MODEL ENSEMBLES

"Getting something out of nothing"

CIML Ch. 13, PRML Ch. 14

0) Averaging (Voting)

If you have access to multiple high-quality models that disagree with one another, model averaging is a good idea.

(We've seen this kind of, with voted perceptron)

How do we get access to multiple models?

1) BOOTSTRAP AGGREGATION ("Bagging")

ML: Training Set → Model

Bootstrapping: Training Set → Set of (Weighted) training sets

Voting over bootstrap samples of training set: bootstrap aggregation.

Important observation: weights of samples change, but do not depend on the quality of the other models.

2) LET'S WIN THE NETFLIX PRIZE

2a) What are the weights used by model averaging?

How can we do better?

- $\|\vec{\hat{y}}_i - \vec{y}\|^2 = \varepsilon_i$ (Sq. error)
- $y^* = \sum_i \varphi_i \vec{\hat{y}}_i$ $\quad \vec{\hat{y}} = (\hat{y}_0\ \hat{y}_1\ \hat{y}_2 \dots)$
- $\underset{\vec{\varphi}}{\operatorname{argmin}} \|y^* - y\| = \langle y^* - y, y^* - y \rangle$

$$\nabla_{\vec{\varphi}} \|y^* - y\| = 0$$

$$\vec{\hat{y}}^T(\vec{\hat{y}}\vec{\varphi} - y) = 0$$

$$\vec{\hat{y}}^T\vec{\hat{y}}\vec{\varphi} = \hat{y}^T y$$

What now! This seems impossible...?

$$\|\hat{y}_i - y\|^2 = \varepsilon_i \qquad \langle \hat{y}_i, \hat{y}_i \rangle - 2\langle \hat{y}_i, y \rangle + \langle y, y \rangle = \varepsilon_i$$

Weights change depending on model quality (and model covariance)

2b) Why are we focusing on linear weights?

The fundamental problem we seek to solve is to predict y from $\hat{y}_i$. What can we do about this...?

It's ML all the way down. (And it works!)

3) BOOSTING

If we create our models sequentially, then we have the opportunity to use the results of one model to change the way we train the next model.

How is that?

Basic intuition: start with a model, but monitor its mistakes. For every correctly predicted sample, decrease its weights. For every incorrectly predicted sample, increase its weights. Build a new model, repeat N times, and final model is a weighted vote of models based on their overall quality.

Algorithm 32 AdaBoost($\mathcal{W}$, $\mathcal{D}$, K)

1: $\boldsymbol{d}^{(0)} \leftarrow \langle \frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N} \rangle$ // Initialize uniform importance to each example
2: **for** $k = 1 \ldots K$ **do**
3: $f^{(k)} \leftarrow \mathcal{W}(\mathcal{D}, \boldsymbol{d}^{(k-1)})$ // Train kth classifier on weighted data
4: $\hat{y}_n \leftarrow f^{(k)}(\boldsymbol{x}_n), \forall n$ // Make predictions on training data
5: $\hat{\epsilon}^{(k)} \leftarrow \sum_n d_n^{(k-1)} [y_n \neq \hat{y}_n]$ // Compute weighted training error
6: $\alpha^{(k)} \leftarrow \frac{1}{2} \log \left(\frac{1-\hat{\epsilon}^{(k)}}{\hat{\epsilon}^{(k)}} \right)$ // Compute "adaptive" parameter
7: $d_n^{(k)} \leftarrow \frac{1}{Z} d_n^{(k-1)} \exp[-\alpha^{(k)} y_n \hat{y}_n], \forall n$ // Re-weight examples and normalize
8: **end for**
9: **return** $f(\hat{\boldsymbol{x}}) = \text{sgn}\left[\sum_k \alpha^{(k)} f^{(k)}(\hat{\boldsymbol{x}})\right]$ // Return (weighted) voted classifier