CSE4502/5717: Big Data Analytics

Prof. Sanguthevar Rajasekaran
Notes by Zigeng Wang (TA)

April 11th, 2018

__Recap from last class__:

When we employ minibatches, we can replace matrix-vector multiplications with matrix-matrix multiplications. Consider the case where each level has *n* neurons and there is an edge from every neuron in any level to every neuron in the next level. We showed that if the minibatch size is *b,* then we can compute the activation values of every node in level *l* (given the activation values from level *l*-1), with q^2 matrix multiplications, each involving two $b \times b$ matrices. Here *q=n/b.* Thus,

$$Total\ Time = O(q^2 \cdot b^{2.373}) = O\left(\frac{n^2}{b^2} \cdot b^{2.373}\right) = O(n^2 \cdot b^{0.373})$$

while the total time for the naïve algorithm is $O(n^2 \cdot b)$.

1. To Improve Test Accuracy

1.1 Ensemble Learning

- Use multiple models, the final output will be based on the outputs from the different models.

__Example__:

Consider k different models for the same input.

- Let ϵ_i be the error from model i, $1 \le i \le k$.
- Let ϵ_i be generated from zero mean multivariable normal distributions.
- Let the variance for ϵ_i be

$$E[\epsilon_i] = v \quad for\ 1 \le i \le k$$

- Also, let

$$E\left[\epsilon_i \epsilon_j\right] = c \quad \forall i, j;\ i \neq j$$

One possible way of combining the outputs from the different models is to take an average of the k outputs.

In this case,

$$the\ average\ error = \frac{1}{k}\sum_{i=1}^{k} \epsilon_i$$

and

$$Expected\ squared\ error = E\left[\left(\frac{1}{k}\sum_{i=1}^{k}\epsilon_i\right)^2\right] = \frac{1}{k^2}\left[\sum_{i=1}^{k}E[\epsilon_i^2] + \sum_{i\neq j}E[\epsilon_i\epsilon_j]\right]$$

$$= \frac{1}{k^2}[k\cdot v + k(k-1)\cdot c] = \frac{1}{k}v + \frac{k-1}{k}c$$

- If the errors are perfectly correlated, and $c = v$, then the expected squared error $= v$.
- If the ϵ_is are perfectly uncorrelated with $c = 0$, then the expected squared error $= \frac{1}{k}v$.

Thus, if we can employ many possibly uncorrelated models, we can improve the accuracy. Also, note that the expected squared mean error will not exceed *v*. In other words, we cannot worsen the accuracy with the employment of multiple models.

<u>**Techniques for generating models**</u>:

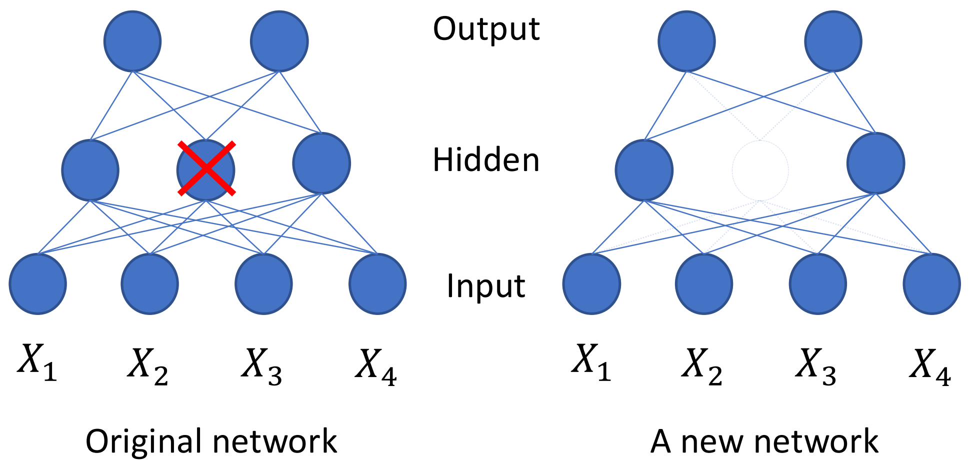
- **Bagging** (Bootstrap Aggregating)

  Here we use the same model but different datasets for training. Given an input data D, we generate new datasets $D_1, D_2, \dots, D_k$ by sampling from D with replacement such that

  $$|D| = |D_i| \quad for\ 1 \le i \le k$$

- **Drop-Out**

  Pick some number of nodes from the neural network randomly and this gives a new neural network (and a model). We can repeat this process *k* times to get *k* different models (for a suitable value of *k*).



1.2 Regularization Techniques

- Aim to decrease the test error possibly by increasing the training error.

These are normally used on *Point Estimators*.

- *Point Estimator:* A point estimator tries to get the best value for a parameter or a set of parameters.
- Let θ be a parameter of interest.
- Bias in estimating $\theta = E\left[\hat{\theta}_m\right] - \theta$, where the expectation is over the input data and θ is the true value.

Example:

- Let X be a Bernoulli variable with mean θ.
- Let $x_1, x_2, \ldots, x_m$ be samples from X.

One estimator for θ could be $\frac{1}{m}\sum_{i=1}^{m} x_i$.

The bias in this estimator can be calculated as,

$$Bias = E\left[\frac{1}{m}\sum_{i=1}^{m} x_i\right] - \theta = \frac{1}{m}\sum_{i=1}^{m} E(x_i) - \theta = \frac{1}{m}\sum_{i=1}^{m}\big(1\cdot\theta + 0\cdot(1-\theta)\big) - \theta = \frac{m\cdot\theta}{m} - \theta = 0$$

In this case,

$$Variance = \frac{1}{m}\theta(1-\theta)$$

Note:

- We want to keep both the bias and the variance small.
- Many of the regularization techniques try to decrease the variance by perhaps increasing the bias.

Ways of Regularization:

- Put some constraints on the model parameters and/or
- Add some additional constraints to the loss function.

Let $(\boldsymbol{X}, \boldsymbol{y})$ be the input. A typical loss function is $L(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y})$.

We can modify the loss function as

$$L'(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = L(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \lambda \cdot \Omega(\boldsymbol{\theta})$$

Where $\Omega(\boldsymbol{\theta})$ is the norm of $\boldsymbol{\theta}$.

In the case of linear regression,

$$L'(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = L(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) + \lambda \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}$$

$$\nabla_{\boldsymbol{w}} L'(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \nabla_{\boldsymbol{w}} L(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) + 2\lambda \boldsymbol{w}$$

$$\Rightarrow \boldsymbol{w}' = \boldsymbol{w} - \alpha(\nabla_{\boldsymbol{w}} L(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) + 2\lambda \boldsymbol{w}) = (1 - 2\alpha\lambda)\boldsymbol{w} - \alpha\nabla_{\boldsymbol{w}} L(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$$

where α is the learning rate. We note that the parameter $\boldsymbol{w}$ shrinks in every step with a factor of $(1 - 2\alpha\lambda)$.

2. Probability Approximately Correct (PAC) Learning

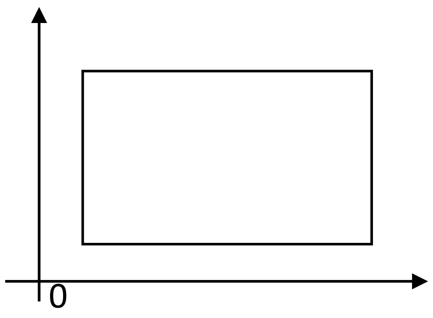
- Let C be a concept to be learnt.
- Let C' be the concept learnt.

We want to make sure that the $distance(C, C') \leq \epsilon$ with a probability $\geq 1 - \delta$.

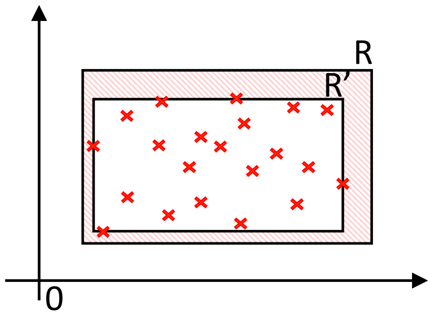
The learning time is a polynomial in $m, \frac{1}{\epsilon}$ and $\frac{1}{\delta}$, where m is the number of examples.

Example:

The concept is an axes parallel rectangle.



Input will be points (×) within the rectangle.



Output the least rectangle that encloses all the input points.

- The difference between the true concept and the output concept can be characterized with the area missed by the output. Let the fraction of area (shadowed) missed be ϵ.
- Let m be the number of examples.

$$Prob[Error\ fraction\ is > \epsilon] \leq (1 - \epsilon)^m$$

We want this probability to be $\leq \delta$, so that

$$(1 - \epsilon)^m \leq \delta$$

$$m \log(1 - \epsilon) \leq \log(\delta)$$

$$m \log\left(\frac{1}{(1 - \epsilon)}\right) \geq \log\left(\frac{1}{\delta}\right)$$

$$\Rightarrow m \geq \frac{\log\left(\frac{1}{\delta}\right)}{\log\left(\frac{1}{(1-\epsilon)}\right)}.$$

Note that our analysis enables us to determine the number of examples needed, for a given ϵ and δ.

Example:

A conjunctive normal form (CNF) Boolean formula is a conjunction of disjunctions. A k-CNF formula is a CNF formula with at most k literals/clause.

- $F = (\bar{X}_2 \vee X_3) \wedge (X_1 \vee X_4) \wedge (\bar{X}_2 \vee \bar{X}_5)$ is a 2-CNF formula.
- A monomial is a 1-CNF formula, like $X_1\bar{X}_2X_5\bar{X}_7$.

Fact:

- We can learn a monomial with positive examples.

Proof:

Let $E_i = (X_i^1, X_i^2, \ldots, X_i^n)$ be the i^{th} example ($1 \leq i \leq m$)

```
Start with $F = X_1\bar{X}_1X_2\bar{X}_2 \ldots X_n\bar{X}_n$
for $1 \leq i \leq m$ do
    for $1 \leq j \leq n$ do
        if $X_i^j = 1$ then
            Eliminate $\bar{X}_j$ from the $F$
          elseif $X_i^j = 0$ then
            Eliminate $X_j$ from the $F$
        end
    end
end
Output the resultant $F$
```

- Let F' be the true formula and F be the output formula.
- Let $D(v)$ be a distribution on all possible assignments.

$$dist(F, F') = \sum_{\substack{v \Rightarrow F \text{ and } v \not\Rightarrow F' \text{ or} \\ v \Rightarrow F' \text{ and } v \not\Rightarrow F \\ \Rightarrow \text{means "SATISFIED"}}} D(v)$$

So that,

$$Prob[all\ the\ m\ examples\ fall\ within\ a\ prob.of\ (1-\epsilon)] = (1-\epsilon)^m$$

If we have n variables, then there are no more than 2^{2n} concepts.

$$Prob[this\ happens\ for\ at\ least\ one\ concept] \leq 2^{2n} \cdot (1-\epsilon)^m$$

We want this to be $\leq \delta$, so that

$$2^{2n} \cdot (1-\epsilon)^m \leq \delta$$
$$2n + m\log(1-\epsilon) \leq \log(\delta)$$

$$-2n + m\log(\frac{1}{1-\epsilon}) \geq \log\left(\frac{1}{\delta}\right)$$

$$\Rightarrow m \geq \frac{2n + \log\left(\frac{1}{\delta}\right)}{\log\left(\frac{1}{1-\epsilon}\right)}.$$

3. Association Rules Mining

- Let D be a database (DB) of transactions.
- A transaction is a set of items.
- Let I be the set of all possible items with $|I| = d$.
- Any transaction $t \in DB$ is a subset of I.

We are interested in finding *Rules* of the form:

$$X \rightarrow Y \text{ where } X \neq \emptyset, Y \neq \emptyset, X \cap Y = \emptyset, X \subseteq I, Y \subseteq I$$

- An itemset is a subset of I.
- A k-itemset is an itemset with k items.

For any itemset X, let $\sigma(X)$ denote the number of transactions in D that contain X.

- $Support$ for the $Rule\ X \rightarrow Y$ is $\frac{\sigma(X \cup Y)}{n}$ where $n = |D|$.
- $Confidence$ for the $Rule\ X \rightarrow Y$ is $\frac{\sigma(X \cup Y)}{\sigma(X)}$.

Problem:

Given $minSupport, minConfidence$ and a database DB of transactions, identify all the $Rules\ X \rightarrow Y$ for which the $support$ is $\geq minSupport$ and the $confidence$ is $\geq minConfidence.$