Evaluation

Andrea Passerini
passerini@disi.unitn.it

Machine Learning

Evaluation

Basic concepts

- Evaluation requires to define performance measures to be optimized
- Performance of learning algorithms cannot be evaluated on entire domain (generalization error) $\rightarrow$ approximation needed
- Performance evaluation is needed for:
  - tuning hyperparameters of learning method (e.g. type of kernel and parameters, learning rate of perceptron)
  - evaluating quality of learned predictor
  - computing statistical significance of difference between learning algorithms

Performance measures

Training Loss and performance measures

- The training loss function measures the cost paid for predicting $f(\mathbf{x})$ for output y
- It is designed to boost effectiveness and efficiency of learning algorithm (e.g. hinge loss for SVM):
  - it is not necessarily the best measure of final performance
  - e.g. misclassification cost is never used as it is piecewise constant (not amenable to gradient descent)
- Multiple performance measures could be used to evaluate different aspects of a learner

Performance measures

Binary classification

True\ Pred	Positive	Negative
Positive	TP	FN
Negative	FP	TN

- The *confusion matrix* reports true (on rows) and predicted (on column) labels
- Each entry contains the number of examples having label in row and predicted as column:

  tp True positives: positives predicted as positives
  tn True negatives: negatives predicted as negatives
  fp False positives: negatives predicted as positives
  fn False negatives: positives predicted as negatives

Binary classification

Accuracy

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

- *Accuracy* is the fraction of correctly labelled examples among all predictions
- It is one minus the misclassification cost

Problem

- For strongly unbalanced datasets (typically negatives much more than positives) it is not informative:
  - Predictions are dominated by the larger class
  - Predicting everything as negative often maximizes accuracy
- One possibility consists of *rebalancing* costs (e.g. a single positive counts as N/P where N=TN+FP and P=TP+FN)

Binary classification

Precision

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

- It is the fraction of positives among examples predicted as positives
- It measures the *precision* of the learner when precting positive

Recall or Sensitivity

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

- It is the fraction of positive examples predicted as positives
- It measures the *coverage* of the learner in returning positive examples

F-measure

$$F_\beta = \frac{(1+\beta^2)(Pre * Rec)}{\beta^2 Pre + Rec}$$

- Precision and recall are complementary: increasing precision typically reduces recall
- F-measure combines the two measures balancing the two aspects
- β is a parameter trading-off precision and recall

F_1

$$F_1 = \frac{2(Pre * Rec)}{Pre + Rec}$$

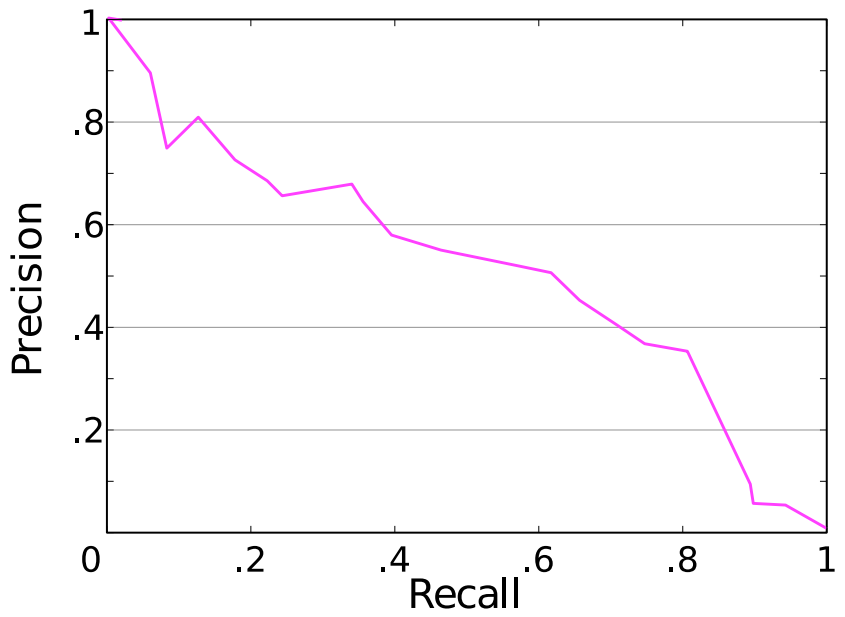
- It is the F-measure for $\beta = 1$
- It is the harmonic mean of precision and recall

Binary Classification

Precision-recall curve

- Classifiers often provide a confidence in the prediction (e.g. margin of SVM)
- A hard decision is made setting a threshold on the classifier (zero for SVM)
- Acc,Pre,Rec,F_1 all measure peformance of a classifier for a specific threshold
- It is possible to change the threshold if interested in maximizing a specific performance (e.g. recall)

Binary Classification



Precision-recall curve

- By varying threshold from min to max possible value, we obtain a curve of performance measures
- This curve can be shown plotting one measure (recall) against the complementary one (precision)
- It is possible to investigate the performance of the learner in different scenarios (e.g. at high precision)

Binary Classification

Area under Pre-Rec curve

- A single aggregate value can be obtained taking the area under the curve
- It combines the performance of the algorithm for all possible thresholds (without preference)

Performance measures

Multiclass classification

T\P	y_1	y_2	y_3
y_1	n_{11}	n_{12}	n_{13}
y_2	n_{21}	n_{22}	n_{23}
y_3	n_{31}	n_{32}	n_{33}

- Confusion matrix is generalized version of binary one
- n_{ij} is the number of examples with class y_i predicted as y_j.
- The main diagonal contains true positives for each class
- The sum of off-diagonal elements along a column is the number of false positives for the column label
- The sum of off-diagonal elements along a row is the number of false negatives for the row label

$$FP_i = \sum_{j \neq i} n_{ji} \qquad FN_i = \sum_{j \neq i} n_{ij}$$

Performance measures

Multiclass classification

- ACC,Pre,Rec,F1 carry over to a per-class measure considering as negatives examples from other classes.
- E.g.:

$$Pre_i = \frac{n_{ii}}{n_{ii} + FP_i} \qquad Rec_i = \frac{n_{ii}}{n_{ii} + FN_i}$$

- *Multiclass accuracy* is the overall fraction of correctly classified examples:

$$MAcc = \frac{\sum_i n_{ii}}{\sum_i \sum_j n_{ij}}$$

Performance measures

Regression

- Root mean squared error (for dataset $\mathcal{D}$ with $n = |\mathcal{D}|$):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{x}_i) - y_i)^2}$$

- Pearson correlation coefficient (random variables X, Y):

$$\rho = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \bar{X})(Y - \bar{Y})]}{\sqrt{E[(X - \bar{X})^2]E[(Y - \bar{Y})^2]}}$$

- Pearson correlation coefficient (for regression on $\mathcal{D}$):

$$\rho = \frac{\sum_{i=1}^{n}(f(\mathbf{x}_i) - \bar{f}(\mathbf{x}_i))(y_i - \bar{y}_i)}{\sqrt{\sum_{i=1}^{n}(f(\mathbf{x}_i) - \bar{f}(\mathbf{x}_i))^2 \sum_{i=1}^{n}(y_i - \bar{y}_i)^2}}$$

- where $\bar{z}$ is the average of z on $\mathcal{D}$.

Performance estimation

Hold-out procedure

- Computing performance measure on training set would be optimistically biased
- Need to retain an independent set on which to compute performance:

  validation set when used to estimate performance of different algorithmic settings (i.e. hyperparameters)

  test set when used to estimate final performance of selected model

- E.g.: split dataset in 40%/30%/30% for training, validation and testing

Problem

- Hold-out procedure depends on the specific test (and validation) set chosen (esp. for small datasets)

Performance estimation

k-fold cross validation

- Split $\mathcal{D}$ in k equal sized disjoint subsets $\mathcal{D}_i$.
- For $i \in [1, k]$
  - train predictor on $\mathcal{T}_i = \mathcal{D} \setminus \mathcal{D}_i$
  - compute score S of predictor $L(\mathcal{T}_i)$ on test set $\mathcal{D}_i$:

$$S_i = S_{\mathcal{D}_i}[L(\mathcal{T}_i)]$$

- return average score across folds

$$\bar{S} = \frac{1}{k} \sum_{i=1}^{k} S_i$$

k-fold cross validation: Variance

- The variance of the average score is computed as (assuming independent folds):

$$Var[\bar{S}] = Var[\frac{S_1 + \cdots + S_k}{k}] = \frac{1}{k^2} \sum_{j=1}^{k} Var[S_j]$$

- We cannot exactly compute $Var[S_j]$, so we approximate it with the *unbiased* variance across folds:

$$Var[S_j] = Var[S_h] \approx \frac{1}{k-1} \sum_{i=1}^{k} (S_i - \bar{S})^2$$

- giving

$$Var[\bar{S}] \approx \frac{1}{k^2} \sum_{j=1}^{k} \frac{1}{k-1} \sum_{i=1}^{k} (S_i - \bar{S})^2 = \frac{1}{k^{\cancel{2}}} \frac{k^{\cancel{}}}{k-1} \sum_{i=1}^{k} (S_i - \bar{S})^2$$

Comparing learning algorithms

Hipothesis testing

- We want to compare generalization performance of two learning algorithms
- We want to know whether observed different in performance is *statistically significant* (and not due to some noisy evaluation)
- Hypothesis testing allows to test the statistical significance of a hypothesis (e.g. the two predictors have different performance)

Hypothesis testing

Test statistic

null hypothesis H_0 default hypothesis, for rejecting which evidence should be provided

test statistic Given a sample of k realizations of random variables $X_1, \ldots, X_k$, a *test statistic* is a statistic $T = h(X_1, \ldots, X_k)$ whose value is used to decide wether to reject H_0 or not.

Example

Given a set of measurements $X_1, \ldots, X_k$, decide wether the actual value to be measured is zero.

null hypothesis the actual value is zero

test statistic sample mean:

$$T = h(X_1, \ldots, X_k) = \frac{1}{k} \sum_{i=1}^{k} X_i = \bar{X}$$

Hypothesis testing

Glossary

tail probability probability that T is at least as great (right tail) or at least as small (left tail) as the observed value t.

p-value the probability of obtaining a value T *at least as extreme* as the one observed t , in case H_0 is true.

Type I error reject the null hypothesis when it's true

Type II error accept the null hypothesis when it's false

significance level the largest acceptable probability for committing a type I error

critical region set of values of T for which we reject the null hypothesis

critical values values on the boundary of the critical region

The test

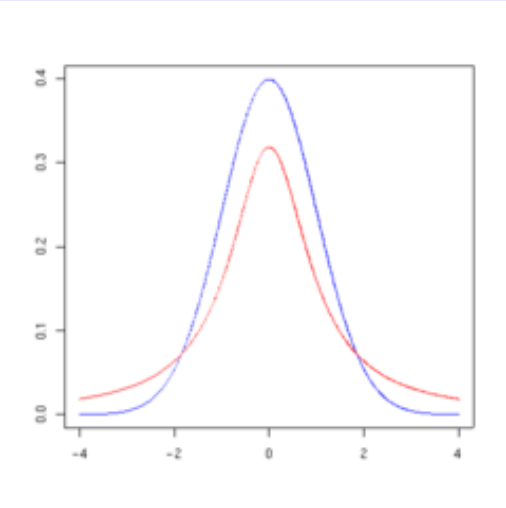
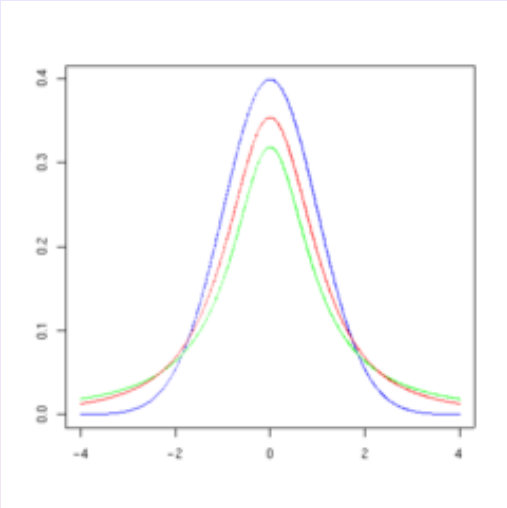
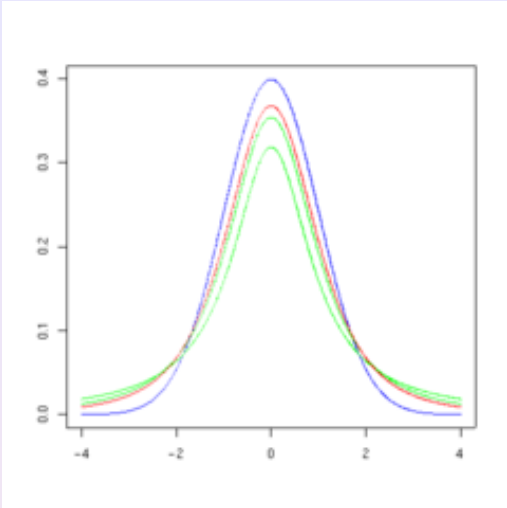
- The test statistics is given by the standardized (also called *studentized*) mean:

$$T = \frac{\bar{X} - \mu_0}{\sqrt{\tilde{Var}[\bar{X}]}}$$

  where $\tilde{Var}[\bar{X}]$ is the approximated variance (using unbiased sample one)
- Assuming the samples come from an unknown Normal distribution, the test statistics has a t_{k-1} distribution under the null hypothesis
- The null hypothesis can be rejected at significance level α if:

$$T \leq -t_{k-1,\alpha/2} \qquad \text{or} \qquad T \geq t_{k-1,\alpha/2}$$

t-test

t_{k-1} distribution

- bell-shaped distribution similar to the Normal one
- wider and shorter: reflects greater variance due to using $\tilde{Var}[\bar{X}]$ instead of the true unknown variance of the distribution.
- $k-1$ is the number of degrees of freedom of the distribution (related to number of independent events observed)
- t_{k-1} tends to the standardized normal z for $k \to \infty$.

Comparing learning algorithms

Hypothesis testing

- Run k-fold cross validation procedure for algorithms A and B
- Compute mean performance difference for the two algorithms:

$$\hat{\delta} = \frac{1}{k}\sum_{i=1}^{k}\delta_i = \frac{1}{k}\sum_{i=1}^{k} S_{\mathcal{D}_i}[L_A(\mathcal{T}_i)] - S_{\mathcal{D}_i}[L_B(\mathcal{T}_i)]$$

- Null hypothesis is that mean difference is zero

Comparing learning algorithms: t-test

t-test

at significance level α:

$$\frac{\bar{\delta}}{\sqrt{\tilde{Var}[\bar{\delta}]}} \leq -t_{k-1,\alpha/2} \quad \text{or} \quad \frac{\bar{\delta}}{\sqrt{\tilde{Var}[\bar{\delta}]}} \geq t_{k-1,\alpha/2}$$

where:

$$\sqrt{\tilde{Var}[\bar{\delta}]} = \sqrt{\frac{1}{k(k-1)} \sum_{i=1}^{k} (\delta_i - \bar{\delta})^2}$$

Note

paired test the two hypotheses where evaluated over identical samples

two-tailed test if no prior knowledge can tell the direction of difference (otherwise use *one-tailed* test)

t-test example

10-fold cross validation

- Test errors:

$\mathcal{D}_i$	$S_{\mathcal{D}_i}[L_A(\mathcal{T}_i)]$	$S_{\mathcal{D}_i}[L_B(\mathcal{T}_i)]$	δ_i
$\mathcal{D}_1$	0.81	0.80	0.01
$\mathcal{D}_2$	0.82	0.77	0.05
$\mathcal{D}_3$	0.84	0.70	0.14
$\mathcal{D}_4$	0.78	0.83	-0.05
$\mathcal{D}_5$	0.85	0.80	0.05
$\mathcal{D}_6$	0.86	0.78	0.08
$\mathcal{D}_7$	0.82	0.75	0.07
$\mathcal{D}_8$	0.83	0.80	0.03
$\mathcal{D}_9$	0.82	0.78	0.04
$\mathcal{D}_{10}$	0.81	0.77	0.04

- Average error difference:

$$\bar{\delta} = \frac{1}{10} \sum_{i=1}^{10} \delta_i = 0.046$$

t-test example

10-fold cross validation

- Unbiased estimate of standard deviation:

$$\sqrt{\tilde{Var}[\bar{\delta}]} = \sqrt{\frac{1}{10 \cdot 9} \sum_{i=1}^{10} (\delta_i - \bar{\delta})^2} = 0.0154344$$

- Standardized mean error difference:

$$\frac{\bar{\delta}}{\sqrt{\tilde{Var}[\bar{\delta}]}} = \frac{0.046}{0.0154344} = 2.98$$

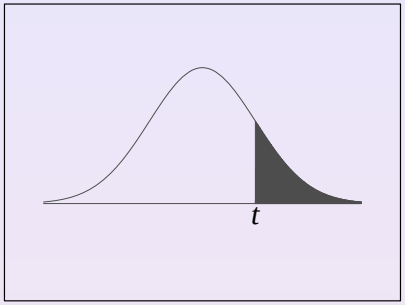
- t distribution for $\alpha = 0.05$ and $k = 10$:

$$t_{k-1,\alpha/2} = t_{9,0.025} = 2.262 < 2.98$$

- Null hypothesis rejected, classifiers are different

t-test example

t-Distribution Table



The shaded area is equal to α for $t = t_\alpha$.

df	$t_{.100}$	$t_{.050}$	$t_{.025}$	$t_{.010}$	$t_{.005}$
1	3.078	6.314	12.706	31.821	63.657
2	1.886	2.920	4.303	6.965	9.925
3	1.638	2.353	3.182	4.541	5.841
4	1.533	2.132	2.776	3.747	4.604
5	1.476	2.015	2.571	3.365	4.032
6	1.440	1.943	2.447	3.143	3.707
7	1.415	1.895	2.365	2.998	3.499
8	1.397	1.860	2.306	2.896	3.355
9	1.383	1.833	2.262	2.821	3.250
10	1.372	1.812	2.228	2.764	3.169
11	1.363	1.796	2.201	2.718	3.106
12	1.356	1.782	2.179	2.681	3.055
13	1.350	1.771	2.160	2.650	3.012
14	1.345	1.761	2.145	2.624	2.977
15	1.341	1.753	2.131	2.602	2.947
16	1.337	1.746	2.120	2.583	2.921
17	1.333	1.740	2.110	2.567	2.898
18	1.330	1.734	2.101	2.552	2.878
19	1.328	1.729	2.093	2.539	2.861

References

Hypothesis testing T. Mitchell, *Machine Learning*, McGraw Hill, 1997 (chapter 5)