Gradient Descent with Large Datasets

Learning with Large Datasets

問題設定：間違いやすい語をclassifyする，例 (to, too, two), (then, than)

"It's not who has the best algorithm that wins. It's who has the most data."

「勝つのはアルゴリズムでなくデータ量だ」

$m = 100,000,000$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \underbrace{\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}}$$

最初から大きなmでトレーニングするのではなく，たとえば1000個などのもっと小さい数のデータをランダムに選んで試してみることは重要．

[講義中のQuiz]

supervised learningでmが大(1億)．mが大でも全部でトレーニングした方が小数(たとえば千)でやるよりも良さそうだと言うためには？

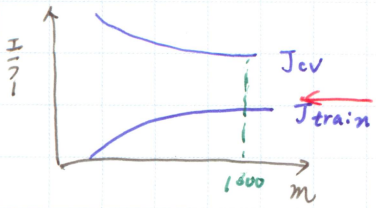
× ① 試す必要はない．大きいことはいいことだ．

× ② $J_{train}(\theta)$ をくりかえし回数を横軸にしてプロットしてみる

× ③ mを横軸にlearning curve ($J_{train}(\theta)$ と $J_{cv}(\theta)$) をプロットしてみて，mが小の時にhigh Bias (underfit) であることを確認する．mが小でunderfitしていることもそもそもダメ

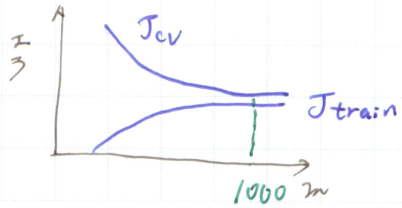
○ ④ mの範囲でlearning curveをプロットして，mが小さい時にhigh variance (overfit) であることを確認する．

m→大でoverfitが解消される



overfit

mを大にすることが役に立つ．



underfit

mを増やしても役に立たない。featureを増やすとかneural networkのhidden unitを増やす方がよい。

Stochastic Gradient Descent

大量のデータセットに対しては gradient descent の計算量が問題となる.

Linear regression with gradient descent

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$$

$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Repeat {

$$\theta_j := \theta_j - \alpha \boxed{\sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}} \quad \frac{\partial J(\theta)}{\partial \theta_j}$$

(for every $j = 0, \cdots, n$)

}

$m = 300.000.000$ の場合 (3億)
Batch gradient descent

Stochastic gradient descent

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$J_{train}(\theta) = \frac{1}{m} \sum_{i=1}^{m} cost(\theta, (x^{(i)}, y^{(i)}))$$

1. データセットをランダムにシャッフルする
2. Repeat {

これを1~10回程度ループさせる

for $i = 1, \cdots, m$ {

$\sum$ が無いことに注意

$$\theta_j := \theta_j - \alpha \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)} \quad \frac{\partial cost(\theta, (x^{(i)}, y^{(i)}))}{\partial \theta_j}$$

(for $j = 1, \cdots, n$)

}

}

[講義中のQuiz]
stochastic gradient descent (確率的～) にあてはまるものを2つ選べ

○ ① mがとても大きい時，普通のgradient descentよりも速い

~~×~~ ○ ② コスト関数 $J_{train} = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$ がbatch～では繰り返しの度に実行されるがstochastic～では必要ない

~~毎回繰り返すのは~~ $\theta_j := \theta_j - \alpha\frac{\partial J}{\partial \theta_j}$ ~~であってコストは毎回計算するだけでもよい~~
batch～では終了判定のためにJの計算が必要．元々$\frac{\partial}{\partial \theta_j}$の計算で$O(m)$の計算をしているので$J$の計算を有効することはない

× ③ stochastic～はlinear regressionにだけ適用できる

単に偏微分を一部のデータについてだけやりましょうという話なので他のアルゴリズムでも適用できる！

~~×~~ ○ ④ main loopの前にデータセットをランダムにシャッフルしておくべき

~~mainループの中で，最初にランダムにシャッフルすべき~~
シャッフルは1回だけでよいみたい

Week 10　　4

Mini_Batch Gradient Descent

Batch gradient descend : 毎回のくり返しで 全ての m データを使う

Stochastic ～ : 1個のデータ

Mini-Batch ～ : b個のデータ
↖ mini-batch size, 例 10～100

Mini_Batch Gradient Descent

1. $b = 10$, $m = 1000$ とする

2. Repeat {

ベクトル化して並列化できる

for $i = 1, \cdots, (m-b+1)$ {

$$\theta_j := \theta_j - \alpha \frac{1}{b} \sum_{k=i}^{i+b-1} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$$

(for every $j = 0, \cdots, n$)

}

}

[講義中のQuiz] m個のデータに対してmini-batch ～を適用する. b = mini-batch size.
これが batch ～と同じなのは?
$b = m$ のとき (答)

Stochastic Gradient Descent Convergence (確率的最急降下法の収束)

収束している事を確認するために すべき事

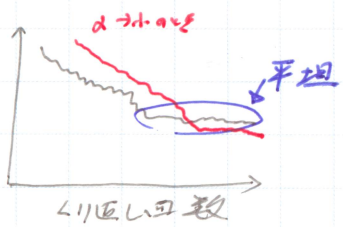
$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

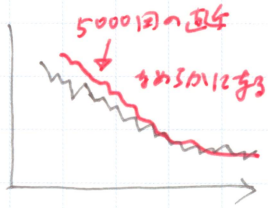
学習中に $(x^{(i)}, y^{(i)})$ を用いて θ を更新する前に $cost(\theta, (x^{(i)}, y^{(i)}))$ を計算する.

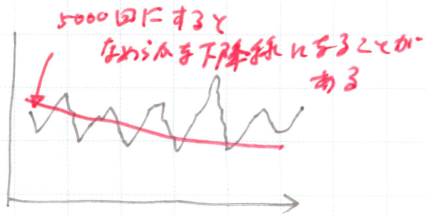
約1000回のくり返し毎に、直近の1000回のexampleで平均した $cost(\theta, (x^{(i)}, y^{(i)}))$ をプロットする

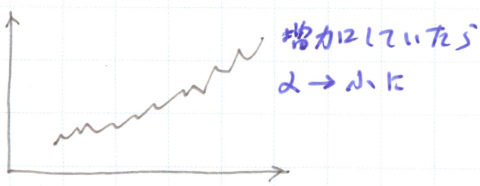
Checking for convergence (収束のチェック)

$cost(\theta, (x^{(i)}, y^{(i)}))$ の直近1000回の平均をプロットする









learning rate (学習率) α は典型的には定数だが、
θ を収束させるために少しずつ小さくすることもできる

例　$\alpha = \dfrac{\text{定数1}}{\text{くり返し回数} + \text{定数2}}$

パラメータが増えて面倒なので普通はやらない

[講義中のQuiz] stochastic ～ にあてはまるのは?

① とても小さいαを選んでも不利な点はなく，学習がspeed upする
× 逆．基本的にはslow downする

② αを減らして，十分まくり返し回数を走らせると，大きなαよりよいパラメータが得られる。 ○

③ (local minimum)の回りを振動しながらうろつくのではなく，収束させられるよう，αを少しずつ小さくしろ．
○ × 「やる人もいるけど，やるな」と言っていた

④ 1000回平均のcost(θ, (x^(i), y^(i)))をプロットして，costが減っていないようなら，αの値が悪いかも
○

Advanced Topics : Online Learning

連続な stream のデータが入力され続けていて、そこから学習させる。

(例1) shopping website : shipping service を提案 (price) → ユーザは $y=\begin{cases}1\\0\end{cases}$ を返す。

features x はユーザの属性，発送元，発送先，料金 を含む。
料金を最適化するために $p(y=1 \mid x;\theta)$ を学習する。

```
Repeat for ever {
    Get (x,y) corresponding user
    Update θ using (x, y)
        θ_j := θ_j − α(h_θ(x) − y)·x_j      (j=0, ⋯, n)
    }
}
```

(例2) Product search (商品を探す)
ユーザは "Android phone 1080p camera" を探す。
店には 100 個の phone がある。 10個の検索結果を返す。

feature x は { 名前が検索ワードと何語マッチするか, 説明が } など

$y=\begin{cases}1 & (\text{ユーザがクリックした}) \text{ click}\\ 0 & \text{otherwise through}\end{cases}$

$p(y=1 \mid x;\theta)$ を学習する。

= CTR 問題
Click Through Rate

(例3) たくさんの news から、そのユーザに提供する news を数個 選ぶ。

[講義中のQuiz] online learning algorithmの利点は？

○ ① userの嗜好の変化に適合できる。$p(y|x;\theta)$がいつも変化する

× ② 学習率αを選ぶ必要がない

○ ③ データ・ストリームから学習できる。各exampleを1度だけ使って、二度と使わない

× ④ 学習時に良いfeatureを選択する必要がない。

Advanced Topics : Map-reduce and data parallelism

Map-reduce

Batch gradient descent : $\theta_j := \theta_j - \alpha \frac{1}{400} \overset{m=400}{\overset{400}{\sum_{i=1}}} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$

Machine 1 : Use $(x^{(1)}, y^{(1)}), \dots, (x^{(100)}, y^{(100)})$ 最初の1/4

$$tmp_j^{(1)} = \sum_{i=1}^{100} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Machine 2 : Use $(x^{(101)}, y^{(101)}), \dots, (x^{(200)}, y^{(200)})$

$$tmp_j^{(2)} = \sum_{i=101}^{200} \sim$$

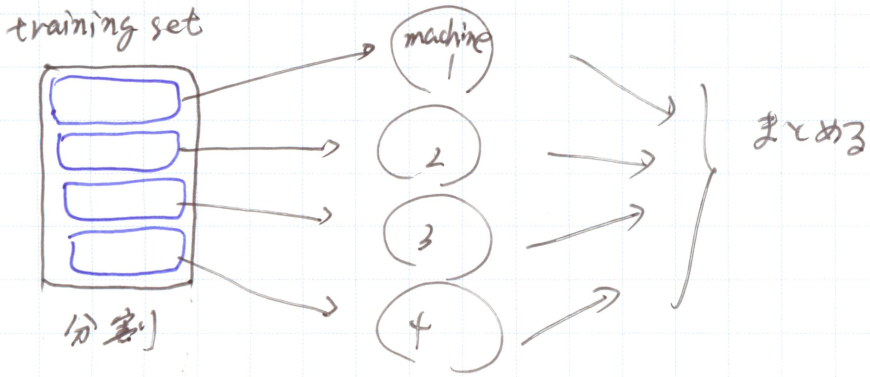
Machine 3 : Use $(x^{(201)}, y^{(201)}), \dots, (x^{(300)}, y^{(300)})$

$$tmp_j^{(3)} =$$

Machine 4 : Use $(x^{(301)}, y^{(301)}), \dots, (x^{(400)}, y^{(400)})$

$$tmp_j^{(4)} =$$

$$\theta_j := \theta_j - \alpha \frac{1}{400} (tmp_j^{(1)} + tmp_j^{(2)} + tmp_j^{(3)} + tmp_j^{(4)})$$



Map-reduce と training set に関する和

学習アルゴリズムの多くが
↓ これを使っている

computing sums of functions over training set

例　logistic regression

$$J_{train}(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) - (1-y^{(i)}) \log (1-h_\theta(x^{(i)}))$$

$$\frac{\partial}{\partial \theta_j} J_{train}(\theta) = \frac{1}{m}\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

datasetを分割して複数のマシンで実行する

1台のマシンでもcpuがmulti-coreならば分割する意味はある

ベクトル化したプログラムを記述しておくと、ライブラリが勝手に並列実行してくれることがある。

[講義中のQuiz] 10台でneural networkの学習を行う（map-reduceを使う）。くり返し毎に各マシンが行うのは

× ① forwardかbackwardのpropergation（データの$\frac{1}{10}$について）

≠0 ? ② forwardとbackwardのpropergation（$\frac{1}{10}$のデータについて）。$\frac{1}{10}$のデータの微分

× ③ $\frac{1}{10}$のデータのforward propergation。一旦集めて、1台でback propegationを行う

× ④ $\frac{1}{10}$のデータのback propergation。1台のマシンがforward propergationを行った後で。