ANLY-601
Advanced Pattern Recognition

Spring 2018

L18 --- Algorithm-Independent Stuff

GEORGETOWN UNIVERSITY

Algorithm-Independent Issues

- Empirical Error Estimates
  - Hold out
  - Cross-validation
    - Leave-one out
    - K-fold cross-validation
  - Boostrap estimates

GEORGETOWN UNIVERSITY

Sampling Issues

- Suppose we have training and test datasets D_{train} and D_{test} respectively.
- We pick some model for a discriminant function $h(x\,;\,\theta)$
  where x is the input feature, and θ is the set of parameters that specify h, such as
  - class priors, parameters in class-conditional density models (means, covariance matrices), total covariance, weights in a neural network, radii in kernel density estimates …

From the training data D_{train} we form <u>estimates</u> of these parameters, and hence an estimate of the discriminant function

$$\hat{h}(x) = h(x;\hat{\theta}) = h(x; D_{train})$$

- This discriminant function defines a classifier function – e.g. the function that returns the class labels {0,1} when given the input feature x

$$\hat{f}(x,\, D_{train}) = \begin{pmatrix} 1, & h(x;\hat{\theta}) > 0 \\ 0, & h(x;\hat{\theta}) < 0 \end{pmatrix}$$

Sampling Issues

- Next we would like to know the error rate for the classifier, the probability that our classifier does not agree with the true class label *l(x)* on the next (independent and previously not seen) sample

$$\mathcal{E}(D_{train}) = \int p(x)\, P(\hat{f} \neq l \mid x)\; dx$$

- However we can't compute this, so we use another data set to estimate it by counting errors

$$\hat{\mathcal{E}}(D_{train}, D_{test} = \{(x_i, l_i), i = 1..N_{test}\}) = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \left[1 - \delta(\hat{f}(x_i, D_{train}), l_i)\right]$$

$$= \frac{N_{error}}{N_{test}}$$

this is called the holdout estimate of error rate.

Sampling Issues

- The estimated classifier $\hat{f}(x, D_{train})$ is a random variable dependent on the particular training set.

- Its estimated error rate $\hat{\mathcal{E}}(D_{train}, D_{test})$ is a random variable, dependent on the particular test set.

- So how do we compare different classifiers on a given problem?

Empirical Error Rate Estimates

We estimate the performance of a classifier by counting errors on a finite test set.

$$\hat{\mathcal{E}} = \frac{\#\ of\ \ errors\ \ on\ \ test\ \ data}{N} \equiv \frac{N_{errors}}{N}$$

Suppose the true error rate is $\mathcal{E}$. Then the number of errors made on a sample of *N* objects follows a binomial distribution

$$P(N_{errors}) = \binom{N}{N_{errors}} \mathcal{E}^{N_{errors}} (1-\mathcal{E})^{N-N_{errors}}$$

The average number of errors is $E[N_{errors}] = N\,\mathcal{E}$ so $E[\hat{\mathcal{E}}] = \mathcal{E}$

The variance is

$$\mathrm{var}(\hat{\mathcal{E}}) = \frac{1}{N^2}\,\mathrm{var}(N_{errors}) = \frac{1}{N^2}\,N\,\mathcal{E}\,(1-\mathcal{E}) = \frac{\mathcal{E}\,(1-\mathcal{E})}{N}$$

and can be substantial for *N* relatively small, or $\mathcal{E}$ near ½.

Error Estimates

Problems with holdout method:

- Usually have only one dataset. Partition it into training (D_{train}) and test (D_{test}). This gives ONE measurement of the error rather than the true error rate.

  Since the empirical error estimate is <u>unbiased</u> it's clear

$$E_{D_{test}}[\hat{\mathcal{E}}(D_{train}, D_{test})] = \mathcal{E}(D_{train})$$

- We'd like to use as much of the data as possible for training, as this gives a more accurate (lower variance) estimator of the classifier.

Error Estimates

We'd like to use as much of the data as possible for training, as this gives a more accurate (lower variance) estimator of the classifier.

One approach is leave-one-out.

Start with N data samples.

1. Choose one sample and remove it.
2. Design classifier based on remaining N-1 samples
3. Test on single removed sample.

but this increases variance of error rate estimate.

Cross Validation

Both the hold-out and the leave-one out provide a single measurement. We really want an average over datasets, but we have only one dataset!

Solution

> Generate many splits into training and test sets.
> Measure the empirical error rate on each of these splits, and average.

Leave One Out Cross-Validation

- Start with N data samples.
    1. Choose one sample and remove it.
    2. Design classifier based on remaining N-1 samples
    3. Test on single removed sample.

  Repeat 1-3 for all N different choices of single-sample test sets. Average the error rate of all splits.

- Leave-one-out is <u>expensive</u> for any technique that requires iterative training (neural networks, mixture model fitting) since you must learn *N* different models.

- However, leave-one-out is <u>cheap</u> for memory-based techniques like k-NN, kernel methods etc.

- All classifiers have very similar training sets – similar to total training set. (Bias of error estimate $\hat{\mathcal{E}}(D_{train}, D_{test})]$ is low)

K-Fold Cross Validation

- Divide data into *k* disjoint sets of equal size *N/k*.
- Train the classifier *k* times, each time with a different set held out to estimate the performance.
- Estimate error rate as mean of the rate measured for each of the *k* splits. (Reduction in amount of training data biases the error rate upward. Variance is lower than in leave-one-out.)
- Cross-validation (leave-one-out, and k-fold) are useful for picking hyper-parameters and architectures
  - Number of components in a Gaussian mixture model.
  - Radius of kernel in kernel density estimates.
  - Number of neighbors in k-NN.
  - Number of layers and hidden neurons in a neural network.

GEORGETOWN UNIVERSITY

Resampling

- Cross-validation attempts to approximate averages over training and test sets. It is a means of ameliorating the variance of estimates due to limited data set size.
  It is one example of a <u>resampling</u> technique.

- Bootstrap – another resampling technique, allows even more data sets to be averaged.

  Bootstrap data set
  – Start with our data set of N samples D

  – Randomly select N samples, <u>with replacement</u> ➔ select a sample at random, copy it into the new dataset, and return the sample to the original bucket of data. (On average, .632xN distinct samples.)
  – Generates independent datasets drawn from the empirical density

$$\hat{p}(x) = \frac{1}{N}\sum_{i=1}^{N}\delta(x - x_i) \qquad (Dirac\ delta)$$

Bootstrap Error Estimate

Generate *B* bootstrap datasets D^b, $b=1, \ldots, B$. Train a classifier to each of the bootstrap datasets – denote these classifiers

$$\hat{f}^{\,b}(x)$$

Evaluate each of the bootstrap classifier on the original complete data set – less the samples present in that particular bootstrap training set

$$\hat{\mathcal{E}}_{boot} = \frac{1}{B}\frac{1}{N'}\sum_{b}\sum_{i=1}^{N'} \hat{\mathcal{E}}(D^b, D-(D\wedge D^b))$$

Since have, on average, only 0.632x*N* distinct samples, error rate has bias similar to 2-fold cross-validation. The ".632 estimator" is designed to alleviate this bias

$$\hat{\mathcal{E}}_{.632} = 0.632\,\hat{\mathcal{E}}_{boot} + 0.368\,E(D_{train}, D_{train})$$

Bootstrap Aggregates

- Committee machines, or aggregates, use several component classifiers and vote them for a final decision. If the errors between the individual component classifiers are uncorrelated (and this can take some work), then they may be expected to cancel out during the voting.

- Bootstrap Aggregation – or *bagging* – constructs the component classifiers by training on bootstrap replicates.

GEORGETOWN UNIVERSITY