Machine Learning - Week 10

赵燕

目录

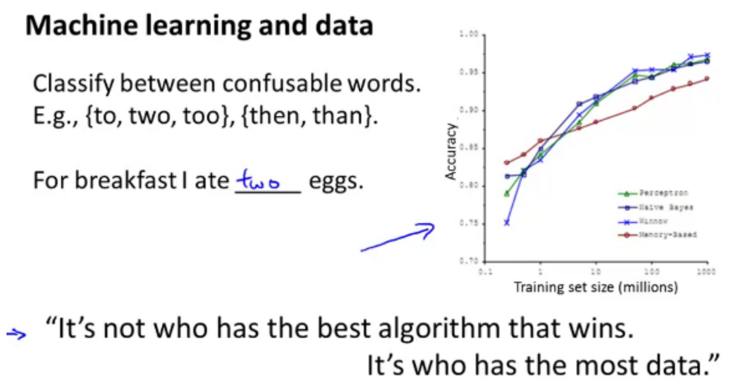
Part I
Large Scale Machine Learning

大规模机器学习

1 Gradient Descent with Large Datasets

1.1 Learning With Large Datasets

大型数据集的学习:



Machine learning and data

Classify between confusable words.
E.g., {to, two, too}, {then, than}.

For breakfast I ate two eggs.

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

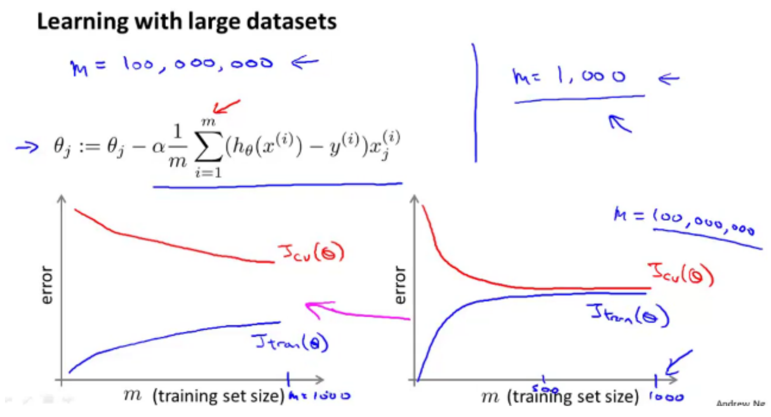
图 1: 机器学习与数据

如果我们有一个低方差的模型,增加数据集的规模可以帮助你获得更好的结果。我们应该怎样应对一个有 100 万条记录的训练集?

以线性回归模型为例,每一次梯度下降迭代,我们都需要计算训练集的误差的平方和,如果我们的学习算法需要有 20 次迭代,这便已经是非常大的计算代价。

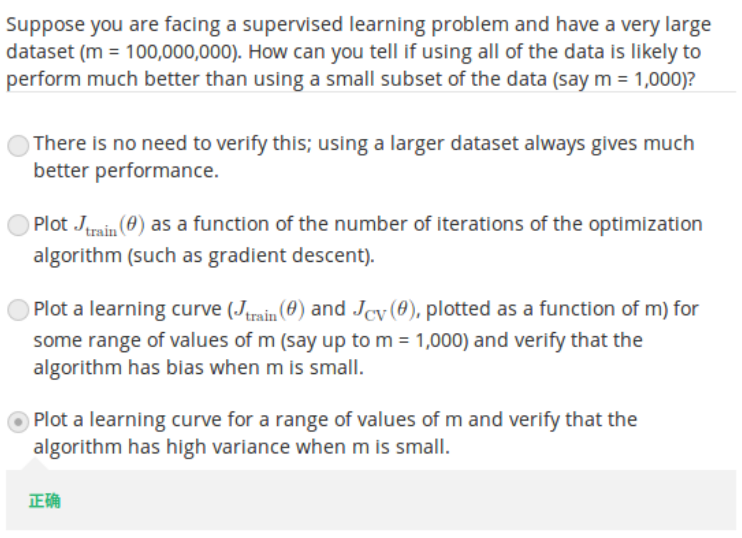
首先应该做的事是去检查一个这么大规模的训练集是否真的必要,也许我们只用 1000个训练集也能获得较好的效果,我们可以绘制学习曲线来帮助判断。

通常的方法是画学习曲线,如果你画了学习曲线而且你的训练目标看上去像这样, 这是$J_{train}()$,如果你的j交叉验证集的目标 $J_{cv}()$,看上去像这个 ,这看起来像高方差的学习算法 ,我们会对增加训练集的大小来提高性能更有信心, 而相比之下如果你画的学习曲线是这样的 ,你的训练目标是这样的, 你的交叉验证是那样的, 这看起来像经典的高偏差学习算法。


Learning with large datasets

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

图 2: 大型数据集的学习

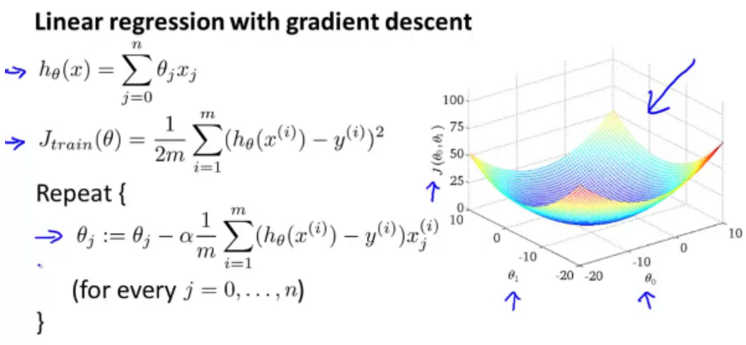

Suppose you are facing a supervised learning problem and have a very large dataset (m = 100,000,000). How can you tell if using all of the data is likely to perform much better than using a small subset of the data (say m = 1,000)?

- There is no need to verify this; using a larger dataset always gives much better performance.
- Plot $J_{\text{train}}(\theta)$ as a function of the number of iterations of the optimization algorithm (such as gradient descent).
- Plot a learning curve ($J_{\text{train}}(\theta)$ and $J_{\text{CV}}(\theta)$, plotted as a function of m) for some range of values of m (say up to m = 1,000) and verify that the algorithm has bias when m is small.
- [✓ Plot a learning curve for a range of values of m and verify that the algorithm has high variance when m is small.]

正确

1.2 Stochastic Gradient Descent

对于很多机器学习算法,包括线性回归、逻辑回归、神经网络等等 ,算法的实现都是通过得出某个代价函数 ,或者某个最优化的目标来实现的 ,然后使用梯度下降这样的方法来求得代价函数的最小值 ,当我们的训练集较大时 ,梯度下降算法则显得计算量非常大。


Linear regression with gradient descent

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

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

Repeat {

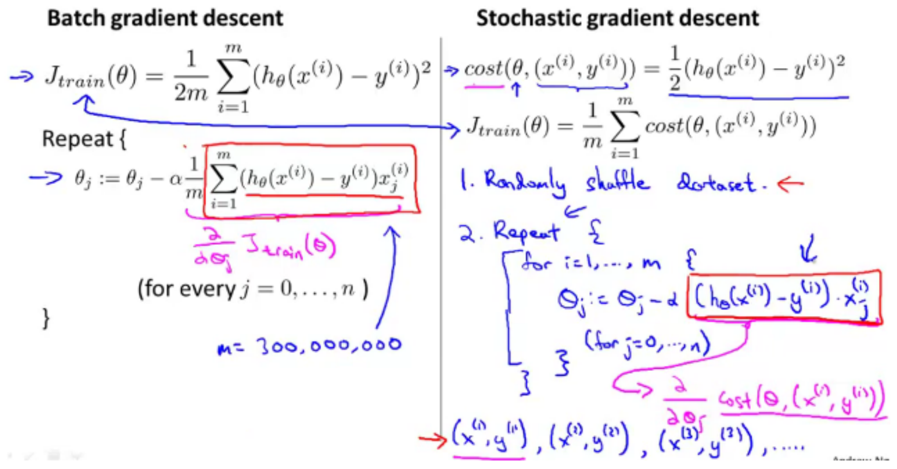
$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

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

}

图 3: 线性回归梯度下降

上述梯度下降也叫作批量梯度下降（Bitch gradient descent）



Batch gradient descent

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

```
Repeat {
  → $\theta_j := \theta_j - \alpha \boxed{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}}$
        $\frac{\partial}{\partial\theta_j} J_{train}(\theta)$
        (for every $j = 0, \dots, n$)
}
```

$m = 300,000,000$

Stochastic gradient descent

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

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

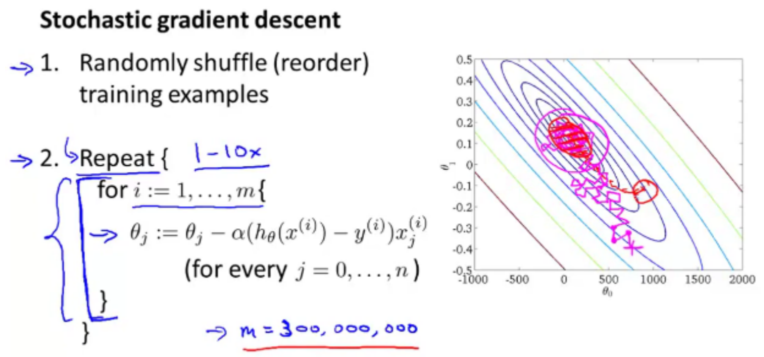
```
1. Randomly shuffle dataset. ←
2. Repeat {
     for i=1,...,m {
        $\theta_j := \theta_j - \alpha \boxed{(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}}$
        (for j=0,...,n)
     }
   }
```

$\frac{\partial}{\partial\theta_j} cost(\theta, (x^{(i)}, y^{(i)}))$

$\rightarrow (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \dots$

图 4: 批量梯度下降与随机梯度下降

随机梯度下降算法在每一次计算之后便更新参数 θ ,而不需要首先将所有的训练集求和,在梯度下降算法还没有完成一次迭代时,随机梯度下降算法便已经走出了很远。但是这样的算法存在的问题是,不是每一步都是朝着”正确”的方向迈出的。因此算法虽然会逐渐走向全局最小值的位置,但是可能无法站到那个最小值的那一点,而是在最小值点附近徘徊。



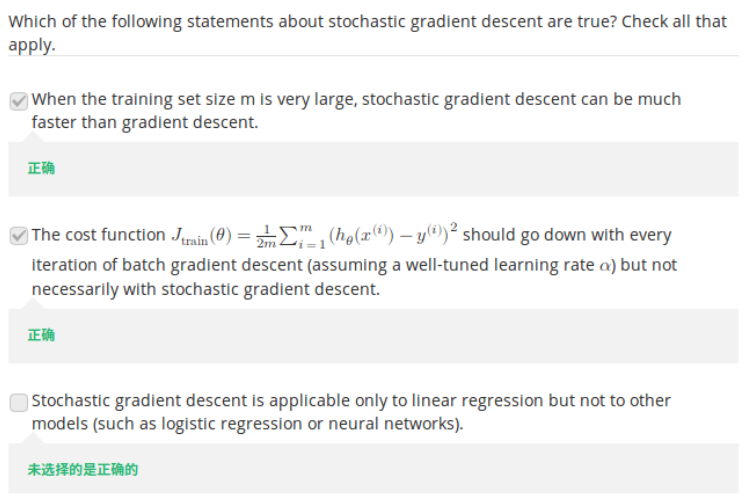
Stochastic gradient descent

```
→ 1. Randomly shuffle (reorder)
     training examples

→ 2. Repeat {   1-10x
       for $i := 1, \dots, m${
         → $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$
              (for every $j = 0, \dots, n$)
       }
     }
```

$\rightarrow m = 300,000,000$

图 5: 随机梯度下降



Which of the following statements about stochastic gradient descent are true? Check all that apply.

- [✓] When the training set size m is very large, stochastic gradient descent can be much faster than gradient descent.

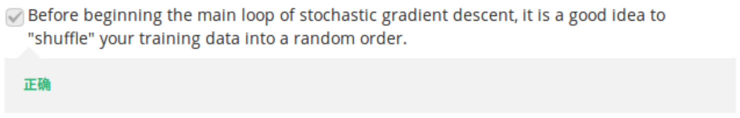
正确

- [✓] The cost function $J_{\text{train}}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$ should go down with every iteration of batch gradient descent (assuming a well-tuned learning rate α) but not necessarily with stochastic gradient descent.

正确

- [ ] Stochastic gradient descent is applicable only to linear regression but not to other models (such as logistic regression or neural networks).

未选择的是正确的



✓ Before beginning the main loop of stochastic gradient descent, it is a good idea to "shuffle" your training data into a random order.

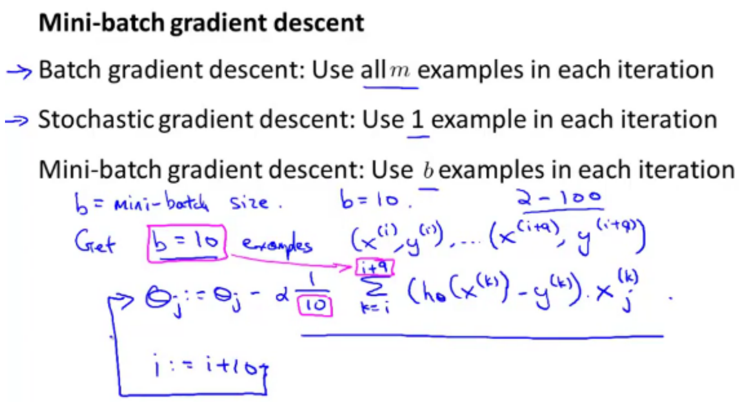
正确

1.3 Mini-Batch Gradient Descent

小批量梯度下降算法是介于批量梯度下降算法和随机梯度下降算法之间的算法,每计算常数 b 次训练实例,便更新一次参数 θ 。

各种梯度下降的一次迭代使用的样本个数

通常我们会令 b 在 2-100 之间。这样做的好处在于,我们可以用向量化的方式来循环b 个训练实例,如果我们用的线性代数函数库比较好,能够支持平行处理,那么算法的总体表现将不受影响(与随机梯度下降相同)。



Mini-batch gradient descent

→ Batch gradient descent: Use all $\underline{m}$ examples in each iteration

→ Stochastic gradient descent: Use $\underline{1}$ example in each iteration

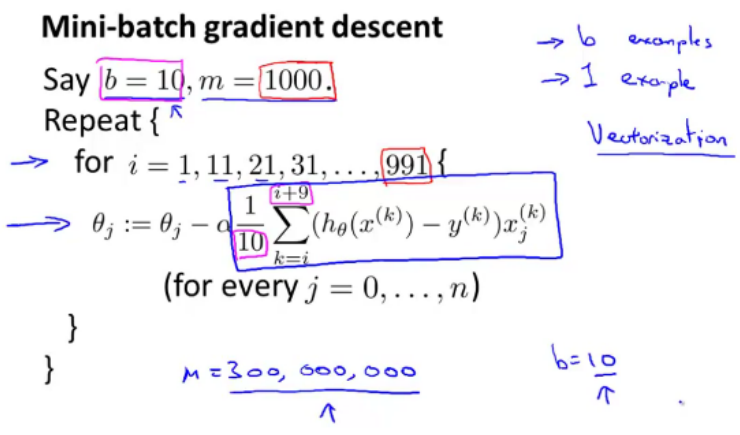
Mini-batch gradient descent: Use b examples in each iteration

b = mini-batch size. $b = 10$. $\underline{2-100}$

Get $\boxed{b=10}$ examples $(x^{(i)}, y^{(i)}), \dots (x^{(i+9)}, y^{(i+9)})$

$$\theta_j := \theta_j - \alpha \frac{1}{\boxed{10}} \underline{\sum_{k=i}^{\boxed{i+9}} \left(h_\theta(x^{(k)}) - y^{(k)}\right) \cdot x_j^{(k)}}$$

$i := i + 10$



Mini-batch gradient descent

```
Say $\boxed{b = 10}$, $\underline{m} = \boxed{1000}$.
Repeat {
    for $i = 1, 11, 21, 31, \dots, \boxed{991}$ {
        $\theta_j := \theta_j - \alpha \boxed{\frac{1}{\boxed{10}} \sum_{k=i}^{\boxed{i+9}} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}}$
            (for every $j = 0, \dots, n$)
    }
}
```

→ b examples

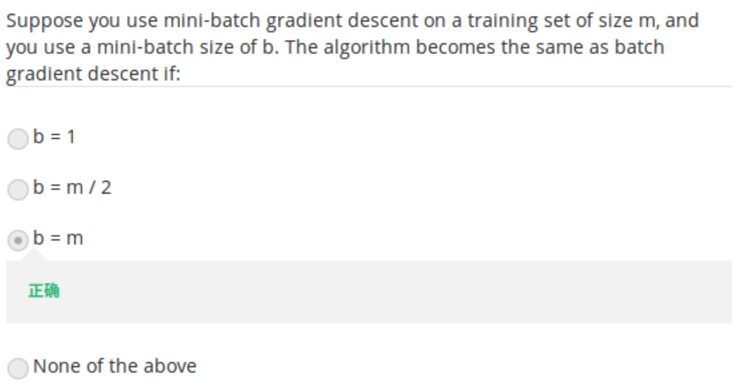
→ 1 example

$\underline{\text{Vectorization}}$

$\underline{m = 300{,}000{,}000}$

$b = 10$

图 6: 批量梯度下降



Suppose you use mini-batch gradient descent on a training set of size m, and you use a mini-batch size of b. The algorithm becomes the same as batch gradient descent if:

- b = 1
- b = m / 2
- [✓ b = m]

正确

- None of the above

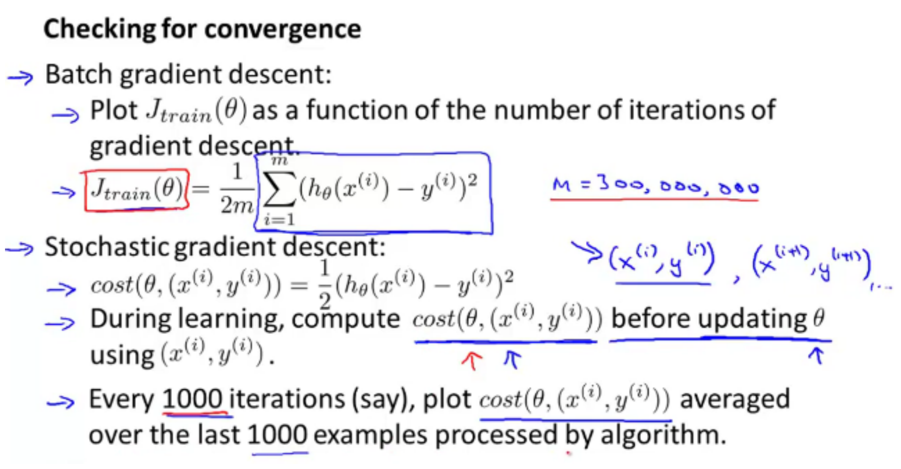
1.4 Stochastic Gradient Descent Convergence

随机梯度下降收敛

现在我们介绍随机梯度下降算法的调试,以及学习率 α 的选取。

数据大的时候使用随机梯度下降，小的时候使用batch Gradient Descent即可。

在批量梯度下降中,我们可以令代价函数 J 为迭代次数的函数,绘制图表,根据图表来判断梯度下降是否收敛。但是,在大规模的训练集的情况下,这是不现实的,因为计算代价太大了。



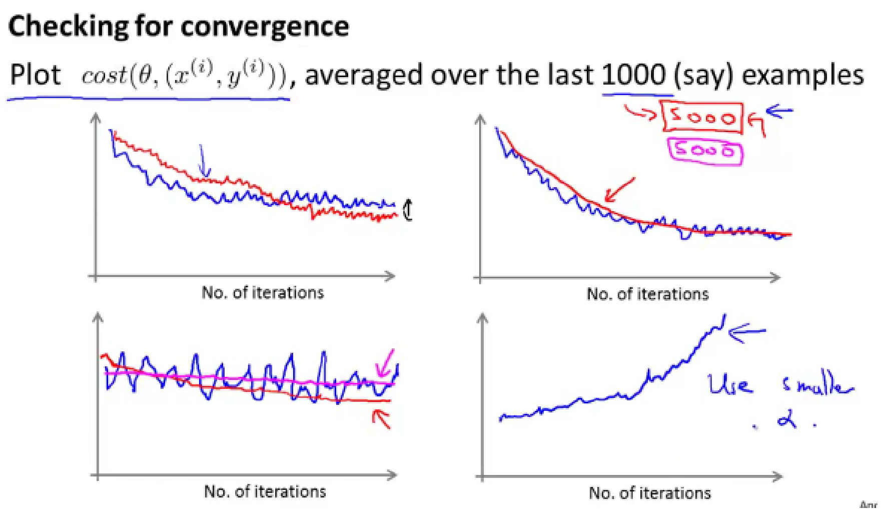
Checking for convergence

- Batch gradient descent:
  - Plot $J_{train}(\theta)$ as a function of the number of iterations of gradient descent.
  - $\boxed{J_{train}(\theta)} = \frac{1}{2m}\boxed{\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2}$ $\underline{M = 300,000,000}$
- Stochastic gradient descent: $\rightarrow \underline{(x^{(i)}, y^{(i)})}, (x^{(i+1)}, y^{(i+1)}) \ldots$
  - $cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$
  - During learning, compute $\underline{cost(\theta, (x^{(i)}, y^{(i)}))}$ $\underline{\text{before updating } \theta}$ using $(x^{(i)}, y^{(i)})$.
  - Every $\underline{1000}$ iterations (say), plot $\underline{cost(\theta, (x^{(i)}, y^{(i)}))}$ averaged over the last $\underline{1000}$ examples processed by algorithm.

在随机梯度下降中,我们在每一次更新 θ 之前都计算一次代价,然后每 X 次迭代后,求出这 X 次对训练实例计算代价的平均值,然后绘制这些平均值与 X 次迭代的次数之间的函数图表。

当我们绘制这样的图表时,可能会得到一个颠簸不平但是不会明显减少的函数图像(如上面左下图中蓝线所示)。我们可以增加你要平均的样本数量来使得函数更加平缓,也许便能看出下降的趋势了(如上面左下图中红线所示);或者可能函数图表仍然是颠簸不平且不下降的(如洋红色线所示),那么我们的模型本身可能存在一些错误。

如果我们得到的曲线如上面右下方所示,代价值是在不断地上升,那么选择一个较小的学习率 α 。

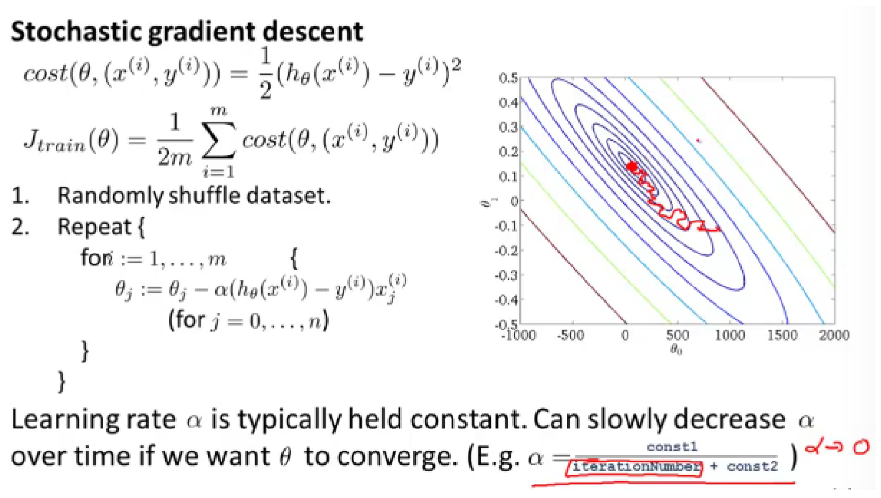


Checking for convergence

Plot $cost(\theta,(x^{(i)},y^{(i)}))$, averaged over the last 1000 (say) examples

我们也可以令学习率随着迭代次数的增加而减小,例如令:

$$\alpha = \frac{const1}{interationNumber + const2} \tag{1}$$



θ_0
θ_1

Stochastic gradient descent

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

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m} cost(\theta,(x^{(i)},y^{(i)}))$$

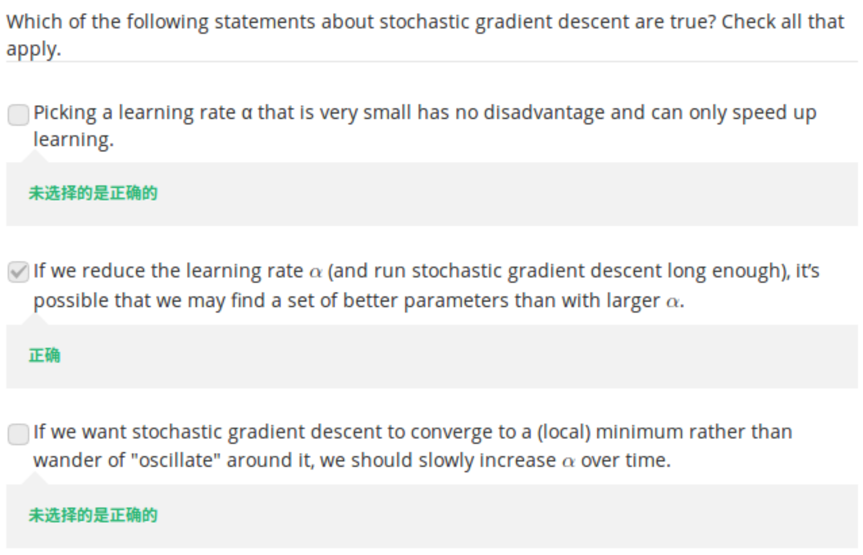
```
1.  Randomly shuffle dataset.
2.  Repeat {
        for $i := 1,\ldots,m$        {
            $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$
                (for $j = 0,\ldots,n$)
        }
    }
```

Learning rate α is typically held constant. Can slowly decrease α over time if we want θ to converge. (E.g. $\alpha = \frac{\text{const1}}{\text{iterationNumber + const2}}$) $\alpha \to 0$

这个公式起作用的原因是: 随着算法的运行 ,迭代次数会越来越大, 因此学习速率 α 会慢慢变小 ,因此你的每一步就会越来越小, 直到最终收敛到全局最小值, 所以 ,如果你慢慢减小 α 的值到0 ,你会最后得到一个更好一点的假设。但由于确定这两个常数需要更多的工作量 , 并且我们通常也对能够很接近全局最小值的参数已经很满意了, 因此我们很少采用逐渐减小 α 的值的方法。

随着我们不断地靠近全局最小值,通过减小学习率,我们迫使算法收敛而非在最小值附近徘徊。 但是通常我们不需要这样做便能有非常好的效果了,对 α 进行调整所耗费的计算通常不值得。

总结下,这段视频中,我们介绍了一种方法,近似地监测出随机梯度下降算法在最优化代价函数中的表现,这种方法不需要定时地扫描整个训练集,来算出整个样本集的代价函数,而是只需要每次对最后 1000 个,或者多少个样本,求一下平均值。应用这种方法,你既可以保证随机梯度下降法正在正常运转和收敛,也可以用它来调整学习速率 α 的大小。



Which of the following statements about stochastic gradient descent are true? Check all that apply.

☐ Picking a learning rate α that is very small has no disadvantage and can only speed up learning.

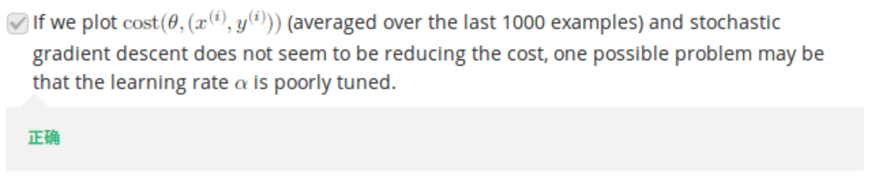
未选择的是正确的

[✓] If we reduce the learning rate α (and run stochastic gradient descent long enough), it's possible that we may find a set of better parameters than with larger α.

正确

☐ If we want stochastic gradient descent to converge to a (local) minimum rather than wander of "oscillate" around it, we should slowly increase α over time.

未选择的是正确的



[✓] If we plot $\mathrm{cost}(\theta,(x^{(i)},y^{(i)}))$ (averaged over the last 1000 examples) and stochastic gradient descent does not seem to be reducing the cost, one possible problem may be that the learning rate α is poorly tuned.

正确

2 Advanced Topics

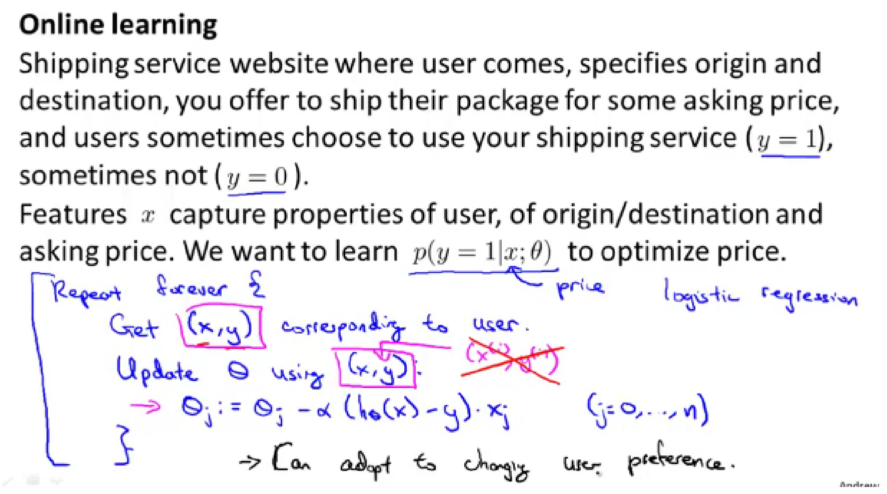
2.1 Online Learning

在线学习机制:

一种新的大规模的机器学习机制,叫做在线学习机制。在线学习机制让我们可以模型化问题。

今天,许多大型网站或者许多大型网络公司,使用不同版本的在线学习机制算法,从大批的涌入又离开网站的用户身上进行学习。特别要提及的是,如果你有一个由连续的用户流引发的连续的数据流,进入你的网站,你能做的是使用一个在线学习机制,从数据流中学习用户的偏好,然后使用这些信息来优化一些关于网站的决策。

假定你有一个提供运输服务的公司,用户们来向你询问把包裹从 A 地运到 B 地的服务,同时假定你有一个网站,让用户们可多次登陆,然后他们告诉你,他们想从哪里寄出包裹,以及包裹要寄到哪里去,也就是出发地与目的地,然后你的网站开出运输包裹的的服务价格。比如,我会收取$50来运输你的包裹,我会收取$20之类的,然后根据你开给用户的这个价 格,用户有时会接受这个运输服务,那么这就是个正样本,有时他们会走掉,然后他们拒绝购买你的运输服务,所以,让我们假定我们想要一个学习算法来帮助我们,优化我们想给用户开出的价格。



Online learning

Shipping service website where user comes, specifies origin and destination, you offer to ship their package for some asking price, and users sometimes choose to use your shipping service ($y = 1$), sometimes not ($y = 0$).

Features x capture properties of user, of origin/destination and asking price. We want to learn $p(y = 1|x;\theta)$ to optimize price.

price   logistic regression

```
Repeat forever {
    Get (x,y) corresponding to user.
    Update θ using (x,y).    ~~(x^(i), y^(i))~~
    → $\theta_j := \theta_j - \alpha(h_\theta(x) - y) \cdot x_j$    $(j = 0, \dots, n)$
}
            → Can adapt to changing user preference.
```

一个算法来从中学习的时候来模型化问题在线学习算法指的是对数据流而非离线的静态数据集的学习。许多在线网站都有持续不断的用户流,对于每一个用户,网站希望能在不将数据存储到数据库中便顺利地进行算法学习。

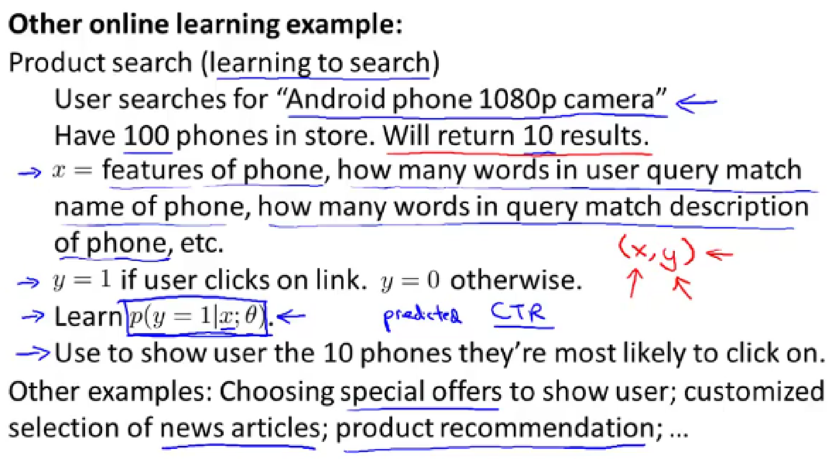
假使我们正在经营一家物流公司,每当一个用户询问从地点 A 至地点 B 的快递费用时,我们给用户一个报价,该用户可能选择接受(y=1)或不接受(y=0)。

现在,我们希望构建一个模型,来预测用户接受报价使用我们的物流服务的可能性。因此报价 是我们的一个特征,其他特征为距离,起始地点,目标地点以及特定的用户数据。模型的输出是 p(y=1)。

在线学习的算法与随机梯度下降算法有些类似,我们对单一的实例进行学习,而非对一个提前定义的训练集进行循环。

一旦对一个数据的学习完成了,我们便可以丢弃该数据,不需要再存储它了。这种方式的好处在于,我们的算法可以很好的适应用户的倾向性,算法可以针对用户的当前行为不断地更新模型以适应该用户。

每次交互事件并不只产生一个数据集,例如,我们一次给用户提供 3 个物流选项,用户选择 2 项,我们实际上可以获得 3 个新的训练实例,因而我们的算法可以一次从 3 个实例中学习并更新模型。



Other online learning example:

Product search (learning to search)

User searches for "Android phone 1080p camera" ←

Have 100 phones in store. Will return 10 results.

→ $x =$ features of phone, how many words in user query match name of phone, how many words in query match description of phone, etc.

→ $y = 1$ if user clicks on link. $y = 0$ otherwise.   (x,y) ←

→ Learn $\boxed{p(y = 1|x;\theta)}$. ←   predicted CTR

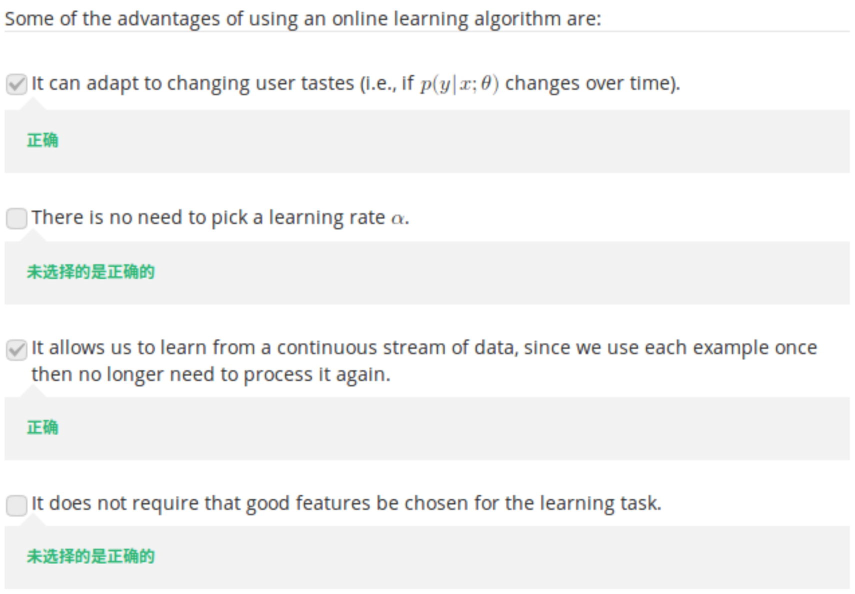
→ Use to show user the 10 phones they're most likely to click on.

Other examples: Choosing special offers to show user; customized selection of news articles; product recommendation; ...

图 7: 学习搜索手机示例

这些问题中的任何一个都可以被归类到标准的,拥有一个固定的样本集的机器学习问题中。或许,你可以运行一个你自己的网站,尝试运行几天,然后保存一个数据集,一个固定的数据集,然后对其运行一个学习算法。

但是这些是实际的问题,在这些问题里,你会看到大公司会获取如此多的数据,真的没有必要来保存一个固定的数据集,取而代之的是你可以使用一个在线学习算法来连续的学习,从这些用户不断产生的数据中来学习。


Some of the advantages of using an online learning algorithm are:

✓ It can adapt to changing user tastes (i.e., if $p(y|x;\theta)$ changes over time).

正确

☐ There is no need to pick a learning rate α.

未选择的是正确的

✓ It allows us to learn from a continuous stream of data, since we use each example once then no longer need to process it again.

正确

☐ It does not require that good features be chosen for the learning task.

未选择的是正确的

这就是在线学习机制,然后就像我们所看到的,我们所使用的这个算法与随机梯度下降算法非常类似,唯一的区别的是,我们不会使用一个固定的数据集,我们会做的是获取一个用户样本,从那个样本中学习,然后丢弃那个样本并继续下去,而且如果你对某一种应用有一个连续的数据流,这样的算法可能会非常值得考虑。

当然,在线学习的一个优点就是,如果你有一个变化的用户群,又或者你在尝试预测的事情,在缓慢变化,就像你的用户的品味在缓慢变化,这个在线学习算法,可以慢慢地调试你所学习到的假设,将其调节更新到最新的用户行为。

2.2 Map Reduce and Data Parallelism

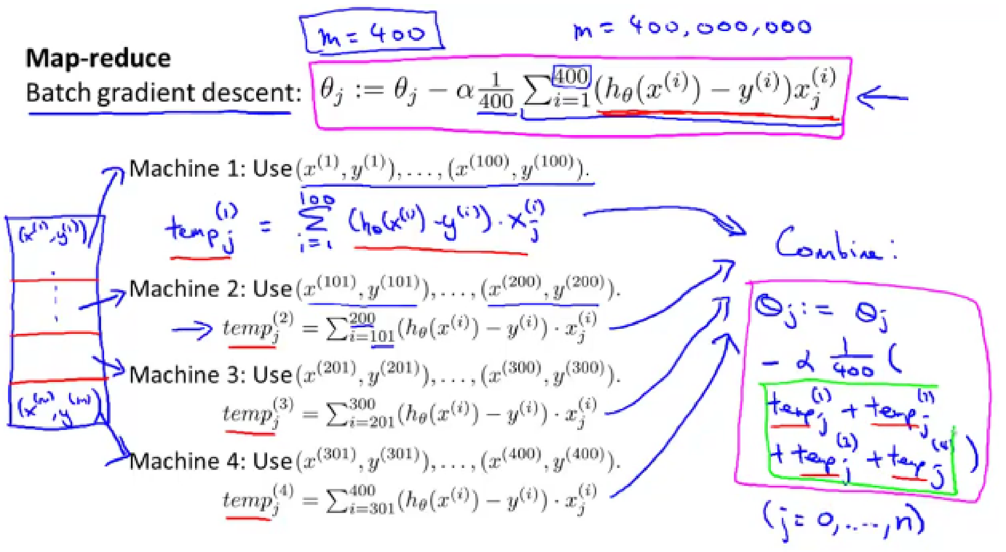
进行大规模机器学习的另一种方法：映射（化简）约减和数据并行

有些算法太大，太复杂以至于不能在一台计算机上运行

映射化简和数据并行对于大规模机器学习问题而言是非常重要的概念。之前提到,如果我们用批量梯度下降算法来求解大规模数据集的最优解,我们需要对整个训练集进行循环,计算偏导数和代价,再求和,计算代价非常大。如果我们能够将我们的数据集分配给不多台计算机,让每一台计算机处理数据集的一个子集,然后我们将计所的结果汇总在求和。这样的方法叫做映射简化。

具体而言,如果任何学习算法能够表达为,对训练集的函数的求和,那么便能将这个任分配给多台计算机(或者同一台计算机的不同 CPU 核心),以达到加速处理的目的。

例如,我们有 400 个训练实例,我们可以将批量梯度下降的求和任务分配给 4 台计算机进行处理:



Map-reduce

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

$\boxed{m = 400}$ $m = 400{,}000{,}000$

Machine 1: Use $(x^{(1)}, y^{(1)}), \ldots, (x^{(100)}, y^{(100)})$.

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

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

$temp_j^{(2)} = \sum_{i=101}^{200} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

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

$temp_j^{(3)} = \sum_{i=201}^{300} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

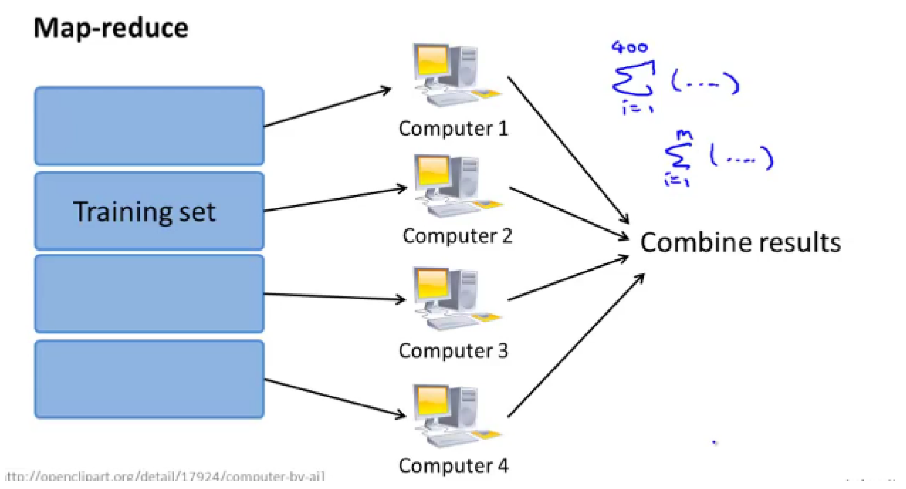
Machine 4: Use $(x^{(301)}, y^{(301)}), \ldots, (x^{(400)}, y^{(400)})$.

$temp_j^{(4)} = \sum_{i=301}^{400} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

Combine:

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

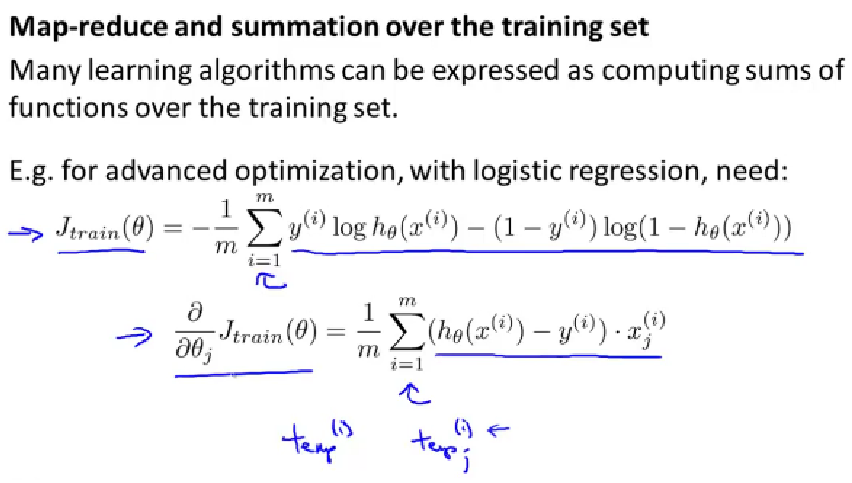
$(j = 0, \ldots, n)$



Map-reduce

Training set
Computer 1
Computer 2
Computer 3
Computer 4
Combine results
$\sum_{i=1}^{400} (\ldots)$
$\sum_{i=1}^{m} (\ldots)$
ttp://openclipart.org/detail/17924/computer-by-aj

图 8: 映射化简



Map-reduce and summation over the training set

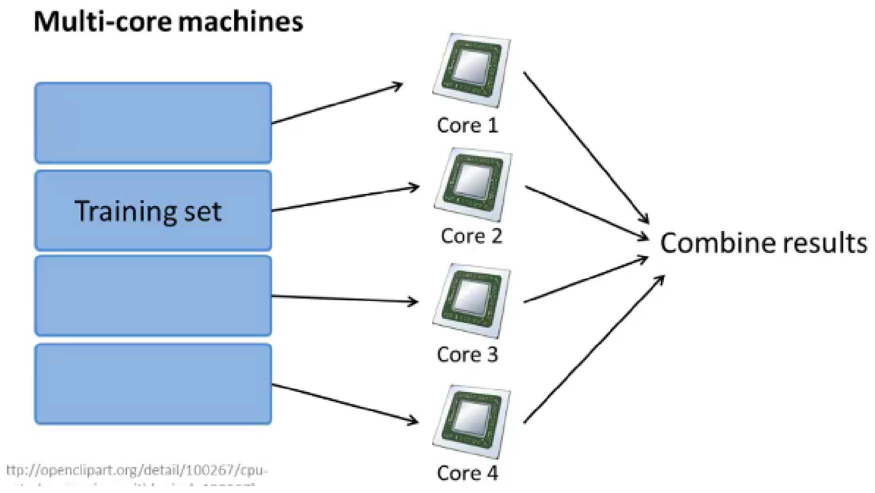
Many learning algorithms can be expressed as computing sums of functions over the training set.

E.g. for advanced optimization, with logistic regression, need:

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

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

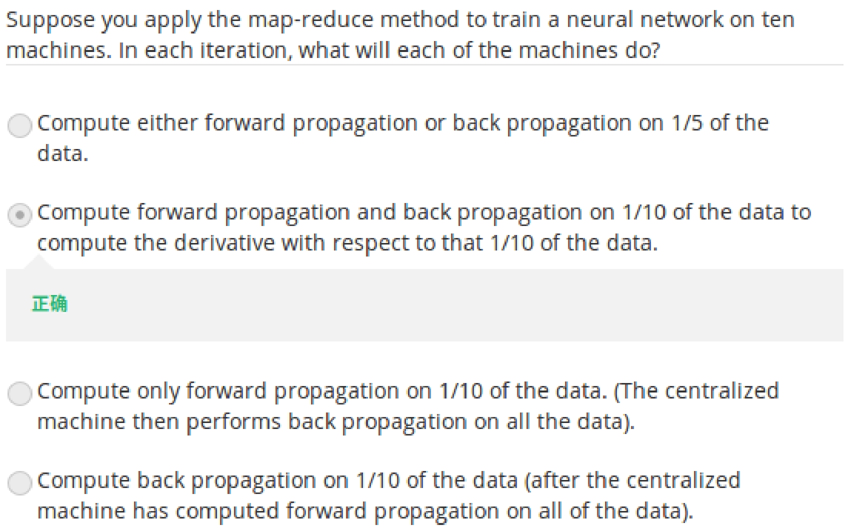
$temp^{(i)}$ $temp_j^{(i)}$



映射化简(map reduce)技术,它可以通过将数据分配到多台计算机的方式来并行化机器学习算法,实际上这种方法 也可以利用 单台计算机的多个核。

网上有许多优秀的开源映射化简实现,实际上一个称为Hadoop 的开源系统,已经拥有了众多的用户,通过自己实现映射化简算法 或者使用别人的开源实现,你就可以利用映射化简技术来并行机器学习算法,这样你的算法将能够处理,单台计算机处理不了的大数据 。

很多高级的线性代数函数库已经能够利用多核 CPU 的多个核心来并行地处理矩阵运算,这也是算法的向量化实现如此重要的缘故(比调用循环快)。


Suppose you apply the map-reduce method to train a neural network on ten machines. In each iteration, what will each of the machines do?

- Compute either forward propagation or back propagation on 1/5 of the data.
- [✓ Compute forward propagation and back propagation on 1/10 of the data to compute the derivative with respect to that 1/10 of the data.]

正确

- Compute only forward propagation on 1/10 of the data. (The centralized machine then performs back propagation on all the data).
- Compute back propagation on 1/10 of the data (after the centralized machine has computed forward propagation on all of the data).

1。
Suppose you are training a logistic regression classifier using stochastic gradient descent. You find that the cost (say, $cost(\theta, (x^{(i)}, y^{(i)}))$, averaged over the last 500 examples), plotted as a function of the number of iterations, is slowly increasing over time. Which of the following changes are likely to help?

- ○ Try using a larger learning rate α.
- [✓ Try using a smaller learning rate α.]
- ○ This is not an issue, as we expect this to occur with stochastic gradient descent.
- ○ Try averaging the cost over a larger number of examples (say 1000 examples instead of 500) in the plot.

2. Which of the following statements about stochastic gradient

descent are true? Check all that apply.
答案CD

- ☐ Suppose you are using stochastic gradient descent to train a linear regression classifier. The cost function $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$ is guaranteed to decrease after every iteration of the stochastic gradient descent algorithm.
- ☐ In order to make sure stochastic gradient descent is converging, we typically compute $J_{\text{train}}(\theta)$ after each iteration (and plot it) in order to make sure that the cost function is generally decreasing.
- ☐ You can use the method of numerical gradient checking to verify that your stochastic gradient descent implementation is bug-free. (One step of stochastic gradient descent computes the partial derivative $\frac{\partial}{\partial \theta_j} cost(\theta, (x^{(i)}, y^{(i)}))$)
- ☐ Before running stochastic gradient descent, you should randomly shuffle (reorder) the training set.

2。
Which of the following statements about stochastic gradient

descent are true? Check all that apply.

- [✓ In each iteration of stochastic gradient descent, the algorithm needs to examine/use only one training example.]
- ☐ Stochastic gradient descent is particularly well suited to problems with small training set sizes; in these problems, stochastic gradient descent is often preferred to batch gradient descent.
- [✓ One of the advantages of stochastic gradient descent is that it can start progress in improving the parameters θ after looking at just a single training example; in contrast, batch gradient descent needs to take a pass over the entire training set before it starts to make progress in improving the parameters' values.]
- ☐ Suppose you are using stochastic gradient descent to train a linear regression classifier. The cost function $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$ is guaranteed to decrease after every iteration of the stochastic gradient descent algorithm.

3.
Which of the following statements about online learning are true? Check all that apply.

- One of the disadvantages of online learning is that it requires a large amount of computer memory/disk space to store all the training examples we have seen.
- [✓ In the approach to online learning discussed in the lecture video, we repeatedly get a single training example, take one step of stochastic gradient descent using that example, and then move on to the next example.]
- One of the advantages of online learning is that there is no need to pick a learning rate α.
- [✓ When using online learning, in each step we get a new example (x, y), perform one step of (essentially stochastic gradient descent) learning on that example, and then discard that example and move on to the next.]

4.
Assuming that you have a very large training set, which of the

following algorithms do you think can be parallelized using

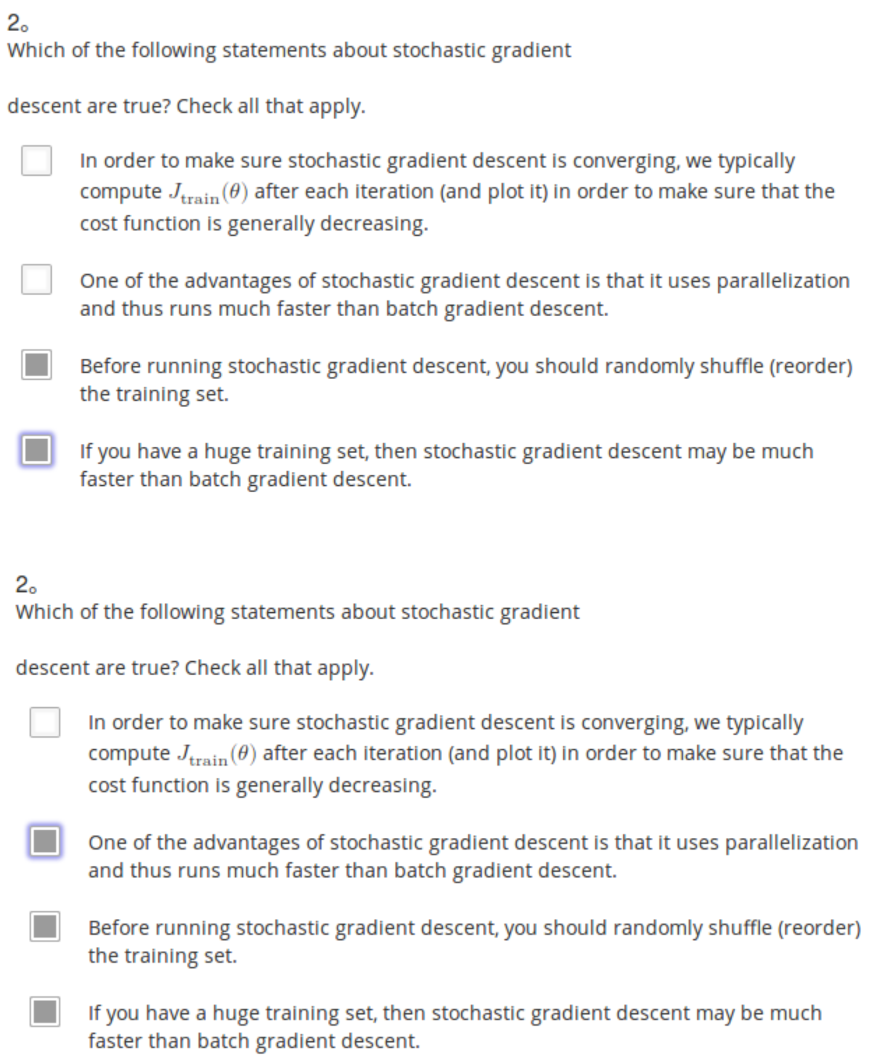
map-reduce and splitting the training set across different

machines? Check all that apply.

- Logistic regression trained using stochastic gradient descent.
- [✓ Logistic regression trained using batch gradient descent.]
- Linear regression trained using stochastic gradient descent.
- [✓ Computing the average of all the features in your training set $\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$ (say in order to perform mean normalization).]

5.
Which of the following statements about map-reduce are true? Check all that apply.

- If you have only 1 computer with 1 computing core, then map-reduce is unlikely to help.
- [✓ Because of network latency and other overhead associated with map-reduce, if we run map-reduce using N computers, we might get less than an N-fold speedup compared to using 1 computer.]
- If we run map-reduce using N computers, then we will always get at least an N-fold speedup compared to using 1 computer.
- [✓ When using map-reduce with gradient descent, we usually use a single machine that accumulates the gradients from each of the map-reduce machines, in order to compute the parameter update for that iteration.]



2。
Which of the following statements about stochastic gradient

descent are true? Check all that apply.

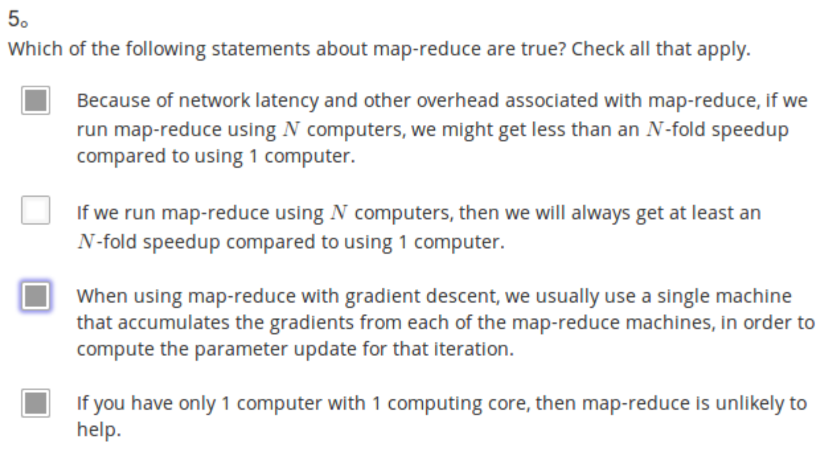
- [ ] In order to make sure stochastic gradient descent is converging, we typically compute $J_{\text{train}}(\theta)$ after each iteration (and plot it) in order to make sure that the cost function is generally decreasing.
- [ ] One of the advantages of stochastic gradient descent is that it uses parallelization and thus runs much faster than batch gradient descent.
- [✓] Before running stochastic gradient descent, you should randomly shuffle (reorder) the training set.
- [✓] If you have a huge training set, then stochastic gradient descent may be much faster than batch gradient descent.

2。
Which of the following statements about stochastic gradient

descent are true? Check all that apply.

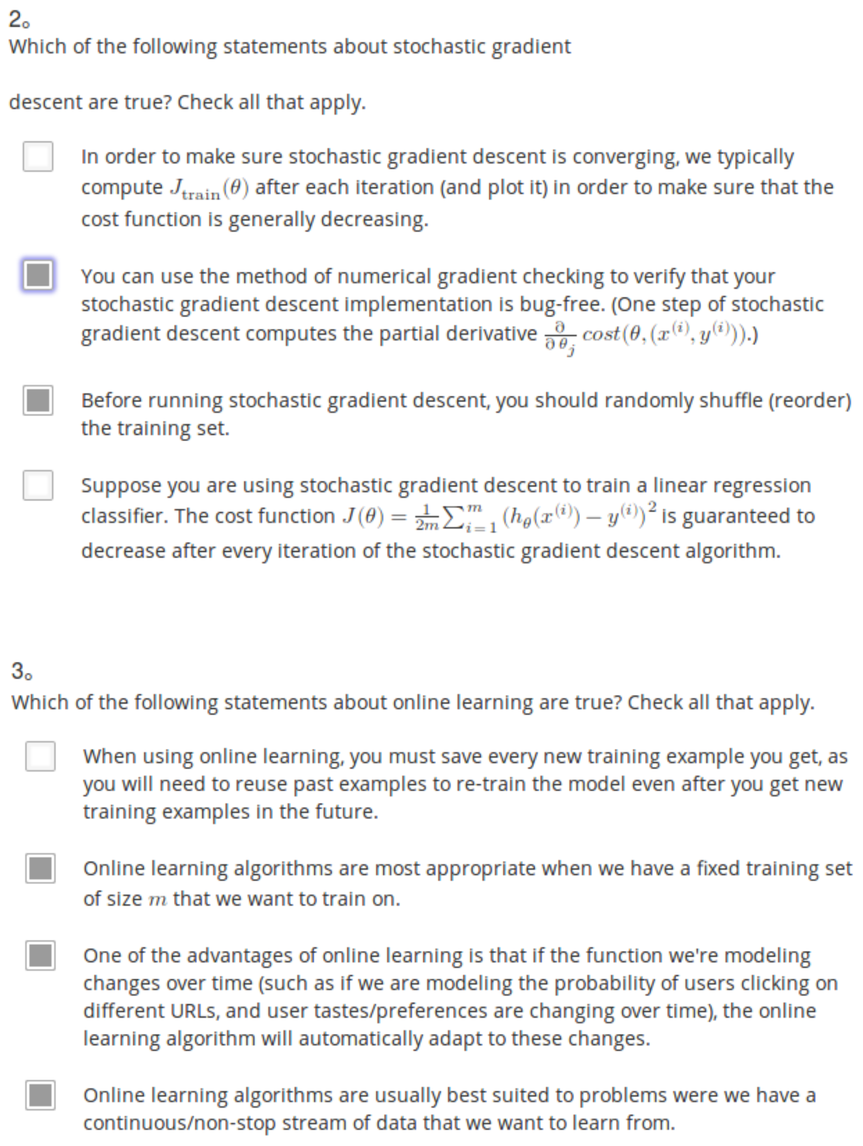
- [ ] In order to make sure stochastic gradient descent is converging, we typically compute $J_{\text{train}}(\theta)$ after each iteration (and plot it) in order to make sure that the cost function is generally decreasing.
- [✓] One of the advantages of stochastic gradient descent is that it uses parallelization and thus runs much faster than batch gradient descent.
- [✓] Before running stochastic gradient descent, you should randomly shuffle (reorder) the training set.
- [✓] If you have a huge training set, then stochastic gradient descent may be much faster than batch gradient descent.

图 9: 2题答案有错误



5。
Which of the following statements about map-reduce are true? Check all that apply.

- [✓] Because of network latency and other overhead associated with map-reduce, if we run map-reduce using N computers, we might get less than an N-fold speedup compared to using 1 computer.
- [ ] If we run map-reduce using N computers, then we will always get at least an N-fold speedup compared to using 1 computer.
- [✓] When using map-reduce with gradient descent, we usually use a single machine that accumulates the gradients from each of the map-reduce machines, in order to compute the parameter update for that iteration.
- [✓] If you have only 1 computer with 1 computing core, then map-reduce is unlikely to help.



2。
Which of the following statements about stochastic gradient

descent are true? Check all that apply.

- [ ] In order to make sure stochastic gradient descent is converging, we typically compute $J_{\text{train}}(\theta)$ after each iteration (and plot it) in order to make sure that the cost function is generally decreasing.
- [✓] You can use the method of numerical gradient checking to verify that your stochastic gradient descent implementation is bug-free. (One step of stochastic gradient descent computes the partial derivative $\frac{\partial}{\partial \theta_j} cost(\theta, (x^{(i)}, y^{(i)}))$.)
- [✓] Before running stochastic gradient descent, you should randomly shuffle (reorder) the training set.
- [ ] Suppose you are using stochastic gradient descent to train a linear regression classifier. The cost function $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$ is guaranteed to decrease after every iteration of the stochastic gradient descent algorithm.

3。
Which of the following statements about online learning are true? Check all that apply.

- [ ] When using online learning, you must save every new training example you get, as you will need to reuse past examples to re-train the model even after you get new training examples in the future.
- [✓] Online learning algorithms are most appropriate when we have a fixed training set of size m that we want to train on.
- [✓] One of the advantages of online learning is that if the function we're modeling changes over time (such as if we are modeling the probability of users clicking on different URLs, and user tastes/preferences are changing over time), the online learning algorithm will automatically adapt to these changes.
- [✓] Online learning algorithms are usually best suited to problems were we have a continuous/non-stop stream of data that we want to learn from.

图 10: 此题答案有误