Machine Learning
Lecture 12

Lecture by Dr. Shai Fine
Typeset by Steven Karas

2018-01-14
Last edited 21:14:28 2018-01-14

Disclaimer These lecture notes are based on the lecture for the course Machine Learning, taught by Dr. Shai Fine at IDC Herzliyah in the fall semester of 2017/2018. Sections may be based on the lecture slides written by Dr. Shai Fine.

Homework 5 HW 5 was published today. It involves running the entire pipeline we built up until now on new unlabeled data.

The distribution of each class in the feature space has not changed, but the distribution of classes has.

Agenda

- Ensemble Methods

1 Ensemble Methods

In a nutshell, uses multiple models to gather an ensemble of hypotheses and combine their predictions.

No free lunch theorem proves that accuracy in one area must come at the cost of another area.

Disagreements between models are important, and what provides us with the value. If the error regions of the models overlap, they do not provide additional value.

An ensemble method is comprised of two components:

1. Generation of component classifiers
2. Method of combining predictions

Bias and Variance Generally speaking, *bias* is a measure of the underfitting of a given model. Similarly, *variance* is a measure of the overfitting of a given model.

For a given model, the more model parameters there are, and the more they are trained to the data, the more variance there is. Too few parameters, however, introduces bias.

Ensemble methods sidestep this and can reduce both variance and bias.

1.1 Classifier Fusion

Trains the component classifiers on all the feature space. Merges the predictions of the weaker classifiers into a single expert prediction.

1.2 Classifier Selection

Each classifier is trained on a local area of the feature space. One or more local experts may be used.

1.3 Combination Rules

1.3.1 Weighted Voting

A vote is a linear combination over weights $w_j \geq 0$ for classifier d_j ($\sum_j w_j = 1$):

$$y = f(x) = \sum_j w_j d_j(x)$$

Example: Majority Vote Given 5 fully independent classifiers with 70% accuracy, the overall majority vote accuracy is $10(.7^3)(.3^2) + 5(.7^4) + (.7^5) = 0.837$.

Note that this is the binomial distribution:

$$\Pr(k; p, n) = \binom{n}{k} p^k (1-p)^{n-k}$$

1.3.2 Partitioning

Divide the feature space into regions that models are trained for. This allows them to specialize, learning exactly the cases for which they are picked.

1.3.3 Routing

When predicting a new instance, it is routed to the relevant expert models.

Mixture of Experts Presented by Jacobs, et al[3]

$$y = f(x) = \sum_j w_j(x) d_j(x)$$

Where the gating function $w_j(x)$ gives the weight of the prediction $d_j(x)$ (where experts get more weight for regions they are experts in).

1.4 Bagging

First presented by Breiman, et al in 1996[1].

Creates ensembles by bootstrap aggregation: repeatedly resampling the training data, training multiple models on the bags, and averages the decisions.

This reduces variance, without touching bias. It tends naturally towards being embarrassingly parallelizeable, which speeds up training.

1.4.1 Random Forests

Presented by Breiman, et al in 2001[2].

Trains decision trees such that each tree depends on different subsets of features. This effectively performs bagging on the features.

This approach is comparable in results to deep learning, but is much simpler. It is relatively robust to outliers as well (if the noise is uniform, then the noise errors cancels out). Because the training is independent, overfit is avoided.

1.5 Boosting

Strong learners are models with arbitrary accuracy. Effectively the objective of Machine Learning.

Weak learners on the other hand are only slightly more accurate than random guessing.

Key Ideas Instead of sampling, simply reweigh examples. In each iteration, train a new weak learner. After training, reweigh the training set for the instances which were errors.

Prediction for the ensemble is the weighted average of their predictions (weighted by the training weights).

1.5.1 AdaBoost

Given a labeled data set $(x_1, y_1), \dots, (x_m, y_m); x_i \in X, y_i \in \{-1, +1\}$.

Initialize $D_1(i) = \frac{1}{m}$.

For each training round $t \in \{1, \dots, T\}$, train a weak learner using weights D_t to get a hypothesis h_t. Calculate the error ϵ_t with respect to weights D_t:

$$\epsilon_t = \sum_{i=1}^{m} D_t(i) I[y_i \neq h_t(x_i)]$$

If the error is 0 or over one half, then break out early. Choose some $\alpha_t \in \mathbb{R}$, typically:

$$\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}$$

Then update the weights using a normalization factor Z_t such that $\sum_i D_t(i) = 1$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

After all these rounds, the final prediction is:

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

Example: Decision Stump Use AdaBoost to train a classifier for some toy data in 2 dimensions using decision stumps (single level decision trees).

This example can be followed along on slides 24-33 of the lecture slides.

Upper bound on Training Error The upper bound on the training error of H:

$$\frac{1}{m} \sum_i [H(x_i) \neq y_i] \leq \prod_{t=1}^{T} Z_t$$

Proof is by unraveling the update rule:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \\ &= \frac{\exp(-\sum_t \alpha_t y_i h_t(x_i))}{m \prod_t Z_t} \end{aligned}$$

Recall that $f(x_i) = \sum_t \alpha_t h_t(x_i)$, then:

$$\left(\prod_t Z_t\right) D_{T+1}(i) = \frac{1}{m} \exp(-y_i f(x_i))$$

If $H(x_i) \neq y_i$, then $y_i f(x_i) \leq 0$, which implies that $\exp(-y_i f(x_i)) > 1$. Therefore:

$$\begin{aligned} \frac{1}{m} \sum_i [H(x_i) \neq y_i] &\leq \frac{1}{m} \sum_i \exp(-y_i f(x_i)) \\ &= \sum_i \left(\prod_t Z_t\right) D_{T+1}(i) \\ &= \prod_{t=1}^{T} Z_t \end{aligned}$$

Loss function It's important to note that instead of minimizing the training error, AdaBoost minimizes the exponential loss:

$$Z_t = \sum_i D_T(i) \exp(-\alpha_t y_i h_t(x_i))$$

Fro this, we find the optimal α_t by setting the derivative to 0:

$$\begin{aligned} \frac{dZ}{d\alpha} - \sum_{i=1}^{m} D(i) y_i h(x_i) e^{-y_i \alpha_i h(x_i)} &= 0 \\ -\sum_{i:y_i=h(x_i)} D(i)e^{-\alpha} + \sum_{i:y_i \neq h(x_i)} D(i)e^{\alpha} &= 0 \\ -e^{\alpha}(1-\epsilon) + e^{\alpha}\epsilon &= 0 \\ \alpha_t = \frac{1}{2}\log\frac{(1-\epsilon_t)}{\epsilon_t} \end{aligned}$$

Proof of boosting

$$\begin{aligned} Z_t &= \sum_{i=1}^{m} D_t(i) e^{-y_i \alpha_i h_t(x_i)} \\ &= \sum_{i:y_i=h_t(x_i)} D_t(i)e^{-\alpha_t} + \sum_{i:y_i \neq h_t(x_i)} D_t(i)e^{\alpha_t} \\ &= (1-\epsilon_t)e^{-\alpha_t} + \epsilon_t e^{\alpha_t} \\ &= 2\sqrt{\epsilon_t(1-\epsilon_t)} \end{aligned}$$

Which shows we minimize Z_t by selecting the h_t with minimal weighted error ϵ_t.
Denote the advantage over random classification as $\gamma_t = \frac{1}{2} - \epsilon_t$.

$$\prod_t Z_t = \prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] = \prod_t \sqrt{1-4\gamma_t^2} \leq \exp\left(-2\sum_t \gamma_t^2\right)$$

If each base classifier is slightly better than random, then for all t, $\gamma_t \geq \gamma > 0$, then the training error drops exponentially fast:

$$\frac{1}{m}\sum_i [H(x_i) \neq y_i] = \prod_{t=1}^{T} Z_t \leq \exp\left(-2\sum_t \gamma_t^2\right) \leq \exp(-2\gamma^2 T)$$

Effect on bias and variance Empirical experiments have shown that AdaBoost tends to reduce bias in the early iterations, and variance in the later iterations.

References

[1] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[2] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[3] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.