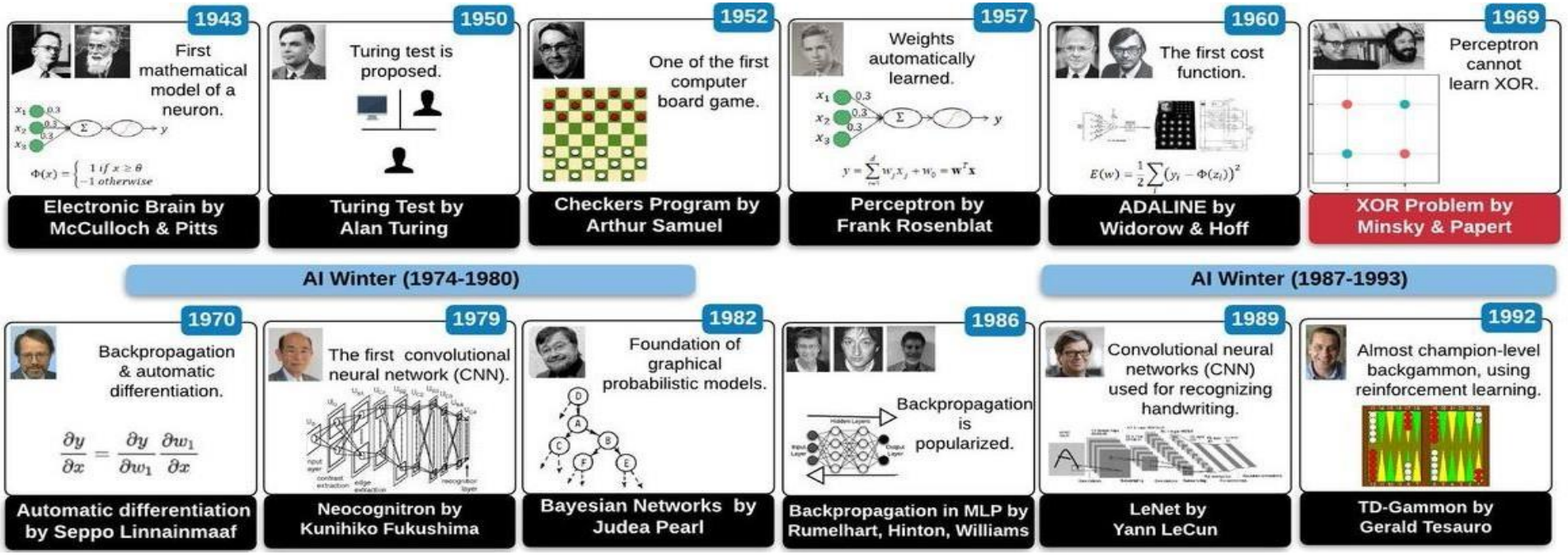
第六章 机器学习II—基本方法

Chao Yu （余超）

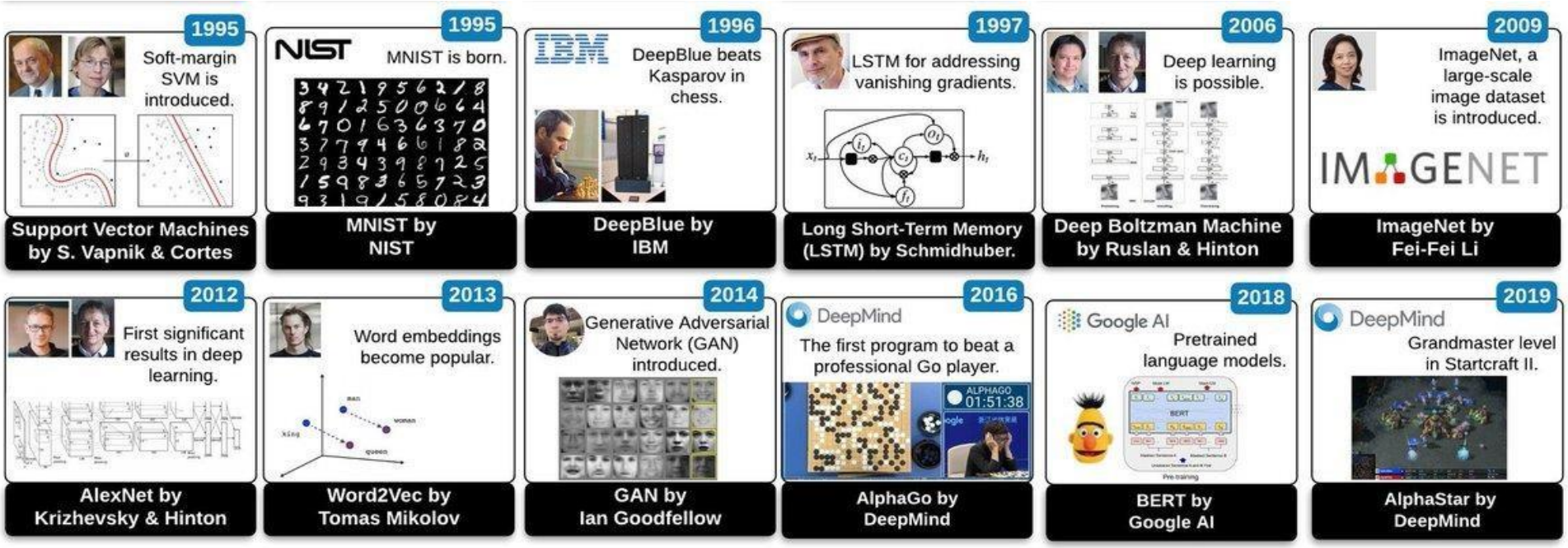
School of Computer Science and Engineering
Sun Yat-Sen University

[Based on Hung-yi Lee Deep Learning Lecture]

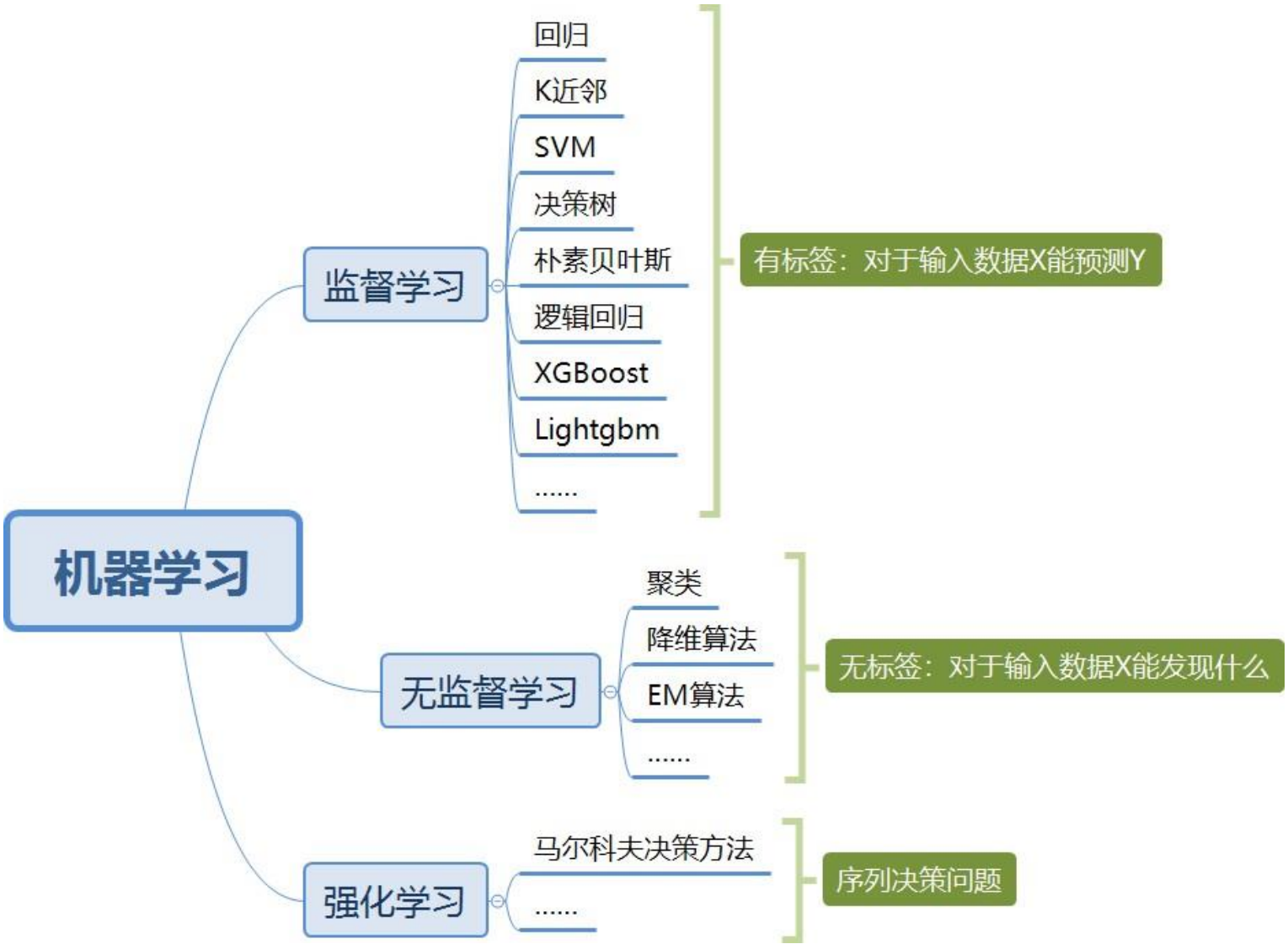
机器学习发展史


1943
First mathematical model of a neuron.
Electronic Brain by McCulloch & Pitts
1950
Turing test is proposed.
Turing Test by Alan Turing
1952
One of the first computer board game.
Checkers Program by Arthur Samuel
1957
Weights automatically learned.
Perceptron by Frank Rosenblat
1960
The first cost function.
ADALINE by Widorow & Hoff
1969
Perceptron cannot learn XOR.
XOR Problem by Minsky & Papert
AI Winter (1974-1980)
AI Winter (1987-1993)
1970
Backpropagation & automatic differentiation.
Automatic differentiation by Seppo Linnainmaaf
1979
The first convolutional neural network (CNN).
Neocognitron by Kunihiko Fukushima
1982
Foundation of graphical probabilistic models.
Bayesian Networks by Judea Pearl
1986
Backpropagation is popularized.
Backpropagation in MLP by Rumelhart, Hinton, Williams
1989
Convolutional neural networks (CNN) used for recognizing handwriting.
LeNet by Yann LeCun
1992
Almost champion-level backgammon, using reinforcement learning.
TD-Gammon by Gerald Tesauro

机器学习发展史



机器学习类型


机器学习
监督学习
回归
K近邻
SVM
决策树
朴素贝叶斯
逻辑回归
XGBoost
Lightgbm
.....
有标签：对于输入数据X能预测Y
无监督学习
聚类
降维算法
EM算法
.....
无标签：对于输入数据X能发现什么
强化学习
马尔科夫决策方法
.....
序列决策问题

机器学习背景知识

- 数学基础
  - ✓ 高等数学：导数、微分、泰勒公式…
  - ✓ 线性代数：向量、矩阵…
  - ✓ 概率论与数理统计：概率基本公式、常见分布、期望、协方差…
- Python基础
  - ✓ 环境安装：Anaconda、Jupyter Notebook、Pycharm、VSCode
  - ✓ Python基础
  - ✓ Python库：numpy、pandas、scipy、matplotlib、scikit-learn…

模型

机器学习首先要考虑使用什么样的模型。

模型的类别，大致有两种：一是概率模型(Probabilistic Model)和非概率模型(Non-Probabilistic Model)。

决策树、朴素贝叶斯、隐马尔科夫模型、高斯混合模型属于概率模型。

感知机、支持向量机、KNN、AdaBoost、K-means以及神经网络均属于非概率模型。

直观理解：拟合数据的分布函数

假设数据的特征是， $X=(x_1,x_2,...,x_n)$

我们需要找到一个函数能表示所有数据：$y=WX=w_1x_1+w_2x_2+...+w_nx_n$

例如，猫的特征是 X=(10，20，30)，$y=WX=w_1x_1+w_2x_2+w_3x_3=w_1\times10+w_2\times20+w_3\times30=\begin{cases}0\\1\end{cases}$

如何学习这些权重 $W=(w_1,w_2,...,w_n)$?

度量预测的 y 和真实的 $\bar{y}$ 之间的差异，即计算loss值，通过使得loss最小化使得差异最小化

损失函数loss

1. 0-1损失函数(0-1 Loss Function)

$$L(Y, f(X)) = \begin{cases} 1, & Y \neq f(X) \\ 0, & Y = f(X) \end{cases}$$

2. 平方损失函数(Quadratic Loss Function)

$$L(Y, f(X)) = (Y - f(X))^2$$

3. 绝对损失函数(Absolute Loss Function)

$$L(Y, f(X)) = |Y - f(X)|$$

4. 对数损失函数(Logarithmic Loss Function)

$$L(Y, P(Y|X)) = -\log P(Y|X)$$

损失函数

根据上述损失函数模型，我们可知，损失函数值越小，模型性能越好。给定一个数据集，我们将训练数据集的平均损失称为**经验风险**。基于经验风险最小化原则，可构建全局损失函数求解最优化问题：

$$\min_{f} \frac{1}{N}\sum_{n=1}^{N} \mathrm{L}(y_n, f(x_n))$$

当样本数量足够大时，根据大数定理，经验风险会近似于模型的期望风险。此时，经验风险最小化能确保有好的学习性能。然而，当样本数量不足时，单单利用经验风险最小化可能会导致 “过拟合”的问题。为此，我们再原有基础上加上用于控制模型复杂度的正则项(Regularizer)，得到结构最小化准则。

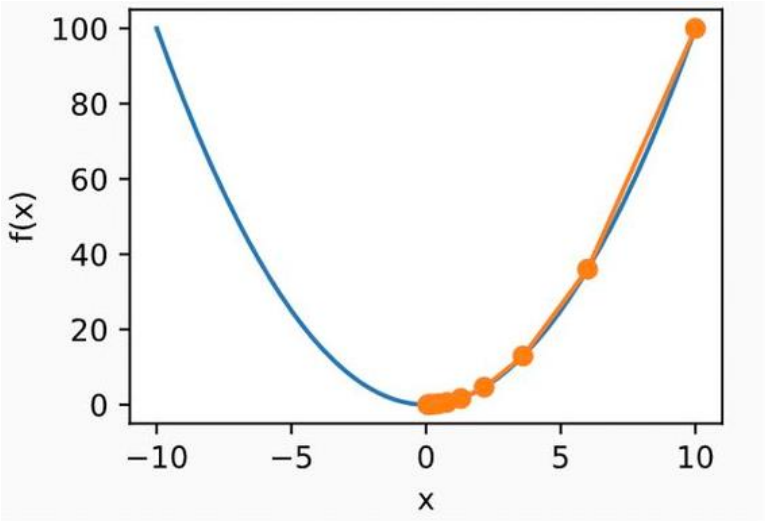
优化算法

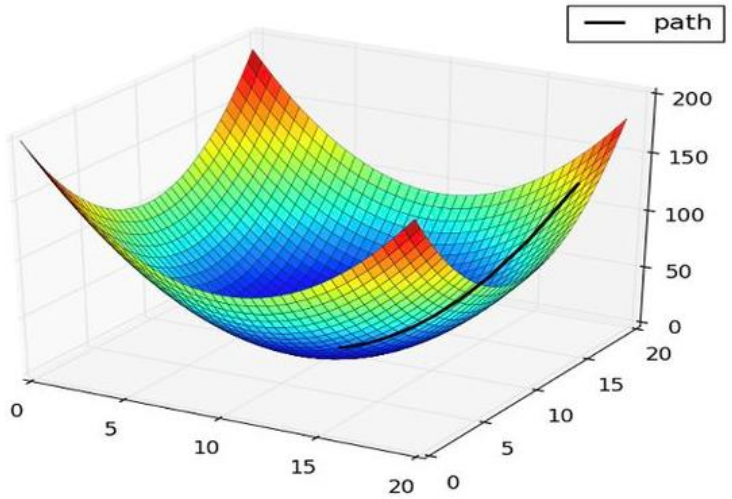
如何使得loss最小化？

算法指的是模型学习中的具体计算方法。一般来说，基于参数模型构建的统计学习问题都为最优化问题，它们都具有显式的解析解。

现有的优化方法主要有：梯度下降法、牛顿法、拟牛顿法、ADAM等等。

梯度下降法：一阶导数为0的点，不断下降寻找极小值



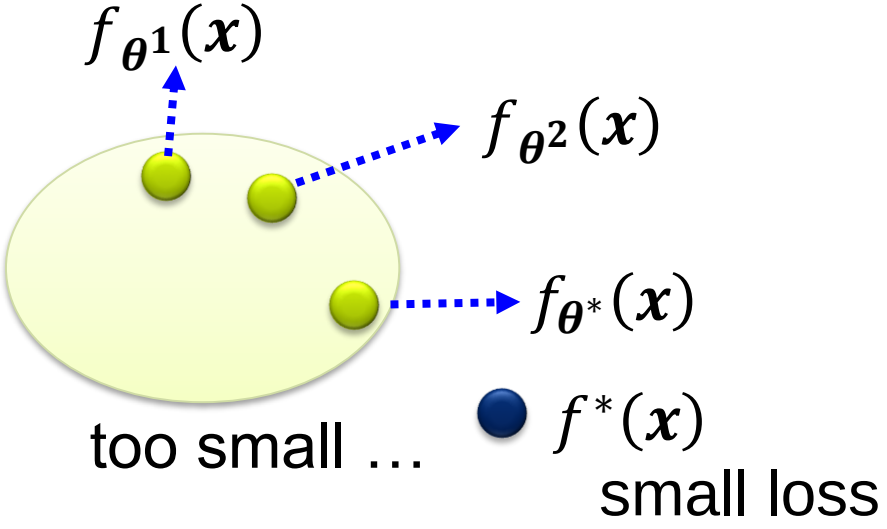


优化算法

Model Bias

find a needle in a haystack …

… but there is no needle

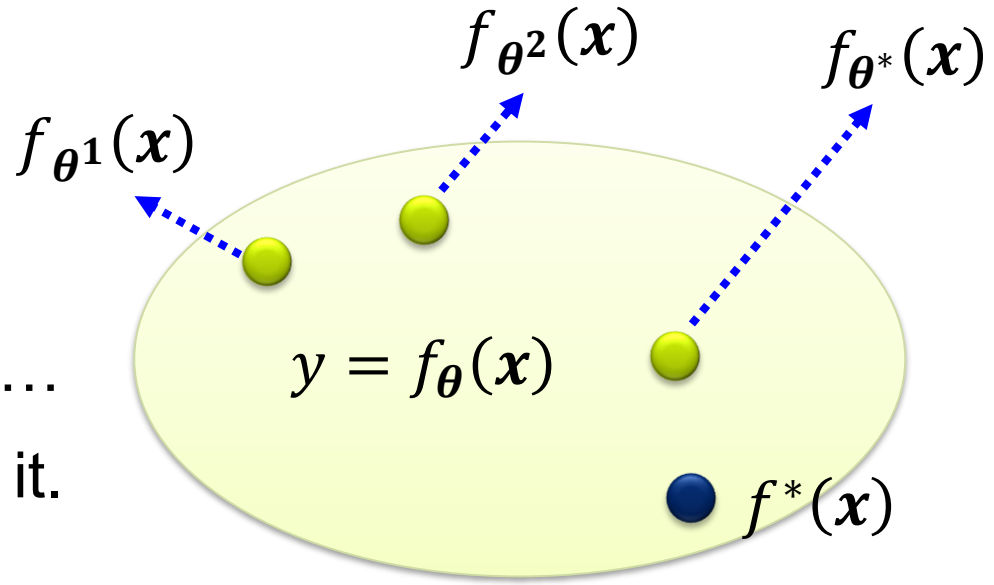


Which one???

Optimization Issue

A needle is in a haystack …

… Just cannot find it.



优化算法

- Gaining the insights from comparison
- Start from shallower networks (or other models), which are easier to optimize.
- If deeper networks do not obtain smaller loss on **training data**, then there is optimization issue.

	1 layer	2 layer	3 layer	4 layer	5 layer
2017 – 2020	0.28k	0.18k	0.14k	0.10k	0.34k

- Solution: More powerful optimization technology
  - Momentum
  - Adaptive Learning Rate

模型评估

当损失函数给定时，我们将基于模型训练数据的误差(Training Error)和测试数据的误差(Testing Error)作为模型评估的标准。

测试误差的具体定义为：$E_{test} = \frac{1}{N'}\sum_{n=1}^{N'} \mathrm{L}\left(y_n, \hat{f}(x_n)\right)$

其中，N'为测试数据数量，$\mathrm{L}(y_n, \hat{f}(x_n))$是损失函数，$y_n$代表真实标签，$\hat{f}(x_n)$代表预测标签。

一般来说，若我们模型学习的效果好，则训练误差和测试误差接近一致。

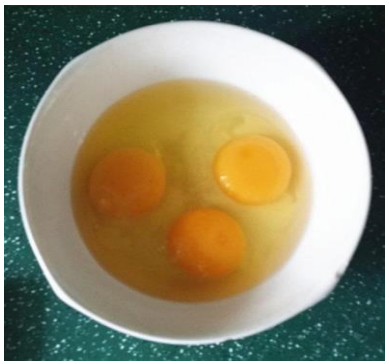
机器学习的开发流程

数据搜集 → 数据清洗 → 特征工程 → 数据建模







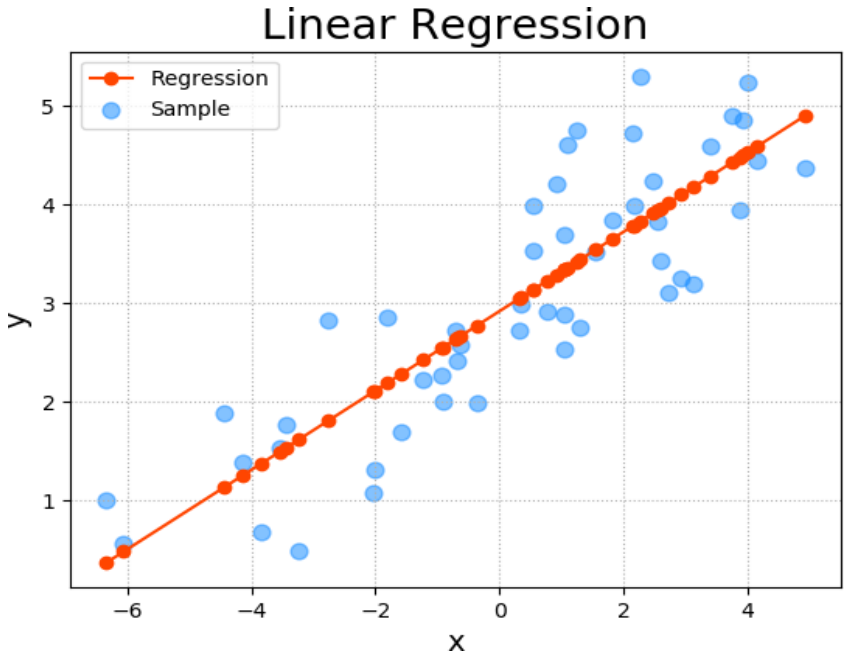


机器学习主要方法

人工智能课程

分类和回归

- 线性回归（Linear Regression）
  - 是一种通过属性的线性组合来进行预测的线性模型，其目的是找到一条直线或者一个平面或者更高维的超平面，使得预测值与真实值之间的误差最小化。



分类和回归

符号定义

m 代表训练集中样本的数量

n 代表特征的数量

x 代表特征/输入变量

y 代表目标变量/输出变量

(x, y) 代表训练集中的样本

$(x^{(i)}, y^{(i)})$ 代表第i个观察样本

h 代表学习算法的解决方案或函数也称为假设（**hypothesis**）

$\hat{y} = h(x)$,代表预测的值

建筑面积	总层数	楼层	实用面积	房价
143.7	31	10	105	**36200**
162.2	31	8	118	**37000**
199.5	10	10	170	**42500**
96.5	31	13	74	**31200**
……	……	……	……	……

$x^{(i)}$是特征矩阵中的第i行，是一个**向量**。

上图的：$x^{(2)} = \begin{bmatrix} 162.2 \\ 31 \\ 8 \\ 118 \end{bmatrix}$ $y^{(2)} = 37000$

$x_j^{(i)}$代表特征矩阵中第 i 行的第 j 个特征

上图的$x_2^{(2)} = 31, x_3^{(2)} = 8$

分类和回归

$$h(x) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

损失函数采用平方和损失：

$$l(x^{(i)}) = \frac{1}{2}(h(x^{(i)}) - y^{(i)})^2$$

要找到一组 $w(w_0, w_1, w_2, \ldots, w_n)$ ，

使得$J(w) = \frac{1}{2}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2$

(残差平方和) 最小

损失函数(Loss Function)度量单样本预测的错误程度，损失函数值越小，模型就越好。常用的损失函数包括：0-1损失函数、平方损失函数、绝对损失函数、对数损失函数等。

代价函数(Cost Function)度量全部样本集的平均误差。常用的代价函数包括均方误差、均方根误差、平均绝对误差等。

目标函数(Object Function)代价函数和正则化函数，最终要优化的函数。

备注：损失函数的系数1/2是为了便于计算，使对平方项求导后的常数系数为1，这样在形式上稍微简单一些。有些教科书把系数设为1/2，有些设置为1，这些都不影响结果。

分类和回归

均方误差（Mean Square Error,MSE）

$$\mathrm{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2$$

均方根误差 RMSE(Root Mean Square Error,RMSE)

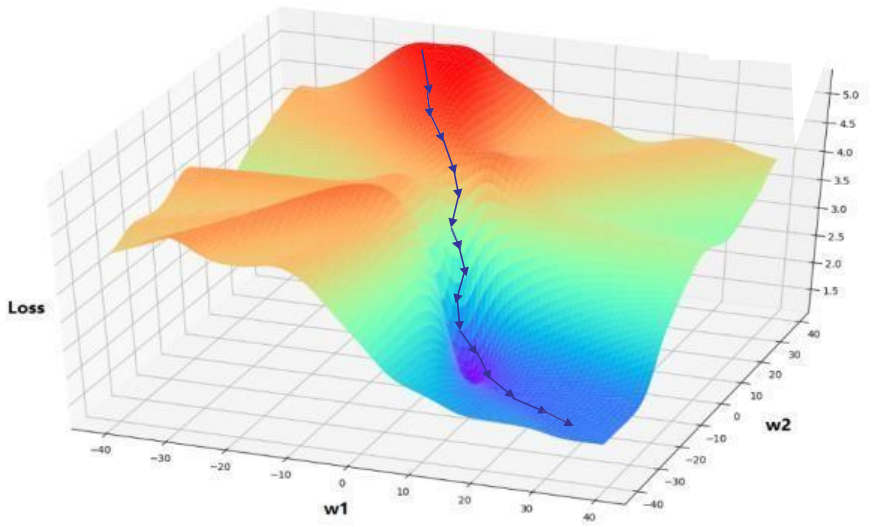
$$\mathrm{RMSE}(y,\hat{y}) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2}$$

平均绝对误差（Mean Absolute Error,MAE）

$$\mathrm{MAE}(y,\hat{y}) = \frac{1}{m}\sum_{i=1}^{n}\left|y^{(i)} - \hat{y}^{(i)}\right|$$

其中，$y^{(i)}$ 和$\hat{y}^{(i)}$分别表示第i个样本的真实值和预测值，m 为样本个数。

梯度下降



学习率

α

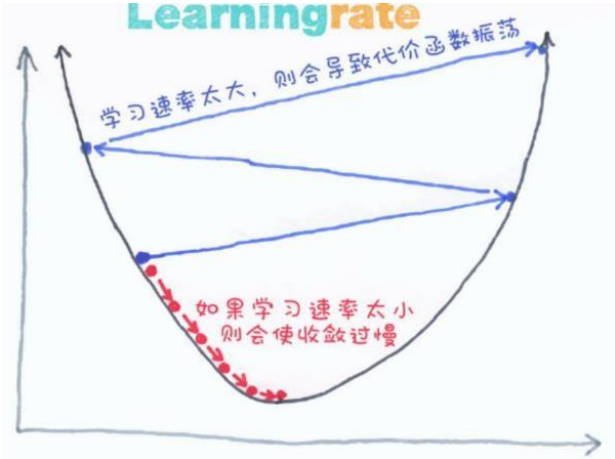
步长

参数更新

学习率　　梯度

$$w_j := w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(\left(h\left(x^{(i)}\right) - y^{(i)} \right) \cdot x_j^{(i)} \right)$$

(同步更新w_j，$(j=0,1,\ldots,n)$)



分类和回归

- 批量梯度下降（Batch Gradient Descent,BGD）
  - 梯度下降的每一步中，都用到了所有的训练样本
- 随机梯度下降（Stochastic Gradient Descent,SGD）
  - 梯度下降的每一步中，用到一个样本，在每一次计算之后便更新参数，而不需要首先将所有的训练集求和
- 小批量梯度下降（Mini-Batch Gradient Descent,MBGD）
  - 梯度下降的每一步中，用到了一定批量的训练样本

$$w_j := w_j - \alpha \frac{1}{b} \sum_{k=i}^{i+b-1} \left(h\left(x^{(k)}\right) - y^{(k)} \right) x_j^{(k)}$$

(同步更新w_j，$(j=0,1,\ldots,n)$)

b=1（随机梯度下降,SGD）
b=m（批量梯度下降,BGD）
b=*batch_size*，通常是2的指数倍，常见有32,64,128等。（小批量梯度下降,MBGD）

数据标准化和归一化

- **归一化：**数据归一化的目的是使得各特征对目标变量的影响一致，会将特征数据进行伸缩变化，所以数据归一化是会改变特征数据分布的。
- 将数据映射到[0,1]区间

$$x^* = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

- **Z-Score ：**数据标准化为了不同特征之间具备可比性，经过标准化变换之后的特征数据分布没有发生改变。
- 就是当数据特征取值范围或单位差异较大时，最好是做一下标准化处理
- 处理后的数据均值为0，方差为1

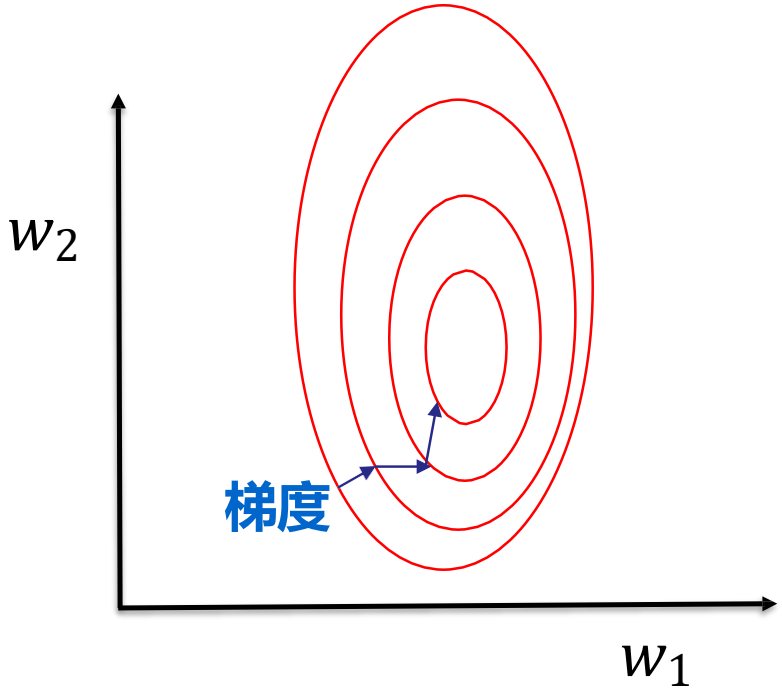
$$x^* = \frac{x - \mu}{\sigma}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)^2$$

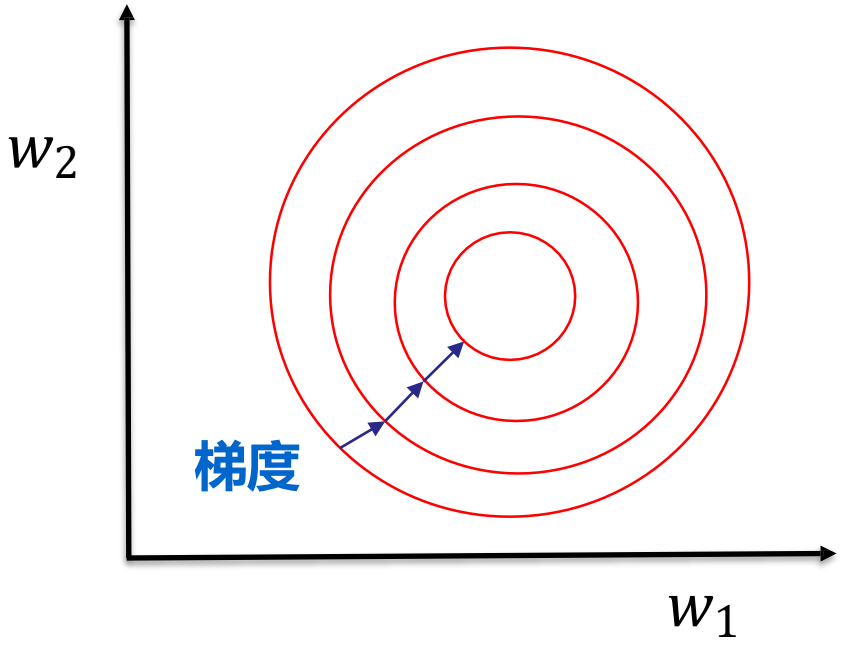
$$\mu = \frac{1}{m}\sum_{i=1}^{m}x^{(i)}$$

数据标准化和归一化

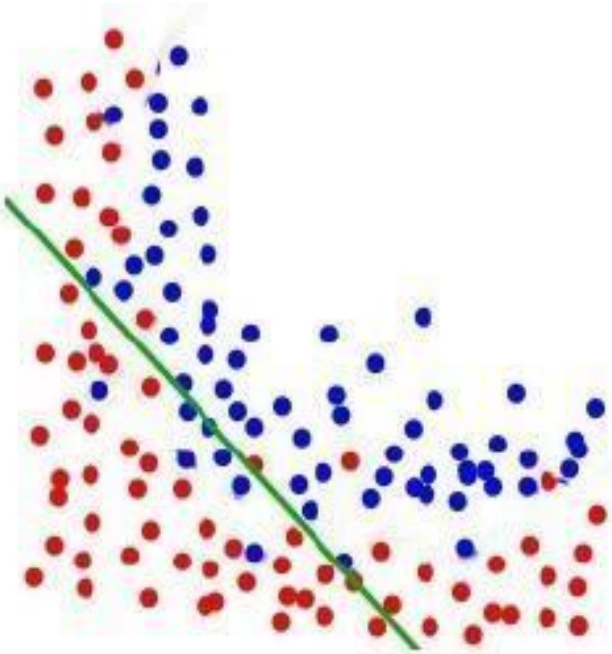
- **提升模型精度：**不同维度之间的特征在数值上有一定比较性，可以大大提高分类器的准确性



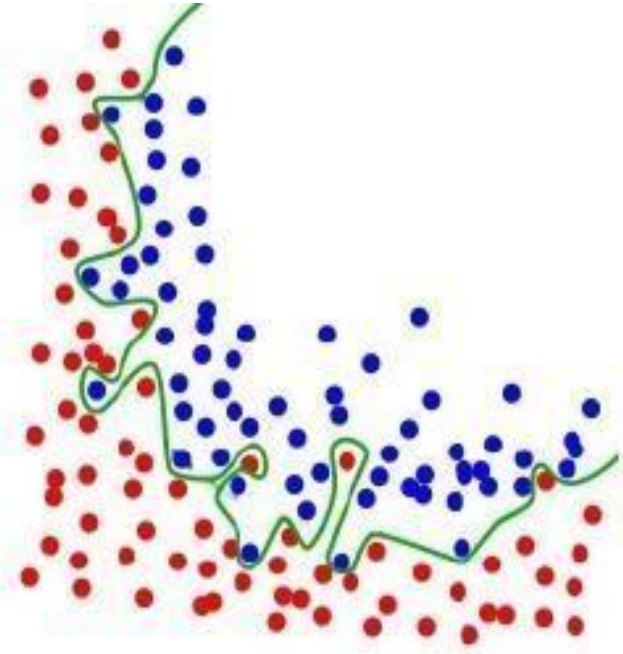
- **加速模型收敛：**最优解的寻优过程明显会变得平缓，更容易正确的收敛到最优解



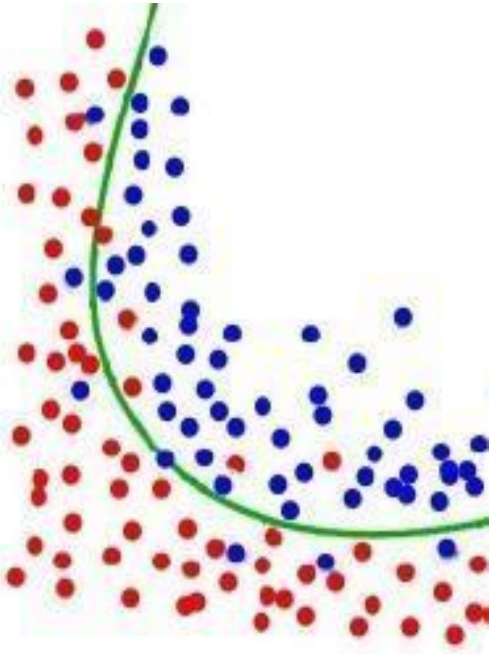
过拟合和欠拟合



欠拟合



过拟合



正合适

过拟合和欠拟合

- **过拟合的处理**

1.**获得更多的训练数据。**使用更多的训练数据是解决过拟合问题最有效的手段，因为更多的样本能够让模型学习到更多更有效的特征，减小噪声的影响

2.**降维。**即丢弃一些不能帮助我们正确预测的特征。可以是手工选择保留哪些特征，或者使用一些模型选择的算法来帮忙（例如PCA）。

3.**正则化。**正则化(regularization)的技术，保留所有的特征，但是减少参数的大小（magnitude），它可以改善或者减少过拟合问题。

4.**集成学习方法。**集成学习是把多个模型集成在一起，来降低单一模型的过拟合风险。

过拟合和欠拟合

- **欠拟合的处理**

  1.**添加新特征。**当特征不足或者现有特征与样本标签的相关性不强时，模型容易出现欠拟合。通过挖掘组合特征等新的特征，往往能够取得更好的效果。

  2.**增加模型复杂度。**简单模型的学习能力较差，通过增加模型的复杂度可以使模型拥有更强的拟合能力。例如，在线性模型中添加高次项，在神经网络模型中增加网络层数或神经元个数等。

  3.**减小正则化系数。**正则化是用来防止过拟合的，但当模型出现欠拟合现象时，则需要有针对性地减小正则化系数。

正则化

L_1正则化：$J(w)=\frac{1}{2}\sum_{i=1}^{m}(h(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{n}|w_j|$，Lasso Regression（Lasso回归）

L_2正则化：$J(w)=\frac{1}{2}\sum_{i=1}^{m}(h(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{n}w_j^2$，Ridge Regression （岭回归）

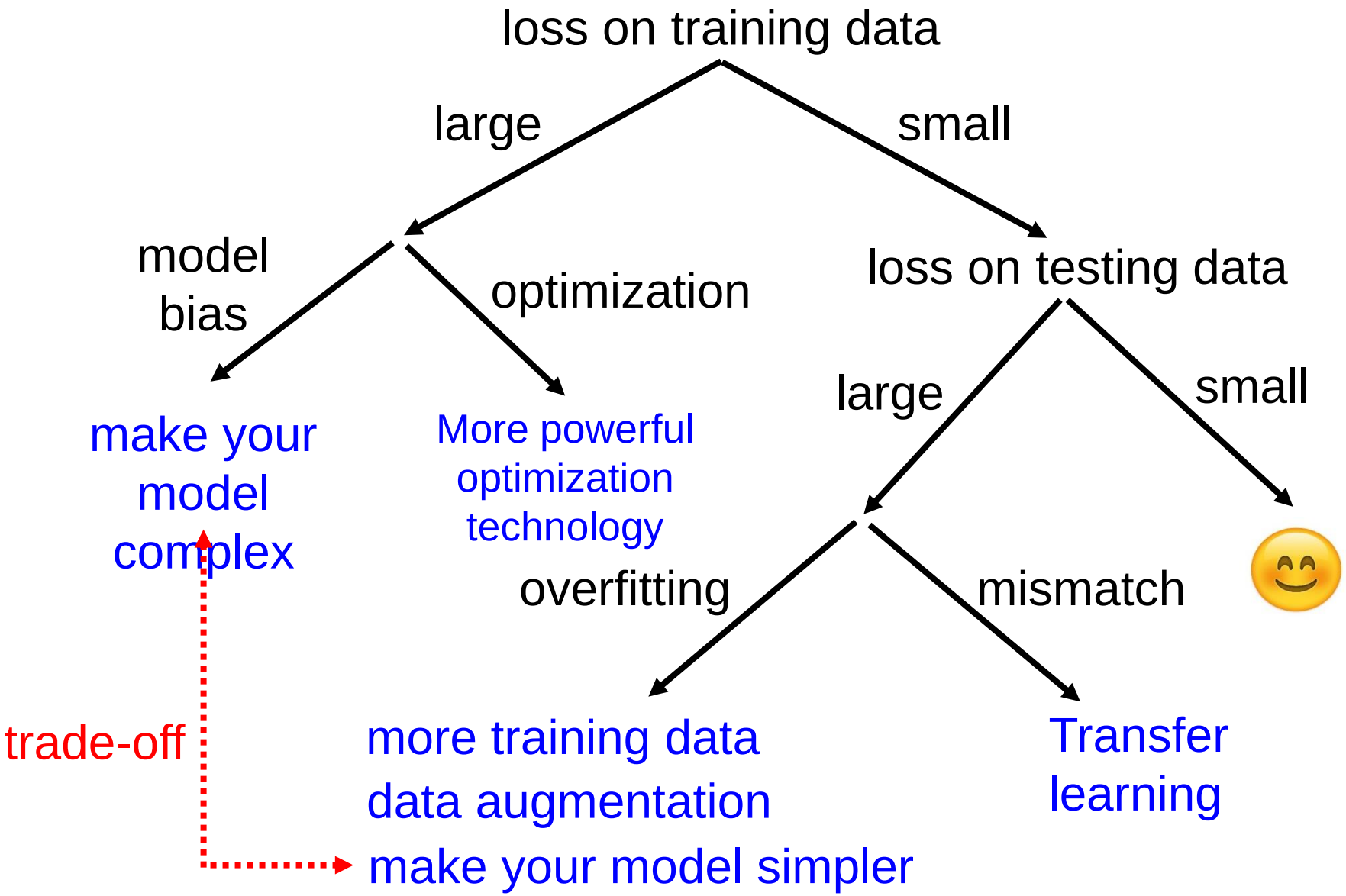
Elastic Net：$J(w)=\frac{1}{2}\sum_{i=1}^{m}(h(x^{(i)})-y^{(i)})^2+\lambda(\rho\cdot\sum_{j=1}^{n}|w_j|+(1-\rho)\cdot\sum_{j=1}^{n}w_j^2)$

(弹性网络)

其中：

- λ为正则化系数，调整正则化项与训练误差的比例，λ>0。
- 1≥ρ≥0为比例系数，调整L_1正则化与L_2正则化的比例。

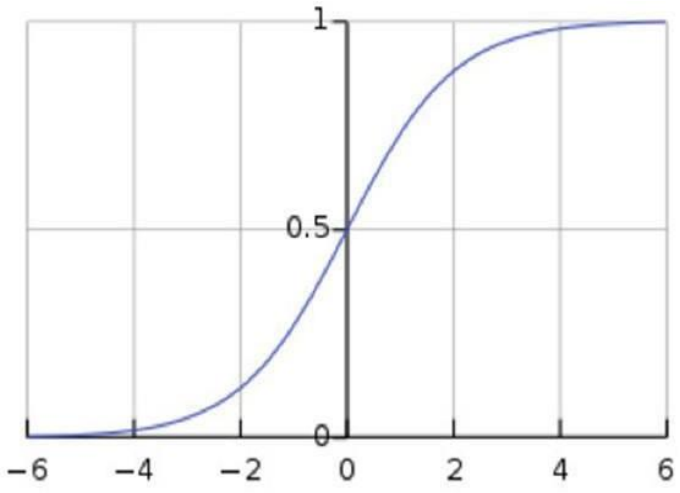
分类和回归


loss on training data
large
small
model bias
optimization
loss on testing data
make your model complex
More powerful optimization technology
large
small
overfitting
mismatch
trade-off
more training data
data augmentation
make your model simpler
Transfer learning

分类和回归

- ✓ 逻辑回归（分类问题）
  - ✓ $\sigma(z)$代表一个常用的逻辑函数（logistic function）为S形函数 (Sigmoid function)

$$\sigma(z) = g(z) = \frac{1}{1+e^{-z}}, z = w^T x + b \qquad \sigma'(z) = \sigma(z)(1-\sigma(z))$$



- ✓ $\sigma(z)$大于等于0.5时，预测y=1
- ✓ $\sigma(z)$小于0.5时，预测y=0

分类和回归

- ✓ 逻辑回归求解
  - ✓ 假设一个二分类模型：$p(y=1 \mid x ; w)=h(x)$

    $p(y=0 \mid x ; w)=1-h(x)$
  - ✓ 逻辑回归模型的假设是：$p(y \mid x; w) = (h(x))^{y}(1-h(x))^{1-y}$
  - ✓ 逻辑函数（logistic function）公式为

$$h(x) = g(w^{T}x) = g(z)$$

  - ✓ 损失函数

$$g(z) = \frac{1}{1+e^{-z}}, g'(z) = g(z)(1-g(z))$$

$$\mathrm{L}(\hat{y}, y) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$$

分类和回归

- ✓ 逻辑回归求解
  - ✓ 损失函数

$$J(w) = -\frac{1}{m} l(w) = -\frac{1}{m} \sum_{i=1}^{m} (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$

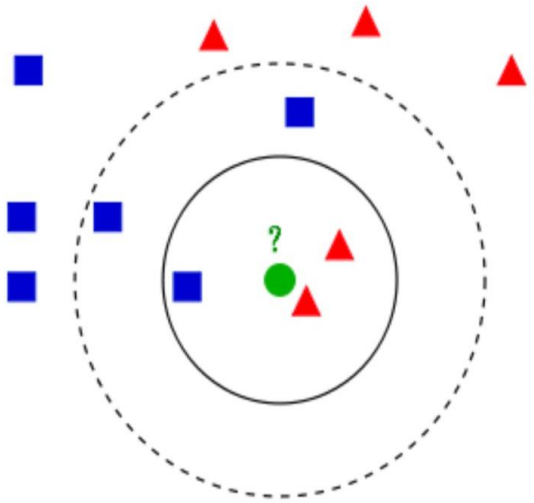
$$\hat{y}^{(i)} = \sigma(z^{(i)}) = \frac{1}{1 + e^{-z^{(i)}}}, z^{(i)} = w^T x^{(i)} + b$$

  - ✓ 梯度下降

$$w_j := w_j - \alpha \frac{\partial J(w)}{\partial w} \qquad \frac{\partial}{\partial w_j} J(w) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) x_j^{(i)}$$

分类和回归

✓ KNN（最近邻算法）



半径大小 表示 K值大小

k-nearest neighbours **classifier**：

其中:

$\Phi_{X,k}(q)$：返回训练集X中距离q最近的k个样本

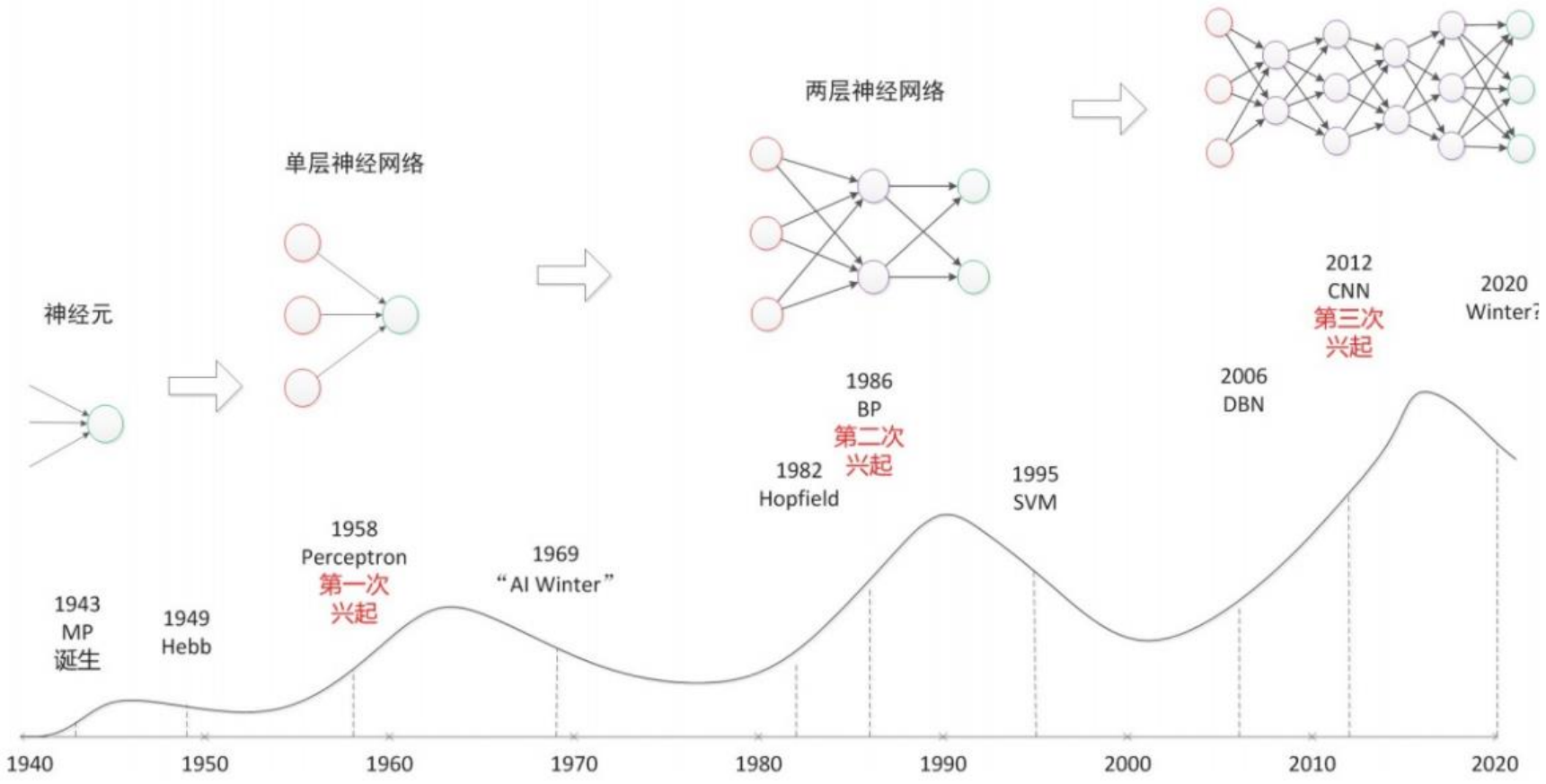
$g(\cdot)$：返回（训练）样本的标签

$maj(\cdot)$：返回众数

分类和回归

✓ SVM（支持向量机）

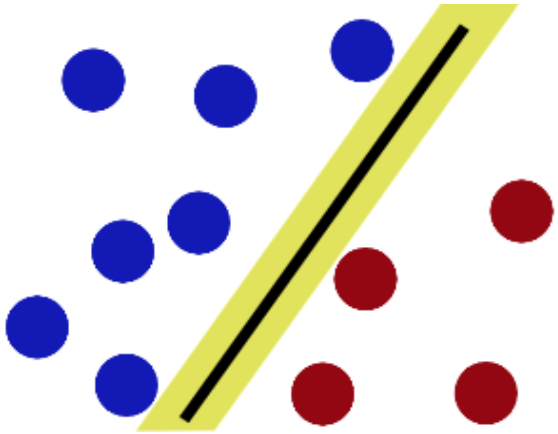
1995，由Corinna Cortes及Vladimir Vapanik等人发明的SVM（Support Vector Machines，支持向量机）算法诞生，迅速打败了神经网络算法成为主流。

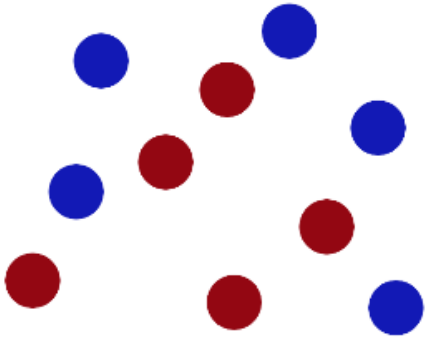


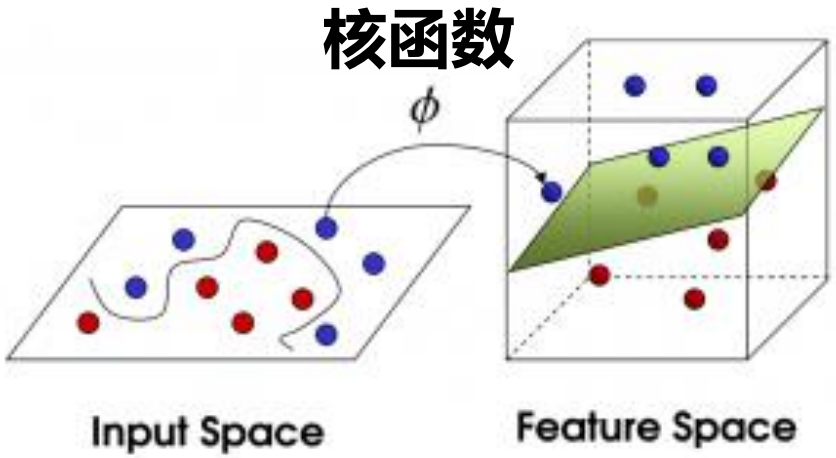
分类和回归

✓ SVM（支持向量机）

SVM就是试图把棍放在最佳位置，好让在棍的两边有尽可能大的间隙

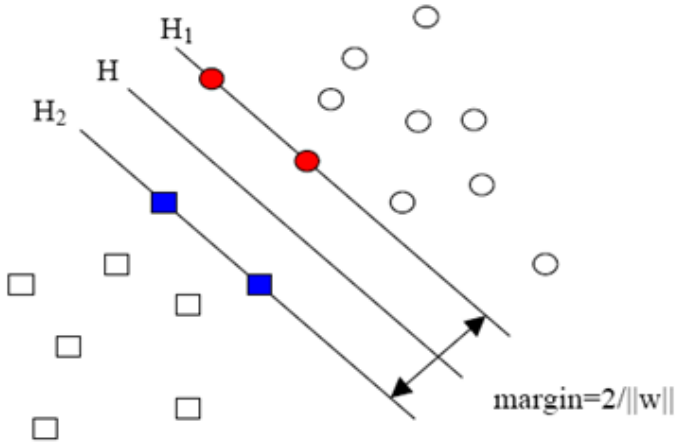






分类和回归

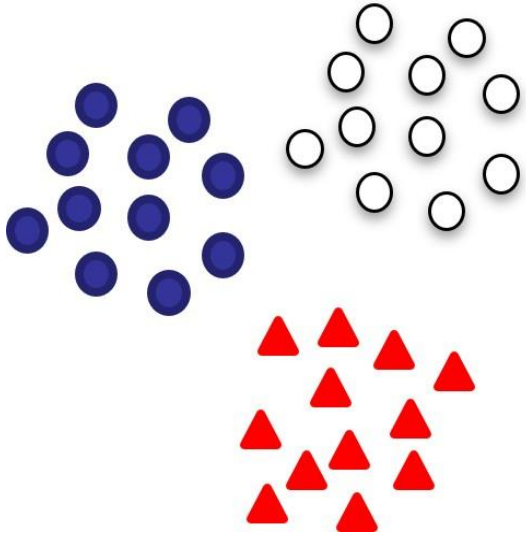
- ✓ SVM（支持向量机）
  - ➢ 全名：Support Vector Machine（支持向量机）
    - ➢ 支持向量：支撑平面上把两类类别划分开来的超平面的向量点
    - ➢ 机：一个算法



  - ➢ 基于统计学习理论的一种机器学习方法。简单的说，就是将数据单元表示在多维空间中，然后对这个空间做划分的算法。

聚类

- ✓ 图中的数据可以分成三个分开的**点集**(称为**簇**)， 一个能够分出这些点集的算法， 就被称为**聚类算法**。



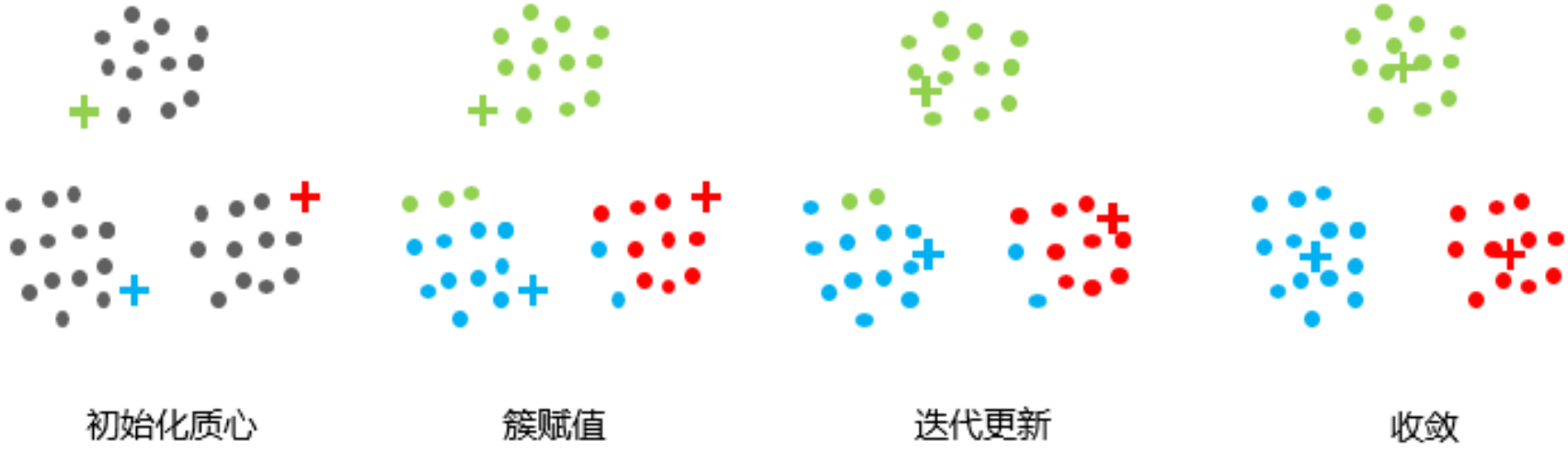
- ✓ 主要算法
  - ✓ K-means、密度聚类、层次聚类
- ✓ 主要应用
  - ✓ 市场细分、 文档聚类、 图像分割、 图像压缩、 聚类分析、 特征学习或者词典学习、 确定犯罪易发地区、 保险欺诈检测、 公共交通数据分析、 IT资产集群、 客户细分、 识别癌症数据、 搜索引擎应用、 医疗应用、 药物活性预测……

算法概述

- ✓ K-means算法是一种无监督学习方法，是最普及的聚类算法，算法使用一个没有标签的数据集，然后将数据聚类成不同的组。
- ✓ K-means算法具有一个迭代过程，在这个过程中，数据集被分组成若干个预定义的不重叠的聚类或子组，使簇的内部点尽可能相似，同时试图保持簇在不同的空间，它将数据点分配给簇，以便簇的质心和数据点之间的平方距离之和最小，在这个位置，簇的质心是簇中数据点的算术平均值

算法流程

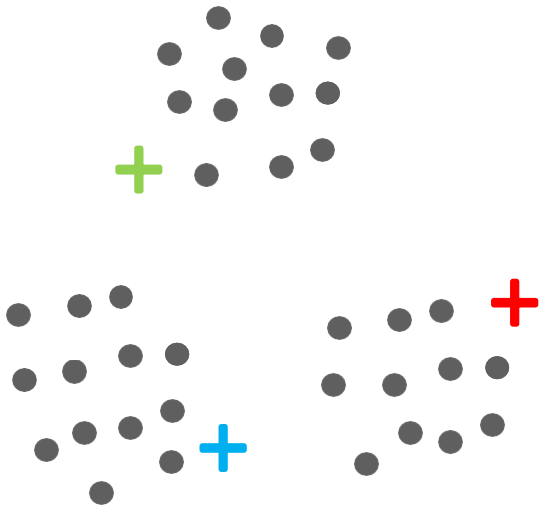
1. 选择K个点作为初始质心。
2. 将每个点指派到最近的质心， 形成K个簇。
3. 对于上一步聚类的结果， 进行平均计算， 得出该簇的新的聚类中心（新的质心）。
4. 重复上述两步/直到迭代结束：质心不发生变化。



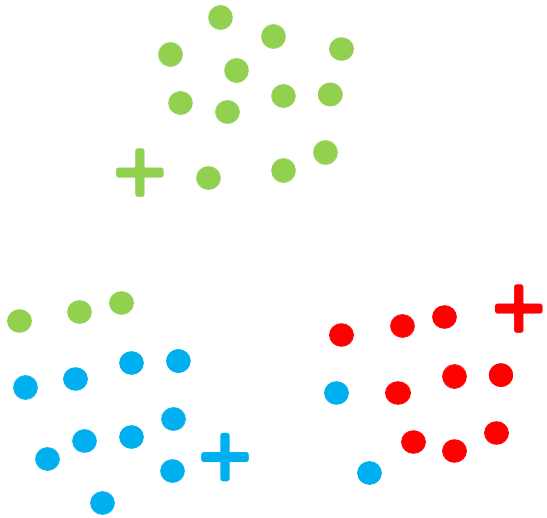
算法流程

1.初试化簇质心为的任意点。初试化时，必须注意簇的质心必须小于训练数据点的数目。

2.遍历所有数据点，计算所有质心与数据点的距离。这些簇将根据质心的最小距离而形成。



初始化质心

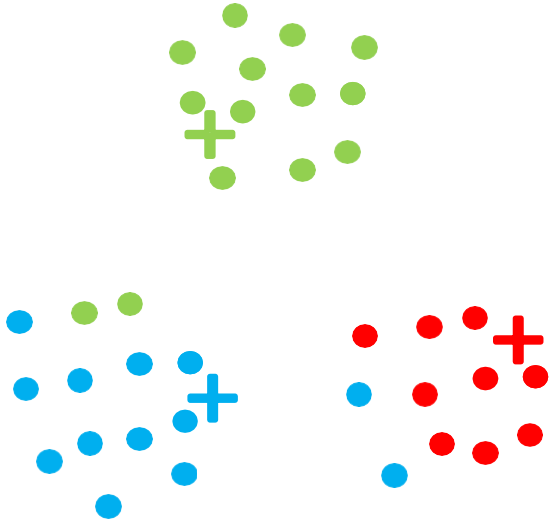


簇赋值

算法流程

3.移动质心，因为上面步骤中形成的簇没有优化，所以需要形成优化的簇。为此需要迭代地将质心移动到一个新位置。取一个簇的数据点，计算平均值，然后将簇的质心移动到这个新位置。所有簇重复相同的步骤。

4.重复上述步骤，直至收敛 。



迭代更新

聚类

优点

- ✓ 原理简单，实现容易，收敛速度快
- ✓ 聚类效果较优
- ✓ 算法的可解释度比较强
- ✓ 主要需要调参的参数仅仅时簇数K

缺点

- ✓ 需要预先指定簇的数量
- ✓ 如果有两个高度重叠的数据，那么它就无法区分，也不能判断有两个簇
- ✓ 欧几里得距离限制了能处理的数据变量类型
- ✓ 随机选择质心并不能带来理想的结果
- ✓ 无法处理异常值和噪声数据
- ✓ 不适用于非线性数据
- ✓ 对特征尺度敏感
- ✓ 如果遇到非常大的数据集，那么计算机可能回崩溃

Thank you