About this class

k-Nearest-Neighbors

Bagging

Boosting

Nearest-Neighbor Methods

Store all training examples

Given a new test example, find the k that are closest to it in feature space (distance: Euclidean/Mahalanobis?)

Return majority classification among those k points

Curse of dimensionality – irrelevant features can dominate classification

Training is trivial, but efficiency of finding k nearest points?

Use intelligent data structures like kd-trees. Worst case behavior is bad for nearest-neighor search ($O(l)$) but much better on average (although distribution dependent). Initial fixed cost to building the tree

Big caveat: search cost seems to scale badly with the number of dimensions in the feature space!

Very simple but efective algorithm!

Bagging

Bootstrap Aggregating (Breiman, 1994)

Key idea: Build t independent replicates of the training set L by sampling *with replacement*

Train classifier on each of them

Predict the majority of all these classifiers

In the case of a regression problem, predict the average

For decision trees: significant improvement in accuracy, but a loss in comprehensibility

Works well for unstable algorithms. Intuition: unstable algorithms can change their predictions substantially based on small changes in the training set, which is essentially what each replicate training set is doing. When you average over multiple sets of training data, you're getting a more stable predictor.

Let f_A be the aggregated predictor. Then $f_A(x)$ attempts to approximate $E_L f(x)$

How different are the training sets? The probability that a given example is not in a given subset is $(1 - 1/n)^n \rightarrow 1/e = 0.368$ as $n \rightarrow \infty$.

Empirically, 50 replicates give all the benefit of bagging, often a 20% to 40% reduction in error rate.

Each trained model has higher initial variance since it is trained on a smaller training set.

Bagging stable classifiers can somewhat degrade performance

What would happen with linear regression or Naive Bayes?

Boosting

Basic question: Can we take an algorithm that learns *weak* hypotheses that perform somewhat better than chance and make it into a strong learner?

Answer: yes (Freund and Schapire, various papers)

We'll again build an ensemble classifier, but, unlike bagging, members of the ensemble will have different weights

Bagging reduces variance (albeit slower than $1/n$ because the training set is replicated), but boosting reduces bias by making the hypothesis space more flexible

AdaBoost Algorithm

Given:

Training examples $(x_i, y_i), \ldots, (x_m, y_m)$

A weak learning algorithm, guaranteed to make error $\epsilon \leq \frac{1}{2} - \gamma$

Maintain a weight distribution D over training examples. Initialize $D(i) = 1/m$.

Now repeat for a number of rounds T:

1. Train weak learner using distribution D. This gives a weak hypothesis $h_t : X \rightarrow \{\pm 1\}$. h_t has error

$$\epsilon_t = \Pr_{i \sim D_t} [h(x_i) \neq y_i]$$

2. $\alpha_t \leftarrow \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

3. Update:

$$D(i) \leftarrow \frac{D(i)}{Z} e^{-\alpha_t} \text{ if } h_t(x_i) = y_i$$

$$D(i) \leftarrow \frac{D(i)}{Z} e^{\alpha_t} \text{ if } h_t(x_i) \neq y_i$$

where Z is a normalization factor

Return final hypothesis:

$$H(x) = \text{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Caveat: We need a weak learner that can learn even on hard weight distributions!

Training Error

First let's bound the weight distribution:

$$D_{T+1}(i) = \frac{D_T(i)}{Z_T} \exp(-\alpha_T h_T(x_i) y_i)$$

$$D_{T+1}(i) = \frac{1}{n} \prod_{t=1}^{T} \frac{1}{Z_t} \exp(-\alpha_T h_T(x_i) y_i)$$

$$= \frac{1}{n} \frac{\exp \sum_{t=1}^{T}(-\alpha_t h_t(x_i) y_i)}{\prod_{t=1}^{T} Z_t}$$

Now for the training error:

$$\epsilon = \frac{1}{n} \sum_i I[y_i \sum_t \alpha_t h_t(x_i) \leq 0]$$

$$\leq \frac{1}{n} \sum_i \exp(-y_i \sum_t \alpha_t h_t(x_i))$$

(because $e^{-z} \geq 1$ if $z \leq 0$)

Substituting from above,

$$\epsilon \leq \sum_i D_{T+1}(i) \prod_t Z_t$$

$$= \prod_t Z_t$$

Finally,

$$Z_t = \sum_{i:h_t(x_i)=y_i} D_t(i) e^{-\alpha_t} + \sum_{i:h_t(x_i)\neq y_i} D_t(i) e^{\alpha_t}$$

$$= e^{-\alpha_t}(1 - \epsilon_t) + e^{\alpha_t}(\epsilon_t)$$

$$= 2\sqrt{\epsilon_t(1 - \epsilon_t)}$$

$$= \sqrt{1 - 4\gamma_t^2}$$

Then

$$\epsilon \leq \exp(-2 \sum_t \gamma_t^2)$$

So, proof that we can boost weak learners that meet the requisite conditions into strong learners!

Generalization and Empirical Properties

Fairly robust to overfitting. In fact, often test error keeps decreasing even after training error has converged

Works well with a range of hypotheses, including decision trees, stumps, and Naive Bayes

Relation to SVMs? Can think of boosting as maximizing a different margin, and of using multiple weak learners to go to a high dimensional space, instead of using a kernel like SVMs do. Computationally, boosting is easier (LP as opposed to QP)