

Mllib 算法



東北大學
Northeastern University



HORIZON
昊宸科技



1 分类& 回归

2 协同过滤

3 降维

4 评估

5 模型导入导出

6 优化

- Classification and regression
 - linear models (SVMs, logistic regression, linear regression)
 - naive Bayes
 - decision trees
 - ensembles of trees (Random Forests and Gradient-Boosted Trees)
 - isotonic regression
- Collaborative filtering
 - alternating least squares (ALS)
- Clustering
 - k-means
 - Gaussian mixture
 - power iteration clustering (PIC)
 - latent Dirichlet allocation (LDA)
 - bisecting k-means
 - streaming k-means
- Dimensionality reduction
 - singular value decomposition (SVD)
 - principal component analysis (PCA)
- Feature extraction and transformation
- Frequent pattern mining
 - FP-growth
 - association rules
 - PrefixSpan
- Evaluation metrics
- PMML model export
- Optimization (developer)
 - stochastic gradient descent
 - limited-memory BFGS (L-BFGS)

Classification and Regression



- Linear models
 - classification (SVMs, logistic regression)
 - linear regression (least squares, Lasso, ridge)
- Decision trees
- Ensembles of decision trees
 - random forests
 - gradient-boosted trees
- Naive Bayes
- Isotonic regression



Linear Methods - spark.mllib

- Mathematical formulation
 - Loss functions
 - Regularizers
 - Optimization
- Classification
 - Linear Support Vector Machines (SVMs)
 - Logistic regression
- Regression
 - Linear least squares, Lasso, and ridge regression
 - Streaming linear regression
- Implementation (developer)

- 线性支持向量机([Linear Support Vector Machines \(SVMs\)](#))

分类超平面，核函数，函数Margin，几何Margin，核函数，极大似然估计，拉格朗日乘子法

- 逻辑回归（[Logistic regression](#)）

逻辑损失函数（loss func），正则化参数

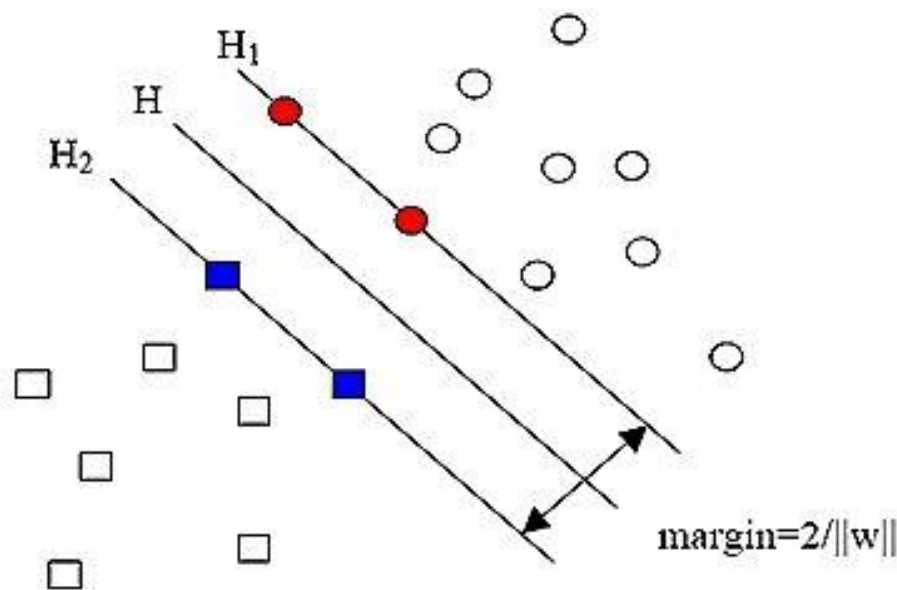
分类—SVM(Support Vector Machine)



➤ 线性支持向量机(Linear Support Vector Machines (SVMs))

在图中中间这条线就是SVM 算法的分类超平面，分类超平面力争做到距离两边最近的点最远。这样分类的信心就最大

margin: 分类间隔



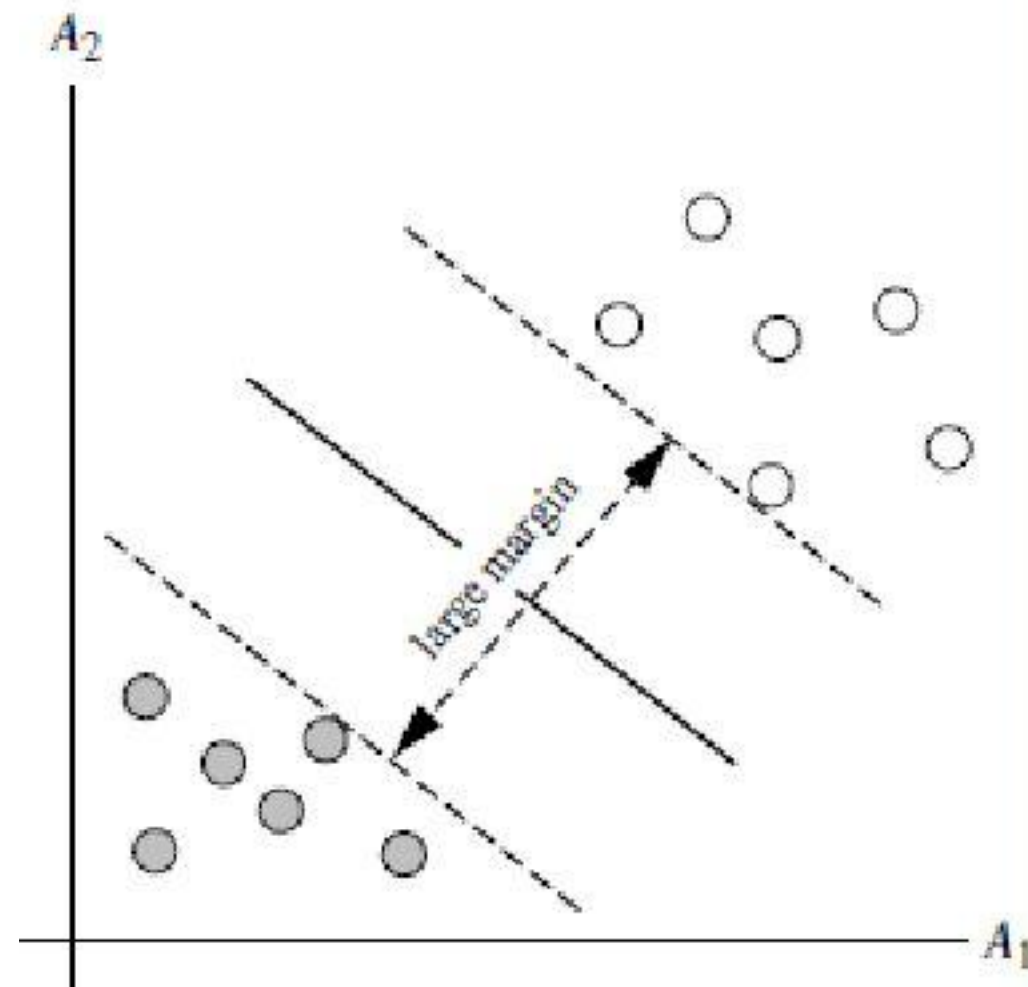
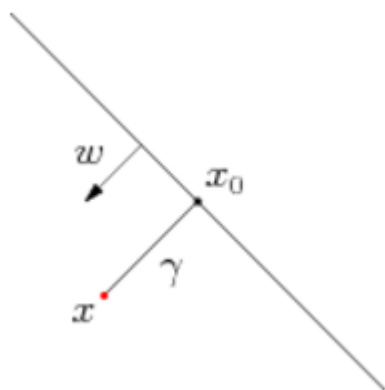
线性可分情况下的最优分类线

分类—SVM(Support Vector Machine)

➤ 线性支持向量机(Linear Support Vector Machines (SVMs))

functional Margin $\hat{\gamma} = y(w^T x + b) = yf(x)$

geometric Margin



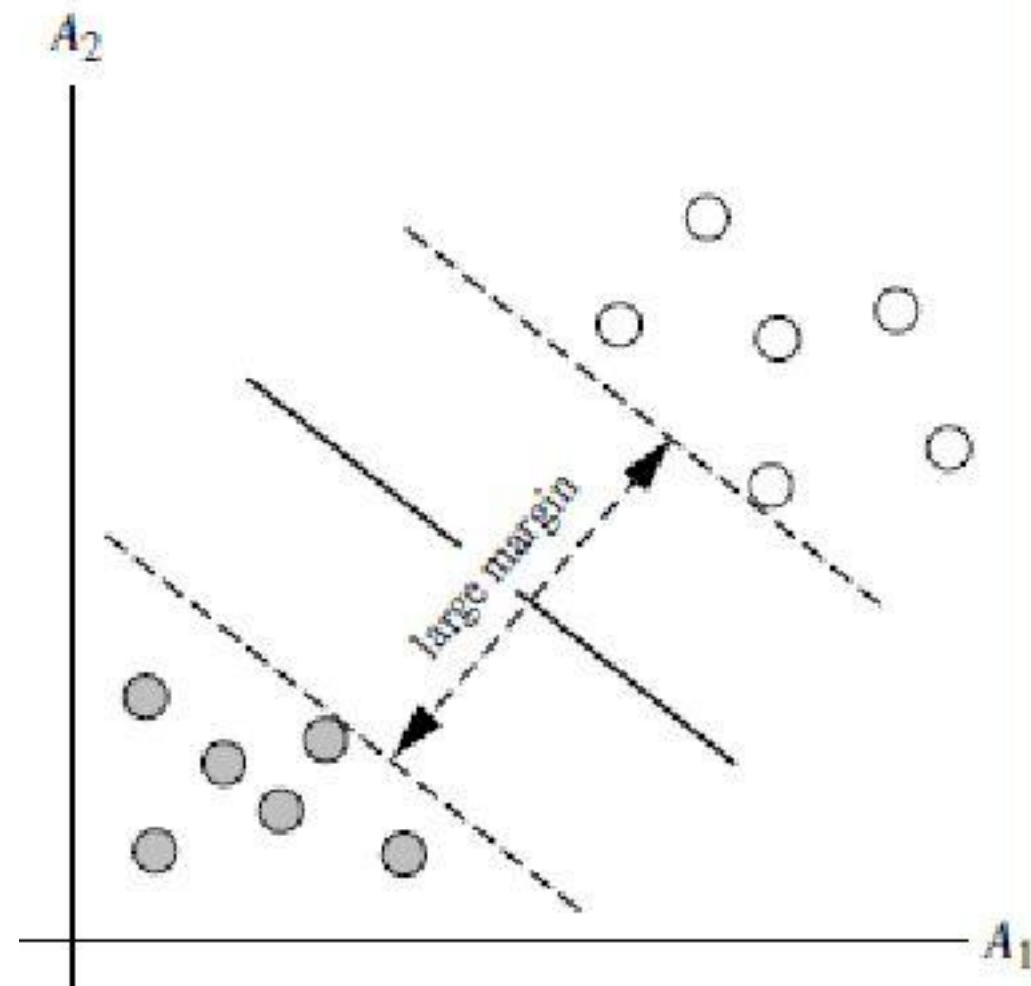
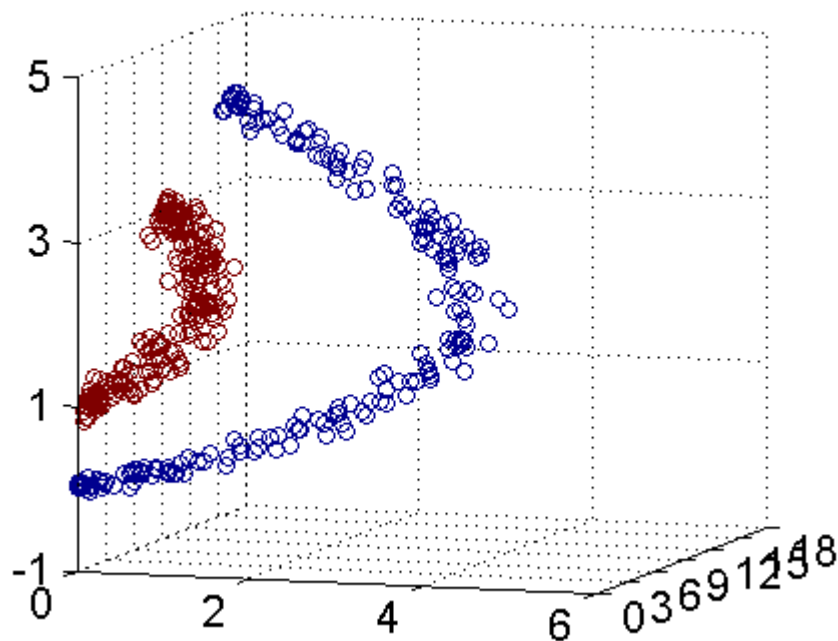
分类—SVM(Support Vector Machine)

➤ 线性支持向量机(Linear Support Vector Machines (SVMs))

优化最大间隔分类器，用到了拉格朗日乘子法；

KKT 条件

核函数用于将SVM 算法用于多维空间中的分类

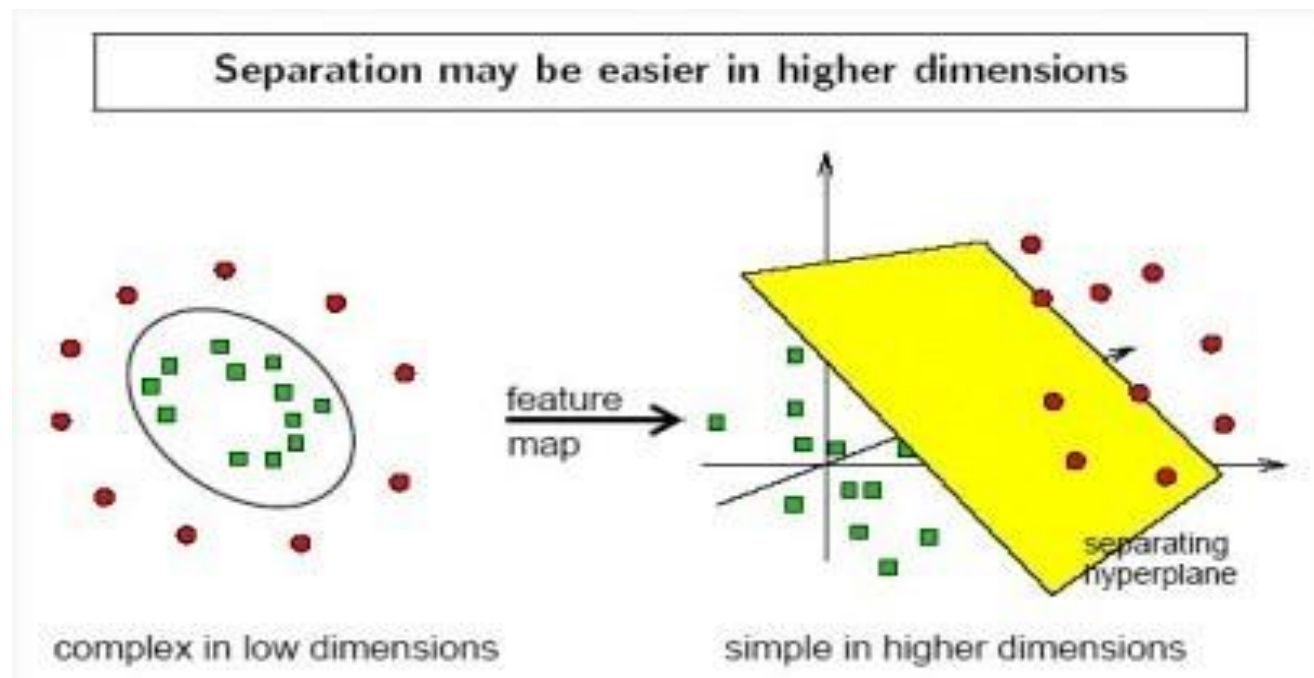


分类—SVM(Support Vector Machine)

➤ 常用的核函数多项式核，多项式核，高斯核

实际中，我们会经常遇到线性不可分的样例，此时，我们的常用做法是把样例特征映射到高维空间中去(如图所示，映射到高维空间后，相关特征便被分开了，也就达到了分类的目的)；

但进一步，如果凡是遇到线性不可分的样例，一律映射到高维空间，那么这个维度大小是会高到可怕的，此时，核函数就隆重登场了，核函数的价值在于它虽然也是讲特征进行从低维到高维的转换，但核函数绝就绝在它事先在低维上进行计算，而将实质上的分类效果表现在了高维上，也就如上文所说的避免了直接在高维空间中的复杂计算。

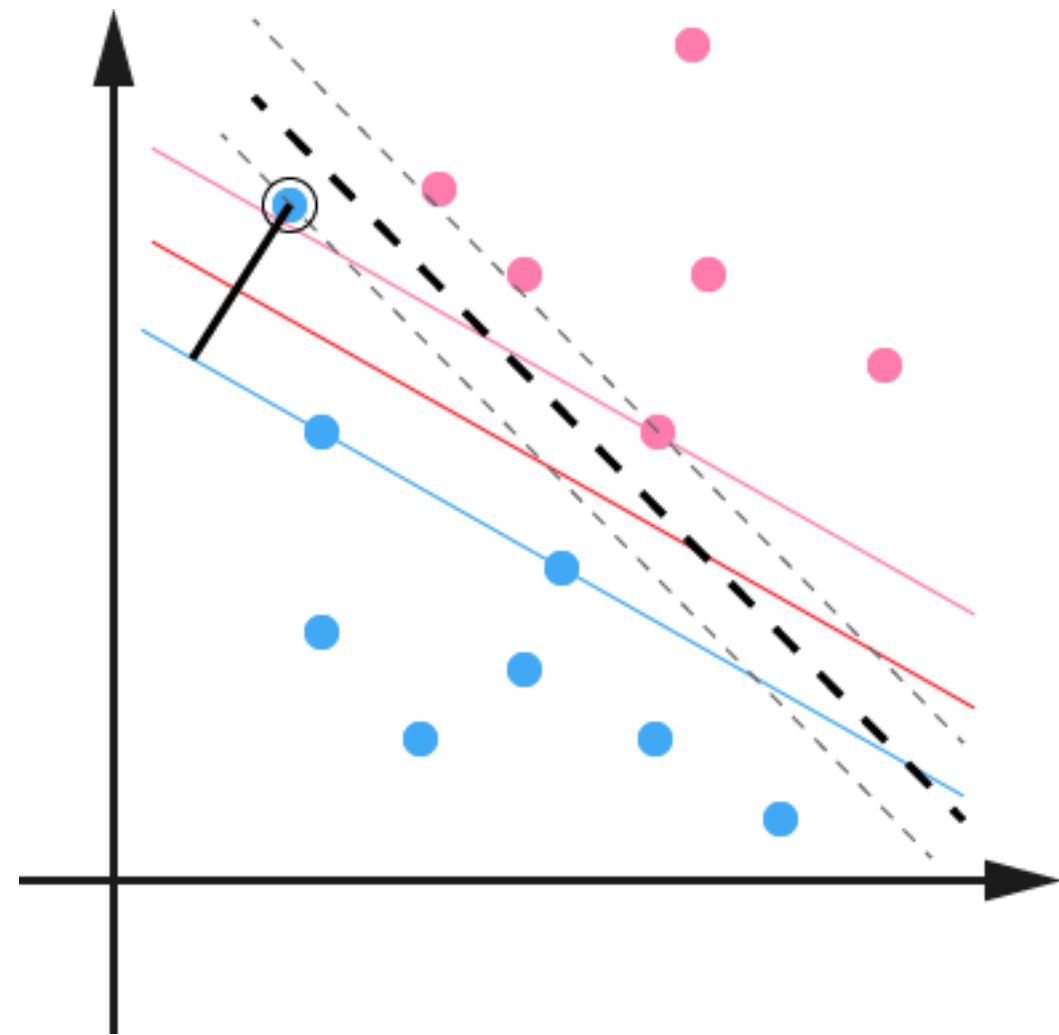


分类—SVM(Support Vector Machine)



➤ 使用松弛变量处理 outliers 方法

使用SVM前提我们假定数据是线性可分的，即我们可以找到一个可行的超平面将数据完全分开。后来为了处理非线性数据，使用 Kernel 方法对原来的线性 SVM 进行推广，使得非线性的情况也能处理。虽然通过映射 将原始数据映射到高维空间之后，能够线性分隔的概率大大增加，但是对于某些情况还是很难处理。



➤ 使用松弛变量处理 outliers 方法

例如可能并不是因为数据本身是非线性结构的，而只是因为数据有噪音。对于这种偏离正常位置很远的数据点，我们称之为 outlier，在我们原来的 SVM 模型里，outlier 的存在有可能造成很大的影响，因为超平面本身就是只有少数几个 support vector 组成的，如果这些 support vector 里又存在 outlier 的话，其影响就很大了。

用黑圈圈起来的那个蓝点是一个 outlier，它偏离了自己原本所应该在那个半空间，如果直接忽略掉它的话，原来的分隔超平面还是挺好的，但是由于这个 outlier 的出现，导致分隔超平面不得被挤歪了，变成途中黑色虚线所示（这只是一个示意图，并没有严格计算精确坐标），同时 margin 也相应变小了。当然，更严重的情况是，如果这个 outlier 再往右上移动一些距离的话，我们将无法构造出能将数据分开的超平面来。

为了处理这种情况，SVM 允许数据点在一定程度上偏离一下超平面。例如上图中，黑色实线所对应的距离，就是该 outlier 偏离的距离，如果把它移动回来，就刚好落在原来的超平面上，而不会使得超平面发生变形了。

换言之，在有松弛的情况下 outlier 点也属于支持向量

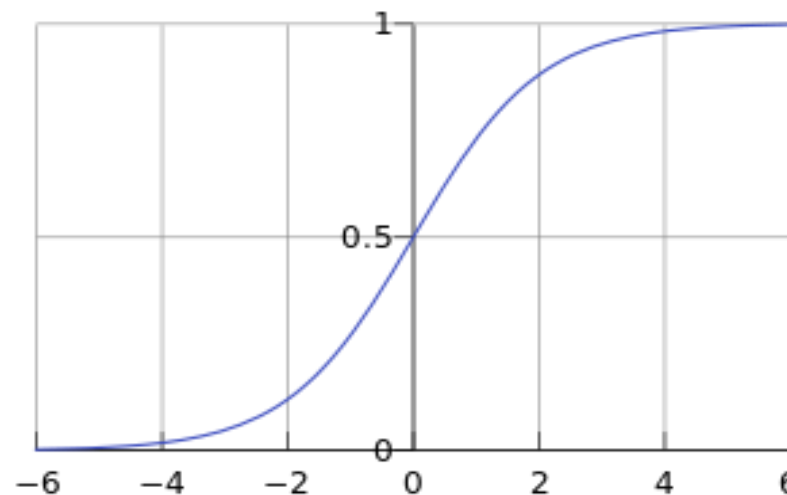
分类/回归—逻辑回归(LogisticRegression)



- 首先，Logistic回归虽然名字里带“回归”，但是它实际上是一种分类方法，主要用于两分类问题，利用Logistic函数（或称为Sigmoid函数），自变量取值范围为 $(-\infty, \infty)$ ，因变量的取值范围为 $(0,1)$ ，函数形式为：

$$g(z) = \frac{1}{1 + e^{-z}}, \text{ 其中 } z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n = \sum_{i=1}^n \theta_i x_i = \theta^T x$$

- 由于sigmoid函数的定义域是 $(-\infty, +\infty)$ ，而值域为 $(0, 1)$ 。因此最基本的LR分类器适合于对两分类（类0，类1）目标进行分类。Sigmoid函数是个很漂亮的“S”形，如下图所示：



分类/回归—逻辑回归(LogisticRegression)



- LR分类器(Logistic Regression Classifier)目的就是从训练数据特征学习出一个0/1分类模型--这个模型以样本特征的线性组合 $\sum_{i=1}^n \theta_i$ 作为自变量，使用logistic函数将因变量映射到(0,1)上。因此LR分类器的求解就是求解一组权值 $\theta_0, \theta_1, \theta_2, \dots, \theta_n$ (θ_0 为常数)，并代入Logistic函数构造出一个预测函数：

$$h_{\theta}(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

- 函数的值表示结果为1的概率，就是特征属于y=1的概率。因此对于输入x分类结果为类别1和类别0的概率分别为：
- $$P(y = 1 | x; \theta) = h_{\theta}(x)$$
- $$P(y = 0 | x; \theta) = 1 - h_{\theta}(x)$$

分类/回归—逻辑回归(LogisticRegression)



➤ 当我们要判别一个新来的特征属于哪个类时，按照下式求出一个z值：

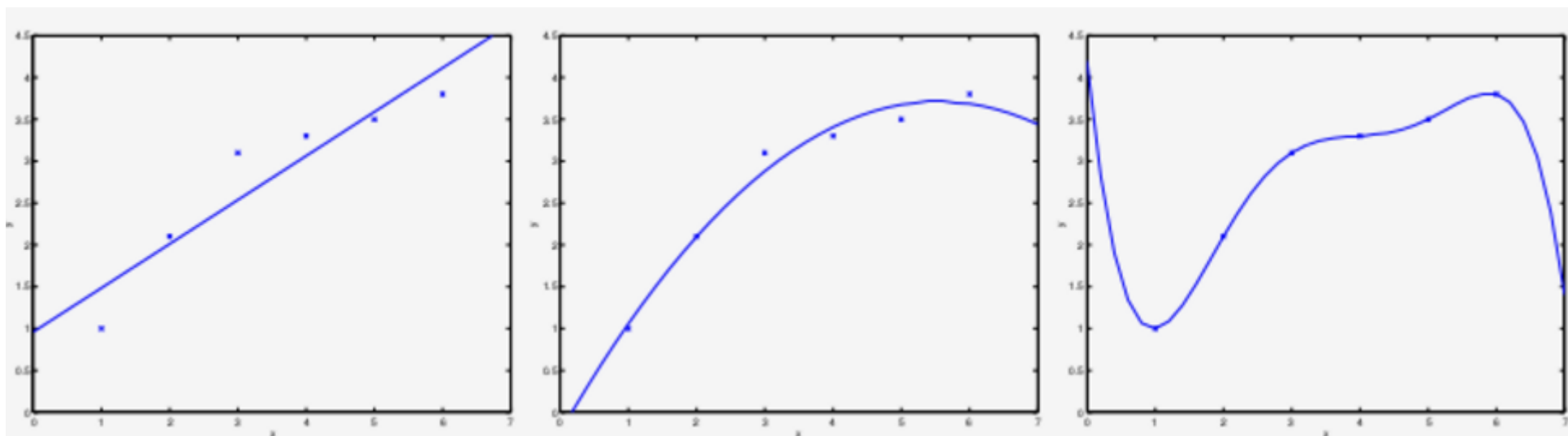
➤ $z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n = \sum_{i=1} \theta_i x_i = \theta^T x$ (x_1, x_2, \dots, x_n 是某样本数据的各个特征，维度为 n) 进而求出 $h_\theta(x)$ --- 若大于0.5就是y=1的类，反之属于y=0类。（注意：这里依然假设统计样本是均匀分布的，所以设阈值为0.5）。LR分类器的这一组权值如何求得的呢？这就需要涉及到极大似然估计MLE和优化算法的概念了，数学中最优化算法常用的就是**梯度上升（下降）算法**。

分类/回归—逻辑回归(LogisticRegression)



➤ 拟合

对于以下三种拟合，第一种用一次函数拟合，第二种用二次函数拟合，第三种用7次函数。显然，二次函数的拟合比较恰当，一次函数为欠拟合，7次函数为过拟合。



监督机器学习问题无非就是“minimize your error while regularizing your parameters”，也就是在规则化参数的同时最小化误差。最小化误差是为了让我们的模型拟合我们的训练数据，而规则化参数是防止我们的模型过分拟合我们的训练数据。

分类/回归—逻辑回归(LogisticRegression)损失函数



➤ 一般来说，监督学习可以看做最小化下面的目标函数。

$$w^* = \arg \min_w \sum_i L(y_i, f(x_i; w)) + \lambda \Omega(w)$$

其中，第一项 $L(y_i, f(x_i; w))$ 衡量我们的模型（分类或者回归）对第 i 个样本的预测值 $f(x_i; w)$ 和真实的标签 y_i 之前的误差。因为我们的模型是要拟合我们的训练样本的嘛，所以我们要求这一项最小，也就是要求我们的模型尽可能的拟合我们的训练数据。但正如上面所言，我们不仅要保证训练误差最小，我们更希望我们的模型测试误差小，所以我们需要加上第二项，也就是对参数 w 的规则化函数 $\Omega(w)$ 去约束我们的模型尽可能的简单。

对于第一项Loss函数，如果是Square loss，那就是最小二乘了；如果是Hinge Loss，那就是著名的SVM了；如果是exp-Loss，那就是牛逼的 Boosting了；如果是log-Loss，那就是Logistic Regression了；还有等等。不同的loss函数，具有不同的拟合特性，这个也得就具体问题具体分析。

分类/回归—逻辑回归(LogisticRegression)损失函数



- 一般来说，监督学习可以看做最小化下面的目标函数。

$$w^* = \arg \min_w \sum_i L(y_i, f(x_i; w)) + \lambda \Omega(w)$$

	正则化因子 $R(w)$
零（未正则化）	0
L2 范数	$\frac{1}{2} \ \mathbf{w}\ _2^2$
L1 范数	$\ \mathbf{w}\ _1$

- Spark Mllib 针对这些算法实现了随机梯度下降(Stochastic Gradient Descent)的版本,这些算法需要传递正则化参数 (regparam : none L1 , L2)
- SVMWithSGD
- LogisticRegressionWithLBFGS
- LogisticRegressionWithSGD
- LinearRegressionWithSGD
- RidgeRegressionWithSGD
- LassoWithSGD

Collaborative Filtering



➤ 协同过滤算法，ALS(Alternating least squares)

- Classification and regression
 - linear models (SVMs, logistic regression, linear regression)
 - naive Bayes
 - decision trees
 - ensembles of trees (Random Forests and Gradient-Boosted Trees)
 - isotonic regression
- Collaborative filtering
 - alternating least squares (ALS)
- Clustering
 - k-means
 - Gaussian mixture
 - power iteration clustering (PIC)
 - latent Dirichlet allocation (LDA)
 - bisecting k-means
 - streaming k-means
- Dimensionality reduction
 - singular value decomposition (SVD)
 - principal component analysis (PCA)
- Feature extraction and transformation
- Frequent pattern mining
 - FP-growth
 - association rules
 - PrefixSpan
- Evaluation metrics
- PMML model export
- Optimization (developer)
 - stochastic gradient descent
 - limited-memory BFGS (L-BFGS)



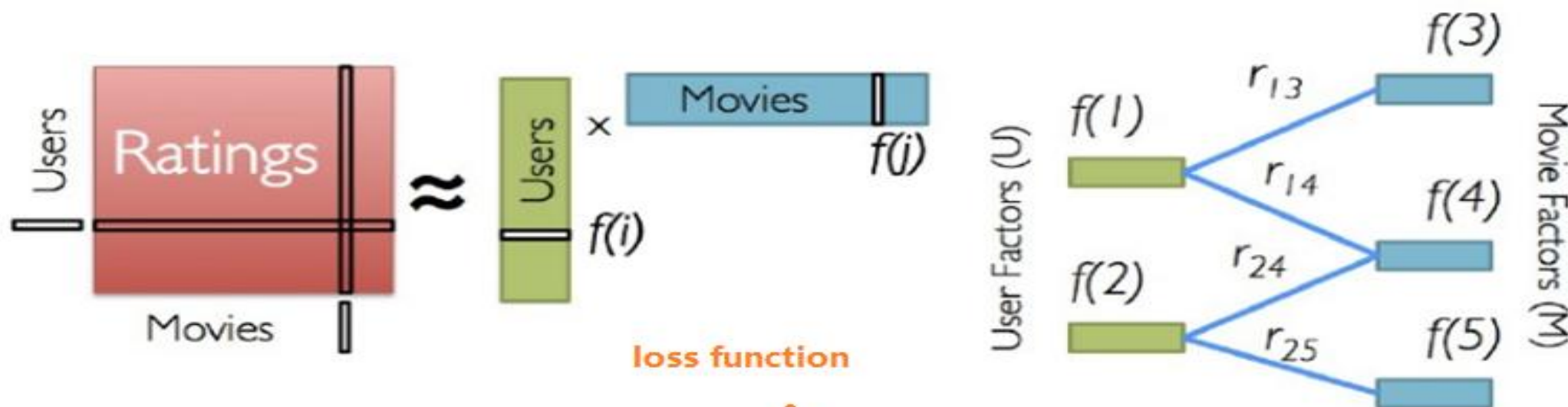
Collaborative Filtering - spark.mllib

- Collaborative filtering
 - Explicit vs. implicit feedback
 - Scaling of the regularization parameter
- Examples
- Tutorial

协同过滤算法 ALS

- 协同过滤算法，ALS(Alternating least squares)，核心思想，把稀疏矩阵拆分小矩阵求解

Low-Rank Matrix Factorization:



loss function

Iterate:

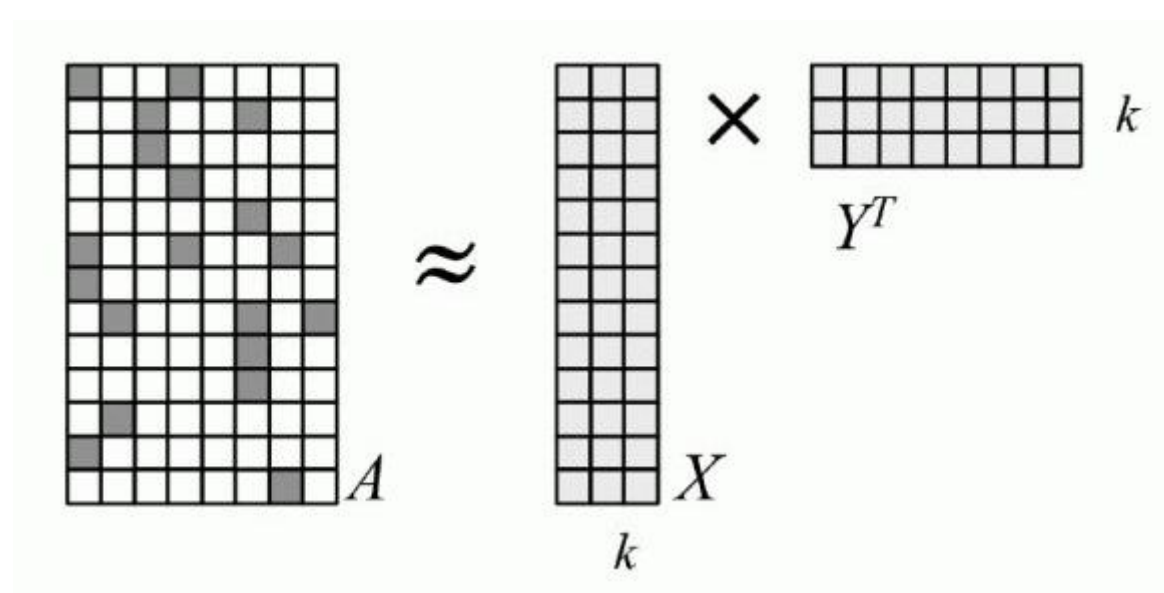
$$f[i] = \arg \min_{w \in \mathbb{R}^d} \sum_{j \in \text{Nbrs}(i)} \left(r_{ij} - w^T f[j] \right)^2 + \lambda ||w||_2^2$$

正则化参数

ALS 矩阵分解思想



- 原始矩阵 A 可能非常稀疏，但乘积 XY^T 是稠密的；两个矩阵分别有一行对应每个用户和每个艺术家，每行的值很少只有 k 个。每个值代表对应模型的一个隐含特征。因此行表示了用户和艺术家怎么关联到这些隐含特征，而隐含特征可能对应偏好或者类别。于是问题就简化为用户-特征矩阵 * 特征-艺术家矩阵，该乘积的结果是整个稠密的用户-艺术家相互关系矩阵的完整估计。



协同过滤算法 ALS—算法参数



- Rank：对应ALS 模型中因子个数，也就是在低阶近似矩阵中的隐含特征个数。因子个数一般越多越好，但会直接影响模型训练和保存的内存开销，尤其是在用户和物品都很多的时候。因此实践中该参数常作为训练效果与系统开销之间的调节参数。通常其合理取值为10~200
- Iterations: 对应运行时的迭代次数。ALS 能确保每次迭代都能降低评级矩阵的重建误差，但一般经少数迭代后ALS 模型便已经收敛为一个比较好的模型，这样大多数情况下没必要迭代太多次（10次左右）
- Lamda: 该参数控制模型的正则化过程，从而控制模型的过拟合情况，其值越高正则化越严厉。该参数的赋值与实际数据的大小，特征、稀疏程度有关。和其他机器学习模型一样，正则化参数应该通过非样本的测试数据进行交叉验证来调整。
- Rating 结果，Mllib 包的ALS 算法接收RDD[Rating]，Rating 结构是<用户，产品，评分>

```
case class Rating @Since("0.8.0") (  
    @Since("0.8.0") user: Int,  
    @Since("0.8.0") product: Int,  
    @Since("0.8.0") rating: Double)
```


协同过滤算法 ALS—构建推荐系统



- 推荐系统属于大规模机器学习，在当今的网站上很容易见到，从社交网络到视频网站，再到在线零售都用到了推荐引擎。
- 数据集描述：user_artist_data.txt 包含 141000个用户和160 万个艺术家，记录了2420万条用户播放艺术家歌曲的信息，其中包括播放次数信息。
- 数据集artist_data.txt 文件中给出了每个艺术家的ID 和对应的名字，请注意记录播放信息时客户端应用提交的是艺术家的名字。名字如果拼写有错误，或使用了非标准的名称，事后才能被发现，比如 “The Smiths” “Smiths , The” 和 “the smiths” 看似代表了不同的艺术家,但其实是指同一个艺术家，因此为了将拼写错误的艺术家 对应到规范的艺术名称，数据集提供了artist_alias.txt

练习—用ALS 算法构建推荐系统符合下列要求



- 1.使用user_artist_data.txt , artist_data.txt , artist_alias.txt 三个数据集构建推荐系统
- 2.过滤掉数据集artist_data.txt中少量非法的行，有些没有制表符，有些不小心加入了换行符。
- 3.将不正确的艺术家ID映射为正确的，例如将ID 为6803336 的艺术家映射为1000010的艺术家，通过广播变量，将artist_alias.txt 读入内存转为map 广播出去
- 4.选择一个用户id，提取该用户收听过的艺术家ID并打印名字
- 5.对上面选择的用户进行推荐
- 6.评价推荐质量，通过AUC曲线
- 7.通过超参数的方式找到最优解

Clustering



- Classification and regression
 - linear models (SVMs, logistic regression, linear regression)
 - naive Bayes
 - decision trees
 - ensembles of trees (Random Forests and Gradient-Boosted Trees)
 - isotonic regression
- Collaborative filtering
 - alternating least squares (ALS)
- Clustering
 - k-means
 - Gaussian mixture
 - power iteration clustering (PIC)
 - latent Dirichlet allocation (LDA)
 - bisecting k-means
 - streaming k-means
- Dimensionality reduction
 - singular value decomposition (SVD)
 - principal component analysis (PCA)
- Feature extraction and transformation
- Frequent pattern mining
 - FP-growth
 - association rules
 - PrefixSpan
- Evaluation metrics
- PMML model export
- Optimization (developer)
 - stochastic gradient descent
 - limited-memory BFGS (L-BFGS)



Clustering - spark.mllib

Clustering is an unsupervised learning problem whereby we aim to group data points based on a notion of similarity. Clustering is often used for exploratory analysis as a learning pipeline (in which distinct classifiers or regression models are trained on different clusters of data).

The spark.mllib package supports the following models:

- K-means
- Gaussian mixture
- Power iteration clustering (PIC)
- Latent Dirichlet allocation (LDA)
- Bisecting k-means
- Streaming k-means

Dimensionality Reduction



- Classification and regression
 - linear models (SVMs, logistic regression, linear regression)
 - naive Bayes
 - decision trees
 - ensembles of trees (Random Forests and Gradient-Boosted Trees)
 - isotonic regression
- Collaborative filtering
 - alternating least squares (ALS)
- Clustering
 - k-means
 - Gaussian mixture
 - power iteration clustering (PIC)
 - latent Dirichlet allocation (LDA)
 - bisecting k-means
 - streaming k-means
- Dimensionality reduction
 - singular value decomposition (SVD)
 - principal component analysis (PCA)
- Feature extraction and transformation
- Frequent pattern mining
 - FP-growth
 - association rules
 - PrefixSpan
- Evaluation metrics
- PMML model export
- Optimization (developer)
 - stochastic gradient descent
 - limited-memory BFGS (L-BFGS)



Dimensionality Reduction - spark.mllib

- Singular value decomposition (SVD)
 - Performance
 - SVD Example
- Principal component analysis (PCA)

[Dimensionality reduction](#) is the process of reducing the number of variables under consideration. It can be used to extract latent features from raw and noisy features or compress data while maintaining the structure. `spark.mllib` provides support for dimensionality reduction on the `RowMatrix` class.

奇异值分解



➤ 奇异值分解，singular value decomposition (SVD) 是线性代数中一种重要的矩阵分解方法

➤ 设有一个对角矩阵如下

➤
$$A = \begin{pmatrix} 100 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

➤ 其特征值为100， 10和1， 当一个向量与之相乘时：

➤
$$\begin{pmatrix} 100 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 100x \\ 10y \\ z \end{pmatrix}$$

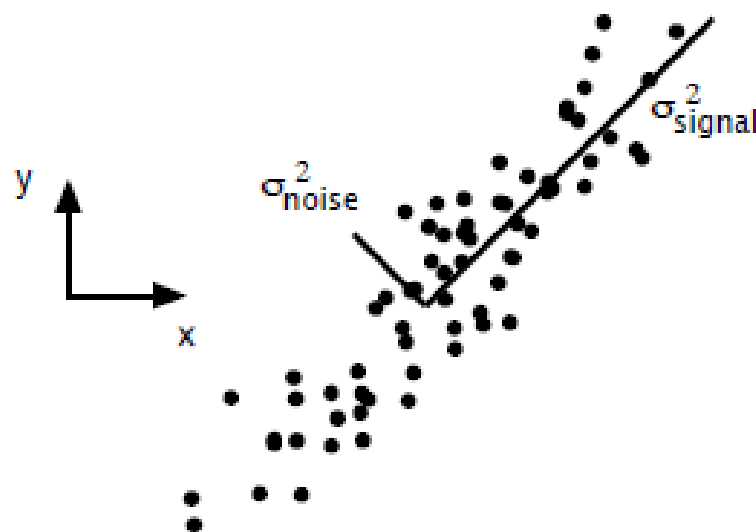
➤ 这个式子有何意义呢？一个典型的场景是降维（**Dimensionality Reduction**）。如何降维呢？从上面可以看到，当xyz发生变化时，x的变化将被扩大100倍，y的变化被扩大10倍，而z则不会被扩大。那么当计算的结果不需要十分精确时，z这个变量对于我们来说意义是十分小的。当处理的数据维度十分巨大的时候，计算量变得很大，这时候就可以通过降维来去除不是那么重要的维度（如本例中的z维度），这些维度对最终的计算结果的影响远远小于其它的维度。

主成分分析



➤ 主成分分析，Principal components analysis (PCA) 是一种分析、简化数据集的技术。主成分分析经常用于减少数据集的维数，同时保持数据集中的对方差贡献最大的特征。其方法主要是通过对协方差矩阵进行特征分解，以得出数据的主成分（即特征向量）与它们的权值（即特征值）。

PCA的数学定义是：一个正交化线性变换，把数据变换到一个新的坐标系统中，使得这一数据的任何投影的第一大方差在第一个坐标（称为第一主成分）上，第二大方差在第二个坐标（第二主成分）上，依次类推如下图所示，通过变化将X-Y坐标系映射到signal和noise上：



Feature Extraction and Transformation



- Classification and regression
 - linear models (SVMs, logistic regression, linear regression)
 - naive Bayes
 - decision trees
 - ensembles of trees (Random Forests and Gradient-Boosted Trees)
 - isotonic regression
- Collaborative filtering
 - alternating least squares (ALS)
- Clustering
 - k-means
 - Gaussian mixture
 - power iteration clustering (PIC)
 - latent Dirichlet allocation (LDA)
 - bisecting k-means
 - streaming k-means
- Dimensionality reduction
 - singular value decomposition (SVD)
 - principal component analysis (PCA)
- Feature extraction and transformation
- Frequent pattern mining
 - FP-growth
 - association rules
 - PrefixSpan
- Evaluation metrics
- PMML model export
- Optimization (developer)
 - stochastic gradient descent
 - limited-memory BFGS (L-BFGS)



Feature Extraction and Transformation - spark.mllib

- TF-IDF
- Word2Vec
 - Model
 - Example
- StandardScaler
 - Model Fitting
 - Example
- Normalizer
 - Example
- ChiSqSelector
 - Model Fitting
 - Example
- ElementwiseProduct
 - Example
- PCA
 - Example

Evaluation



Evaluation Metrics - spark.mllib



- Classification and regression
 - linear models (SVMs, logistic regression, linear regression)
 - naive Bayes
 - decision trees
 - ensembles of trees (Random Forests and Gradient-Boosted Trees)
 - isotonic regression
- Collaborative filtering
 - alternating least squares (ALS)
- Clustering
 - k-means
 - Gaussian mixture
 - power iteration clustering (PIC)
 - latent Dirichlet allocation (LDA)
 - bisecting k-means
 - streaming k-means
- Dimensionality reduction
 - singular value decomposition (SVD)
 - principal component analysis (PCA)
- Feature extraction and transformation
- Frequent pattern mining
 - FP-growth
 - association rules
 - PrefixSpan
- Evaluation metrics
- PMML model export
- Optimization (developer)
 - stochastic gradient descent
 - limited-memory BFGS (L-BFGS)



Evaluation Metrics - spark.mllib

- Classification model evaluation
 - Binary classification
 - Threshold tuning
 - Multiclass classification
 - Label based metrics
 - Multilabel classification
 - Ranking systems
- Regression model evaluation

➤ 二分类评估

1. 二分类用于将给定数据集分成2类（判断欺诈），二分类可以看做多分类的特殊情况。
2. 许多分类模型输出的结果是评分（score，代表分类）

➤ 多分类评估

➤ 多标签分类

➤ 评分系统

➤ 回归模型评估

MSE, RMSE, MAE, R², Explained Variance

模型评估——准确率和召回率(Precision & Recall)



➤ 准确率和召回率，准确率用来评价结果质量，而召回率用于评价结果完整性。

在二分类问题中准确率（精确率）precision定义为真阳性的数目除以真阳性和假阳性的总数，其中真阳性是指被正确预测的类别为1的样本，假阳性是错误预测为类别为1的样本。

召回率(recall)定义为真阳性数目除以真阳性和假阴性的和，其中假阴性是类别为1却被预测为0的样本。

TP: label 是正的预测结果也是正的

TN:label 是负的预测也是负的

FP:label 是负的但是预测结果是正的

FN: label是正的但是预测结果是负的

精确率(precision)的公式是 $P = \frac{TP}{TP+FP}$

召回率(recall)的公式是 $R = \frac{TP}{TP+FN}$

F-measure认为精确率和召回率的权重是一样的,但有些场景下,我们可能认为精确率会更加重要,调整参数a,使用Fa-measure可以帮助我们更好的evaluate结果.



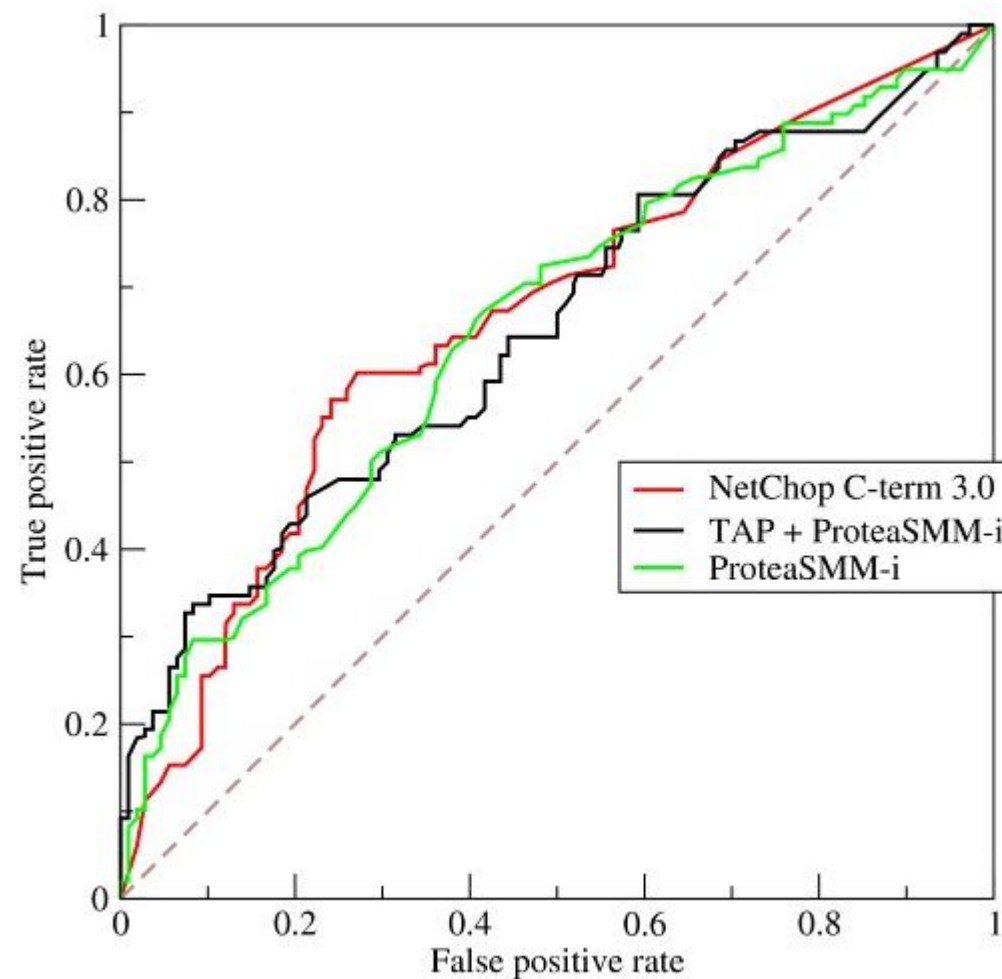
➤ ROC & AUC 评估二分类结果

许多分类模型对每个类实际上输出一个“得分”（很多时候是一个概率），其中分数越高越相似。在二元分类情况下，模型可能对每个类别输出一个概率。 $P(Y=1|X)$ $P(Y=0|X)$ 。不同于简单地使用较高的概率，在某些情况下，模型需要被调整以便他预测一个类别的时候，属于这个类需要特别高的概率（例如对于二元分类，我们设定的阈值为90%，则>90%则为正，小于<90%都为负），因此，模型的输出的类别取决于模型的阈值。

Available metrics

Metric	Definition
Precision (Postive Predictive Value)	$PPV = \frac{TP}{TP+FP}$
Recall (True Positive Rate)	$TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$
F-measure	$F(\beta) = (1 + \beta^2) \cdot \left(\frac{PPV \cdot TPR}{\beta^2 \cdot PPV + TPR} \right)$
Receiver Operating Characteristic (ROC)	$FPR(T) = \int_T^\infty P_0(T) dT$ $TPR(T) = \int_T^\infty P_1(T) dT$
Area Under ROC Curve	$AUROC = \int_0^1 \frac{TP}{P} d\left(\frac{FP}{N}\right)$
Area Under Precision-Recall Curve	$AUPRC = \int_0^1 \frac{TP}{TP+FP} d\left(\frac{TP}{P}\right)$

- 真阳性率(TPR) 是真阳性的样本数除以真阳性和假阴性的样本之和。TPR是真阳性数目占有所有正样本的比例。
- 假阳性率 (FPR) 是假阳性的样本数除以假阳性和真阴性的样本之和。FPR是假阳性样本数占有所有负样本总数的比例。
- ROC 曲线表示分类器性能在不同决策阈值下的TPR对FPR的折衷
- ROC 下的面积称为AUC



➤ MSE: Mean Squared Error （针对数据集）

均方误差是指参数估计值与参数真值之差平方的期望值；MSE可以评价数据的变化程度，MSE的值越小，说明预测模型描述实验数据具有更好的精确度。

➤ MAE: Mean Absolute Error （针对数据集）

MAE 和MSE的选择主要是取决于你的应用场景，因为MSE会对离平均较远的点给一个更大的惩罚值（有平方），而MAE则是给一个相对更小的，MAE反映预测值的实际情况。

➤ RMSE:均方根误差是均方误差的算术平方根

➤ Explained Variance ： 解释方差

拟合结果解释了多少的原数据

回归模型评估—Regression Model Evaluation (RegressionMetricsDemo)



- R^2 : (可决系数/确定系数) 可决系数可以作为综合度量回归模型对样本观测值拟合优度的度量指标(0~1)。可决系数越大, 说明在总变差中由模型作出了解释的部分占的比重越大, 模型拟合优度越好。反之可决系数小, 说明模型对样本观测值的拟合程度越差。

Available metrics

Metric	Definition
Mean Squared Error (MSE)	$MSE = \frac{\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}{N}$
Root Mean Squared Error (RMSE)	$RMSE = \sqrt{\frac{\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}{N}}$
Mean Absolutue Error (MAE)	$MAE = \sum_{i=0}^{N-1} y_i - \hat{y}_i $
Coefficient of Determination (R^2)	$R^2 = 1 - \frac{MSE}{\text{VAR}(\mathbf{y}) \cdot (N-1)} = 1 - \frac{\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{N-1} (y_i - \bar{y})^2}$
Explained Variance	$1 - \frac{\text{VAR}(\mathbf{y} - \hat{\mathbf{y}})}{\text{VAR}(\mathbf{y})}$

Model Export



- Spark mllib 包中的众多算法提供了模型的save 和 load 方法，方便使用离线数据进行建模后，分享到其他平台进行预测。

```
extends GeneralizedLinearModel(weights, intercept) with RegressionModel with Serializable  
with Saveable with PMMLExportable {
```

- PMML(Predictive Model Markup Language) 是一个开放的工业标准，它以 XML 为载体将上述数据挖掘任务标准化，可以把某一产品所创建的数据挖掘方案应用于任何其它遵从 PMML 标准的产品或平台中，而不需考虑分析和预测过程中的具体实现细节。

- Classification and regression
 - linear models (SVMs, logistic regression, linear regression)
 - naive Bayes
 - decision trees
 - ensembles of trees (Random Forests and Gradient-Boosted Trees)
 - isotonic regression
- Collaborative filtering
 - alternating least squares (ALS)
- Clustering
 - k-means
 - Gaussian mixture
 - power iteration clustering (PIC)
 - latent Dirichlet allocation (LDA)
 - bisecting k-means
 - streaming k-means
- Dimensionality reduction
 - singular value decomposition (SVD)
 - principal component analysis (PCA)
- Feature extraction and transformation
- Frequent pattern mining
 - FP-growth
 - association rules
 - PrefixSpan
- Evaluation metrics
- PMML model export
- Optimization (developer)
 - stochastic gradient descent
 - limited-memory BFGS (L-BFGS)



PMML model export - spark.mllib

- [spark.mllib supported models](#)
- [Examples](#)

spark.mllib supported models


spark.mllib supports model export to Predictive Model Markup Language ([PMML](#)).

The table below outlines the `spark.mllib` models that can be exported to PMML and their equivalent PMML model.

<code>spark.mllib</code> model	PMML model
KMeansModel	ClusteringModel
LinearRegressionModel	RegressionModel (functionName="regression")
RidgeRegressionModel	RegressionModel (functionName="regression")
LassoModel	RegressionModel (functionName="regression")
SVMModel	RegressionModel (functionName="classification" normalizationMethod="none")
Binary LogisticRegressionModel	RegressionModel (functionName="classification" normalizationMethod="logit")

模型优化



- Classification and regression
 - linear models (SVMs, logistic regression, linear regression)
 - naive Bayes
 - decision trees
 - ensembles of trees (Random Forests and Gradient-Boosted Trees)
 - isotonic regression
 - Collaborative filtering
 - alternating least squares (ALS)
 - Clustering
 - k-means
 - Gaussian mixture
 - power iteration clustering (PIC)
 - latent Dirichlet allocation (LDA)
 - bisecting k-means
 - streaming k-means
 - Dimensionality reduction
 - singular value decomposition (SVD)
 - principal component analysis (PCA)
 - Feature extraction and transformation
 - Frequent pattern mining
 - FP-growth
 - association rules
 - PrefixSpan
 - Evaluation metrics
 - PMML model export
 - Optimization (developer)
 - stochastic gradient descent
 - limited-memory BFGS (L-BFGS)
- 

Optimization - spark.mllib

- Mathematical description
 - Gradient descent
 - Stochastic gradient descent (SGD)
 - Update schemes for distributed SGD
 - Limited-memory BFGS (L-BFGS)
 - Choosing an Optimization Method
- Implementation in MLlib
 - Gradient descent and stochastic gradient descent
 - L-BFGS
- Developer's notes

➤ 以房交会问题为例，假设我们回归问题的训练集如下：

1.m 代表训练集中实例的数量

2.x 代表特征 / 输入变量

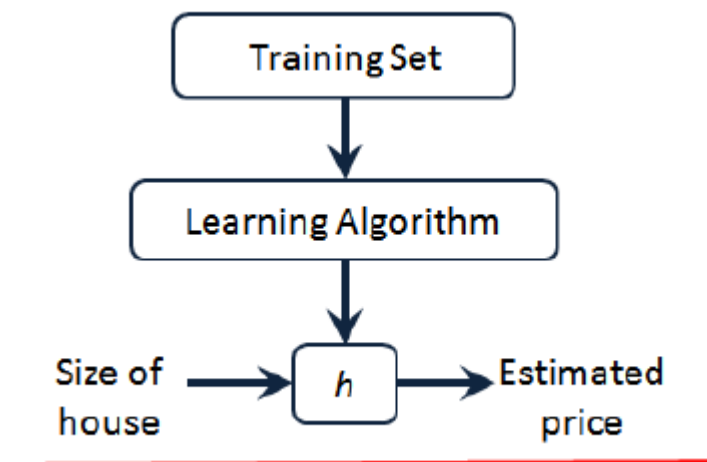
3.y 代表目标变量 / 输出变量

4.(x,y) 代表训练集中的实例

5.(x (i) ,y (i)) 代表第 i 个观察实例

6. h 代表学习算法的解决方案或函数也称为 假设
(hypothesis)

Size in feet ² (x)	Price (\$) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...



➤ 要解决房价预测问题，我们实际上是要将训练集“喂”给我们的学习算法，进而学习得一个假设 h ，然后将我们要预测的房屋尺寸作为输入变量输入给 h ，预测出该房屋的交易价格作为输出变量输出为结果。

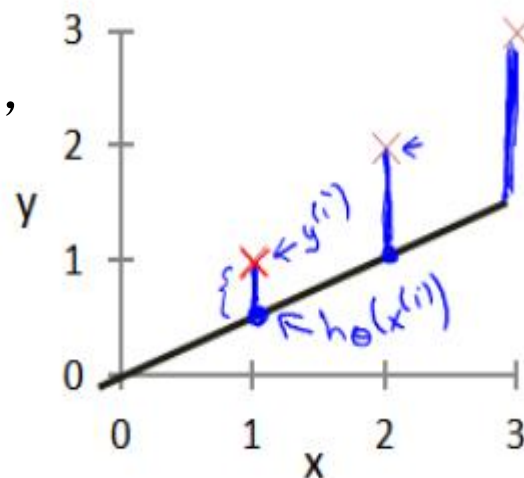
➤ 一种可能的表达方式：
$$h_{\theta} = \theta_0 + \theta_1 x$$

➤ 我们现在要做的便是为我们的模型选择合适的参数（parameters） θ_0 和 θ_1 ，在房价问题这个例子中便是直线的斜率和在 y 轴上的截距。

➤ 我们选择的参数决定了我们得到的直线相对于我们的训练集的准确程度，

➤ 模型所预测的值与训练集中实际值之间的差距（下图中蓝线所指）就是

➤ 建模误差（modeling error）。



模型优化—LossFunction

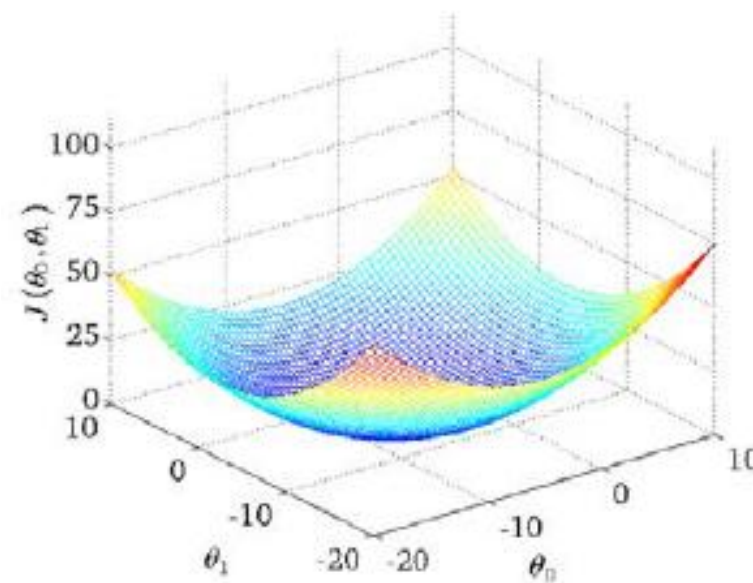


- 我们的目标便是选择出可以使得建模误差的平方和能够最小的模型参数。

即损失函数

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \text{ 最小。}$$

- 可以看出在三维空间中存在一个使得 $J(\theta_0, \theta_1)$ 最小的点, 梯度下降用于寻找全局最小值。



模型优化—梯度下降(Gradient descent)



➤ 梯度下降是一个用来求函数最小值的算法，我们将使用梯度下降算法来求出代价函数 $J(\theta_0, \theta_1)$ 的最小值。

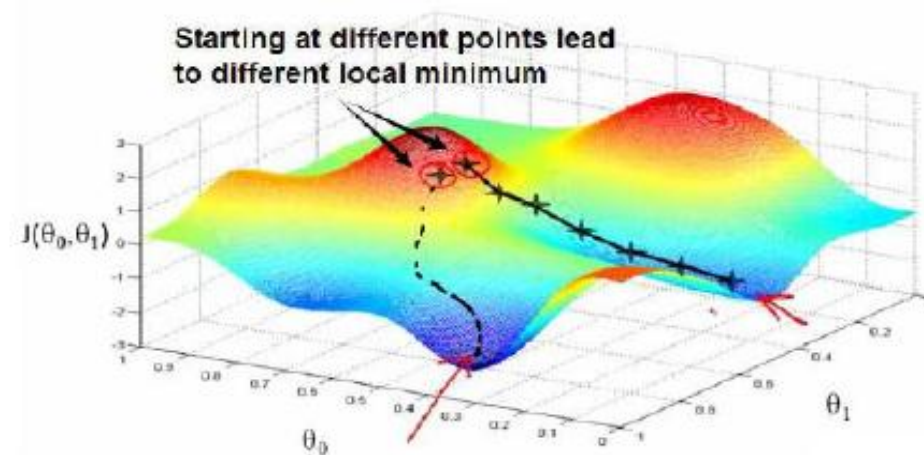
➤ 梯度下降背后的思想是：开始时我们随机选择一个参数的组合 $(\theta_0, \theta_1, \dots, \theta_n)$ ，计算代价函数，然后我们寻找下一个能让代价函数值下降最多的参数组合。

将 $J(\theta)$ 对 θ 求偏导，得到每个 θ 对应的的梯度

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

由于是要最小化风险函数，所以按每个参数 θ 的梯度负方向，来更新每个 θ

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



从上面公式可以注意到，它得到的是一个全局最优解，但是每迭代一步，都要用到训练集所有的数据，如果 m 很大，可想而知这种方法的迭代速度！所以，这就引入了另外一种方法，随机梯度下降。

模型优化—梯度下降(Gradient descent)



- 批量梯度下降(batch gradient descent)算法的公式为
$$\begin{array}{l} \text{repeat until } \underline{\text{convergence}} \{ \\ \quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1) \\ \} \end{array}$$
- 其中 α 是 学习率 (learning rate) , 它决定了我们沿着能让代价函数下降程度最大的方向向下迈出的步子有多大, 在批量梯度下降中, 我们每一次都同时让所有的参数减去学习速率乘以代价函数的导数。

模型优化—随机梯度下降(Stochastic Gradient descent)



- 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，而是有可能在最小值点附近徘徊。
- i 是样本编号下标， j 是样本维数下标， m 为样例数目， n 为特征数目。所以更新一个 θ_j 只需要一个样本就可以。

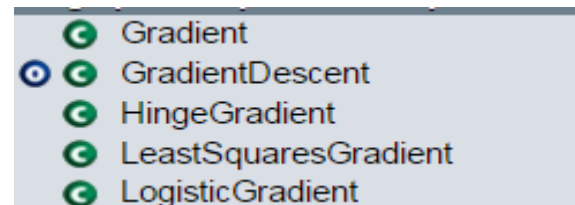
```
Loop {  
    for i=1 to m, {  
         $\theta_j := \theta_j + \alpha (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)}$     (for every  $j$ ).  
    }  
}
```

模型优化—Spark 中的实现

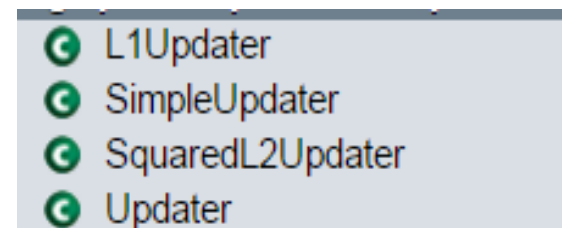


- Spark 中的 Gradient 实现有三种 LogisticGradient ; LeastSquaresGradient ; HingeGradient 对应三种 LossFunction.

Class used to compute the gradient for a loss function, given a single data point.



- Updater 也有三种
SimpleUpdater;
L1Updater;
SquaredL2Updater



- L-BFGS—局部优化法 (Broyden fletcher goldfarb shann,BFGS) 和 有限内存局部优化法 (LBFGS)

L-BFGS is currently **only a low-level optimization primitive in MLlib**. If you want to use L-BFGS in various ML algorithms such as Linear Regression, and Logistic Regression, you have to pass the gradient of objective function, and updater into optimizer .

- 1.线性回归
- 2.逻辑回归
- 3.svm
- 4.als
- 以上算法进行试验并提交试验报告，简述建模和评估的过程