
Evaluating Hypotheses

Part II

Recap on Evaluation

- Sample error

$$\text{error}_S(h) \equiv 1/n \cdot \sum_{x \in S} \delta(c(x) \neq h(x))$$

where $\delta(c(x) \neq h(x)) = 1$ if $c(x) \neq h(x)$; 0 otherwise

- Generalisation error

$$\text{error}_D(h) \equiv \Pr_{x \in D} [c(x) \neq h(x)]$$

- Approximate $\text{error}_D(h)$ using $\text{error}_S(h)$

- Use S independent from generation of h (in any way)
- Ensure S drawn from D if reporting predictive ability of h on D
- When wrapper-based model selection is necessary, avoid using S for such purposes (use parts of the training data)

- $\text{error}_S(h)$ is a random variable following a Binomial Distribution

- Can be approximated via Normal Distribution (when n large and p not close to 0 or 1)

Holdout Testing

- Most straightforward approach to hypothesis evaluation
 - Given data S^*
 - ◆ Reserve $S \subset S^*$ for evaluating $\text{error}_S(h)$ – i.e., test/validation set
 - ◆ Utilise $S^* \setminus S = S'$ for training h – i.e., training set

- S is now independent of the data used to train h
 - $\text{error}_S(h)$ not an optimistic approximation of $\text{error}_D(h)$
 - Representative of unobserved instances drawn from D

Problems with Holdout Testing

- Data is limited
 - Collecting data can be expensive
 - Labelling is expensive
 - Supervised learning datasets typically not large
- When measuring $\text{error}_S(h)$, we must account for the various sources of variance
- Interval size for $\text{error}_S(h)$ inversely related to n

Using Data More Efficiently

- Reducing variance
 - Increase sample sizes
 - Aggregating results over repeated experiments ((re)using limited data)
- Repeated Holdout
- Cross-validation
- Bootstrapping

Repeated Holdout Testing

- Simply repeat holdout testing
 - Each iteration, randomly selection new sample to train h
 - Use remaining data to evaluate h
 - Average the error rates
- Not optimal use of the data
 - Overlap between data from test/validation samples
- How can we prevent this overlap?

k-Fold Cross-Validation

- Divide the dataset S^* into k mutually exclusive subsets
 - $fold_1, \dots, fold_k$
 - Given
 - ◆ $|S^*| = n; m = \lfloor n / k \rfloor$
 - ◆ $p = n \% k; q = k - p$
 - There are
 - ◆ p folds with $m + 1$ instances
 - ◆ q folds with m instances
 - k hypotheses (h_1, \dots, h_k) are trained and evaluated
 - Each h_i ($1 \leq i \leq k$)
 - ◆ Trained using $S^* \setminus fold_i$
 - ◆ Evaluated using $fold_i$
 - Report average across the k results
 - Every instance evaluated once
- Example
 - ◆ Given a dataset S^* such that
 - ◆ $n = 134$
 - ◆ $k = 10$
 - We have
 - ◆ $m = \lfloor 134 / 10 \rfloor = 13$
 - ◆ $p = 134 \% 10 = 4$
 - ◆ $q = 10 - 4 = 6$
 - Thus
 - ◆ 4 folds with 14 instances
 - ◆ 6 folds with 13 instances

Issues with k -Fold Cross-Validation

1. How many splits are possible?

$$\left[\prod_{i=1}^p \binom{n - (m+1)(i-1)}{m+1} \right] \times \left[\prod_{j=1}^q \binom{n - (m+1)p - m(j-1)}{m} \right] / [(p!)(q!)]$$

- Still many possible permutations of k -folds
2. What is the class distribution across the folds?
 - Example
 - ◆ S^* contains 100 instances
 - ◆ 20 with label 1; 80 with label 0
 - ◆ Assume 10-fold cross-validation
 - What is the greatest class imbalance possible with this example?

Stratified k -fold Cross-Validation

- Ensures that each fold approximately has the same class distribution as the given dataset
- Does *not* randomly assigning each instance in S^* to a fold
- Separate S^* into label-specific subsets
 - S_{y_i} : all instance from S^* with label y_i
 - Divide S_{y_i} into the k folds (similar to how we did it over S^*)
 - Repeat for all classes ensuring all folds differ in size by at most 1
- Variance can further be decreased by running cross-validation or stratified cross-validation repeatedly (m times)
 - i.e., $m \times k$ -fold cross-validation or $m \times k$ -fold stratified cross-validation

leave-one-out Cross-Validation

- Let $k = n$
 - Each fold contains exactly 1 instance
- Train h using all data except 1 instance
- Evaluate h on left out instance
- How many possible sets of folds here?
 - One; it is deterministic
- Issues
 - Have to train n hypotheses; computational cost may be high
 - Strictly, each fold cannot follow class distribution in S^*

leave-one-out Cross-Validation

□ Example

- Random dataset split equally into 2 classes
- 100 instances
- 50 with label 1; 50 with label 0

□ Evaluate mode / majority-class h

- Always classifies instances based on most-frequent class in training set

□ With leave-one-out cross-validation

- Each training fold contains 50 instances of one label (y) and 49 of the other (y')
- Instance left out for testing has label y'
- Evaluation gives 100% error across the n folds!
 - ◆ Expected error should be 50%

The 0.632 Bootstrap

- Randomly sample (with replacement) n instances from S^*
 - Let this sample be S'
 - Use S' to train h
- Use $S = S^* \setminus S'$ to evaluate h
- When sampling without replacement, the probability of picking each $x \in S^*$ is not uniform
 - Sample drawn not *iid*
 - Bootstrapping always samples each instance with the same probability

The 0.632 Bootstrap

- Chance of picking a particular instance is $1/n$
- Chance of not picking a particular instance is $1 - 1/n$
- Probability that an instance is not chosen in the sample of size n is

$$(1 - 1/n)^n = e^{-1} = 0.368$$

- Recall that we are sampling with replacement so each draw is independent
- This where the name comes from: $1 - 0.368 = 0.632$
 - The probability that an instance is chosen to train h

The 0.632 Bootstrap

- h is only trained with about 63.2% of the original data
- Error estimate can be quite pessimistic
- Recommendation is to combine it with error on the data used in training

$$0.632 \cdot \text{error}_S(h) + 0.368 \cdot \text{error}_{S'}(h)$$

- Repeat the process several times and take the average
- 0.632 Bootstrap is especially good with very small datasets

Issues with 0.632 Bootstrap

- Same example as before
 - Random dataset split equally into 2 classes
 - 100 instances
 - 50 with label 1; 50 with label 0

- Evaluate non-pruned decision tree h
 - h is consistent with training data, so error over data used in training is 0
 - Assume that true error rate of h is 0.5

- With 0.632 bootstrap evaluation
 - Bootstrap estimate is
$$0.632 * 0.5 + 0.368 * 0 = 0.316$$
 - True error is 0.5, but the bootstrap error is 0.316
 - Just under 20% difference!

Revisiting Wrapper-Methods

- Recall that when measuring some $\text{error}_S(h)$ we must avoid using S for any part of the training/selection of h
 - Feature selection
 - Hyperparameter tuning
- To facilitate this, we use 3 independent samples
 - **Training set** – trains various h_1, \dots, h_m (e.g., each with different features, hyperparameters, etc.)
 - **Validation set** – evaluates each h with the purpose of choosing between them
 - **Test set** – evaluates the chosen h
- Once evaluation is done, use all data on chosen **learning scheme**
- How can we do this with cross-validation?

Nested Cross-Validation

- “Outer” k -fold cross-validation to estimate quality of entire learning process
- With each of the k iterations
 - Consider training data as a new dataset and perform an “inner” p -fold cross-validation
 - Results of “inner” p -fold cross-validation guide model choice
 - “Inner” cross-validation is part of the learning scheme!
- Suppose you have a dataset with 100 instances
 - Assume 10-fold outer and inner cross-validation
 - What is the size of each sample?
 - ◆ Test set: 10 instances (100 / 10)
 - ◆ Validation set: 9 instances (90 / 10)
 - ◆ Training set: 81 instances (90 - 9)

Performance & Benchmarks

- Suppose you observe 10% error on h
 - Is this a good result?
- Application considerations
 - Needs analysis can inform us about the necessary accuracy required
 - Evaluation provides evidence for h
- Comparing against simple hypotheses (baselines)
 - Mode classifier – are you at least doing better than always guessing the most frequent class
 - ◆ 10% error when the mode classifier gets 89.5% error means your hypothesis isn't looking too good
 - Landmarkers – is h better than simple models?
 - ◆ Decision stump (decision tree with only 1 attribute split)
 - ◆ 1-nearest neighbour (a very simple instance-based learner)
 - ◆ Naive bayes

Performance & Benchmarks

□ Versus state of the art (SOTA)

- Show that your learning scheme (that produced h) is the best
- Requires good evidence to convince people

□ Comparing learning schemes

- Want to show that Scheme X is better than Scheme Y for a particular problem domain
 - ◆ For a given amount of training data
 - ◆ On average, across all possible training sets from that domain
- Simple approach assuming infinite data from domain available
 - ◆ Sample infinitely many samples of specified size
 - ◆ Use a method of evaluation – e.g., cross-validation
 - ◆ Compare mean error of Scheme X against mean error for Scheme Y

Paired t-test

- Data is limited
 - Limited estimates to compute means over
- *Student's t-test* helps determine if the mean of two samples are significantly different
 - The two sample means are computed over the k cross-validation estimates
 - One set for Scheme X and the other for Scheme Y
 - Data used to evaluate each scheme is the same (i.e., same training-validation-test sets)
 - ◆ i.e., the samples are paired
- Example
 - Given Scheme X and Y
 - Divide the dataset into k folds
 - Each iteration we use the same fold to test Scheme X and Scheme Y while the rest is used by both schemes to train h
 - here are thus k paired values under each scheme

Distribution of Means

- Given two samples of values:
 - Over Scheme X: $\text{err}_{x,1}; \text{err}_{x,2}; \dots; \text{err}_{x,k}$
 - Over Scheme Y: $\text{err}_{y,1}; \text{err}_{y,2}; \dots; \text{err}_{y,k}$
- Assume that each $\text{err}_{x,i}$ and $\text{err}_{y,i}$ pair is measured with
 - Same training data
 - Same test data
 - Only the scheme differs
- m_x and m_y are the respective means for Scheme X and Scheme Y
- From the Central Limit Theorem
 - With large k , m_x and m_y are normally distributed
 - Variances are σ_x^2/k and σ_y^2/k
 - If μ_x and μ_y are the true means, then the following are approximately normally distributed with mean 0 and variance 1

$$\frac{m_x - \mu_x}{\sqrt{\sigma_x^2 / k}} \quad \frac{m_y - \mu_y}{\sqrt{\sigma_y^2 / k}}$$

Student's Distribution

- With small sample sizes ($k < 100$) the mean follows *Student's distribution with $k - 1$ degrees of freedom*

Assuming
we have
10 estimates

9 degrees of freedom

$\Pr[T \geq z]$	z
0.1%	4.30
0.5%	3.25
1%	2.82
5%	1.83
10%	1.38
20%	0.88

normal distribution

$\Pr[T \geq z]$	z
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84

Distribution of the Differences

- Let $m_d = m_x - m_y$
- The difference of the means (m_d) also has a Student's distribution with $k-1$ degrees of freedom
- Let σ_d^2 be the variance of the differences
- The standardised version of m_d is called the t -statistic

$$t = \frac{m_d}{\sqrt{\sigma_d^2 / k}}$$

- We use t to perform the paired t -test

Performing the Test

- Fix a significance level
 - If a difference is significant at the $\alpha\%$ level, there is a $(100 - \alpha)\%$ chance that the true means differ
- Divide the significance level by two because the test is two-tailed
 - i.e., the true difference can be +ve or – ve
 - There are thus k paired values under each scheme
- Look up the value for z that corresponds to $\alpha/2$
- Compute the value of t based on the observed performance estimates for the schemes being compared
- If $t \leq -z$ or $t \geq z$ then the difference is significant
 - i.e., the *null hypothesis* (that the difference is zero) can be rejected

Confusion Matrix

- Important to analysis the kinds of errors made by h

Truth	Predicted	
	Positive	Negative
Positive	True Positive	False Negative (Type II Error)
Negative	False Positive (Type I Error)	True Negative

- Many domains have their own specific measures, which are based on the values in the confusion matrix
 - Find out the ones used in your domain, and why they are used

Questions?
