華南理工大學

South China University of Technology

《机器学习》课程实验报告

学　　院　软件学院

专　　业　软件工程

组　　员　饶浩聪

学　　号　201530612644

邮　　箱　568302203@qq.com

指导教师 吴庆耀

提交日期 2017 年 12 月 15 日

1. 实验题目: 逻辑回归、线性分类与随机梯度下降

2. 实验时间：2017 年 12 月 10 日

3. 报告人: 饶浩聪

4. 实验目的:

对比理解梯度下降和随机梯度下降的区别与联系。

对比理解逻辑回归和线性分类的区别与联系。

进一步理解 SVM 的原理并在较大数据上实践。

5. 数据集以及数据分析：

实验使用的是 LIBSVM Data 的中的 a9a 数据，包含 32561 / 16281(testing)个样本，每个样本有 123/123 (testing)个属性。请自行下载训练集和验证集。

6. 实验步骤:

逻辑回归与随机梯度下降

1. 读取实验训练集和验证集。
2. 逻辑回归模型参数初始化，可以考虑全零初始化，随机初始化或者正态分布初始化。
3. 选择Loss函数及对其求导，过程详见课件ppt。
4. 求得**部分样本**对Loss函数的梯度G。
5. **使用不同的优化方法更新模型参数（NAG，RMSProp，AdaDelta和Adam）**。
6. 选择合适的阈值，将验证集中计算结果**大于阈值的标记为正类，反之为负类**。在验证集上测试并得到不同优化方法的Loss函数值L_{NAG}，$L_{RMSProp}$，$L_{AdaDelta}$和L_{Adam}。
7. 重复步骤4-6若干次，**画出L_{NAG}，$L_{RMSProp}$，$L_{AdaDelta}$和L_{Adam}随迭代次数的变化图。**

线性分类与随机梯度下降

1. 读取实验训练集和验证集。
2. 支持向量机模型参数初始化，可以考虑全零初始化，随机初始化或者正态分布初始化。
3. 选择Loss函数及对其求导，过程详见课件ppt。
4. 求得**部分样本**对Loss函数的梯度G。
5. **使用不同的优化方法更新模型参数（NAG，RMSProp，AdaDelta和Adam）**。
6. 选择合适的阈值，将验证集中计算结果**大于阈值的标记为正类，反之为负类**。在验证集上测试并得到不同优化方法的Loss函数值L_{NAG}，$L_{RMSProp}$，$L_{AdaDelta}$和L_{Adam}。
7. 重复步骤4-6若干次，**画出L_{NAG}，$L_{RMSProp}$，$L_{AdaDelta}$和L_{Adam}随迭代次数的变化图。**

7. 代码内容:

（针对逻辑回归和线性分类分别填写 8-11 内容）

8. 模型参数的初始化方法:

9.选择的 loss 函数及导数:

1.逻辑回归选择的 loss 函数是(对数损失函数):

$$= \sum_{n} -\underbrace{\left[\hat{y}^n ln f_{w,b}(x^n) + (1-\hat{y}^n) ln\left(1-f_{w,b}(x^n)\right)\right]}_{\text{Cross entropy between two Bernoulli distribution}}$$

权重更新的公式是(包括导数):

$$w_i \leftarrow w_i - \eta \sum_{n} -\left(\underline{\hat{y}^n - f_{w,b}(x^n)}\right) x_i^n$$

2.线性分类 SVM 选择的 loss 函数是:

下面是课程 PPT 里面关于 SVM 的 loss 函数(注意是总的，还没有取平均)及其求导,与我实际用的有所差别，可以对比一下

$$L(f)=\sum_n l(f(x^n),\hat{y}^n) \qquad l(f(x^n),\hat{y}^n)=max\left(0,1-\hat{y}^n f(x^n)\right)$$

$$\frac{\partial l(f(x^n),\hat{y}^n)}{\partial w_i}=\frac{\partial l(f(x^n),\hat{y}^n)}{\partial f(x^n)}\boxed{\frac{\partial f(x^n)}{\partial w_i}}x_i^n \qquad \boxed{\begin{array}{l} f(x^n) \\ = w^T\cdot x^n\end{array}}$$

$$\frac{\partial max\left(0,1-\hat{y}^n f(x^n)\right)}{\partial f(x^n)}=\begin{cases}-\hat{y}^n & \text{If } \hat{y}^n f(x^n)<1 \\ 0 & \text{otherwise}\end{cases}$$

$$\frac{\partial L(f)}{\partial w_i}=\sum_n \underbrace{-\delta(\hat{y}^n f(x^n)<1)\hat{y}^n x_i}_{c^n(w)} \qquad w_i \leftarrow w_i-\eta\sum_n c^n(w)x_i^n$$

我实际所用的 **loss** 函数为：

$$L(f)=\frac{|w|\wedge 2}{2}+\frac{1}{n}\sum_{i=1}^{n}\max(0,1-y(i)f(x(i)))$$

实际上所用的 **w** 更新权重公式为(最右边为求的导)：

$$w_j := w_j - \eta\left(\mathrm{w_j}+\sum_{i=1}^{n}-\delta(y(i)*f(x(i)))*y(i)*x(i)_j\right)$$

其中，**n** 为训练样本总数量，**x(i)**为第 **i** 个样本的 **x** 向量(总共有 **14** 个特征)

y(i)为第 **i** 个样本的预测结果，**x(i)j** 为第 **i** 个样本的 **x** 向量第 **j** 个分向量，同理 **wj** 为 **w** 向量的第 **j** 个分向量，**f(x)**为分类函数即 **wx+b**,若大于 **0** 则 **y=1** 若小于 **0** 则 **y=-1**

10. **实验结果和曲线图:**（各种梯度下降方式分别填写此项）

超参数选择：

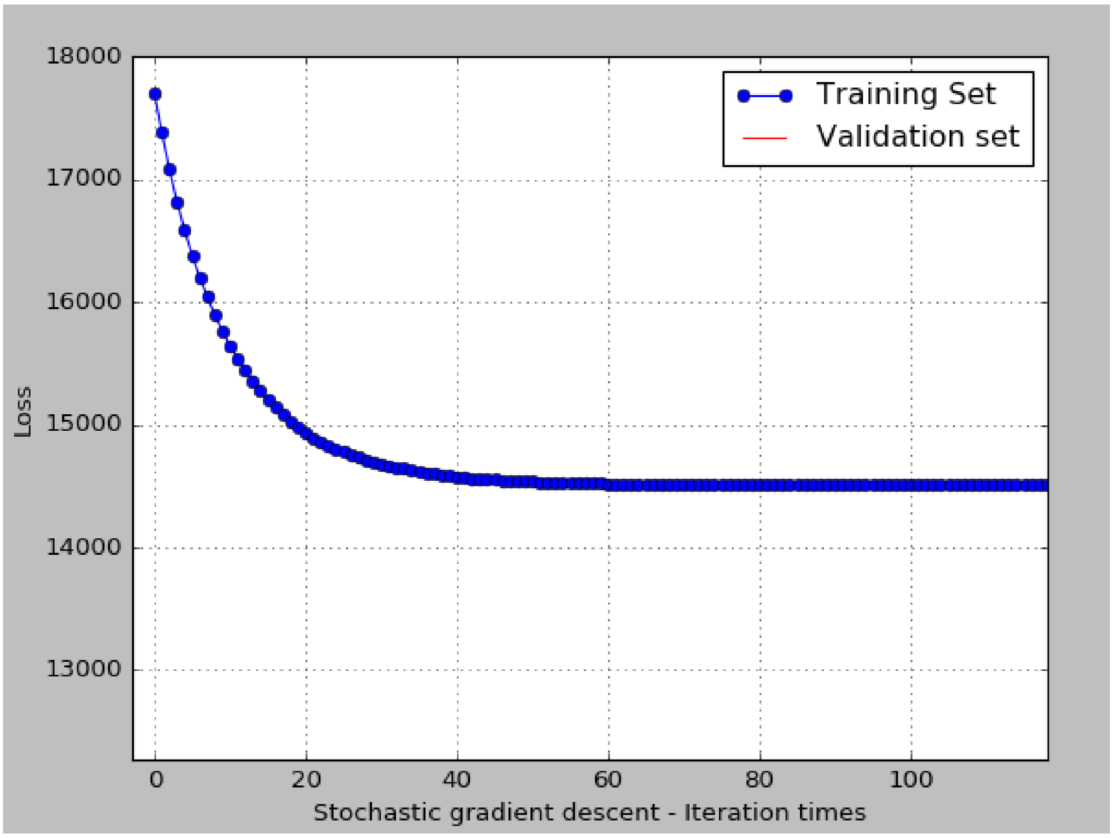
逻辑回归和线性分类实验中，w 采用全零初始化(经过测试发现这个效果最佳)，然后全局学习参数(若有)，逻辑回归为 a = 0.0010，SVM 为 aa = 0.00096

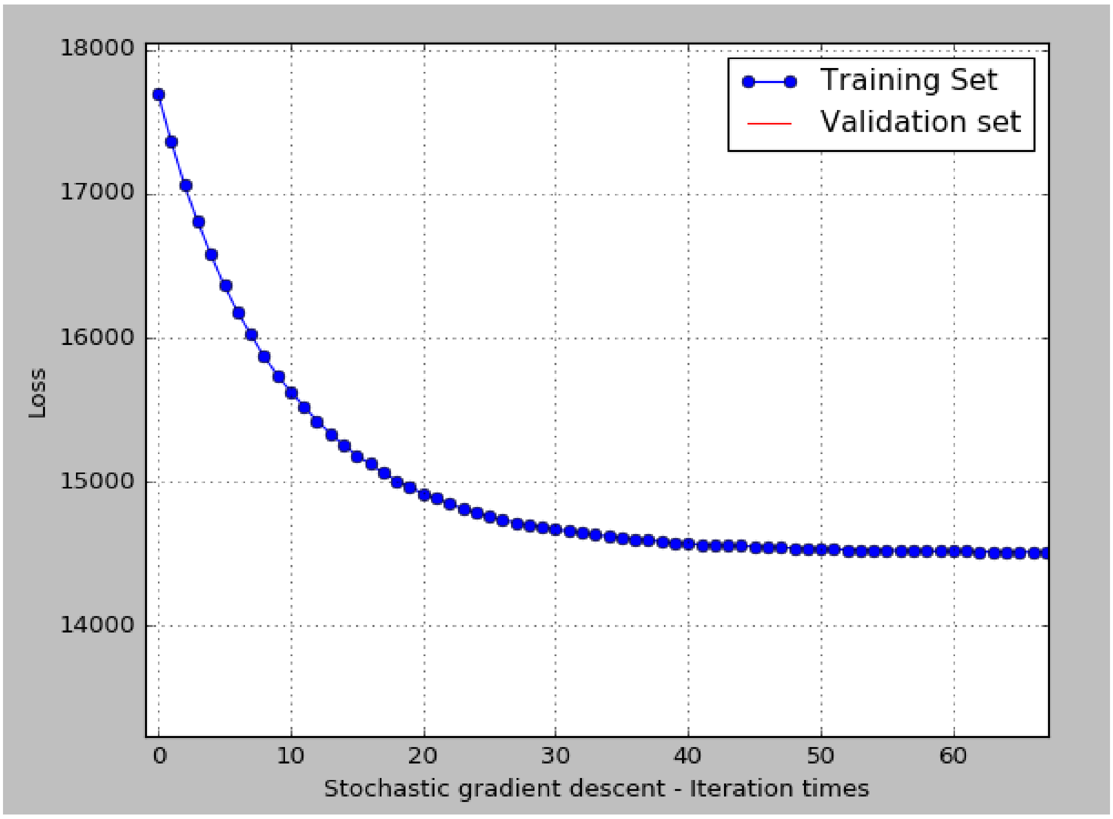
loss 曲线图：

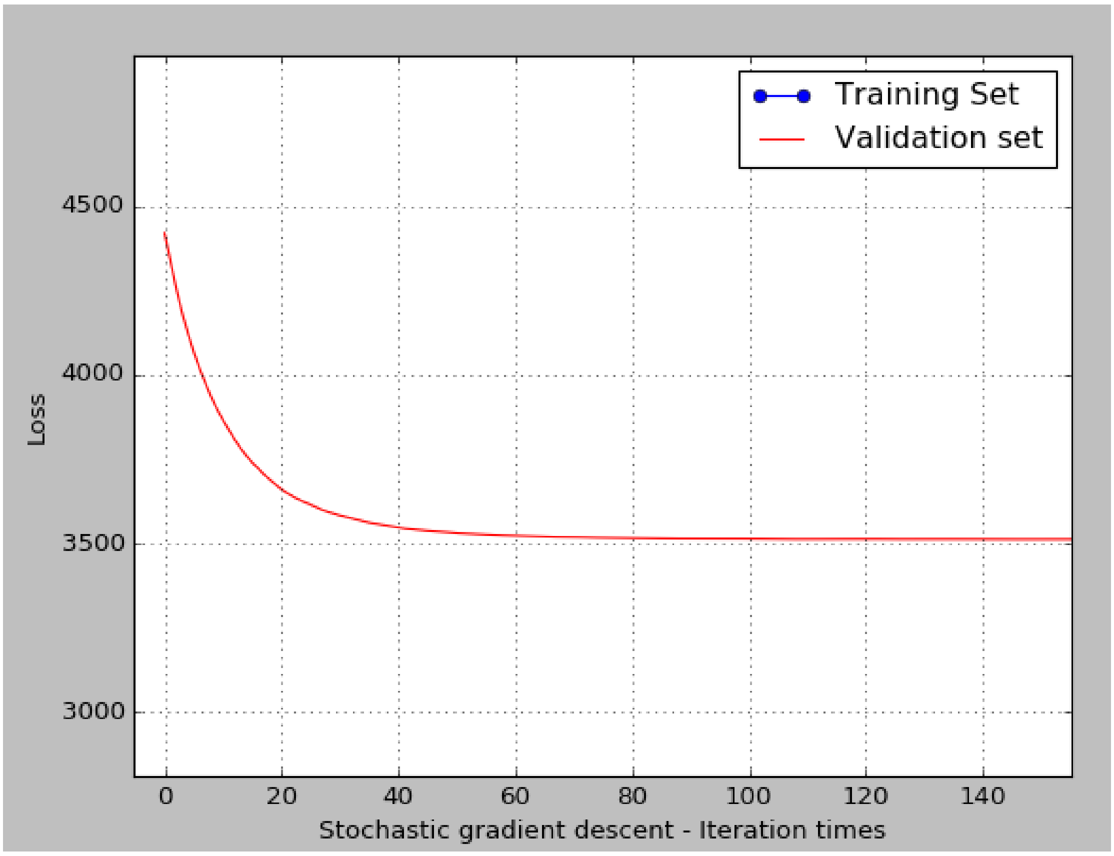
分别如下所示(调试时使用 Pycharm 输出，与 jupyter 输出的图像风格不太一致，在此说明一下)

SGD

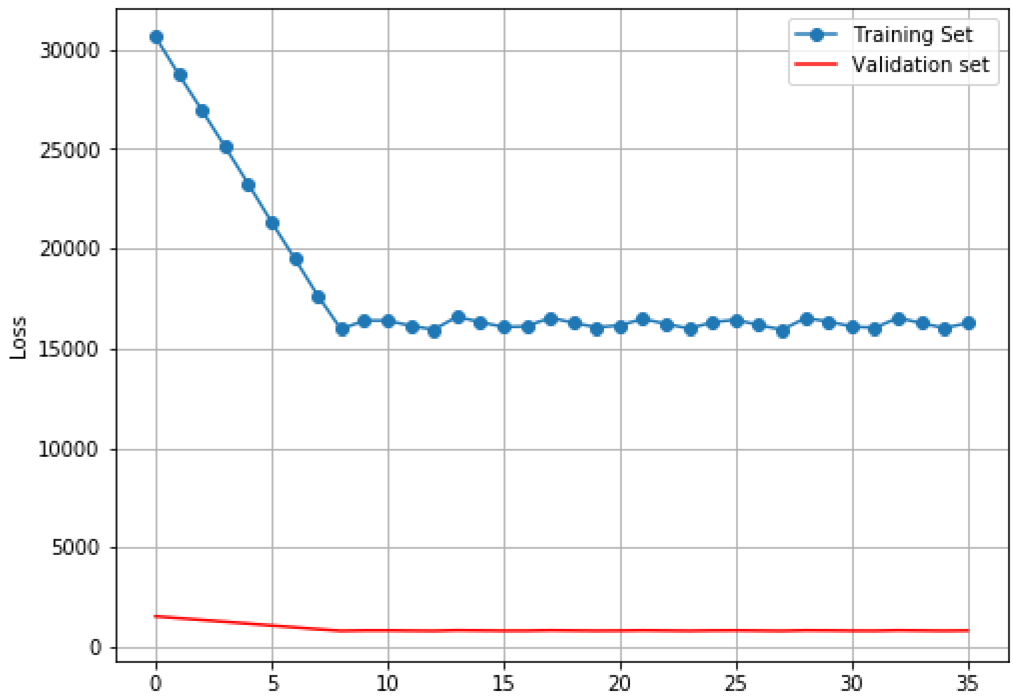
逻辑回归:




18000
17000
16000
15000
14000
Loss
0
10
20
30
40
50
60
Stochastic gradient descent - Iteration times
Training Set
Validation set


4500
4000
3500
3000
Loss
0
20
40
60
80
100
120
140
Stochastic gradient descent - Iteration times
Training Set
Validation set

SVM：



Nesterov accelerated gradient

Nesterov 提出的加速梯度下降（Nesterov’s accelerated gradient）是凸优化的一种最优算法 [1]，其收敛速度可以达到 $O(1/t^2)$，而不是$O(1/t)$。尽管在使用 Caffe 训练深度神经网络时很难满足$O(1/t^2)$收敛条件（例如，由于非平滑 non-smoothness 、非凸 non-convexity），但实际中 NAG 对于某些特定结构的深度学习模型仍是一个非常有效的方法[2]。

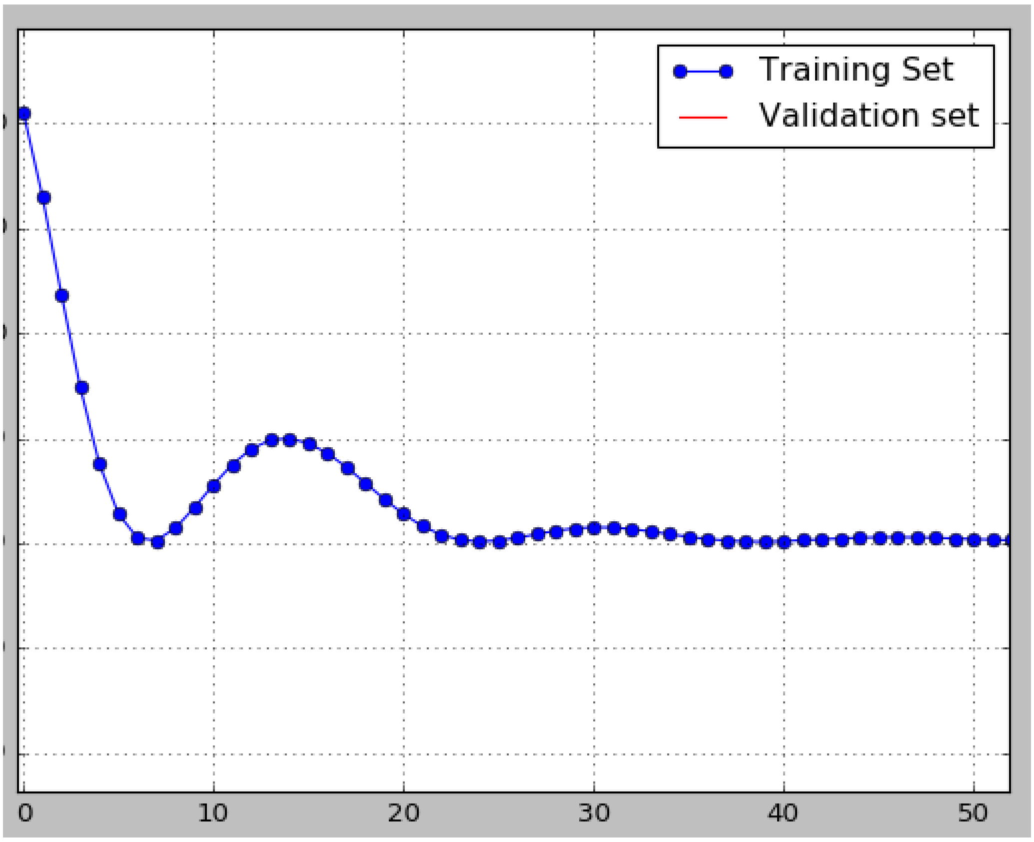
权重 weight 更新参数与随机梯度下降（Stochastic gradient）非常相似：

$$V_{t+1} = \mu V_t - \alpha \nabla L(W_t + \mu V_t)$$

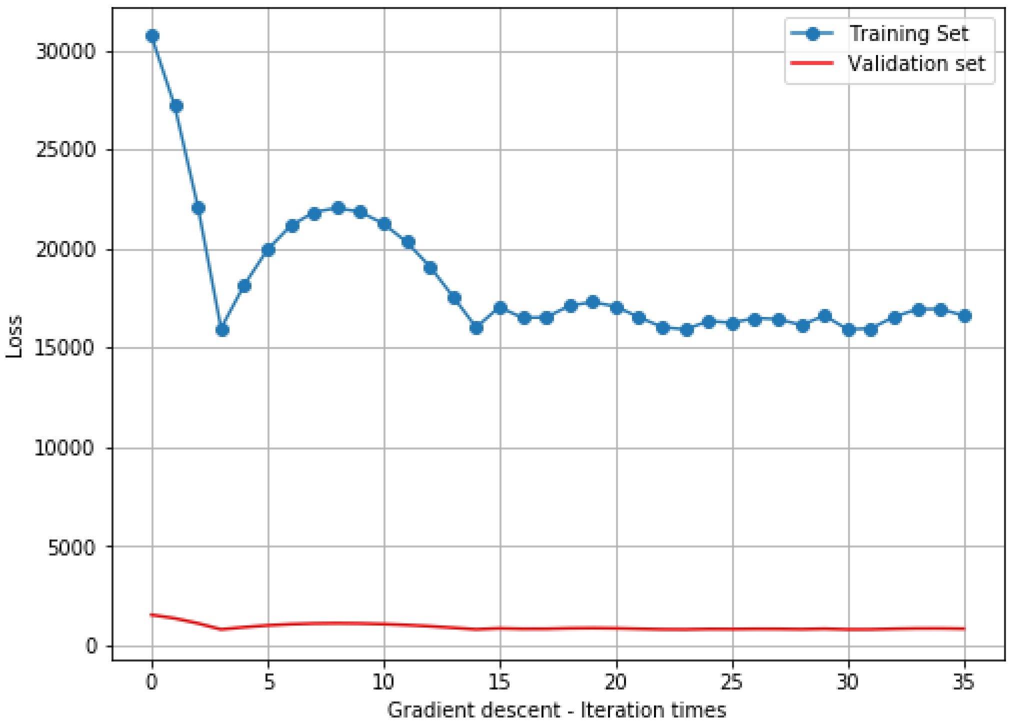
$$W_{t+1} = W_t + V_{t+1}$$

与 SGD 的不同之处在于梯度$\nabla L(W)$项中取值不同：在 NAG 中， 我们取当前权重和动量之和的梯度$\nabla L(W_t + \mu V_t)$；在 SGD 中，只是简单的计算当前权重的动量$\nabla L(W_t)$。

逻辑回归：

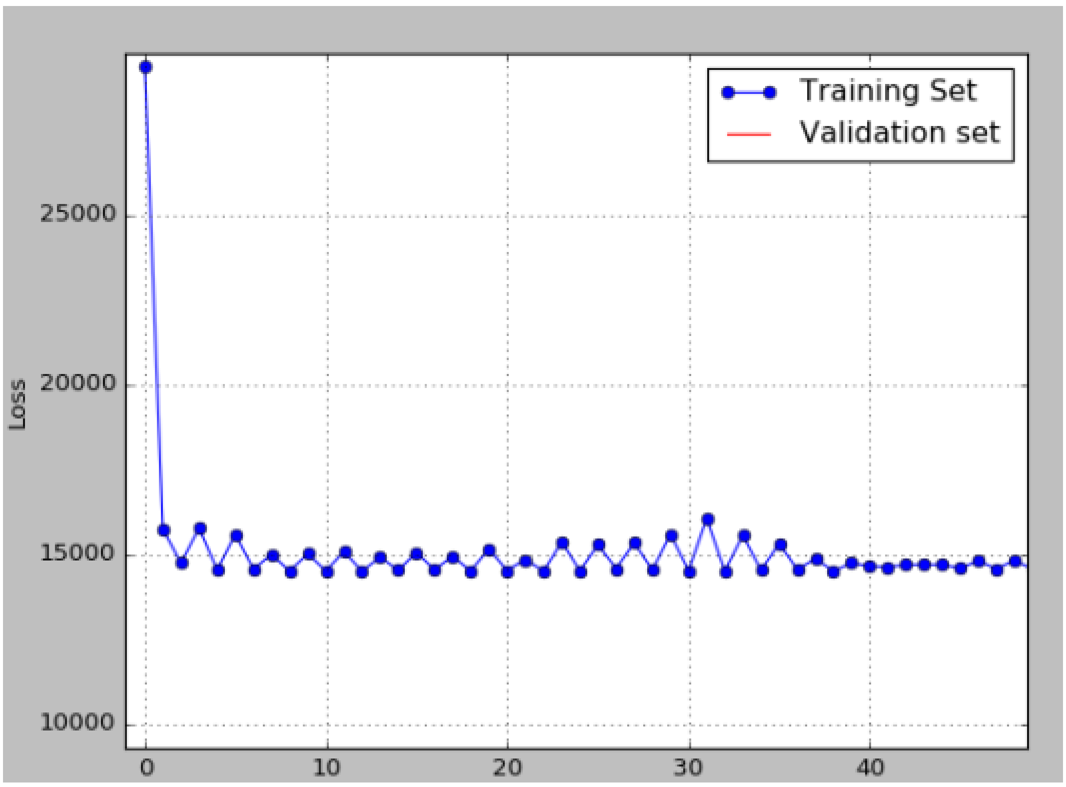


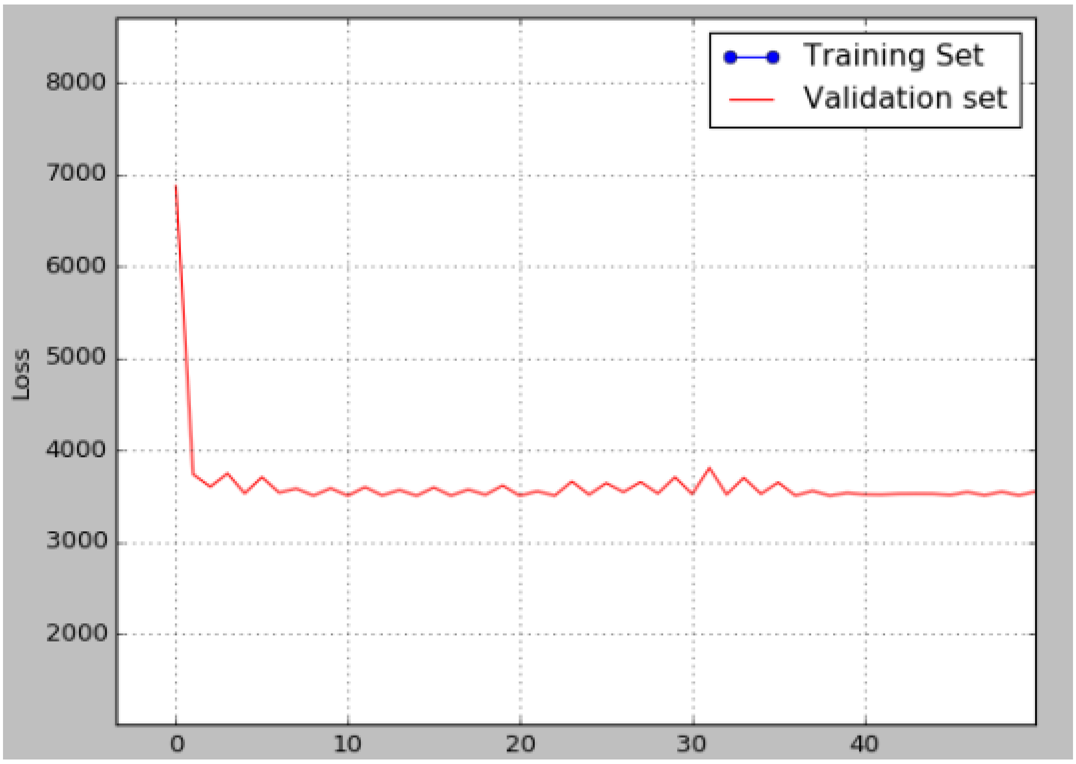
SVM：



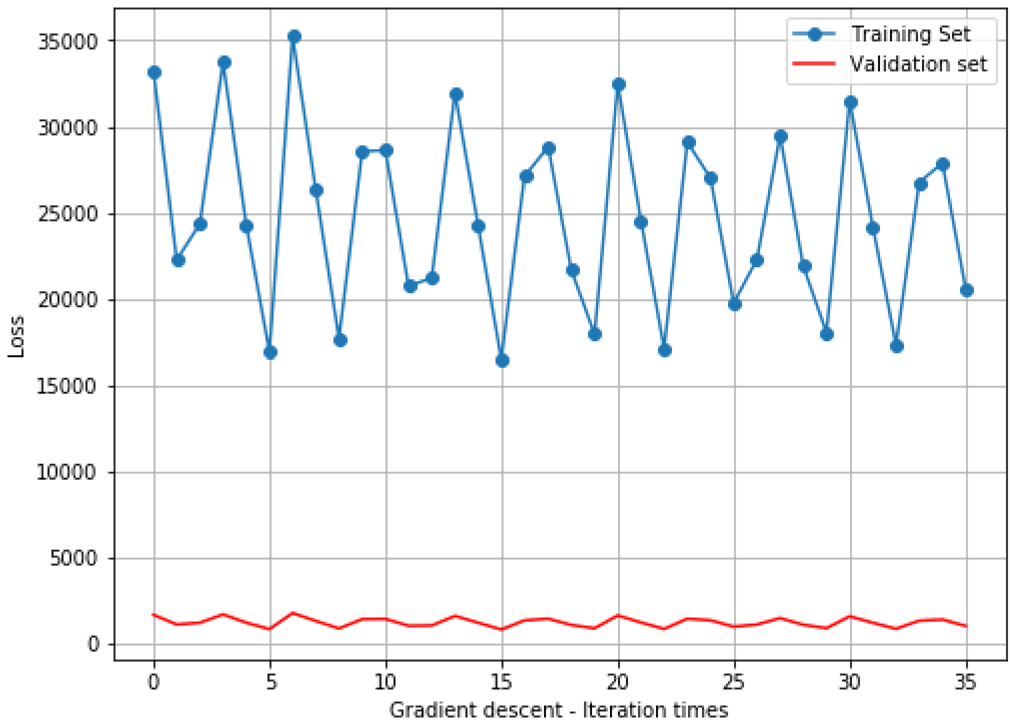
RMSprop

逻辑回归：




Training Set
Validation set
Loss
8000
7000
6000
5000
4000
3000
2000
0
10
20
30
40

SVM:


Training Set
Validation set
Loss
35000
30000
25000
20000
15000
10000
5000
0
5
10
15
20
25
30
35
Gradient descent - Iteration times

RMSprop可以算作Adadelta的一个特例：

当 $\rho = 0.5$ 时， $E|g^2|_t = \rho * E|g^2|_{t-1} + (1-\rho) * g_t^2$ 就变为了求梯度平方和的平均数。

如果再求根的话，就变成了RMS(均方根)：

$$RMS|g|_t = \sqrt{E|g^2|_t + \epsilon}$$

此时，这个RMS就可以作为学习率 η 的一个约束：

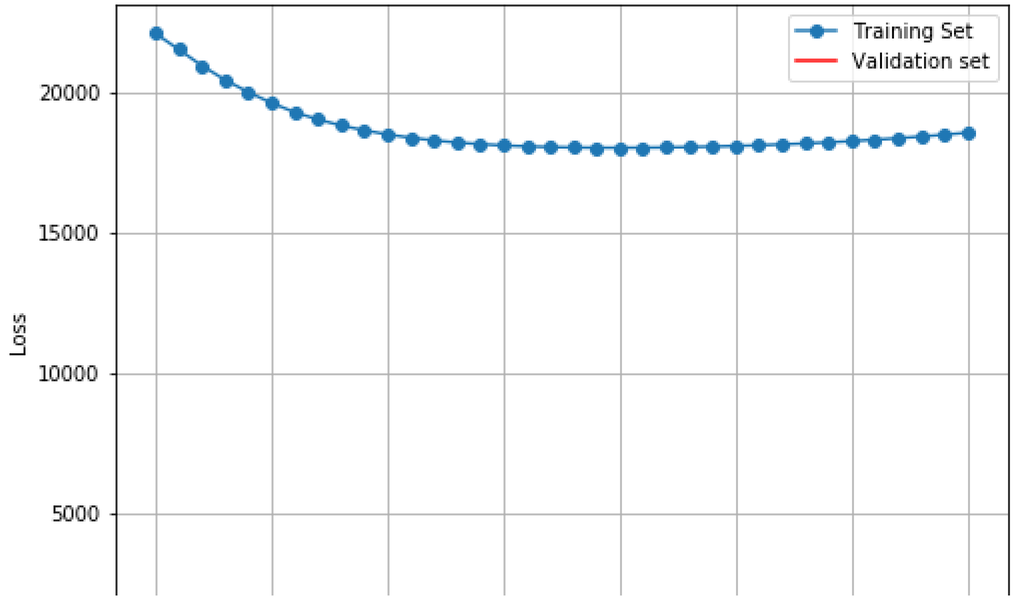
$$\Delta x_t = -\frac{\eta}{RMS|g|_t} * g_t$$

特点：

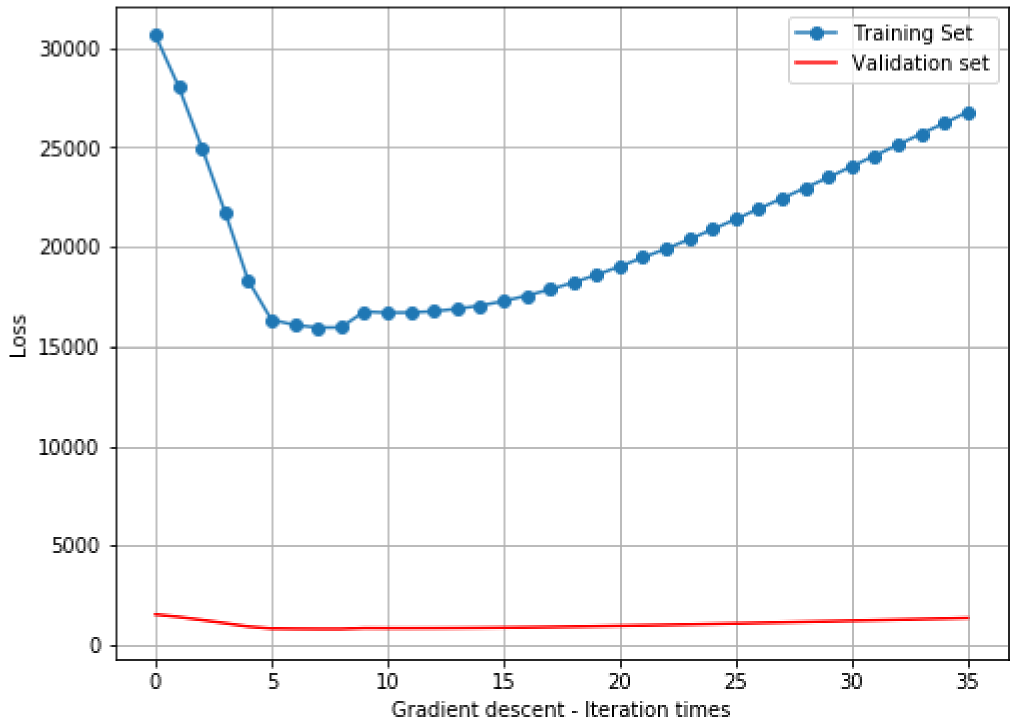
- 其实RMSprop依然依赖于全局学习率
- RMSprop算是Adagrad的一种发展，和Adadelta的变体，效果趋于二者之间
- 适合处理非平稳目标 - 对于RNN效果很好

4. Adam

逻辑回归：



SVM：



Adaptive Moment Estimation (自适应矩估计 Adam)是另外一种为每个参数提供自适应学习率的方法。

同 RMSprop、Adadelta 相比，Adam 在对梯度平方(二阶矩)估计的基础上增加了对梯度(一阶矩)的估计，使得整个学习过程更加稳定。

$$E[g]_t = \beta_1 E[g]_{t-1} + (1-\beta_1) g_t \qquad \hat{E[g]}_t = \frac{E[g]_t}{1-\beta_1}$$

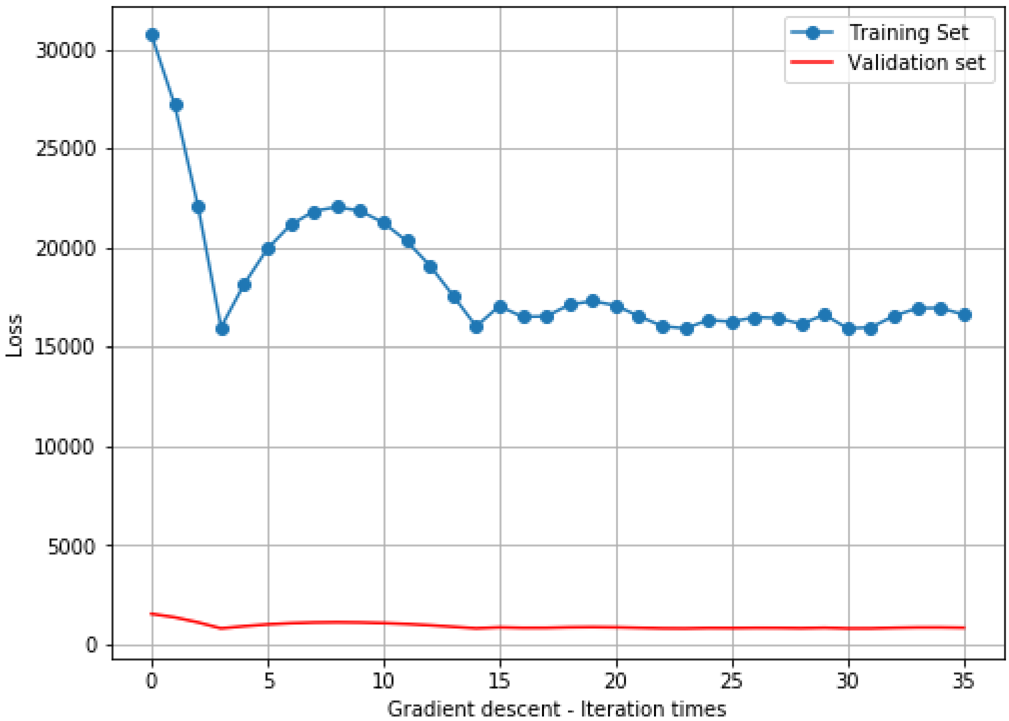
$$E[g^2]_t = \beta_2 E[g^2]_{t-1} + (1-\beta_2) g_t^2 \qquad \hat{E[g^2]}_t = \frac{E[g^2]_t}{1-\beta_2}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{E[g^2]}_t + \epsilon}} \hat{E[g]}_t$$

通常 β_1, β_2 分别被设置为 0.9 和 0.999.

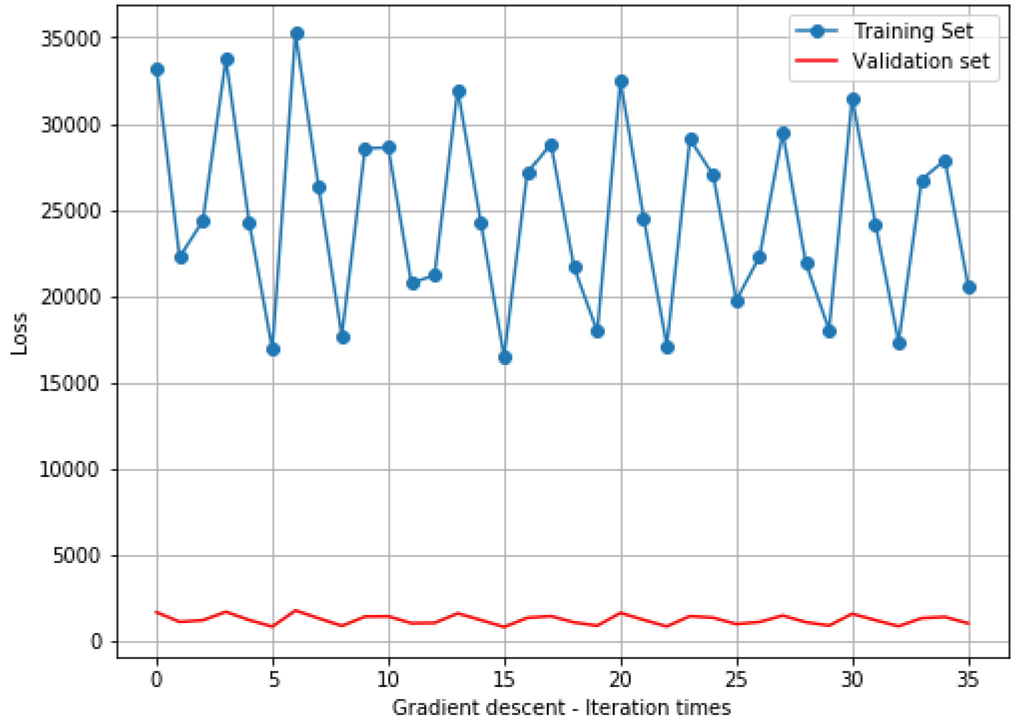
11.实验结果分析:

NAG



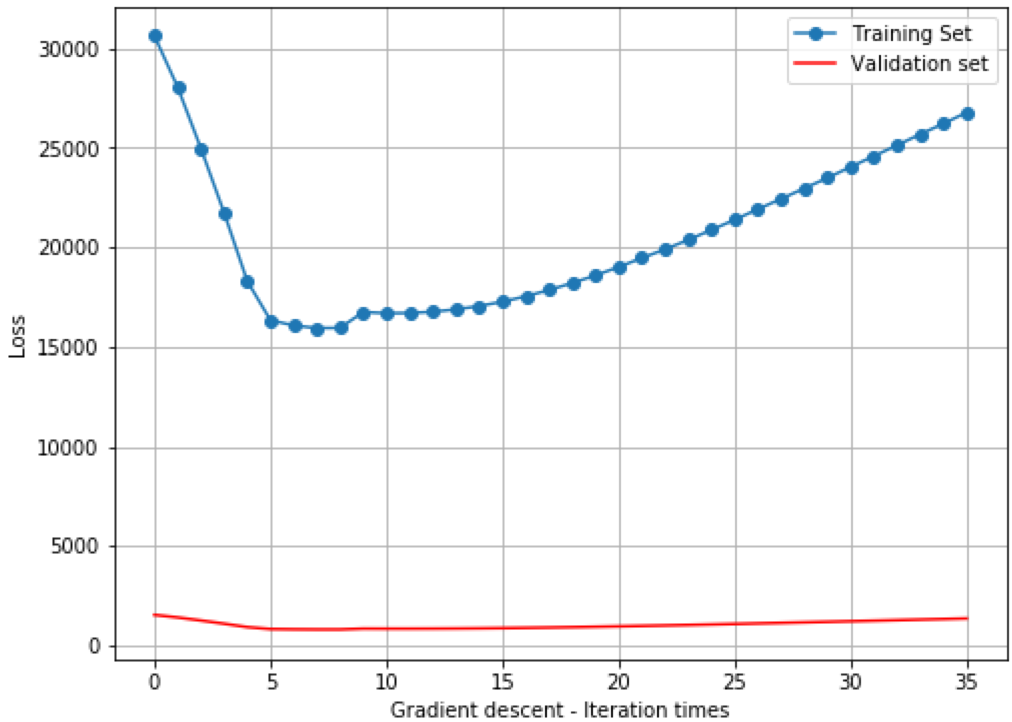
NAG 学习参数的更新可以自动调整学习速度，如图所示，刚开始速度加快(还没收敛)，到后来的震荡程度越来越小，在实际应用中能够增大跳出局部最优值的可能性。

RMSProp



RMSprop 无论是否达到收敛点，都会不断地震荡，即不断更新方向寻找更优的收敛点，越到后面可以明显看出越接近全局最优点。

Adam



Adam 在前期大幅加快了随机梯度下降的速度，后期则放慢速度，若陷入局部最优点之后就比较难以出来，从图像可见，已经往错误的梯度方向继续进行。

12.对比逻辑回归和线性分类的异同点：

相同点：

第一，LR 和 SVM 都是分类算法。

第二，如果不考虑核函数，LR 和 SVM 都是线性分类算法，也就是说他们的分类决策面都是线性的。LR 也是可以用核函数的。原始的 LR 和 SVM 都是线性分类器

第三，LR 和 SVM 都是监督学习算法。

第四，LR 和 SVM 都是判别模型。

判别模型会生成一个表示 P(Y|X)的判别函数（或预测模型），而生成模型先计算联合概率 p(Y,X)然后通过贝叶斯公式转化为条件概率。简单来说，在计算判别模型时，不会计算联合概率，而在计算生成模型时，必须先计算联合概率。或

者这样理解：生成算法尝试去找到底这个数据是怎么生成的（产生的），然后再对一个信号进行分类。基于你的生成假设，那么那个类别最有可能产生这个信号，这个信号就属于那个类别。判别模型不关心数据是怎么生成的，它只关心信号之间的差别，然后用差别来简单对给定的一个信号进行分类。常见的判别模型有：KNN、SVM、LR，常见的生成模型有：朴素贝叶斯，隐马尔可夫模型。当然，这也是为什么很少有人问你朴素贝叶斯和 LR 以及朴素贝叶斯和 SVM 有什么区别（哈哈，废话是不是太多）。

第五，LR 和 SVM 在学术界和工业界都广为人知并且应用广泛。

LR 和 SVM 的不同点：

第一，本质上是其 loss function 不同。

第二，支持向量机只考虑局部的边界线附近的点，而逻辑回归考虑全局（远离的点对边界线的确定也起作用）。

支持向量机改变非支持向量样本并不会引起决策面的变化

逻辑回归中改变任何样本都会引起决策面的变化

线性 SVM 不直接依赖于数据分布，分类平面不受一类点影响；LR 则受所有数据点的影响，如果数据不同类别 strongly unbalance，一般需要先对数据做 balancing。

第三，在解决非线性问题时，支持向量机采用核函数的机制，而 LR 通常不采用核函数的方法。

第四，线性 SVM 依赖数据表达的距离测度，所以需要对数据先做 normalization，LR 不受其影响

第五，SVM 的损失函数就自带正则(损失函数中的 1/2||w||^2 项），

13.实验总结：

1.通过本次实验，既掌握了随机梯度下降的方法（在实际应用中采用随机 mini-batch 的方法进行梯度的更新项计算，在本次实验中采取样本数量的 1/8 进

行随机更新，详情可见代码），又尝试了很多不同的学习参数更新方法，比如 RMSprop, Adam, NAG,每一种方法都有优劣，在实践的过程中，通过对 Loss 图像的分析，进一步领悟了各种学习参数更新方法的特性，有助于以后机器学习的深入开展。

2.本次实验在实验一的基础上对代码进行了效率上的提高，实验一采用是计算方式是 **python** 自带的 **list**，效率不高，实验二和实验三开始全部使用 **numpy** 的计算，例如矩阵的计算 **numpy.dot(x,y)**，还有 **log** 函数的计算 **numpy.log** 等，在一定程度上减少了循环的次数，可以达到一次高效计算实现目的的效果。

3.进一步总结了 **LR** 和 **SVM** 的异同，更深入地思考了关于不同 **Loss** 函数之间的对比(例如逻辑回归的方差 **loss** 函数和对数 **loss** 函数的优劣)，对于以后 **loss** 函数的选择具有重要的意义，同时加强了对大规模数据处理的分类能力。