Bayesian learning:

Machine learning comes from Bayesian decision theory in statistics. There we want to minimize the expected value of the loss function.

Let y be the true label and y' be the predicted value. Popular loss functions are

Hinge: max(0, 1 -y(y'))
Linear regression: (y - y')^2
Logistic: log(1 + e^(-y(y')))
0/1 loss: max(0, sign(-y(y'))

Each loss function gives rise to a different empirical risk. The empirical risk is just the sum of the loss function across all training data points. For example empirical risk for linear regression (or least squares) is

Empirical risk for least squares = $\Sigma_i (y_i - y_i')$ where i is across all data points.

If we let yi' = wTxi + w0 then each of the above loss terms becomes a linear classifier loss.

The above is useful to know. We study it to better understand classification from a theoretical perspective. For example we can ask: how well does the empirical estimate the expected risk?

Moving on to classification we actually use Bayes rule to make decisions (review Bayes rule).

Bayesian learning:

We use Bayes rule. This means we classify a datapoint x into one of two classes C1 and C2 by comparing P(C1|x) and P(C2|x). But what is P(C1|x)? Bayes rule says it is

P(C1|x) = P(x|C1) P(C1) / P(x)
(posterior) (likelihood) (prior) (normalization - not important)

In multivariate Gaussian learning we assumed that the likelihood was Gaussian and prior was uniform and that gave us a way to classify with Bayes rule. In Bayesian learning we have to assume some form of the likelihood and prior. People use Gamma distributions for priors as well.

When we implemented this we ran into singularity problems because we have to invert the covariance matrix. But if we assume that the covariance matrices are the same for classes C1 and C2 then our multivariate Gaussian simplifies to nearest means.

Nearest means and Naive Bayes

Nearest means is a linear classifier, which means it is a hyperplane in some space. Naive Bayes is also linear but does not completely ignore the variances. It actually only assumes that covariances are 0. Both are very fast and easy to implement.

LInear separators (hyperplanes)

Remember that a linear classifier is just a hyperplane

SVM

A linear classifier that maximizes the margin. The margin is the minimum distance of points to the plane. Points which are correctly classified have a positive margin and incorrect ones have negative.

The SVM objective is $\Sigma_i max(0, 1-(y_i y_i')) = \Sigma_i max(0, 1-(y_i(w^T x_i + w_0))$ where i is across all data points. This is, however, unregularized. The regularized version would be

$\Sigma_i max(0, 1-(y_i(w^T x_i + w_0) \ + \ \lambda\|w\|^2$

Logistic regression and least squares

We have listed these two loss functions above.

Kernel methods

What is kernel value K(x,y) of vectors x and y? What does it mean?

K(x,y) is the dot product of x and y in some feature space $\Phi(x)$. Specifically $K(x, y) = \Phi(x)\Phi^T(y)$

For example suppose x=(x1,x2) and $\Phi(x) \ = \ (1,\ x1,\ x2,\ x1^2\ ,\ x2^2,\ x1x2)$

So K(x,y) in this case is the dot product of x and y as given by the above Phi function.

We do non-linear classification by linearizing the data by creating new features. But for most linear classifiers we only need the dot product. This means we can use kernels to do linear classifiers in new linearized spaces without having to explicitly create the new feature space.

Neural networks

Creating new feature representation with hidden layers and solved with a gradient approach. The objective is smooth but non-convex (which means many local minima). And so we have to run the classifier several times and pick the one with the best objective.

For images the convolution operator usually improves the accuracy.

Decision stump: hyperplane parallel to a given axes (dimension)

Decision tree:

A collection of decision stumps arranged in a tree structure. We can write the empirical hinge risk for decision trees as

$\Sigma_i max(0, 1-(y_i y_i'))$

The regularized decision tree risk would be

$\Sigma_i max(0, 1-(y_i y_i')) + (number\ of\ nodes\ in\ tree)$

Random forest

Bagged decision trees: A collection of decision trees. Each decision tree is built on a bootstrapped sample of the training dataset. Bootstrapping is a method to create a new random sample from a given dataset. If the data has n rows we randomly sample with replacement n rows to create the bootstrapped dataset. Random sample by default usually means without replacement.

To classify a test datapoint x we predict its label with all the decision trees in the ensemble and pick the majority vote.

Random forest: A random forest will also randomly sample a fraction of the columns. Why? We do this to reduce covariance between the classifiers in the ensemble which in turn reduces the variance of the ensemble. If the ensemble has classes ci (i=0 to 99) then the variance of the ensemble is

Variance(ensemble) = Σ_i Variance(ci) + Σ_{ij} Covariance(ci,cj)

Boost decision trees
Two differences from random forest:

1. The dataset is not bootstrapped (except for first round). In subsequent rounds we select datapoints according to probabilities given by error in the previous round
2. In the final classification random forest picks majority. This is the same as saying every classifier has the same weight. In boosting we weight the classifier according to its error on training data.

Bias and variance

Bias of a classifier: E(prediction - truelabel)

Variance of a classifier: Var(prediction - truelabel)

We can measure bias by determining the average prediction on many different bootstraps and measuring the difference between the average and the truelabel. We do the same thing for the variance. We measure the variance of the average predicted value from the true label.

So now we have low and high bias and low and high variance.

Class1: low bias and low variance - indicates stability. We generally desire low bias and variance, but does it mean it will high low test error also? No. There are counterexamples.

Class 2: high bias and low variance

Class 3: low bias and high variance - unstable. Classifier may be unstable or the data is unclean

Class 4: high bias and high variance

Remember that both bias and variance are on training data. Ultimately we are more interested in generalization and test error.

Non-linear methods

1. Decision tree, random forest, boosted trees
2. Kernel methods
3. Neural networks
4. K-nearest neighbor (K-NN)

Linear methods

1. Nearest means
2. Naive Bayes
3. Least squares
4. Logistic regression
5. Support vector machine

Regression:

Key thing to remember is that regression comes from modifying classification algorithms. We have linear regression (least squares), support vector regression, and ridge regression (which is regularized linear regression).

Unsupervised learning:

K-means is the most popular and fastest clustering program. Knowing and working with this is sufficient. There is also graph-based clustering.

Big data:

To do data science (or machine learning) or big datasets we have several different solutions.

1. Parallel computing
    a. multicore programming with OpenMP
    b. GPU programming with CUDA
    c. Hadoop is a popular tool for distributed computing. Made by Google it implements the MapReduce algorithm
2. Stochastic gradient descent also available in Python scikit learn - usually for supervised linear classifiers, main idea is to use a subset of the training data for the gradient calculation
3. Mini-batch method - usually for clustering, main idea is to select subset of data for initial clustering

----

Feature selection and dimensionality reduction

For feature selection:

1. Pearson correlation (for regression and classification)
2. F-score (for classification only)
3. Chi-square (for categorical data and classification only)
4. Signal to noise ratio (for regression and classification)
5. We can use the w vector also for classification

Dimensionality reduction

1. PCA (unsupervised, very popular, and can be used for data visualization)
2. LDA (supervised but has singularity problems)

Stacking:

A powerful method to make new features. The method is to take the predictions of different classifiers and make a new dataset out of them. This method is also used to win Kaggle contests.

Regularization

A method to control overfitting. Also smoothens the objective and makes the problem easier to solve. For linear classifiers it is the norm of w (or length of w). For a decision tree it is number of nodes or height of the tree. The main idea is to keep the classifier simple so we don't overfit.

Representation learning

1. Learning new features with random hyperplanes
2. Deep learning methods:
    a. Deep networks and advance methods for training in the Keras (methods like dropout)
    b. Convolution and pooling for image data
    c. Applying specific convolution kernels vs learning them
    d. Images before and after convolution (visual and machine learning)
3. Representation for text data: bag of words format
4. Representation of protein data: empirical kernel map