The Big Picture

AI and ML

It is hard to give Artificial Intelligence (AI) a precise definition. In reality, many experts define it as: **to make computers be like humans**. This word, AI, was first proposed in 1956 in Dartmouth summer workshop.

AI has 3 very popular branches

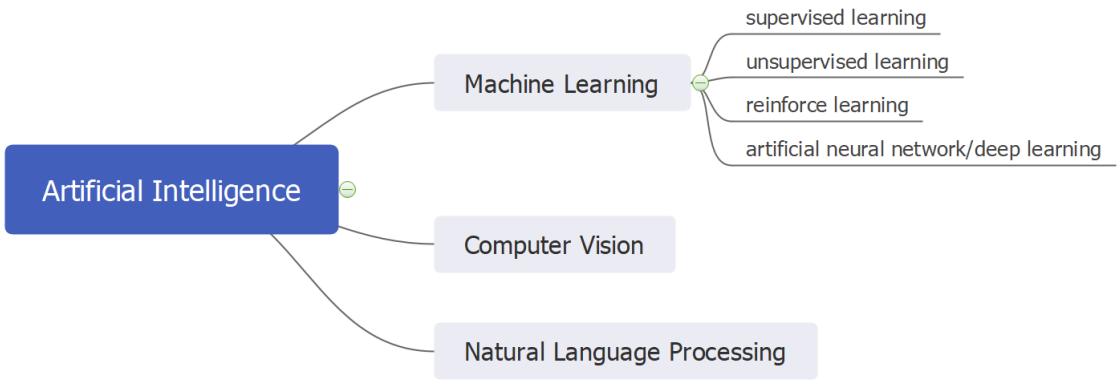
- Machine Learning (ML): **to make computers learn like humans**
  - A practical definition of learning: learning is a process where a system improves performance from experience. This definition is given by Herbert Simon (司马贺).
- Computer Vision (CV): to train computers to interpret and understand the visual world like humans
- Natural Language Processing (NLP): giving computers the ability to understand text and spoken words like humans

ML can be divided into 4 major types

- supervised learning: have human tags
- unsupervised learning: don't have human tags
- reinforce learning: learn from reward
- artificial neural network/deep learning: use artificial neural network

Some people regard reinforce learning and deep learning as same-level-concept to ML, not its subset.

Summary:



监督式学习的基本框架

机器学习、深度学习、强化学习中都是监督式学习最常见，下面是它的基本框架。

数据集D中有n个点和它们的标签：

$$(\vec{x_1}, y_1), (\vec{x_2}, y_2), \ldots, (\vec{x_n}, y_n)$$

每个点有p个特征

$$\vec{x_i} = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{bmatrix}$$

在lab5中，n=10000，p=120。

当y离散时，称之为分类问题。

当y连续时，称之为回归问题。

之后我都将使用这些记号。

西瓜书

CH1、CH2介绍了机器学习的big picture。

CH5简单介绍了深度学习。

CH16简单介绍了强化学习。

这里插一句，AlphaGo和AlphaFold用的方法都是深度学习和强化学习的结合，取得了巨大的成功。

有很多章节介绍了supervised learning

- CH3：线性模型
- CH6：支持向量机（其实支持向量机也是一种线性模型）
- CH4：决策树

有一个章节介绍了unsupervised learning

- CH9：聚类

有两个章节介绍了如何把机器学习看成贝叶斯框架下的东西（机器学习中的很多定理、算法都可以转化到贝叶斯框架下，从而有一些贝叶斯框架下的含义；或者可以转化到概率框架下，有一些概率含义；比如上次我发给你的RELIEF F算法，最开始提出时只是提出者偶然发现它好用，后来的很多研究者给了这个算法一种概率含义。）

- CH7：贝叶斯分类器
- CH14：概率图模型

CH8介绍了集成学习。集成学习可以把一些弱学习器结合起来获得更好的效果。

CH10介绍了降维、CH11介绍了特征选择。现实中，降维或特征选择非常常用。

CH13介绍了半监督学习。

CH12、CH15，我没有仔细看。

接下来我按照章节顺序写summary。

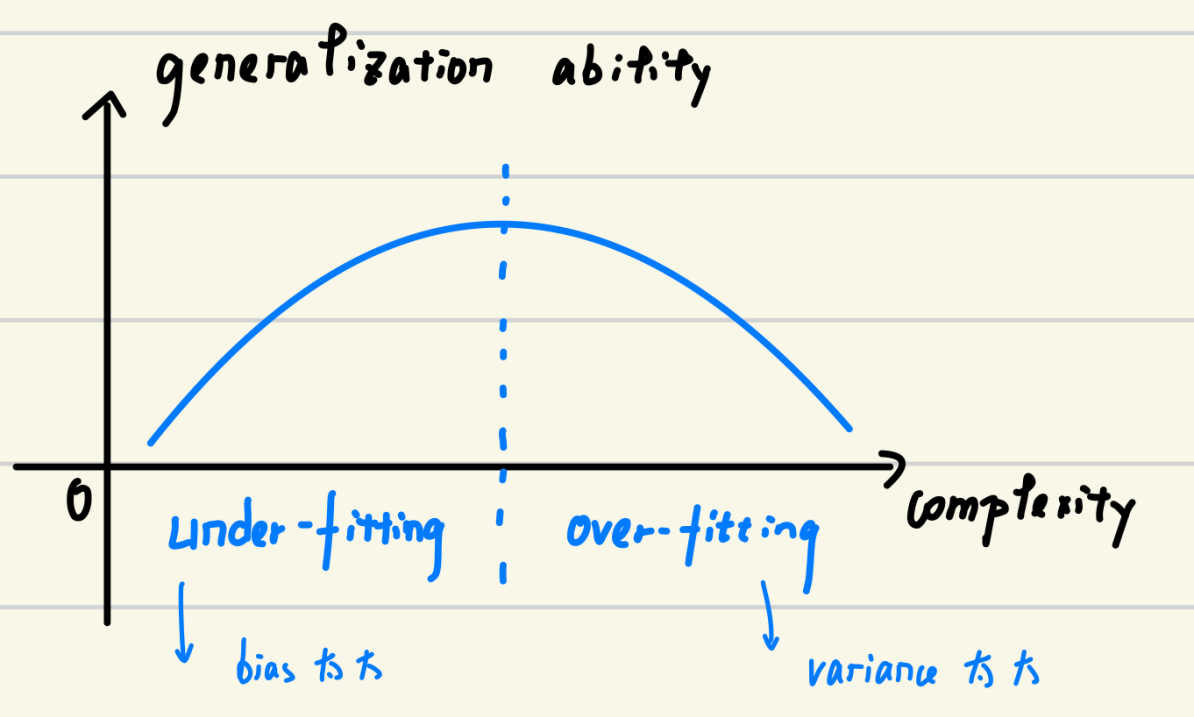
CH1

- Occam's razor: 若有多个假设与观察一致,则选最简单的那个；如无必要，勿增实体。

- No Free Lunch Theorem: Any two optimization algorithms are equivalent when their performance is averaged across all possible problems.

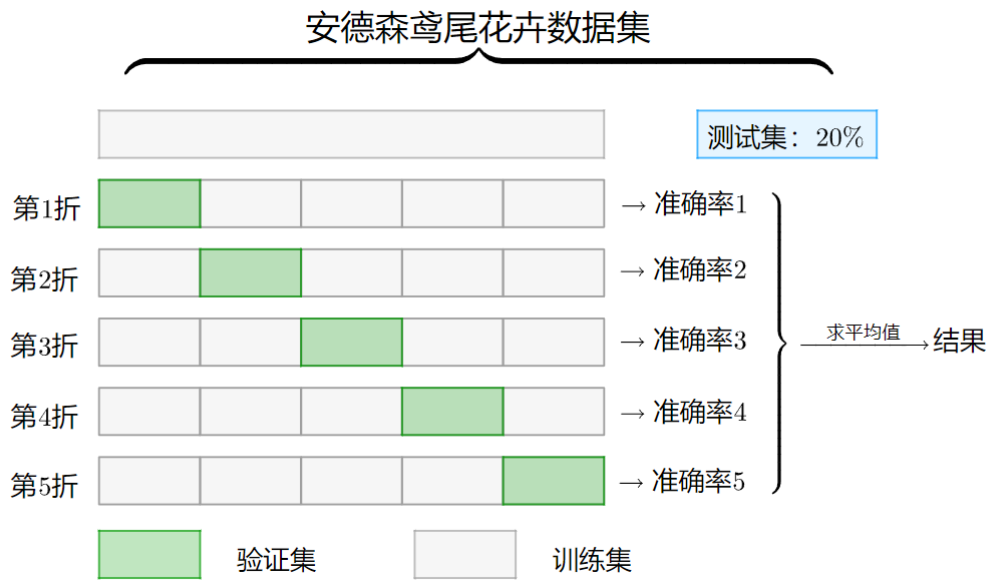
CH2

- generalization ability (泛化能力): 指一个模型对于训练集之外的数据进行预测的能力。
- bias-variance-tradeoff (偏差方差权衡): $MSE = Bias + Variance + Noise$。维基百科 的公式推导写得很好。bias-variance-tradeoff和欠拟合、过拟合有直接关系，如下图所示，欠拟合时bias太大，过拟合时variance太大。因此，我们要让模型有合适的复杂度，在bias和variance中进行权衡。



- performance measure
  - regression: MSE
  - classification
    - **accuracy (最先考察的肯定是accuracy，随后才考察下面的那些。)**
    - precision, recall, Precision-Recall graph
    - ROC, AUC
- hypothesis test: 比较不同模型的泛化能力。
- 划分数据集的方式：留出法（要做多次取平均，比如，按6：2：2随机划分训练集、验证集、测试集，重复100次）、交叉验证法（推荐10次10折交叉验证）、bootstrapping（在集成学习里常用）

注意，交叉验证也是需要留出测试集的。比如，下图中留出了20%的数据当测试集，而其余80%的数据作为训练集+测试集进行5折交叉验证。



CH3 Linear Model

顾名思义，指在监督式学习中只采用线性函数，即采用：

$$f(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

为了方便，我们定义

$$\vec{w'} := \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_p \end{bmatrix}, \vec{x'_i} := \begin{bmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix}$$

于是

$$f(\vec{x'}) = \vec{w'} \cdot \vec{x'}$$

我们也可以定义一个所谓的design matrix

$$X := \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix}$$

我们记

$$Y := \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix}$$

以上就是线性模型的基本框架。

线性模型是最简单的模型之一，通常只能解决线性问题。不过，对于非线性问题，可以通过**线性模型+神经网络**或者**线性模型+映射**来解决，后两者也是很常用的解决非线性问题的方法。核函数方法就属于后者。

有一个小区别需要注意一下：按照线性代数中对线性变换的定义，一维情况下，$y=ax$是线性变换，$y=ax+b$不是线性变换，即不能有偏置项。高维同理。也许把$\vec{w},\vec{x}$重新定义成$\vec{w'},\vec{x'}$的除了方便外的另一个原因是和线代统一。

Normal Linear Regression

以下为了方便，我把$\vec{w'},\vec{x'}$写为$\vec{w},\vec{x}$。

在普通线性回归中，model是

$$f(\vec{x})=\vec{w}\cdot\vec{x}$$

以下所有方法中，我都将用E表示要优化的目标函数。E这个字母很有意思，统计出身的做机器学习的通常把它理解成$Error$，而物理出身的做机器学习的通常把它理解成$Energy$，从而和统计物理扯上关系。也有很多学者把E称为loss function。我们把E理解成要优化的目标函数就好了。

在普通线性回归中，误差定义为

$$E(\vec{w}):=\sum_{i=1}^{n}(y_i-\vec{w}\cdot\vec{x_i})^2$$

如果按上述公式定义误差，那么普通线性回归和最小二乘法就是一个东西。

Dataset给定时，E仅是$\vec{w}$的函数。它是一个凸函数，可以求一阶导数得到全局极小。**（注意，在所有方法中，优化E时Dataset都是已知的，即$(\vec{x_1},y_1),(\vec{x_2},y_2),\dots,(\vec{x_n},y_n)$都已知）**

Generalized Linear Model

在GLM中

$$f(\vec{x})=g^{-1}(\vec{w}\cdot\vec{x})$$

当$g(x)=ln(x)$时，我们称之为log linear regression (对数线性回归/对数线性模型)

$$f(\vec{x})=e^{\vec{w}\cdot\vec{x}}$$

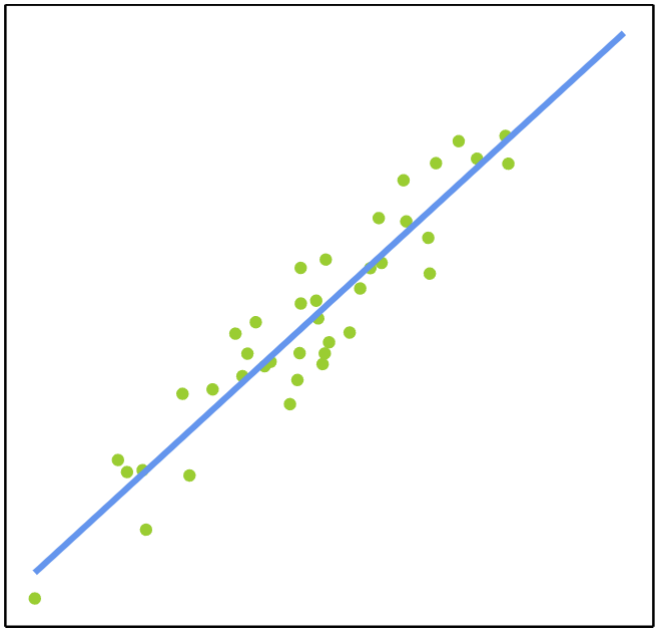
注意，对数线性回归和Logistic Regression（对数几率回归）不是一个东西，后者引入了logit（几率），前者没有。

广义线性模型的维基百科页面写得也很不错，但对考试估计帮助不大，考完试再。

From Regression to Classification

普通线性回归、对数线性回归、广义线性回归都用来解决回归问题。那么怎么用线性模型做分类问题呢?

有两种方法，第一种方法是在这三种方法的基础上，把曲线一侧的划为一类，另一侧的划为另一类，如下图所示。我在lab1中尝试了一下普通线性回归+这种方法，效果还不错，准确率和Logistic Regression差不多（就是我直接用pinv函数，你跟我说这是耍赖的那次）。



第二种方法就是Logistic Regression咯，这种方法给出了一种概率含义，详情见下。

Logistic Regression

In LR,

$$E(\vec{w}) := \sum_{i=1}^{n}(-y_i\vec{w}\cdot\vec{x}_i + ln(1+e^{\vec{w}\cdot\vec{x}_i})), when\ y \in \{0,+1\}$$

It's hard to calculate $\frac{\partial E}{\partial \vec{w}}$ (According to Wikipedia, it doesn't have a close-form solution), so we use numerical method like gradient descent and Newton method to get the optimal solution $\hat{w}$.

E的推导和LR的概率含义见附录。

LASSO and Ridge Regression

为了防止过拟合，我们可以给LS和LR加上惩罚项。惩罚项通过控制$\vec{w}$的范数来降低模型复杂度。

LS+LASSO

$$E(\vec{w}) := \sum_{i=1}^{n}(y_i - \vec{w}\cdot\vec{x}_i)^2 + \lambda||\vec{w}||_1^1$$

LS+Ridge

$$E(\vec{w}) := \sum_{i=1}^{n}(y_i - \vec{w}\cdot\vec{x}_i)^2 + \lambda||\vec{w}||_2^2$$

LR+LASSO

$$E(\vec{w}) := \sum_{i=1}^{n}(-y_i\vec{w}\cdot\vec{x}_i + ln(1+e^{\vec{w}\cdot\vec{x}_i})) + \lambda||\vec{w}||_1^1$$

LR+Ridge

$$E(\vec{w}) := \sum_{i=1}^{n}(-y_i\vec{w}\cdot\vec{x}_i + ln(1+e^{\vec{w}\cdot\vec{x}_i})) + \lambda||\vec{w}||_2^2$$

把惩罚项显式地写出来（注意Ridge Regression的惩罚项是L2范数的平方，别把平方丢了。）

$$||\vec{w}||_1^1 = \sum_{j=1}^{p} |w_j|$$

$$||\vec{w}||_2^2 = \sum_{j=1}^{p} w_j^2$$

PCA

PCA是K.P. Pearson发明的老古董。

PCA的思想：把样本从高维空间投影到低维空间，使得在低维空间中所有样本尽可能分开。

做到这一点的两个思路

- 最小化所有样本到低维空间的总距离
- 最大化所有样本在低维空间的方差

这两个思路是等价的（其实就是勾股定理）。可以看看这个简单直观的YouTube视频。

在PCA中，优化问题见(10.15)或者(10.16)。

利用拉格朗日乘数法，我们可以得到PCA的一个简单的算法：求$X^T X$的所有特征值，从大到小排序，取前d'个特征值，它们的特征向量组成了一个d'维的低维空间。通常，特征值的大小代表了重要程度。

d'也可以取为p，这时我们相当于我们做了一个同维度的线性变换。比如，在lab5中，直接调库会得到$PC_1, PC_2, \ldots, PC_{120}$。

LDA

和PCA类似，LDA是R.A. Fisher发明的老古董。

LDA的思想：把样本从高维空间投影到低维空间，使得在低维空间中同类样本尽可能近且异类样本尽可能远。

在LDA中，优化问题见(3.36)

LASSO、Ridge Regression、PCA、LDA都可以用来做降维or特征选择。

Supporting Vector Machine

SVM你学得比我懂，我这里只列出一些重点：

- 原优化问题
- 对偶优化问题
  - SMO算法
- 软间隔SVM：相当于硬间隔SVM+惩罚项
- 核函数：使得SVM可以处理非线性问题（PCA、LDA也可以用类似的核函数方法）
- SVR：把SVM用于回归

PS：离SVM的分类直线最近的向量称为支持向量，这是SVM名字的来源。

二分类拓展到多分类

三种策略，各有优缺点

- one vs one
- one vs rest
- many vs many

类别不平衡

这个考试应该没法考，但是在实际中是一个很重要的事情，我们在实验室里也需要考虑这个问题。

见书3.6节，书写得还是比较清楚的。

我在lab1中试了一下，lab1中类别不平衡不是特别显著，所以考不考虑影响不大。

CH4 Decision Tree

这一章不难，我迅速带过。

- 我们希望决策树得分支节点所包含的样本尽可能属于同一类别，为此我们需要定量衡量一个集合的纯度。
  - Shannon Entropy
  - Gini index
- 有3个主流的算法
  - ID3：用香农熵增益来选择属性
  - C4.5：用香农熵增益率来选择属性
  - CART：用基尼指数来选择属性
- 剪枝：为了防止过拟合
  - pre-pruning
  - post-pruning

PS：决策树是贪心算法——每次只选择对当前最有利的划分。

CH5 Neural Network

- MP神经元模型: 即$y = f(\vec{w} \cdot \vec{x} + b)$
  - f为激活函数，通常为ReLU或者Sigmoid
  - b可以为0
- 感知机：只有输入层、输出层的神经网络
  - 如果选择Sigmoid为激活函数，那么感知机和Logistic Regression的公式一模一样

    - 感知机、LR都只能处理线性问题。
    - 只有加入隐藏层，Neural Network才能处理非线性问题。
    - 历史上，BP算法发明前，由于感知机只能处理线性问题且加入隐藏层之后的神经网络没法训练，神经网络陷入冰河期。
- 前馈神经网络
    - E见（5.4）
    - 优化E的方法：梯度下降+BP算法
    - 通常用随机梯度下降来避免陷入局部极小。

这一章的内容参考我之前发给的文件。

CH8 Ensemble Learning

- 有两种方法
    - bagging
        - 并行：基学习器之间不存在强依赖关系
        - eg：随机森林
    - boosting
        - 串行：后一个基学习器依赖于前一个基学习器的结果
        - eg: AdaBoost, XGboost
- 投票：见8.4
- 集成学习的理论分析：见8.5，指出了集成学习如何实现"好而不同"

CH9 Clustering

- 聚类是一种无监督学习，也是这本书里唯一的一个无监督学习
- 聚类的性能度量（类似于监督式学习中的E）：见9.2
- 选择合适的距离：见9.3
- 具体方法
    - prototype-based clustering：用一组原型刻画，基于原型聚类
        - K-means
        - Learning Vector Quantization
        - 高斯混合模型
    - density-based clustering：基于密度聚类
        - DBSCAN
        - 缺点：只能聚球形的
    - 层次聚类：我们在数据结构里学过类似的东西
    - lab4（这个方法真的很帅）

CH10 Dimensionality Reduction

特征选择和降维是解决维度诅咒的两个方法。

做过lab5的你对CH10和CH11一定印象深刻，我就列个目录了。它们也都不难，耐心看。

- Multiple Dimensional Scaling
- PCA
- Kernelized PCA
- Manifold Learning
  - Isomap
  - LLE
- Metric Learning

CH11 Feature Selection

- 过滤法
  - RELIEF
- 包裹法
  - LVW
- 嵌入法
  - LASSO
  - 决策树
  - 随机森林

其实我感觉包裹法和嵌入法不用区分得太开

CH11最后介绍了字典学习和压缩感知，你在凸优化是应该学过后者。

CH13 半监督学习

- 需求：我们需要利用没有标签的样本，这个需求在现实中很常见
  - 主动学习：让人来判断
  - 半监督学习：不用人来判断
- 半监督学习（不是很难，耐心看）
  - 生成式模型
  - 半监督SVM
  - 图半监督学习
  - disagreement based methods
  - 半监督聚类

CH7 Bayes

- 生成式模型vs判别式模型：P148
- 假设有p个特征
  - not naive Bayes需要的数据正比于2^p
  - naive Bayes需要的数据正比于p
- naive Bayes
  - 引入属性条件独立假设的贝叶斯分类器称为naive Bayes
  - 没办法，数据就这么多，只能假设条件独立
  - 拉普拉斯平滑：给每个小格子加上1
- semi naive Bayes
  - 在现实中，属性条件独立假设难以成立，于是人们开发出半朴素贝叶斯
  - One Dependent：每个属性仅仅依赖于1个属性
- Bayes Network：不难，耐心看，见7.5
- EM算法：老师说必考，见7.6

CH14 概率图模型

这一章确实较难，但是也确实很有用，我们实验室的几个师兄经常用。

CH14我学得不是很懂，但我相信你能看懂~

附录

Normal Linear Regression的解析解

在普通线性回归中，E的定义如下

$$E(\vec{w}) := \sum_{i=1}^{n}(y_i - \vec{w}\cdot\vec{x}_i)^2 = Y^TY + w^TX^TXw - 2w^TX^TY$$

求一阶导，我们得到

$$\frac{\partial E}{\partial \vec{w}} = 2(X^TXw - X^TY)$$

于是

$$\frac{\partial E}{\partial \vec{w}} = 0 \Rightarrow X^TX\hat{w} = X^TY$$

当X^TX可逆时

$$\hat{w} = (X^TX)^{-1}X^TY = X^{\dagger}Y$$

where $X^{\dagger} := (X^TX)^{-1}X^T$ and $X^{\dagger}$ is Moore-Penrose inverse, as we learned in YYN's class.

当X^TX不可逆时，$X^TX\hat{w}=X^TY$这个线性方程组有多个解，选择哪一个作为结果由算法的偏好决定。

线性代数中有一个很有名的定理，杨亚宁也讲过：X列满秩 $\Leftrightarrow$ X^TX可逆

X的行数越多，线性无关的行向量**很可能**越多。线性无关的行向量越多，行秩、秩、列秩越大，X越容易列满秩。

以下是三个例子：

- **在lab5中，X有10000行和121列，X大概率列满秩。**
- **在波士顿房价数据集中，X有506行和14列，X大概率列满秩。**
- 在生物信息学中，通常feature数目比sample多，X大概率列不满秩。

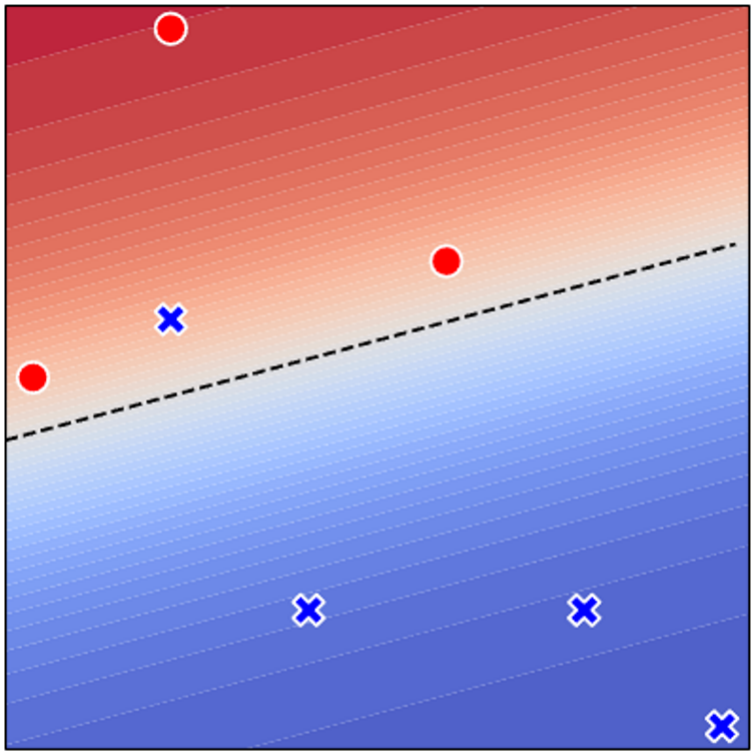
Logistic Regression的概率含义和E的推导

LR中，我们假设

$$\left\{\begin{array}{l} P(y=+1|\vec{x},\vec{w},b)=\frac{1}{1+e^{-\vec{w}\cdot\vec{x}}} \\ P(y=0|\vec{x},\vec{w},b)=\frac{1}{1+e^{+\vec{w}\cdot\vec{x}}} \end{array}\right\}$$

注意，这是一个假设。

正是这个假设赋予了LR概率含义。$P(y=+1|\vec{x},\vec{w},b)$即样本$\vec{x}$是正例的概率，$P(y=0|\vec{x},\vec{w},b)$即样本$\vec{x}$是反例的概率。如下图所示，离分类直线越远，属于对应类的概率越大。



逻辑回归

接下来推导E：

把伯努利分布的两个P写到一起

$$P(y_i|\vec{x}_i,\vec{w},b) = (P(y_i=+1|\vec{x}_i,\vec{w},b))^{y_i}(P(y_i=0|\vec{x}_i,\vec{w},b))^{1-y_i}$$

似然函数为

$$likelihood(Y|X,\vec{w},b) = \prod_{i=1}^{n} P(y_i|\vec{x}_i,\vec{w},b)$$

对数似然函数为

$$\begin{aligned} ln(likelihood(Y|X,\vec{w},b)) &= \sum_{i=1}^{n} ln(P(y_i|\vec{x}_i,\vec{w},b)) \\ &= \sum_{i=1}^{n} y_i\vec{w}\cdot\vec{x}_i - ln(1+e^{\vec{w}\cdot\vec{x}_i}) \end{aligned}$$

最大化对数似然函数等价于最小化负对数似然函数，因此我们可以把E定义为

$$E := \sum_{i=1}^{n} -y_i\vec{w}\cdot\vec{x}_i + ln(1+e^{\vec{w}\cdot\vec{x}_i})$$

懒惰学习 vs 急切学习

kNN是我们学过的唯一一个懒惰学习方法，其它全都是急切学习。

CH7提到了一次懒惰学习，但是没有展开。

一些常见的数据集

- Iris: R.A. Fisher玩鸢尾时候留下的数据集
- handwritten digits: 手写数字识别，很适合神经网络
- 波士顿房价数据集：用线性模型即可
- 泰坦尼克号数据集：适合练练特征选择

还有很多toy dataset，见sklearn的官方说明文档。