Model evaluation

© Victor Kitov

v.v.kitov@yandex.ru

Summer school on Machine Learning in High Energy Physics

in partnership with

Yandex Data Factory

August 2015

Introduction

- Advanced aspects of machine learning
- 8 lectures:
  - Reminder about major algorithms. Model evaluation.
  - Feature selection.
  - Ensemble learning N1.
  - Ensemble learning N2.
  - Linear dimensionality reduction.
  - Non-linear dimensionality reduction.
  - Kernel trick. Kernelized algorithms.
  - Deep learning.

Recommended materials

- **Statistical Pattern Recognition.** 3rd Edition, Andrew R. Webb, Keith D. Copsey, John Wiley & Sons Ltd., 2011.
- **The Elements of Statistical Learning: Data Mining, Inference, and Prediction.** Trevor Hastie, Robert Tibshirani, Jerome Friedman, 2nd Edition, Springer, 2009. http://statweb.stanford.edu/~tibs/ElemStatLearn/
- **Machine Learning: A Probabilistic Perspective.** Kevin P. Murphy. Massachusetts Institute of Technology. 2012.
- **Lectures of Machine Learning Course** (in Russian). Konstantin Vorontsov. `machinelearning.ru`.
- **Additional sources** - wikipedia, articles, tutorials.

Table of Contents

Formal definitions of machine learning

- Machine learning is a field of study that gives computers the ability to learn without being explicitly programmed.
- A computer program is said to learn from **experience E** with respect to some **class of tasks T** and **performance measure P**, if its performance P at tasks in T improves with experience E.

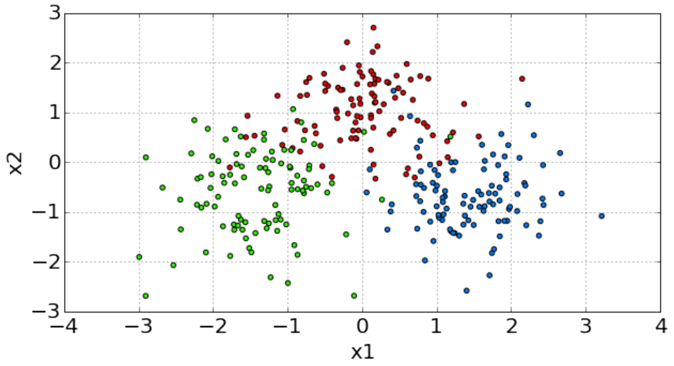
Supervised machine learning

- Find functional relationship between input variables x and output variables y based on expert knowledge and their common observations:

$$(x_1, y_1), (x_2, y_2), ...(x_N, y_N)$$

  - x as a vector is called object, pattern.
  - individual components of x are called features, regressors, inputs.
  - y is called output, target
- if $y \in \mathbb{R}$ => regression
- if $y \in \{\omega_1, \omega_2, ...\omega_C\}$ => classification / pattern recognition

Demonstration



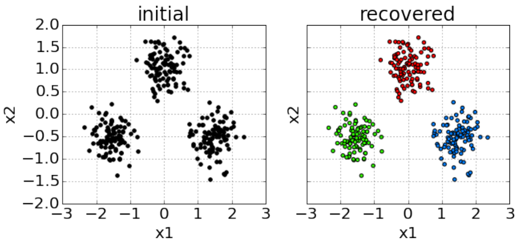
Supervised learning: $x = (x_1, x_2)$, y specified by color

Unsupervised learning

- Find functional relationship between input variables x and output variables y based on expert knowledge and only x observations:

$$x_1, x_2, \dots x_N$$

- Unsupervised learning is also known as clustering (for discrete output)



Unsupervised output recovery

Semi-supervised learning

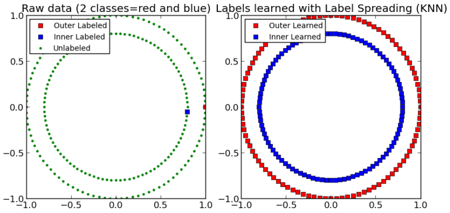
- A small number of joint observations is available:

$$(x_1, y_1), (x_2, y_2), ...(x_N, y_N)$$

- A bigger number of only input observations is also available:

$$x_1, x_2, ...x_M$$

- Recover $x \to y$ relationship



Semi-supervised output recovery

Notation

- $(x_1, y_1), (x_2, y_2), ...(x_N, y_N)$ - training sample, N is number of observations
- $x_i \in \mathbb{R}^D$, D is dimensionality of data
- x_i or (x_i, y_i) - individual sample, pattern, object.
- In case of feature selection or dimensionality reduction: d is output dimensionality
- $\omega_1, \omega_2, ...\omega_C$ - labels of classes, C - total number of classes.

Typical workflow (CrispDM methodology)

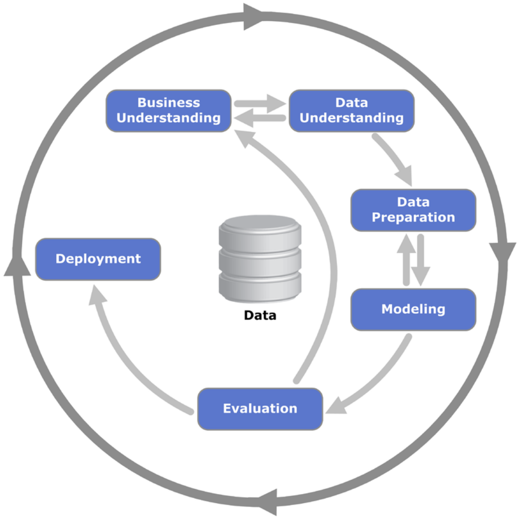


Table of Contents

Comments on some major ML methods

- K-NN (metric selection, search optimization using KD-trees and ball-trees)
- Random Forest
- Extra random trees
- Neural network (later)
- Boosting

Forward stagewise additive modeling

Input: training dataset (x_i, y_i), $i = 1, 2, ...n$; loss function $L(f, y)$, general form of additive classifier $h(x, \gamma)$ (dependent from parameter γ) and the number M of successive additive approximations.

1. Fit initial approximation $f^0(x)$ (might be taken $f^0(x) \equiv 0$)
2. For $m = 1, 2, ...M$:
    1. find next best classifier
    $$(c_m, \gamma_m) = \arg\min \sum_{i=1}^{n} L(f_{m-1}(x_i) + c_m h(x, \gamma_m),\ y_i)$$
    2. set
    $$f_m(x) = f_{m-1}(x) + c_m h(x, \gamma_m)$$

Output: approximation function $f^M(x) = f^0(x) + \sum_{j=1}^{M} c_j h(x, \gamma_m)$

Adaboost algorithm is obtained for $L(y, f(x)) = e^{-yf(x)}$

Adaboost (discrete version)

Assumptions: loss function $L(y, f(x)) = e^{-yf(x)}$, classification task: $y \in \}$
Input: training dataset (x_i, y_i), $i = 1, 2, ...n$; number of additive weak classifiers M, a family of weak classifiers $h(x)$, outputting only +1 or -1 (binary classification) and trainable on weighted datasets.

1. Initialize observation weights $w_i = 1/n$, $i = 1, 2, ...n$.
2. for $m = 1, 2, ...M$:
    1. fit $h^m(x)$ to training data using weights w_i
    2. compute weighted misclassification rate:
    $$E_m = \frac{\sum_{i=1}^{n} w_i \mathbb{I}[h^m(x) \neq y_i]}{\sum_{i=1}^{n} w_i}$$
    3. compute $\alpha_m = \ln\left((1 - E_m)/E_m\right)$
    4. increase all weights, where misclassification with $h^m(x)$ was made:
    $$w_i \leftarrow w_i e^{\alpha_m}, \; i \in \{i : \; h^m(x_i) \neq y_i\}$$

Output: composite classifier $f(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m h^m(x)\right)$

Gradient boosting - regression

Input: training dataset (x_i, y_i), $i = 1, 2, ...n$; loss function $L(f, y)$ and the number M of successive additive approximations.

1. Fit initial approximation $f^0(x)$ (might be taken $f^0(x) \equiv 0$)
2. For each step $m = 1, 2, ...M$:
    1. calculate derivatives $z_i = -\frac{\partial L(r,y)}{\partial r}|_{r=f^{m-1}(x)}$
    2. train additive approximation with classifier h^m on (x_i, z_i), $i = 1, 2, ...n$ with simple loss function, e.g. squared difference $\sum_{i=1}^{n} (h^m(x_i) - z_i)^2$
    3. solve univariate optimization problem:
    $$\sum_{i=1}^{n} L\left(f^{m-1}(x_i) + c_m h^m(x_i), y_i\right) \to \min_{c_m \in \mathbb{R}_+}$$
    4. set $f^m(x) = f^{m-1}(x) + c_m h^m(x)$

Output: approximation function $f^M(x) = f^0(x) + \sum_{m=1}^{M} c_m h^m(x)$

Gradient boosting of trees - regression

Input: training dataset (x_i, y_i), $i = 1, 2, ...n$; loss function $L(f, y)$ and the number M of successive additive approximations.

1. Fit constant initial approximation $f^0(x)$:
   $f^0(x) = \arg\min_\gamma \sum_{i=1}^n L(\gamma,\ y_i)$
2. For each step $m = 1, 2, ...M$:
   1. calculate derivatives $z_i = -\frac{\partial L(r,y)}{\partial r}\big|_{r=f^{m-1}(x)}$
   2. train regression tree h^m on (x_i, z_i), $i = 1, 2, ...n$ with squared loss function $\sum_{i=1}^n (h^m(x_i) - z_i)^2$ and extract terminal regions R_{jm}, $j = 1, 2, ...J_m$.
   3. for each terminal region R_{jm}, $j = 1, 2, ...J_m$ solve univariate optimization problem:
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L(f^{m-1}(x_i) + \gamma,\ y_i)$$
   4. update $f^m(x) = f^{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbb{I}[x \in R_{jm}]$

Output: approximation function $f^M(x)$

Gradient boosting for classification

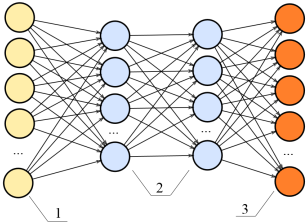
- Suppose we have C classes. Then each class probability may be represented using $C-1$ functions $f_i(x)$:

$$p_i(x) = \begin{cases} \frac{e^{f_i(x)}}{1+\sum_{i=1}^{C-1} e^{f_i(x)}}, & i = 1, 2, \dots C-1 \\ \frac{1}{1+\sum_{i=1}^{C-1} e^{f_i(x)}} & i = C \end{cases}$$

- In classification boosting functions $f_i(x)$, $i = 1, 2, \dots C-1$ are estimated the same way as single regression function $f^m(x)$ in regression boosting - the loop [for $c = 1, 2, \dots C-1$] is inserted inside step 2 loop [for $m = 1, 2, \dots M$].
- More information on boosting can be found in chapter 10 of the book “The Elements of Statistical Learning” (`http://statweb.stanford.edu/~tibs/ElemStatLearn/`)

Neural networks

Structure of neural network



Activation function

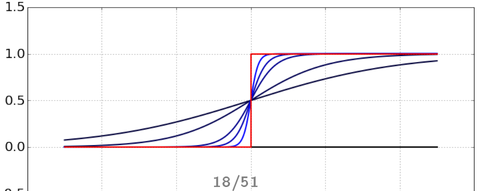


Table of Contents

Discriminative functions

- Classification of two classes ω_1 and ω_2
- Discriminant function: $g_w(x)$ is defined

$$\widehat{\omega} = \begin{cases} \omega_1, & g(x) \geq 0 \\ \omega_2, & g(x) < 0 \end{cases}$$

- Linear discriminant function: $g(x) = w^T x + w_0 =< W, X >$, where $W = [w_0,\ w]$ and $X = [1,\ x]$.
- If we denote classes ω_1 and ω_2 with $y = +1$ and $y = -1$ respectively, we get the decision rule $y = \operatorname{sign} g(x)$.

Margin

- Define margin $M(x, y) = g(x)y$
  - $M(x, y) > 0 <=>$ object x is correctly classified
  - $|M(x, y)| = M'(x) \geq 0$ measures confidence of decision

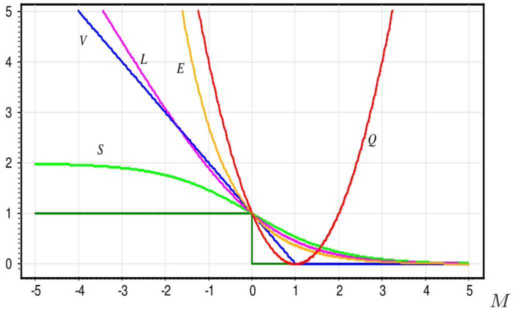
- Upper boundary on misclassification:

$$\begin{aligned} Q_{accurate}(w|X) &= \sum_i \mathbb{I}[M(x_i|w) < 0] \\ &\leq \sum_i \mathcal{L}(M(x_i|w)) = Q_{approx}(w|X) \end{aligned}$$

- Optimization task to get weights:

$$Q_{approx}(w|X) = \sum_{i=1}^{n} \mathcal{L}(M(x_i|w)) = \sum_{i=1}^{n} \mathcal{L}(\langle w, x_i \rangle y_i) \to \min_w$$

Approximating loss functions



$$
\begin{aligned}
Q(M) &= (1-M)^2\\
V(M) &= (1-M)_+\\
S(M) &= 2(1+e^{M})^{-1}\\
L(M) &= \log_2(1+e^{-M})\\
E(M) &= e^{-M}
\end{aligned}
$$

- SVM: $(1-M)_+$, logistic regression: $\ln(1+e^{-M})$
- Sigmoid: more tight approximation, but non-convex.
- Exponential: strongly affected by outliers.

Optimization

- Optimization task to get weights:

$$Q_{approx}(w|X) = \sum_{i=1}^{n} \mathcal{L}(M(x_i|w)) = \sum_{i=1}^{n} \mathcal{L}(\langle w, x_i \rangle y_i) \to \min_w$$

- Gradient descent algorithm:
  - Iteratively until convergence

$$w \leftarrow w - \eta \frac{\partial Q_{approx}(w|X)}{\partial w} = w - \eta \sum_{i=1}^{n} \mathcal{L}'(\langle w, x_i \rangle y_i) x_i y_i$$

  - η - parameter, controlling the speed of convergence.
- Faster convergence when updates are more often - e.g. at each observation. Observations may be taken randomly.

Improved optimization

Stochastic gradient descent algorithm

Calculate $\widehat{Q}_{approx}(w,X)=\sum_{i=1}^{n}\mathcal{L}(M(x_i|w))$

Iteratively, until convergence of $\widehat{Q}_{approx}$ or convergence of w:

1. select random observation (x_i, y_i)
2. adapt weights: $w \leftarrow w - \eta\mathcal{L}'(\langle w, x_i\rangle y_i)x_i y_i$
3. Estimate error: $\varepsilon_i = \mathcal{L}(\langle w, x_i\rangle y_i)$
4. Recalculate $\widehat{Q}_{approx} = (1-\alpha)\widehat{Q}_{approx} + \alpha\varepsilon_i$

Initial weights selection:

- all zeros
- random at $[-\frac{1}{2D}, \frac{1}{2D}]$ (for logistic approximation) or arbitrary random
- $w_i = \frac{\langle x^i, y\rangle}{\langle x^i, x^i\rangle}$

Selection of η

- Larger η => algorithm more prone to diverge.
- Plot $Q_{approx}(w)$ (or $\widehat{Q}_{approx}(w)$) versus iteration number t to control convergence.
- Deterministic scheme:
    - Stochastic gradient descent converges to local optima if
        - $\eta_t \to 0$
        - $\sum_{t=1}^{\infty} \eta_t = \infty$
        - $\sum_{t=1}^{\infty} \eta_t^2 < \infty$
    - Example: $\eta_t = \frac{1}{t}$
- Data dependent scheme:
    - At each step find $\eta_t = \arg\min_{\eta} Q_{approx}(w - \eta \frac{\partial Q_{approx}}{\partial w})$
    - Often analytical solution for such η exists

Comments

- Margins increase robustness, by pushing decision boundary away from the samples.
- Non-symmetrical margin:
  - $(g(x) = \tilde{g}, y = \tilde{y})$ is equivalent to $(g(x) = -\tilde{g}, y = -\tilde{y})$
  - not relevant for non-symmetric losses (example: predicting illness)
  - by introducing $g_y(x) = \begin{cases} g_1(x) & y = +1 \\ g_2(x) & y = -1 \end{cases}$ we can treat non-symmetrical case.

Table of Contents

Regularization

- Useful technique to control the trade-off between bias and variance, can be applied to any algorithm.

$$Q^{regularized}(w) = Q(w) + \tau ||w||_2$$

$$Q^{regularized}(w) = Q(w) + \tau ||w||_1$$

$$||w||_1 = \sum_{d=1}^{D} |w_d|, \quad ||w||_2 = \sum_{d=1}^{D} (w_d)^2$$

Maximum probability estimation

- $X = \{x_1, x_2, ...x_n\}$, $Y = \{y_1, y_2, ...y_n\}$ - training sample of i.i.d. observations, $(x_i, y_i) \sim p(y|x, w)$
- ML estimation $\widehat{w} = \arg\max_w p(Y|X, w)$
- Using independence assumption:

$$\prod_{i=1}^{n} p(y_i|x_i, w) = \sum_{i=1}^{n} \ln p(y_i|x_i, w) \to \max_w$$

- Approximated misclassification:

$$\sum_{i=1}^{n} \mathcal{L}(g(x_i)y_i|w) \to \min_w$$

- Interrelation:

$$\mathcal{L}(g(x_i)y_i|w) = -\ln p(y_i|x_i, w)$$

Maximum a prosteriori estimation

- $X = \{x_1, x_2, ...x_n\}$, $Y = \{y_1, y_2, ...y_n\}$ - training sample of i.i.d. observations, $(x_i, y_i) \sim p(x, y|w)$
- $x_i \sim p(x|w)$
- MAP estimation:
  - w is random with prior probability $p(w)$

$$p(w|X,Y) = \frac{p(X,Y,w)}{p(X,Y)} = \frac{p(X,Y|w)p(w)}{p(X,Y)} \propto p(X,Y|w)p(w)$$

$$w = \arg\max_w p(w|X,Y) = \arg\max_w p(X,Y|w)p(w)$$

$$\sum_{i=1}^{n} \ln p(x_i, y_i|\theta) + \ln p(w) \to \max_w$$

Gaussian prior

- Gaussian prior

$$\ln p(w,\sigma^2) = \ln\left(\frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{||w||_2^2}{2\sigma^2}}\right) = -\frac{1}{2\sigma^2}||w||_2^2 + \text{const}(w)$$

- Laplace prior

$$\ln p(w,C) = \ln\left(\frac{1}{(2C)^n} e^{-\frac{||w||_1}{C}}\right) = -\frac{1}{C}||w||_1 + \text{const}(w)$$

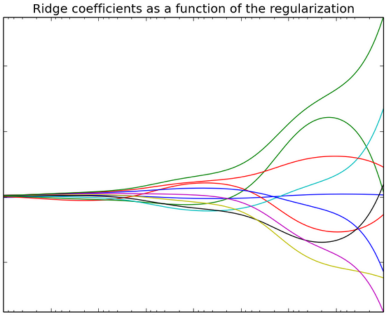
L_1 norm

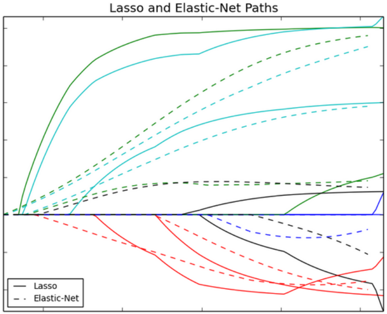
- $||w||_1$ regularizer will do feature selection.
- Consider
$$Q(w) = \sum_{i=1}^{n} \mathcal{L}_i(w) + \frac{1}{C}\sum_{d=1}^{D} |w_d|$$
- if $\frac{1}{C} > \sup_w \left|\frac{\partial \mathcal{L}(w)}{\partial w_i}\right|$, then it becomes optimal to set $w_i = 0$
- For smaller C more inequalities will become active.

Regression

Example: least squares regression
$\sum_{n=1}^{N}(w^T x_n + w_0 - y_n)^2 + R(w) \to \min_{w,w_0}$

- LASSO: least-squares regression, using $||w||_1$
- Ridge: least-squares regression, using $||w||_2$
- Elastic Net: : least-squares regression, using both





Multi-task lasso

K outputs are solved with K regressions:

- in the same feature space
- with constraint that features become included/excluded simulataneously across all tasks.

Optimization problem:

$$\min_{W} \|XW - Y\|_2^2 + \alpha \|W\|_{21}$$

where $X \in \mathbb{R}^{NxD}$, $W \in \mathbb{R}^{DxK}$, $Y \in \mathbb{R}^{NxK}$ and

$$\|W\|_{21} = \sum_{n=1}^{N} \sqrt{\sum_{k=1}^{K} w_{nk}^2}$$

Table of Contents

Model output

- Regression output: $y \in \mathbb{R}$
- Classification output:
    - exact class (e.g: majority voting)
    - score (SVM, nearest centroid)
    - class probability (logistic regression, all tree based methods, K-NN)
- Techniques for transforming score $f(x)$ to probability $p(y=1)$:
    - Platt scaling
    - isotonic regression

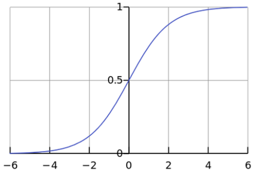
Platt scaling

Platt scaling assumes logistic relationship

$$p(y=1|x)=\frac{1}{1+e^{Af(x)+B}}$$

and fits parameters A, B using maximum likelihood.

Logistic function



- For fitting A, B training set should be different from training set where $f(x)$ was fitted (otherwise overfitting).
- Platt scaling is good for small datasets, otherwise use more general *isotonic regression*.

Isotonic regression

- The following functional relationship is assumed:

$$y_i = m(s_i) + \varepsilon_i$$

- where $\varepsilon_i \sim i.i.d.N(0, \sigma^2)$, s_i is score of classifier for x_i and $m(\cdot)$ is arbitrary monotone function.
- Using (s_i, y_i), $i = 1, 2, ...N$ as training set, find

$$\widehat{m} = \arg\min_m \sum_i (y_i - m(f_i))^2$$

- Should be fitted on separate validation set.
- Piecewise constant solution is found in linear time.

Sample fit of isotonic regression

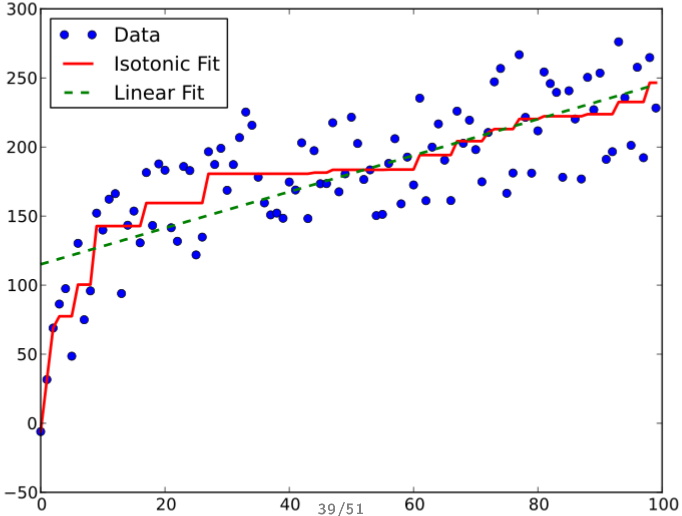


Table of Contents

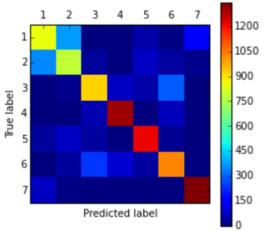
6 Model evaluation
- Evaluation of class assignements

Confusion matrix

Confusion matrix:

$$\begin{array}{cc|c} & & \text{Estimated classes} \\ & & \begin{array}{cccc} 1 & 2 & \cdots & C \end{array} \\ \text{True classes} & \begin{array}{c} 1 \\ 2 \\ \vdots \\ C \end{array} & \begin{bmatrix} n_{11} & n_{12} & & \\ n_{21} & n_{22} & & \\ & & \ddots & \\ & & & n_{CC} \end{bmatrix} \end{array}$$

Visualized confusion matrix



n_{ij} - number of objects, belonging to ω_i but classified as ω_j.

2-class case

Confusion matrix:

		Estimated class	
		+	-
True class	+	True positives	False negatives
	-	False positives	True negatives

2-class case

Confusion matrix:

Estimated class

True class		+	-
	+	True positives	False negatives
	-	False positives	True negatives

Derived performance measures:

Accuracy:	$\frac{TP+TN}{P+N}$	Error rate:	$\frac{FP+FN}{P+N}$
FPR:	$\frac{FP}{N}$	TPR:	$\frac{TP}{P}$
Precision:	$\frac{TP}{TP+FP}$	Recall:	$\frac{TP}{P}$
F-measure:	$\frac{2}{\frac{1}{Precision}+\frac{1}{Recall}}$	F_β-measure:	$\frac{1}{\frac{\beta^2}{1+\beta^2}\frac{1}{Precision}+\frac{1}{1+\beta^2}\frac{1}{Recall}}$

- Accuracy - most intuitive but irrelevant for skewed classes
- All measures require specification of probability / score.

Discriminability vs. reliability

- **Discriminability** measures how well classes are classified
    - Error rate is discriminability measure
- **Reliability** how well class probabilities are estimated
    - Likelihood (y_i is the class of x_i):

$$\prod_{i=1}^{n} \widehat{p}(y_i|x_i)$$

    - Brier score:

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C}\left(\mathbb{I}[x_i \in \omega_c] - \widehat{p}(\omega_c|x_i)\right)^2$$

- Example of good discriminability and poor reliability

Table of Contents

Parametrization of predicted class proportions

Bayes minimum risk solution: assign x to ω_1 if

$$\lambda_1 p(\omega_1)p(x|\omega_1) > \lambda_2 p(\omega_2)p(x|\omega_2)$$

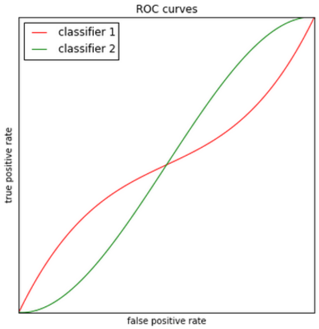
This condition is equivalent to

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} > \frac{\lambda_2 p(\omega_2)}{\lambda_1 p(\omega_1)} = \mu$$

Discriminant functions: assign x to ω_1 if $g_1(x) - g_2(x) > \mu'$.

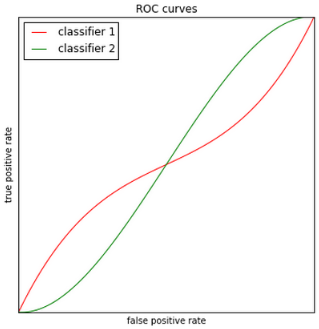
ROC curve

- ROC curve characterizes classifier performance for all values of parameter cut-off.
- As μ decreases, the algorithm becomes more inclined to select class ω_1 (positive class)
    - TPR$=1-\varepsilon_1$ increases
    - FPR$=\varepsilon_2$ also increases



ROC curve

- ROC curve characterizes classifier performance for all values of parameter cut-off.
- As μ decreases, the algorithm becomes more inclined to select class ω_1 (positive class)
    - TPR=$1-\varepsilon_1$ increases
    - FPR=ε_2 also increases



How to compare different classifiers?

ROC properties

- Better ROC curves are more concave
- Diagonal represents random guessing
- Expected loss is equal to

$$L = \lambda_2 p(\omega_2)\varepsilon_2 + \lambda_1 p(\omega_1)\varepsilon_1 = \lambda_2 p(\omega_2)\varepsilon_2 - \lambda_1 p(\omega_1)(1-\varepsilon_1) + \lambda_1 p(\omega_1)$$

- At optimality point iso-loss surface is tangent to ROC curve with slope tangent equal to $\frac{\lambda_2 p(\omega_2)}{\lambda_1 p(\omega_1)}$

ROC quality criteria

- AUC:
  - global performance characteristic
  - equals the probability that for random $x_1 \in \omega_1$ and $x_2 \in \omega_2$ it would be true that: $\widehat{p}(\omega_1|x_1) > \widehat{p}(\omega_2|x)$
- LC index:
  - rescale λ_1 and λ_2 so that $\lambda_1 + \lambda_2 = 1$
  - define $\lambda_1 = \lambda$, $\lambda_2 = 1 - \lambda$
  - for each $\lambda \in [0, 1]$ calculate

    $$L(\lambda) = \begin{cases} +1 & \text{if 1st classifier is better} \\ -1 & \text{if 2nd classifier is better} \end{cases}$$
  - define probability for $p(\lambda)$ (example: triangular)
  - choose 1-st classifier iff $\int_0^1 L(\lambda)p(\lambda)d\lambda > 0$.

Comments on model evaluation

- Bayes minimum error rate - theoretical lower bound for classification
    - need to know $P(x, y)$.
- Training error rate - optimistically biased
- Test error rate - pessimistically biased (since part of data used for error estimation)

Holdout estimate of error rate distribution

Let e be the probability of making error on previously unseen object. Probability of observing k errors on test sample of size n:

$$p(k|e,n) = \binom{n}{k} e^k (1-e)^{n-k}$$

Then

$$p(e|k,n) = \frac{p(e,k|n)}{p(k|n)} = \frac{p(k|e,n)p(e|n)}{\int p(k|n)p(e|n)de}$$

Assuming that $p(e|n) \equiv const$, we obtain

$$p(e|k,n) = \frac{p(k|e,n)}{\int p(k|n)de} \propto e^k (1-e)^{n-k}$$

Since beta-distribution
$Be(x|\alpha,\beta) = [\Gamma(\alpha+\beta)/(\Gamma(\alpha)\Gamma(\beta))]x^{\alpha-1}(1-x)^{\beta-1}$ it follows that

$$p(e|k,n) \sim Be(k+1, n-k+1)$$