Comp 135
Introduction to Machine Learning and Data Mining

Fall 2016

Professor: Roni Khardon

Computer Science
Tufts University

Evaluating machine learning outcomes

- How should we define a good outcome for a machine learning algorithm?
- How do we know when an algorithm is doing well in a particular application?
- How can we compare different classifiers?
- How can we compare different learning algorithms?

Evaluating machine learning outcomes

- More concrete questions:

- What quantity should we measure?
- How can we estimate it?
- Can we get quantitative guarantees for estimates and comparisons?

What to Measure?

- We have so far focused on classification and looked at accuracy
- But this depends on the application and data distribution …

Confusion Matrix

- + - <-- classified as
- TP FN | true label +
- FP TN | true label -

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

- In many applications one class (typically the +) has very low frequency
- Running algorithm to optimize Acc often yields bad results. Why?

IR Terminology

- Context: search for items with QueryTerm
- De-emphasize role of Neg examples
- Aims to measure quality of response

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

- Tradeoff given by Precision/Recall curve
- And by $\text{F} = \frac{2 \cdot R \cdot P}{R + P}$

Medical Community Terminology

- Context: test to identify Med Condition

$$\text{Sensitivity} = \text{Recall} = \frac{TP}{TP + FN}$$

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

- These measure "accuracy" within each class

Signal Detection Termilology

- Context: identify "signal"

$$\text{TPrate} = \text{Recall} = \frac{TP}{TP + FN}$$

$$\text{FPrate} = 1 - \text{Specificity} = \frac{FP}{TN + FP}$$

- The ROC curve (receiver operator characteristic) plots TPrate vs FPrate

ROC Curves

- The AROC (area under ROC curve) has a nice interpretation:

  AROC = probability of correctly ranking a random Pos example above a random Neg example

From Ranking to ROC Curve

- Many learning algorithms (e.g., Naïve Bayes, Perceptron) provide a numerical output that can be used to rank examples in addition to the prediction of +/- label
- This can be used to produce a ROC curve for the algorithm by changing its threshold specifying the transition from Neg to Pos.

How to Measure?

- Validation Set Method:
  - keep aside a portion of the example set
  - Train model on remaining data
  - Measure performance on validation set
- (+) Unbiased estimate of quantity
- (-) Wastes data …
- (-) Variance in estimate due to choice of validation set. Can we fix this?

How to Measure?

- Validation Set Method:
  - keep aside a portion of the example set
  - Train model on remaining data
  - Measure performance on validation set
- Can reduce variance by repeating k times and averaging
- But this introduces bias in estimates because the train/test data in different runs are highly correlated

How to Measure?

- Cross Validation Method:
- Divide data into k portions (called folds)
- Repeat k times
  - Train model on data from all folds except k
  - Measure performance on k'th fold
- Average the performance

- Test data in different folds is disjoint
- But training sets are not

Stratified Cross Validation

- Another source of variance in cross validation is the fact that the test sets across folds do not have exactly the same class frequencies
- Stratified cross validation removes this variation by splitting into random subsets of the same size per class

Implementing Stratified Cross Validation

- Split data according to class
- Randomly permute the examples in each class
- Partition each class (by permuted index) into k equal size portions (up to +/-1)
- Join portions i from all classes to get fold i

- This gives k portions of the same size and having the same class proportions

Leave One Out Method

- Setting k=N (number of examples) we get the leave one out method

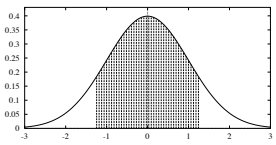
- Often effective but
- High variance in individual estimates
- Expensive if we need to train N times
- But for some algorithms, e.g. kNN, can get away without significant expense.

Model Selection

- Cross validation can be applied to configure parameters in many algorithms during the training phase.
  - Pick k for nearest neighbors
  - Pick prune threshold in DT
- In this case we perform cross validation on training set only
- Using the result we pick best parameter value
- Finally: we train again on the entire data

Quantitative Comparisons

Review notions from Normal distributions



Beware to check if using tables for one-sided bounds or two sided bounds.
These slides use two sided bounds as in the shaded area.
Some other sources use one sided bounds. It is straightforward to extract one from the other.

80% of area (probability) lies in $\mu \pm 1.28\sigma$

N% of area (probability) lies in $\mu \pm z_N \sigma$

$N\%$:	50%	68%	80%	90%	95%	98%	99%
z_N:	0.67	1.00	1.28	1.64	1.96	2.33	2.58

Image from Mitchell text/slides

Quantitative Comparisons

- From probability to estimates

  N% of area (probability) lies in $\mu \pm z_N \sigma$

- Probability Statement

  [with probability $N\%$] $x \in \mu \pm z_N \sigma$

- Confidence interval

  [with confidence $N\%$] $\mu \in x \pm z_N \sigma$

Confidence Interval

[with confidence $N\%$] $\mu \in x \pm z_N \sigma$

- Assume $x \sim \mathcal{N}(\mu, \sigma^2)$
- And we sample to observe value of x

  [with confidence 95%] $\mu \in x \pm 1.96\sigma$

Evaluating one classifier

- We measure the performance on test set of *n* examples
- If the error rate is *p* then the number of mistakes on the test set is distributed as Binomial(n,p)
- Our estimate is $\hat{p} = \frac{\#\text{ mistakes}}{n}$
- ^p distributed approx as $\hat{p} \sim \mathcal{N}(p, \frac{p(1-p)}{n})$

Evaluating one classifier

- ^p distributed approx as $\hat{p} \sim \mathcal{N}(p, \frac{p(1-p)}{n})$
- Therefore, we can apply

  [with confidence $N\%$] $\mu \in x \pm z_N \sigma$

- To get

  [with confidence $N\%$] $p \in \hat{p} \pm z_N \sqrt{\frac{p(1-p)}{n}}$

Evaluating one classifier

- Good news: we have an interval for *p*

  [with confidence $N\%$] $p \in \hat{p} \pm z_N \sqrt{\frac{p(1-p)}{n}}$

- But: we cannot plug in p in variance part
- Solution 1: p(1-p)<=0.25

  [with confidence $N\%$] $p \in \hat{p} \pm z_N \sqrt{\frac{1}{4n}}$

- Solution 2: (not justified)

  [with confidence $N\%$] $p \in \hat{p} \pm z_N \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$

Evaluating Learning Algorithm

- Cross validation error estimate $\hat{e} = \frac{1}{k} \sum e_i$
- Define *e* as the expected error rate when running alg (on data of this size)
- Then we have: $e_i \sim \mathcal{N}(e, \sigma_e^2)$

  $\hat{e} \sim \mathcal{N}(e, \sigma_e^2/k)$

- And we can use

  [with confidence $N\%$] $\mu \in x \pm z_N \sigma$

Evaluating Learning Algorithm

- Cross validation error estimate $\hat{e} = \frac{1}{k}\sum e_i$
- Define e as the expected error rate when running alg (on data of this size)
- Then we have: $e_i \sim \mathcal{N}(e, \sigma_e^2)$

  $\hat{e} \sim \mathcal{N}(e, \sigma_e^2/k)$
- And we can use

$$[\text{with confidence } N\%] \quad e \in \hat{e} \pm z_N \sqrt{\sigma_e^2/k}$$

Evaluating Learning Algorithm

$$[\text{with confidence } N\%] \quad e \in \hat{e} \pm z_N \sqrt{\sigma_e^2/k}$$

- But: we do not know σ_e
- Solution 1: plug in the estimate s instead of σ_e

$$s = \sqrt{\frac{1}{k-1}\sum (e_i - \hat{e})^2}$$

- Solution 2: a more accurate interval through T random variable

Sol2 is more accurate and as easy to implement so avoid Sol1.

Evaluating Learning Algorithm

- It turns out [we skip definitions and details] that $\frac{\hat{e} - e}{(s/\sqrt{k})} \sim T_{k-1}$
- And that this implies

$$[\text{with confidence } N\%] \quad e \in \hat{e} \pm t_{N,k-1} \frac{s}{\sqrt{k}}$$

- And more concretely

$$[\text{with confidence } N\%] \quad e \in \hat{e} \pm t_{N,k-1} \sqrt{\frac{\sum (e_i - \hat{e})^2}{(k-1)k}}$$

Degrees of freedom	p = 0.1	p = 0.05	p = 0.02	p = 0.01	p = 0.002	p = 0.001
1	6.314	12.706	31.821	63.657	318.310	636.620
2	2.920	4.303	6.965	9.925	22.327	31.598
3	2.353	3.182	4.541	5.841	10.214	12.924
4	2.132	2.776	3.747	4.604	7.173	8.610
5	2.015	2.571	3.365	4.032	5.893	6.869
6	1.943	2.447	3.143	3.707	5.208	5.959
7	1.895	2.365	2.998	3.499	4.785	5.408
8	1.860	2.306	2.896	3.355	4.501	5.041
9	1.833	2.262	2.821	3.250	4.297	4.781
10	1.812	2.228	2.764	3.169	4.144	4.587
11	1.796	2.201	2.718	3.106	4.025	4.437
12	1.782	2.179	2.681	3.055	3.930	4.318
13	1.771	2.160	2.650	3.012	3.852	4.221
14	1.761	2.145	2.624	2.977	3.787	4.140
15	1.753	2.131	2.602	2.947	3.733	4.073
16	1.746	2.120	2.583	2.921	3.686	4.015
17	1.740	2.110	2.567	2.898	3.646	3.965
18	1.734	2.101	2.552	2.878	3.610	3.922
19	1.729	2.093	2.539	2.861	3.579	3.883
20	1.725	2.086	2.528	2.845	3.552	3.850
21	1.721	2.080	2.518	2.831	3.527	3.819
22	1.717	2.074	2.508	2.819	3.505	3.792
23	1.714	2.069	2.500	2.807	3.485	3.767
24	1.711	2.064	2.492	2.797	3.467	3.745
25	1.708	2.060	2.485	2.787	3.450	3.725
26	1.706	2.056	2.479	2.779	3.435	3.707
27	1.703	2.052	2.473	2.771	3.421	3.690
28	1.701	2.048	2.467	2.763	3.408	3.674
29	1.699	2.045	2.462	2.756	3.396	3.659
30	1.697	2.042	2.457	2.750	3.385	3.646
40	1.684	2.021	2.423	2.704	3.307	3.551
60	1.671	2.000	2.390	2.660	3.232	3.460
120	1.658	1.980	2.358	2.617	3.160	3.373
∞	1.645	1.960	2.326	2.576	3.090	3.291

Table from http://www.saburchill.com/

Comparing Two Algorithms

- We can calculate an interval for the error of each and say that one is better if the intervals do not overlap
- But we can do much better when variance σ_e is large
- Alg1 Alg2 Diff
- 60 68 8
- 70 77 7
- 80 88 8

Comparing Two Algorithms

- We can calculate an interval for the error of each and say that one is better if the intervals do not overlap
- But we can do much better when variance σ_e is large

→ Compute a confidence interval the difference in performance.

→ Matched tests: estimates come from running algorithms on the same folds. This is valid and reduces variance.

Running many experiments and tests

- We develop 100 new algorithms
- Calculate interval with N=95% confidence for diff over baseline alg
- What is the probability of wrongly claiming a significant difference in at least one of these?

$$1 - 0.95^{100} = 1 - 0.059 = 0.9941$$

Running many experiments and tests

- We develop 100 new algorithms
- Calculate interval with N=95% confidence for diff over baseline alg
- Simple correction (tests are not required to be independent)
- To get 0.95 confidence for all the tests together we need each individual run with N=99.95%

$$p(\text{fail}) \leq 100(1 - 0.9995) = 100 * 0.0005 = 0.05$$

$$p(\text{not fail}) \geq 0.95$$

Evaluating machine learning outcomes

- You should now have some answers to these questions:

- What quantity should we measure?
- How can we estimate it?
- Can we get quantitative guarantees for estimates and comparisons?