CS229 boosting notes

James Chuang

May 2, 2017

Contents

My notes on John Duchi's CS229 supplemental notes on boosting.

1. Boosting

- so far, have seen how to solve classification (and other) problems when we have a data representation already chosen
- in **boosting**, feature representations are automatically chosen
  - the rough idea:
    - take a *weak learning* algorithm (a classifier that is slightly better than random)
    - transform it into a *strong* classifier, which does much better than random
  - intuition:
    - consider a digit recognition problem distinguishing 0 from 1 from images
      - a weak learner might take the middle pixel of the image:
        - if it is colored, call the image a 1
      - if it is blank, call the image a 0
    - boosting procedures take a collection of weak classifiers, and reweight their contributions to form a classifier with much better accuracy than any individual classifier
- problem formulation:
  - one interpretation of boosting is as **a coordinate descent method in an infinite dimensional space**
  - assume:
    - raw input samples $x \in \mathbb{R}^n$ with labels $y \in \{-1, 1\}$
    - an infinite collection of *feature* functions $\phi_j : \mathbb{R}^n \to \{-1, 1\}$
    - an infinite vector $\theta = \begin{bmatrix} \theta_1 & \theta_2 & \cdots \end{bmatrix}^T$ with a finite number of non-zero entries
  - hypothesis:

$$h_\theta(x) = \text{sign}\left(\sum_{j=1}^{\infty} \theta_j \phi_j(x)\right)$$

- define:

$$\theta^T \phi(x) = \sum_{i=1}^{\infty} \theta_j \phi_j(x)$$

- in boosting, the features ϕ_j are called **weak hypotheses**
- given a training set $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(m)}, y^{(m)}\right)$:
  - we call a vector $p = \left(p^{(1)}, \ldots, p^{(m)}\right)$ a distribution on the examples if $p^{(i)} \geq 0 \; \forall \; i$ and

$$\sum_{i=1}^{m} p^{(i)} = 1$$

- we say that there is a *weak learner with margin* $\gamma > 0$ if for any distribution p on the m training examples there exists one weak hypothesis ϕ_j such that

$$\sum_{i=1}^{m} p^{(i)} \mathbf{1}\left\{y^{(i)} \neq \phi_j\left(x^{(i)}\right)\right\} \leq \frac{1}{2} - \gamma$$

- i.e., we assume that there is *some* classifier that does slightly better than random guessing on the dataset
    - the existence of a weak learning algorithm is an assumption
        - however, we can transform any weak learning algorithm into one with perfect accuracy
- in more generality, we assume we have access to a **weak learner**- an algorithm that takes as input a distribution (weights) p on the training examples and returns a classifier doing slightly better than random
    - given access to a weak learning algorithm, boosting can return a classifier with perfect accuracy on the training data (we ignore generalization for now)

1.1 the boosting algorithm

- roughly, boosting begins by assigning each training example in the dataset equal weight
- it then receives a weak hypothesis that does well according to the current weights on training examples, and incorporates it into its current classification model
- it then reweights the trainng examples so that examples on which it makes mistakes receive higher weight, while examples with no mistakes receive lower weight
    - this forces the weak learning algorithm to focus on a classifier doing well on examples poorly classified by the weak hypothesis
- repeated reweighting of the training data coupled with a weak learner doing well on examples for which the classifier currently does poorly yields classifiers with good performance
- specifically, the boosting algorithm performs **coordinate descent** on the exponential loss for classification problems
    - the objective:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \exp\left(-y^{(i)} \theta^T \phi\left(x^{(i)}\right)\right)$$

- coordinate descent algorithm:
    1. choose a coordinate $j \in \mathbb{N}$
    2. update θ_j: $\theta_j \leftarrow \arg\min_{\theta_j} J(\theta)$
        - leave θ_k unchanged for all $k \neq j$
    - iterate until convergence
- derivation of the coordinate update for coordinate k:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \exp\left(-y^{(i)} \theta^T \phi\left(x^{(i)}\right)\right) \quad \text{the objective function}$$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \exp\left(-y^{(i)} \sum_{j \neq k} \theta_j \phi\left(x^{(i)}\right)\right) \exp\left(-y^{(i)} \theta_k \phi\left(x^{(i)}\right)\right) \quad \text{property of exp}$$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} w^{(i)} \exp\left(-y^{(i)} \theta_k \phi\left(x^{(i)}\right)\right) \quad \text{define } w^{(i)} = \exp\left(-y^{(i)} \sum_{j \neq k} \theta_j \phi\left(x^{(i)}\right)\right)$$

to optimize coordinate k :

$$\alpha^* = \arg\min_{\alpha} \sum_{i=1}^{m} w^{(i)} \exp\left(-y^{(i)} \phi_k\left(x^{(i)}\right) \alpha\right) \quad \text{,where } \alpha = \theta^k$$

$$\alpha^* = \arg\min_{\alpha} \sum$$

- define the **weights**:

$$w^{(i)} = \exp\left(-y^{(i)} \sum_{j \neq k} \theta_j \phi_j\left(x^{(i)}\right)\right)$$

- optimizing coordinate k corresponds to minimizing

$$\sum_{i=1}^{m} w^{(i)} \exp\left(-y^{(i)} \phi_k\left(x^{(i)}\right) \alpha\right)$$

- w.r.t. $\alpha = \theta_k$
- define:

$$W^+ := \sum_{i:y^{(i)}\phi_k\left(x^{(i)}\right)=1} w^{(i)} \qquad W^- := \sum_{i:y^{(i)}\phi_k\left(x^{(i)}\right)=-1} w^{(i)}$$

- these are the sums of the weights of examples that ϕ_k classifies correctly and incorrectly, respectively
- finding θ_k is the same as choosing

$$\alpha = \arg\min_{\alpha} \left\{W^+ e^{-\alpha} + W^- e^{\alpha}\right\}$$
$$\alpha = \frac{1}{2} \log \frac{W^+}{W^-}$$