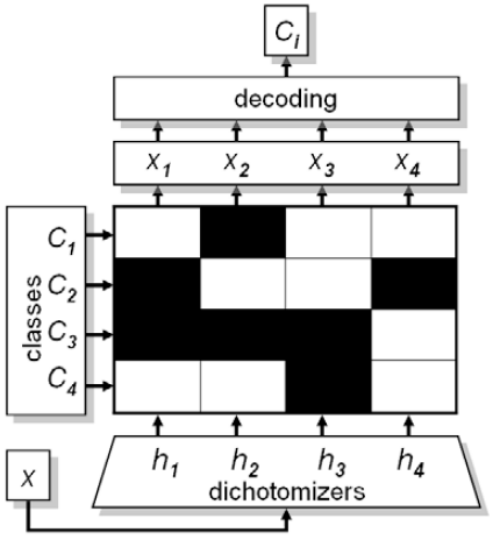
This design exhausts all combinations of class assignments

Random Coding

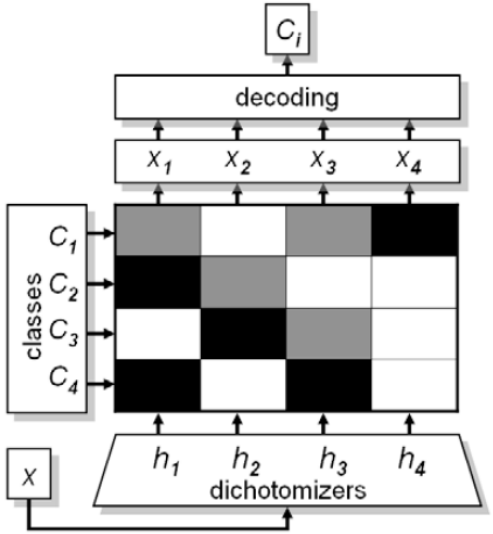
Dense random coding — for each binary learner, classes are randomly assigned into positive or negative classes, with at least one of each type

Sparse random coding — for each binary learner, classes are randomly assigned into positive or negative classes, some classes can be ignored

Dense random



Sparse random



Coding Design Strategies

- One-versus-all
- One-versus-one
- Binary complete
- Ternary complete
- Dense random
- Sparse random
- Custom
  Custom coding design matrix must have a certain form (all rows are unique, can separate any two classes etc.)

The number of binary learners grows with the number of classes
For a problem with many classes, the binary complete and ternary complete coding designs are not efficient

$K \leq 4 \Rightarrow$ ternary complete
$K \leq 5 \Rightarrow$ binary complete

ECOC Decoding Scheme

Suppose that:

Class of hypotheses $\mathscr{H} = \{h : h(x) = \operatorname{sign} \varphi(x, w)\}$,

where $\varphi(x, w)$ is a classification score for object $x \in \mathscr{X}$

Each binary learner $h_l \in \mathscr{H},\ l = 1, ..., L$: $h_l(x) = \operatorname{sign} \varphi_l(x, w_l)$
where $\varphi_l(x, w_l)$ is a classification score of l-th binary learner for object $x \in \mathscr{X}$

Score-based decoding:

$$s_k = \sum_{l=1}^{L} m_{kl} \varphi_l(x, w_l)$$

Score-weighted decoding:

$$s_k = \frac{\sum_{l=1}^{L} m_{kl} \varphi_l(x, w_l)}{\sum_{l=1}^{L} |m_{kl}|}$$

Predicted class: $k^* = \arg \max_{k=\overline{1,K}} s_k$, s_k is a score of k-th class

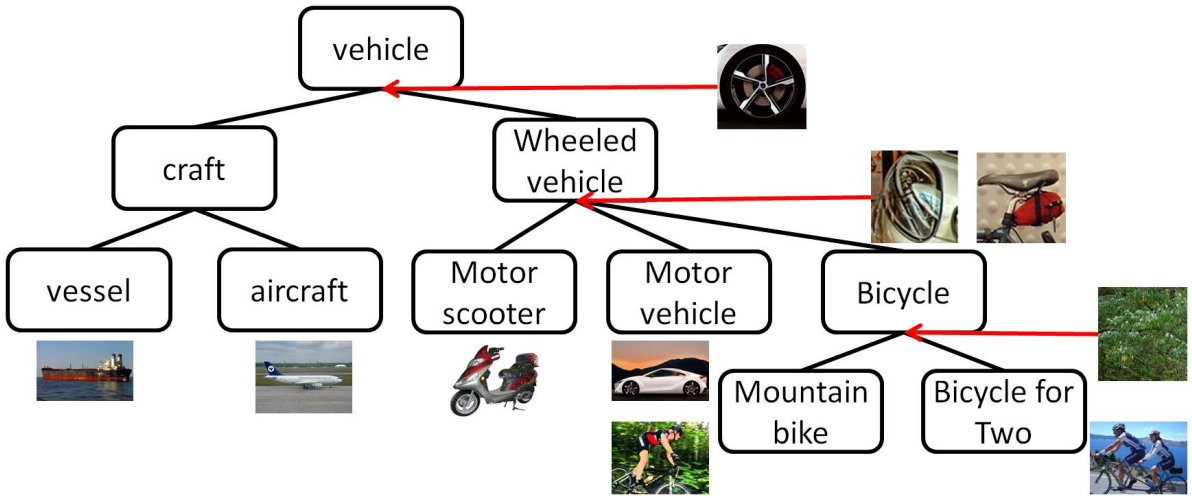
Extension from Binary

Extension from binary approach supposes modifications of the existing binary classifiers to solve multiclass classification problems

- k-nearest neighbours
  Naturally extensible to the multiclass problem
- Naive Bayes classifier
  Naturally extensible to the multiclass problem
- Multiclass support vector machine
  Additional parameters and constraints are added to the SVM optimization problem
- Multiclass logistic regression
  Use softmax functions
- ...

Hierarchical Classification

Hierarchical classification supposes decomposition of the multiclass classification problem to hierarchically organized classification problems



If a case is misclassified at upper level it will continue to be miss-classified at deeper levels too

Hierarchy has a big impact on the final classifier efficiency

Measures of Classification Performance

Approaches to measuring:

- Micro-averaging
  Generalization to multiclass classification

$$Perf_\mu = Perf\left(\sum_k TP_k, \sum_k FP_k, \sum_k TN_k, \sum_k FN_k\right)$$

- Macro-averaging
  Averaging of per-class measures over classes

$$Perf_M = \frac{1}{K}\sum_k Perf(TP_k, FP_k, TN_k, FN_k)$$

$Perf$ — performance measure for binary classifier
TP_k, TN_k, FP_k, FN_k — TP, TN, FP and FN with respect to k-th class: k-th class is considered as positive, rest classes as negative

All micro-averaged and macro-averaged performance measures are based on OVA binary performance measures

Multiclass Confusion Matrix

		Prediction				
		Class 1	Class 2	Class 3	...	Class K
Actual	Class 1	Accurate				
	Class 2		Accurate			
	Class 3			Accurate		
	...				Accurate	
	Class K					Accurate

Positive: 1

		Prediction	
		Positive	Negative
Actual	Positive	TP_1	FN_1
	Negative	FP_1	TN_1

. . . . . .

Positive: K

		Prediction	
		Positive	Negative
Actual	Positive	TP_K	FN_K
	Negative	FP_K	TN_K

Multiclass Error Rate

Binary error rate:

$$ERR = \frac{FP + FN}{TP + FN + FP + TN}$$

Micro-averaged error rate:

$$ERR_\mu = \frac{\sum_k FP_k + \sum_k FN_k}{\sum_k (TP_k + FN_k + FP_k + TN_k)} = \frac{\sum_k FP_k + \sum_k FN_k}{Kn}$$

Macro-averaged error rate:

$$ERR_M = \frac{1}{K} \sum_k \frac{FP_k + FN_k}{TP_k + FN_k + FP_k + TN_k}$$
$$= \frac{\sum_k FP_k + \sum_k FN_k}{Kn} = ERR_\mu$$

Multiclass Accuracy

Binary accuracy:

$$ACC = \frac{TP + TN}{TP + FN + FP + TN} = 1 - ERR$$

Micro-averaged accuracy:

$$ACC_\mu = \frac{\sum_k TP_k + \sum_k TN_k}{\sum_k (TP_k + FN_k + FP_k + TN_k)} = \frac{\sum_k TP_k + \sum_k TN_k}{Kn}$$

Macro-averaged accuracy:

$$ACC_M = \frac{1}{K} \sum_k \frac{TP_k + TN_k}{TP_k + FN_k + FP_k + TN_k}$$
$$= \frac{\sum_k TP_k + \sum_k TN_k}{Kn} = ACC_\mu$$

Multiclass Confusion Matrix Based Measures

Micro-averaged measures:

$$SENS_\mu = REC_\mu = \frac{\sum_k TP_k}{\sum_k (TP_k + FN_k)}$$

$$SPEC_\mu = \frac{\sum_k TN_k}{\sum_k (TN_k + FP_k)}$$

$$PREC_\mu = PPV_\mu = \frac{\sum_k TP_k}{\sum_k (TP_k + FP_k)}$$

Macro-averaged measures:

$$SENS_M = REC_M = \frac{1}{K} \sum_k SENS_k = \frac{1}{K} \sum_k \frac{TP_k}{TP_k + FN_k}$$

$$SPEC_M = \frac{1}{K} \sum_k SPEC_k = \frac{1}{K} \sum_k \frac{TN_k}{TN_k + FP_k}$$

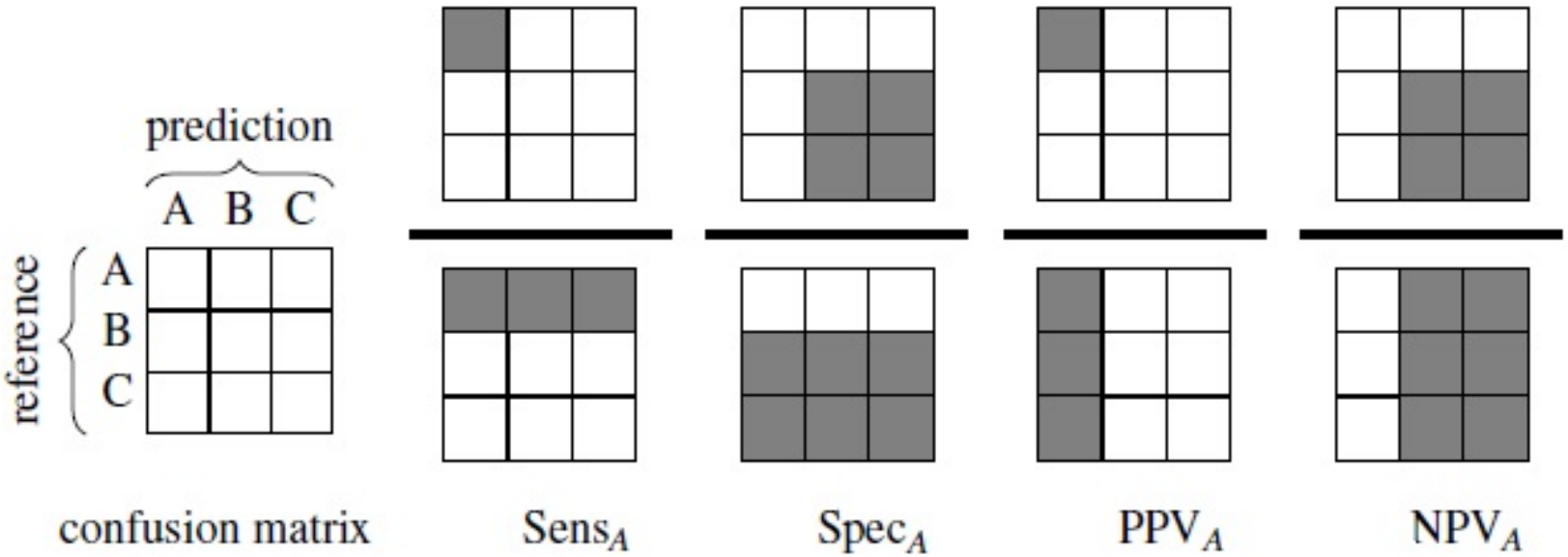
$$PREC_M = PPV_M = \frac{1}{K} \sum_k PREC_k = \frac{1}{K} \sum_k \frac{TP_k}{TP_k + FP_k}$$

Sensitivity, Specificity, Precision and Recall. Illustration

$Sensitivity_k$ ($Recall_k$): How well does the classifier recognize cases from class k?

$Specificity_k$: How well does the classifier recognize that a case does not belong to class k?

$Precision_k$ (PPV_k): Given the prediction is class k, what is the probability that the case truly belongs to class k?



F-score and Cohen's Kappa

Micro-averaged F-score:

$$F_{\mu\beta} = \frac{(1+\beta^2)(PREC_\mu \cdot REC_\mu)}{\beta^2 \cdot PREC_\mu + REC_\mu}$$

Macro-averaged F-score:

$$F_{M\beta} = \frac{1}{K}\sum_{k=1}^{K} \frac{(1+\beta^2)(PREC_k \cdot REC_k)}{\beta^2 \cdot PREC_k + REC_k}$$

Cohen's kappa is a performance measure over a random classifier:

$$\kappa = \frac{ACC - ACC_0}{1 - ACC_0}$$

where ACC_0 is a probability of concordance between given classifier h and "true" classifier y:

$$ACC_0 = \sum_k P(h(x)=k)P(y=k) = \sum_k \frac{(TP_k + FP_k)n_k}{n^2}$$

Some Properties of Multiclass Measures

Property 1

$$\sum_k FP_k = \sum_k FN_k$$

Property 2

$$ACC_\mu = ACC_M, \quad ERR_\mu = ERR_M$$

Property 3

$$ACC_\mu = 1 - ERR_\mu, \quad ACC_M = 1 - ERR_M$$

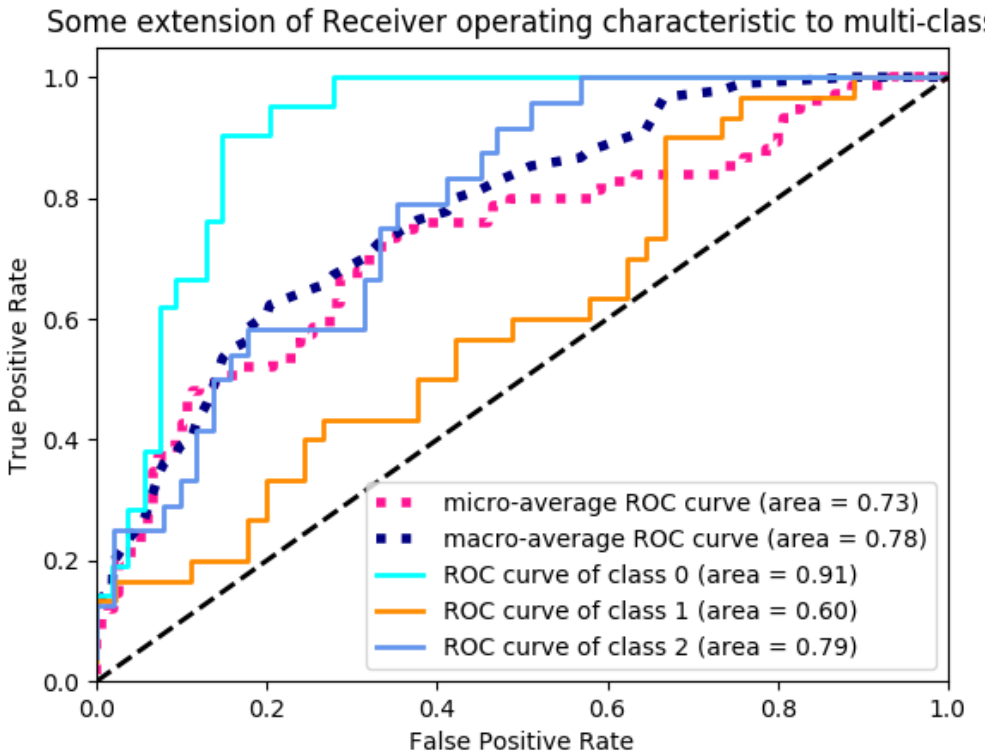
Property 4

$$PREC_\mu = REC_\mu = F_{\mu\beta} = ACC = \frac{\#correctly\ classified}{\#total}$$

Prove it!

Multiclass ROC Analysis

Macro-averaging and micro-averaging approaches to ROC analysis



Micro-averaged AUC: $AUC_\mu = AUC(1 - SPEC_\mu, SENS_\mu)$

Macro-averaged AUC: $AUC_M = \frac{1}{K} \sum_k AUC_k$

Micro-averaging vs Macro-averaging

Macro-averaging gives equal weight to each class
Micro-averaging gives equal weight to each per-class classification decision

Macro-averaging treats all classes equally while micro-averaging favors classes with a larger number of instances

Micro-averaging tends to over-emphasize the performance on the largest classes, while macro-averaging over-emphasizes the performance on the smallest

It's often best to look at both of them to get a good idea of how your data distributes across classes