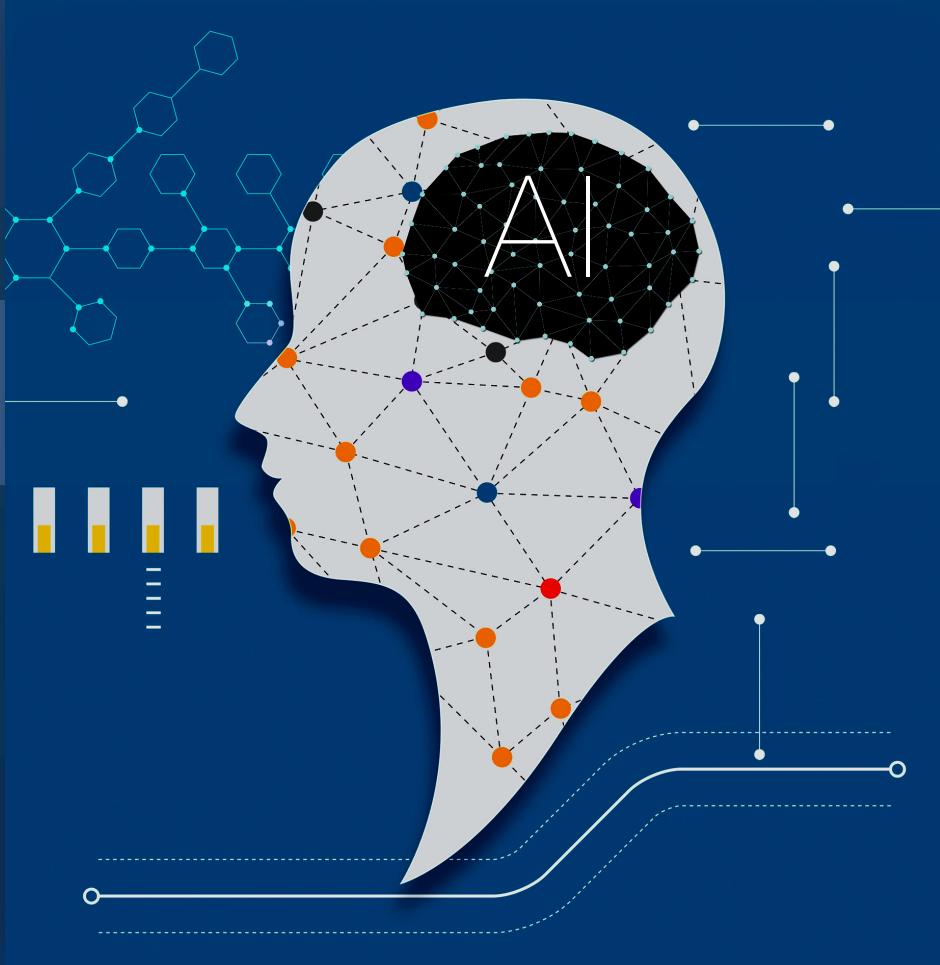




机器学习基础



大纲

- 线性回归
- Basis Expansion
- 正则化Regularization
- Bias-Variance Trade-off
- Ridge, Lasso, ElasticNet
- 逻辑回归
- Softmax Classifier
- 梯度下降算法

线性回归

线性回归的数学定义

- 数据 $(Y_i, X_{i1}, \dots, X_{ip}), i = 1, \dots, n$
- 模型 $Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \dots + \beta_p X_{ip} + \varepsilon_i, \quad i = 1, \dots, n$

线性回归的数学定义(矩阵表达)

- 数据 $(Y_i, X_{i1}, \dots, X_{ip}), i = 1, \dots, n$

- 模型 $Y = X\beta + \varepsilon$

Y 是一个包括了观测值的列向量 Y_1, \dots, Y_n

β 是一个包括了参数值的列向量 β_0, \dots, β_p

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

线性回归的数学解析解(矩阵表达)

- 数据 $(Y_i, X_{i1}, \dots, X_{ip}), i = 1, \dots, n$

- 模型 $Y = X\beta + \varepsilon$

- 解析解 $\hat{\beta} = (X^T X)^{-1} X^T Y$

线性回归解析解的推导

- $\mathcal{L}(\beta) = \|X\beta - Y\|^2 = (X\beta - Y)^T (X\beta - Y) = Y^T Y - Y^T X\beta - \beta^T X^T Y + \beta^T X^T X\beta$
- $\nabla_{\beta} \mathcal{L}(\beta) = \nabla_{\beta} (Y^T Y - Y^T X\beta - \beta^T X^T Y + \beta^T X^T X\beta) = 0 - X^T Y - X^T Y + 2X^T X\beta = 2X^T X\beta - 2X^T Y$
- 令 $\nabla_{\beta} \mathcal{L}(\beta) = 0$, 即 $2X^T X\beta - 2X^T Y = 0$
- $\Rightarrow X^T X\beta = X^T Y$
- $\Rightarrow \beta = (X^T X)^{-1} X^T Y$

线性回归损失函数的梯度

- $J(\beta) = \frac{1}{2n} \sum_{i=1}^n (\beta^T x^i - y^i)^2$
- $\nabla_{\beta} J(\beta) = \nabla_{\beta} \left[\frac{1}{2n} \sum_{i=1}^n (\beta^T x^i - y^i)^2 \right] = \frac{1}{2n} \sum_{i=1}^n \nabla_{\beta} (\beta^T x^i - y^i)^2$
- $= \frac{1}{2n} \sum_{i=1}^n 2(\beta^T x^i - y^i) \nabla_{\beta} (\beta^T x^i - y^i) = \frac{1}{n} \sum_{i=1}^n (\beta^T x^i - y^i) x^i$

线性回归的概率理解

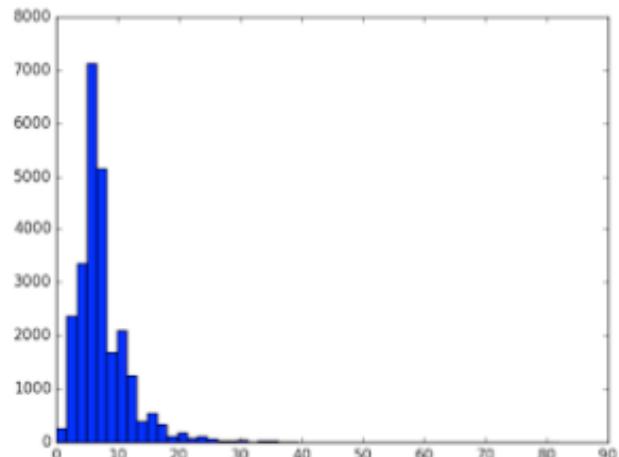
- $y^i = \beta^T x^i + \varepsilon$
- Maximum Likelihood: $P(y^i | x^i, \beta, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y^i - \beta^T x^i)^2}{2\sigma^2}}$
- $\log[P(D|\beta, \sigma)] = \log \left\{ \left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \prod_{i=1}^n e^{-\frac{(y^i - \beta^T x^i)^2}{2\sigma^2}} \right\}$
- 针对 w 最大化
 - $\arg \max_w \log \left\{ \left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \prod_{i=1}^n e^{-\frac{(y^i - \beta^T x^i)^2}{2\sigma^2}} \right\} = \arg \max_w \log \left[\left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \right] + \sum_{i=1}^n -\frac{(y^i - \beta^T x^i)^2}{2\sigma^2}$
 - $= \arg \max_w \sum_{i=1}^n \frac{-(y^i - \beta^T x^i)^2}{2\sigma^2} = \arg \max_w \sum_{i=1}^n -(y^i - \beta^T x^i)^2$
 - $\arg \min_w \sum_{i=1}^n (y^i - \beta^T x^i)^2$

线性回归总结

- **线性假设:** 线性回归假设输入和输出之间的关系是线性的。您可能需要转换数据以使关系成为线性关系（例如，指数关系的对数转换）。
- **消除噪音:** 线性回归假设您的输入和输出变量没有噪声。请考虑使用数据清理操作，以便更好地阐明数据中的信号。噪声对输出变量影响最大，如果可能，您需要删除输出变量 (y) 中的异常值。
- **删除共线性(Collinearity):** 当您具有高度相关的输入变量时，线性回归将过度拟合您的数据。考虑计算输入数据的成对相关性并删除最相关的数据。
- **高斯分布:** 如果输入和输出变量具有高斯分布，则线性回归将进行更可靠的预测。您可以使用变换（例如log或BoxCox）在变量上获得一些好处，使其分布更加高斯。
- **重新缩放输入 :** 如果使用标准化或规范化重新缩放输入变量，则线性回归通常会进行更可靠的预测。

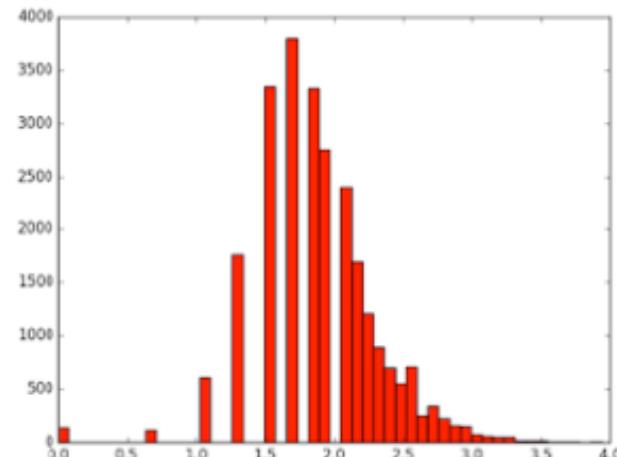
Boxcox

Before



Accuracy = 62%

After



Accuracy = 80%

$$y_i^{(\lambda)} = \begin{cases} \frac{(y_i + \lambda_2)^{\lambda_1} - 1}{\lambda_1} & \text{if } \lambda_1 \neq 0, \\ \ln(y_i + \lambda_2) & \text{if } \lambda_1 = 0, \end{cases}$$

Basis Expansion, Regularization

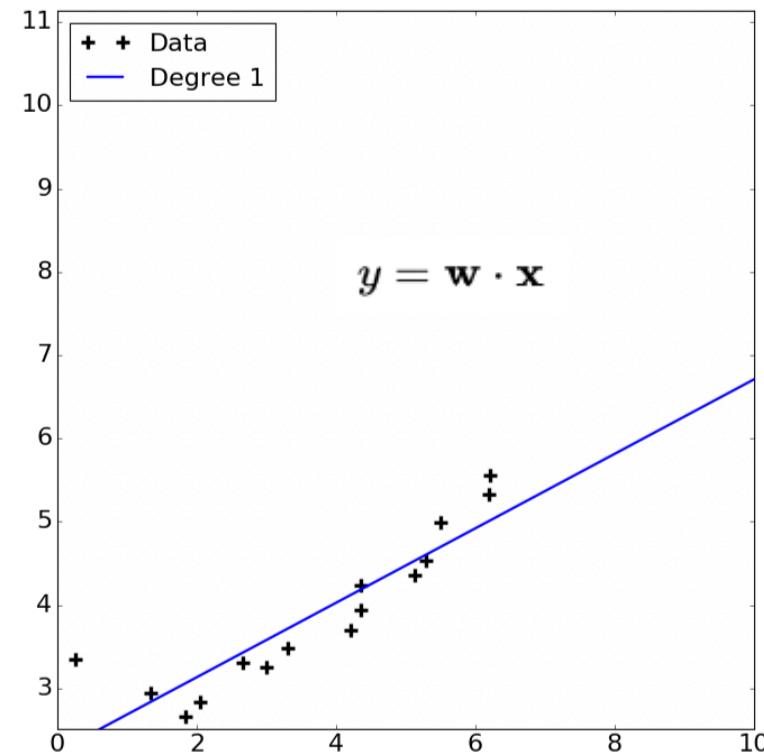
Basis Expansion, Regularization

- 使用 Basis expansion 捕捉自变量和因变量的非线性关系
- 理解 bias-variance 的关系
- 过拟合与正则化 Overfitting and Regularization

Basis Expansion

- Basis Expansion是指通过对数据进行转换来扩充/替换数据集的特征。例如，给定输入特征 X ，Basis Expansion可以将此特征映射到三个特征： $1, X, X^2$ “多项式Basis Expansion”。这种映射允许各种学习算法捕获数据中的非线性趋势，同时仍使用线性模型来分析这些转换后的特征。例如，将多项式Basis Expansion与线性回归结合使用，可以使线性回归找到数据中的多项式（非线性）趋势；这通常称为“多项式回归”。

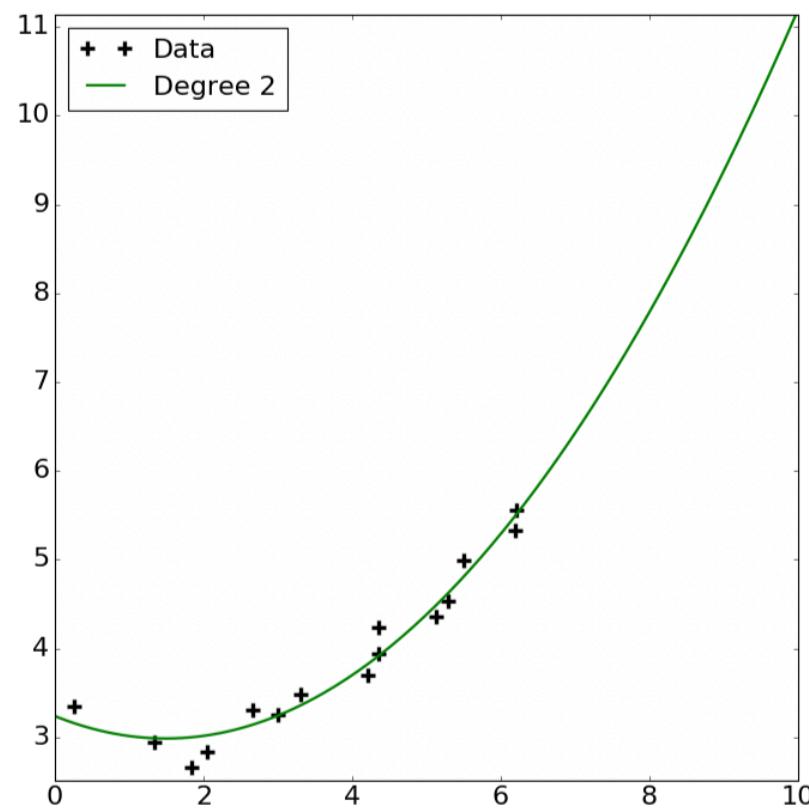
Linear Regression : Polynomial Basis Expansion(1)



Linear Regression : Polynomial Basis Expansion (2)

$$\phi(x) = [1, x, x^2]$$

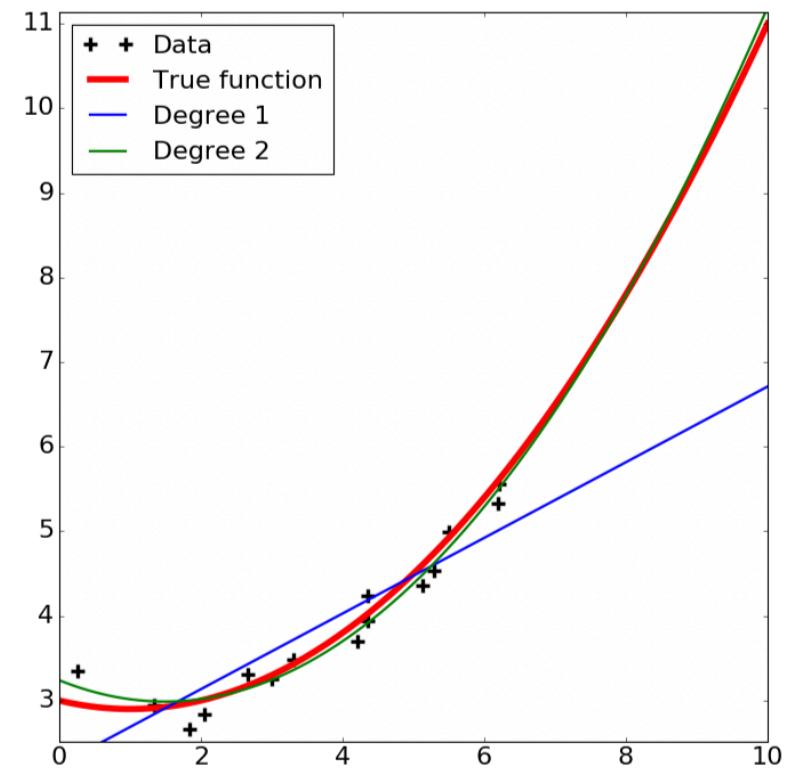
$$w_0 + w_1x + w_2x^2 = \phi(x) \cdot [w_0, w_1, w_2]$$



Linear Regression : Polynomial Basis Expansion (3)

$$\phi(x) = [1, x, x^2]$$

$$w_0 + w_1 x + w_2 x^2 = \phi(x) \cdot [w_0, w_1, w_2]$$

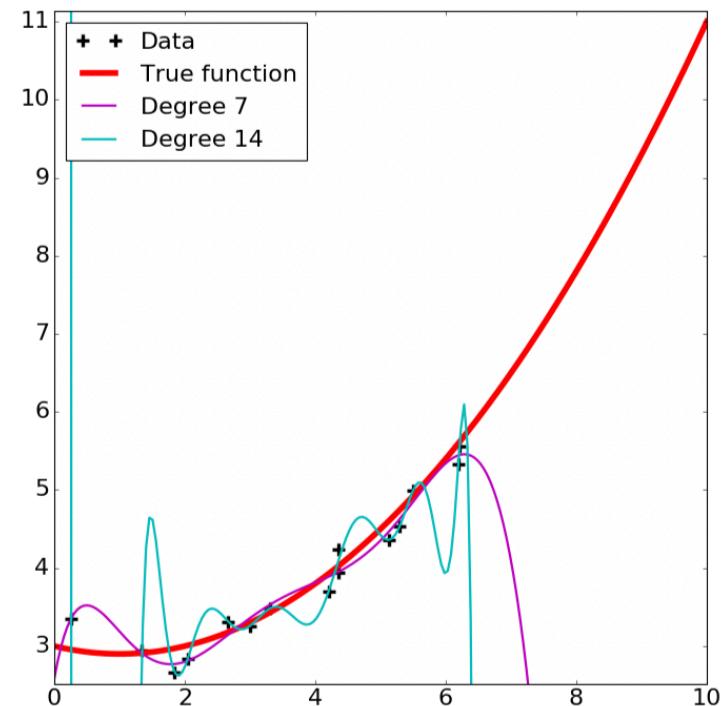


Linear Regression : Polynomial Basis Expansion (4)

$$\phi(x) = [1, x, x^2, \dots, x^d]$$

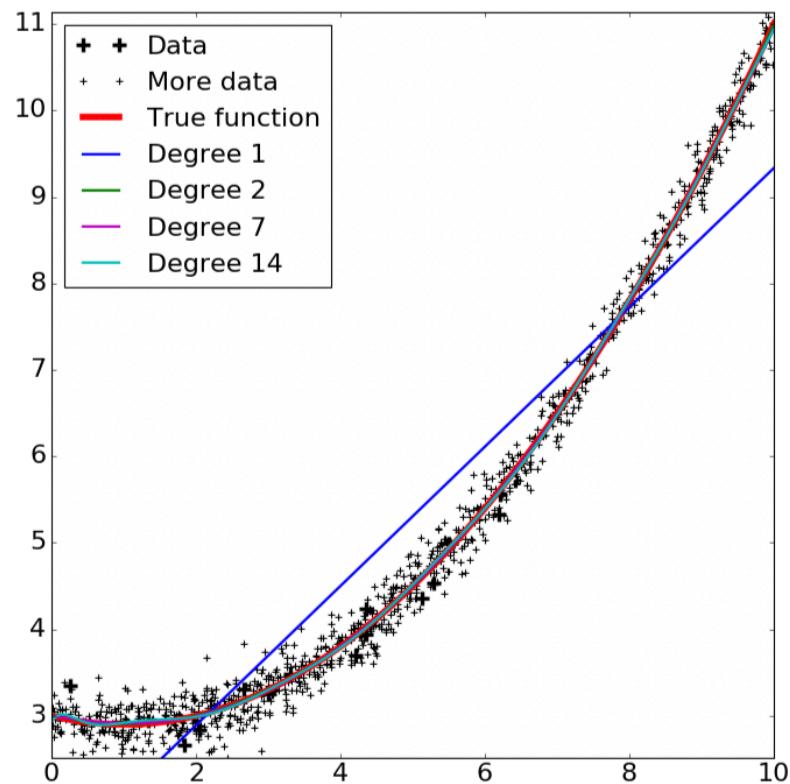
$$\text{Model } y = \mathbf{w}^\top \phi(x) + \epsilon$$

Here $\mathbf{w} \in \mathbb{R}^M$, where M is the number for expanded features



Linear Regression : Polynomial Basis Expansion (5)

- 获取更多训练数据防止过拟合.

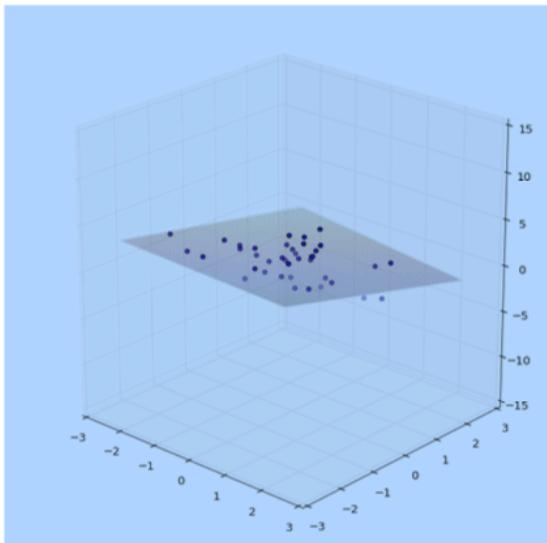


Polynomial Basis Expansion in Higher Dimensions

$$y = \mathbf{w} \cdot \phi(\mathbf{x}) + \epsilon$$

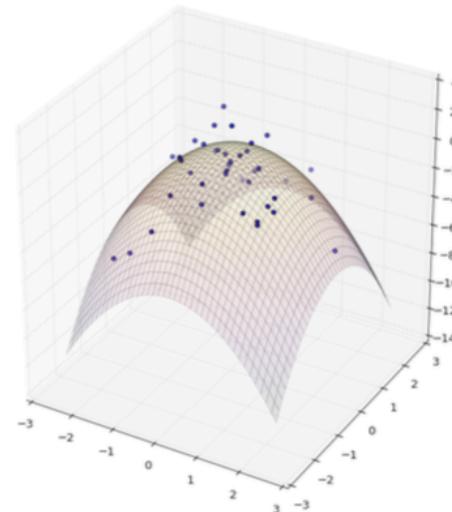
Linear Model

$$\phi(\mathbf{x}) = [1, x_1, x_2]$$



Quadratic Model

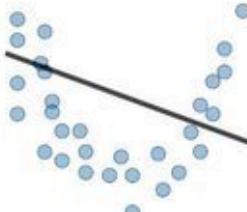
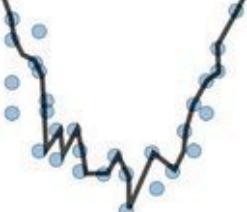
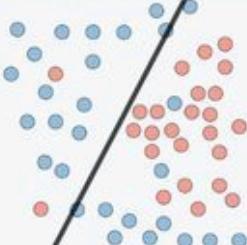
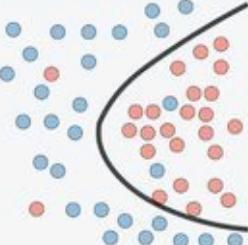
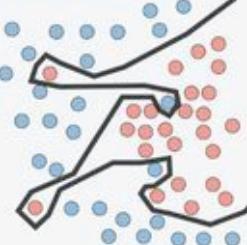
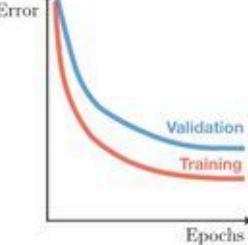
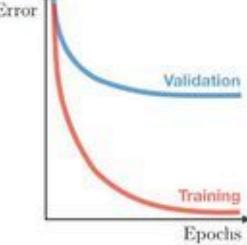
$$\phi(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_2^2, x_1 x_2]$$



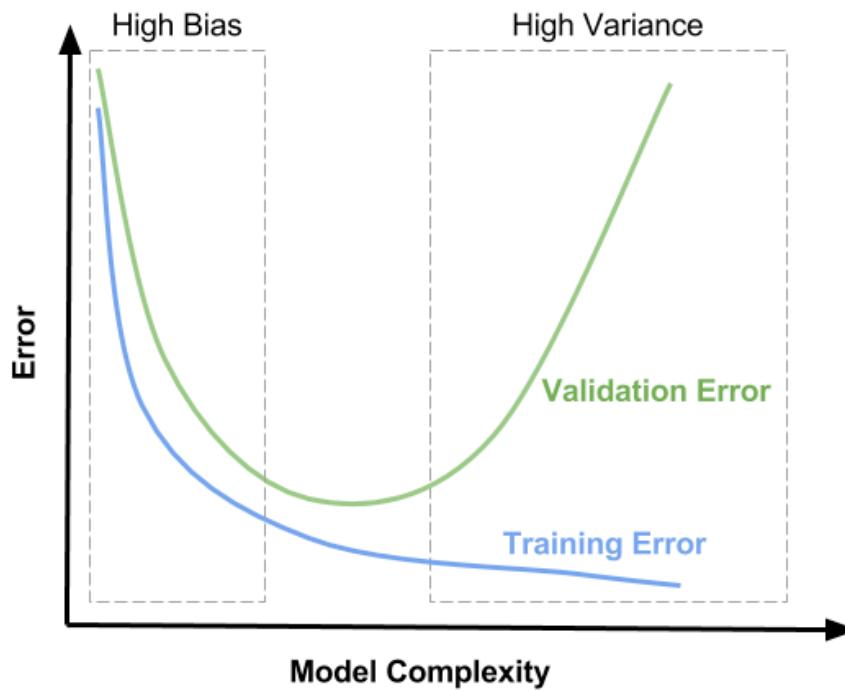
Using degree d polynomials in D dimensions results in $\approx D^d$ features!

Bias-Variance Trade-off

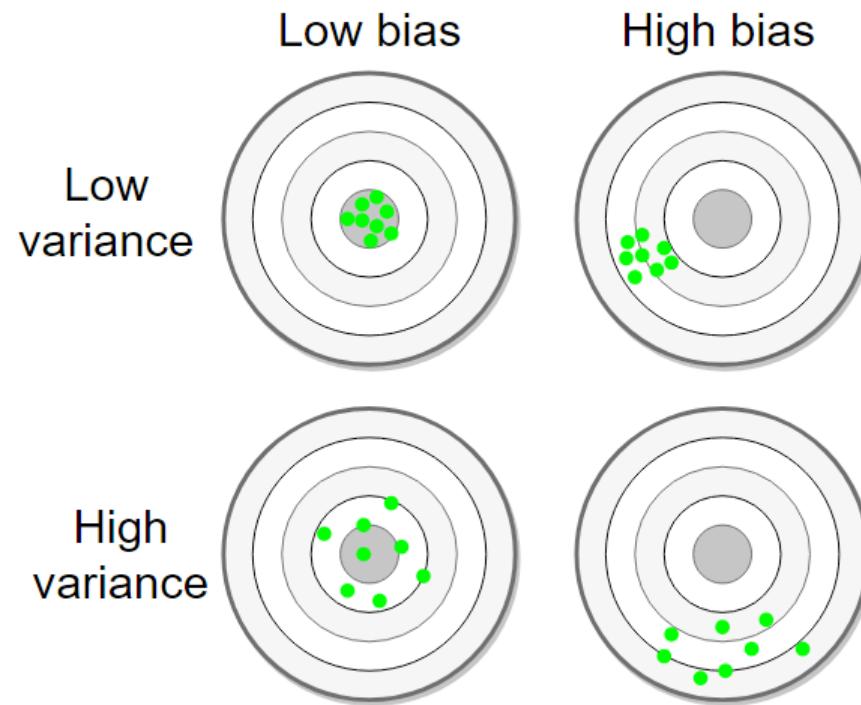
Underfitting Overfitting Bias Variance

| | Underfitting | Just right | Overfitting |
|-----------------------------|--|---|---|
| Symptoms | <ul style="list-style-type: none">• High training error• Training error close to test error• High bias | <ul style="list-style-type: none">• Training error slightly lower than test error | <ul style="list-style-type: none">• Very low training error• Training error much lower than test error• High variance |
| Regression illustration |  |  |  |
| Classification illustration |  |  |  |
| Deep learning illustration |  |  |  |
| Possible remedies | <ul style="list-style-type: none">• Complexify model• Add more features• Train longer | | <ul style="list-style-type: none">• Perform regularization• Get more data |

Bias/Variance V. S. Model Complexity



Bias/Variance



每一个小绿点代表使用不同训练数据,
在同一个模型下训练以后得到的结果

Bias

- 给定保护 n 个数据的数据集 D
- 在不同的数据集 D 上训练出一个不同的模型 $h(x)$
- 期望值 (在所有的 $h(x)$ 上的平均): $E_D[h(x)]$
- Bias: 期望值与真实值的差异
 - 衡量模型和真实情况的差异
 - 随着模型复杂度增加, Bias减少

$$\text{bias}^2 = \int_x \{E_D[h(x)] - t(x)\}^2 p(x) dx$$

Variance

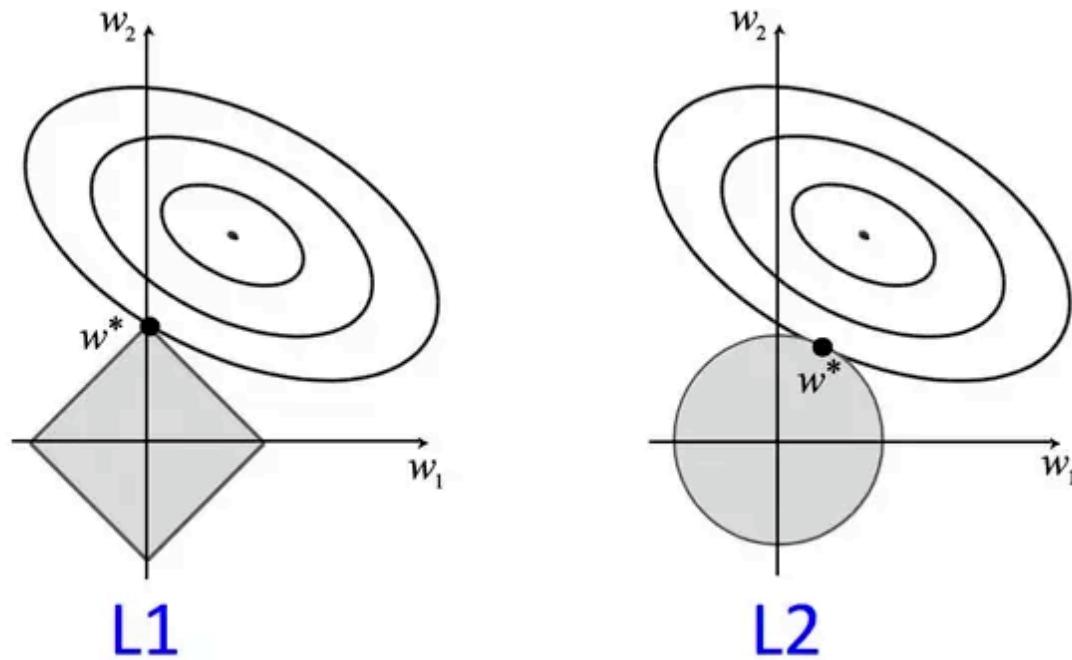
- 给定保护 n 个数据的数据集 D
- 在不同的数据集 D 上训练出一个不同的模型 $h(x)$
- 期望值 (在所有的 $h(x)$ 上的平均): $E_D[h(x)]$
- Variance: 在一个特定数据集上训练的模型与所有模型的期望的差异
 - 衡量模型对特定数据集的变化的敏感度
 - 随着模型复杂度增加, Variance增加

$$\overline{h(x)} = E_D[h(x)]$$

$$variance = \int_x E_D \left[(h(x) - \overline{h(x)})^2 \right] p(x) dx$$

正则化(Regularization)

正则化：对机器学习算法的任何修改，以减少其泛化错误 (generalization error)，但不减少其训练错误 (training error)



Regularization

- **Cost function = Loss + Regularization term**
- 在等式中添加正则化项，通过抑制模型中的系数值防止overfitting。正则化的假设是较小的权重对应更简单的模型，从而有助于防止overfitting。
- 由于正则化项的添加，使得权重的值减小，假设具有较小权重的模型对应更简单的模型。因此，它将在很大程度上减少overfitting。

Regularization(续)

$$J(\mathbf{w}) = L(\mathbf{w}) + \|\mathbf{w}\|_p$$

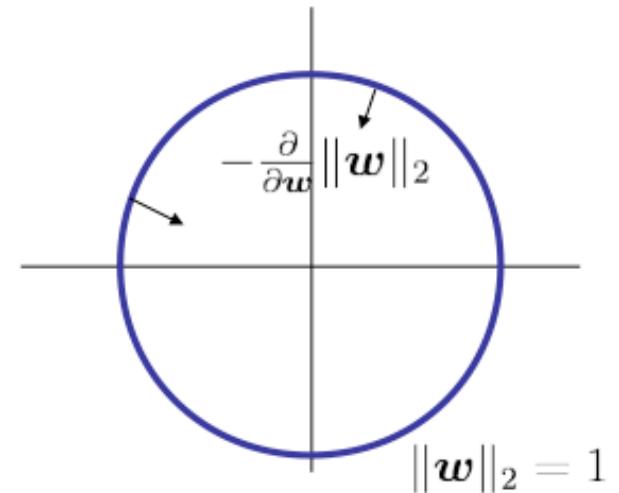
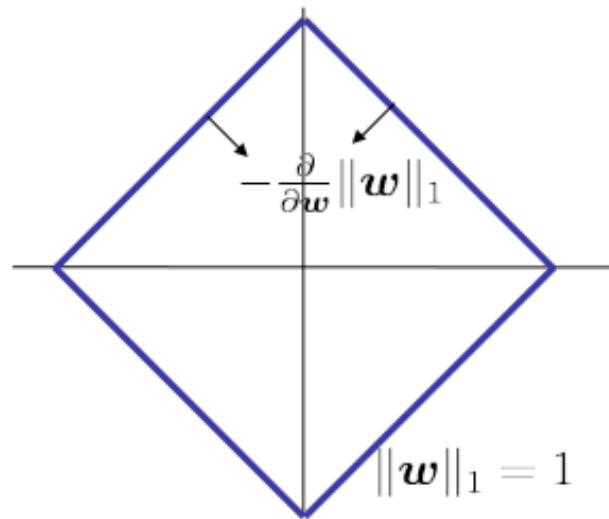
- **Norm** 是一种定义向量大小的方法。标准的一般表示法是这样的

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}$$

L1/L2

$$\|\mathbf{w}\|_1 = \sum_{f=0}^d |w_f|$$

$$\|\mathbf{w}\|_2 = \sqrt{\sum_{f=0}^d w_f^2}$$



$$\frac{dL_1(w)}{dw} = sign(w), \text{ where } sign(w) = \left(\frac{w_1}{|w_1|}, \frac{w_2}{|w_2|}, \dots, \frac{w_m}{|w_m|} \right)$$

$$\frac{dL_2(w)}{dw} = w$$

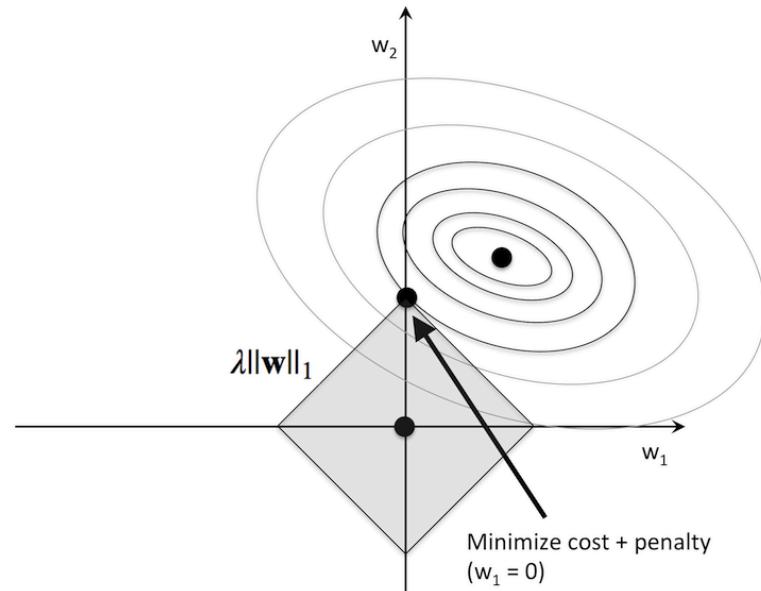
L1-regularization 特点

- 不容易计算, 在零点连续但不可导, 需要分段求导。
- L1模型可以将一些权值缩小到零 (稀疏) 。
- 执行隐式变量选择。这意味着一些变量值对结果的影响将为零, 就像删除它们一样。
- 其中一些预测因子对应较大的权值, 而其余的 (几乎) 归零。
- 由于它可以提供稀疏的解决方案, 因此通常是建模特征数量巨大时的首选模型。在这种情况下, 获得稀疏解决方案具有很大的计算优势, 因为可以简单地忽略具有零系数的特征。
- 它任意选择高度相关的特征中的任何一个, 并将其余特征对应的系数减少到零。此外, 所选特征随模型参数的变化而随机变化。
- 与脊回归(ridge regression)相比, 通常效果不佳。
- L1范数函数对异常值更具抵抗力。

L1 Regularization

- 使得参数中的许多值为0

$$\sum_{i=1}^n \left(Y_i - \sum_{j=1}^p X_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^p |\beta_j|$$



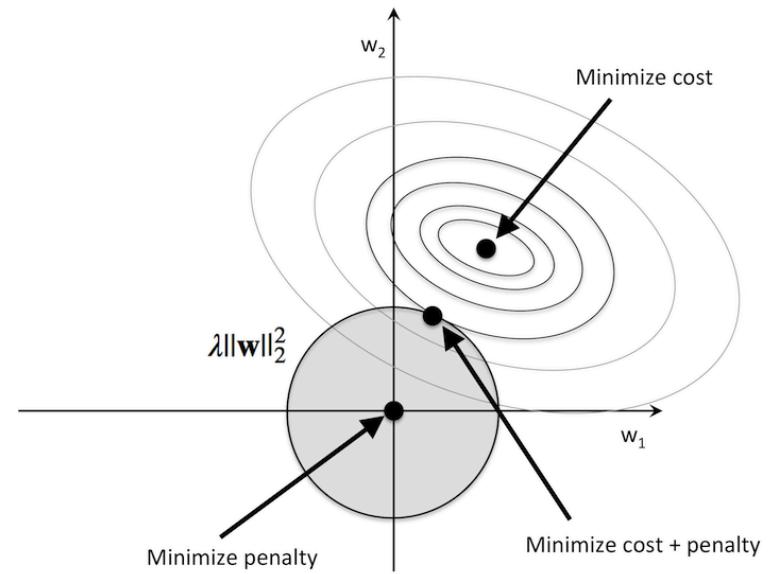
L2- regularization 特点

- 容易计算, 可导, 适合基于梯度的方法
- 将一些权值缩小到接近0
- 相关的预测特征对应的系数值相似
- 当特征的数量巨大时, 计算量会比较大
- 对于有相关特征存在的情况, 它会包含所有这些相关的特征, 但是相关特征的权值的分布取决于相关性
- 对outliers(异常值) 非常敏感
- 相对于L1正则化会更加精确

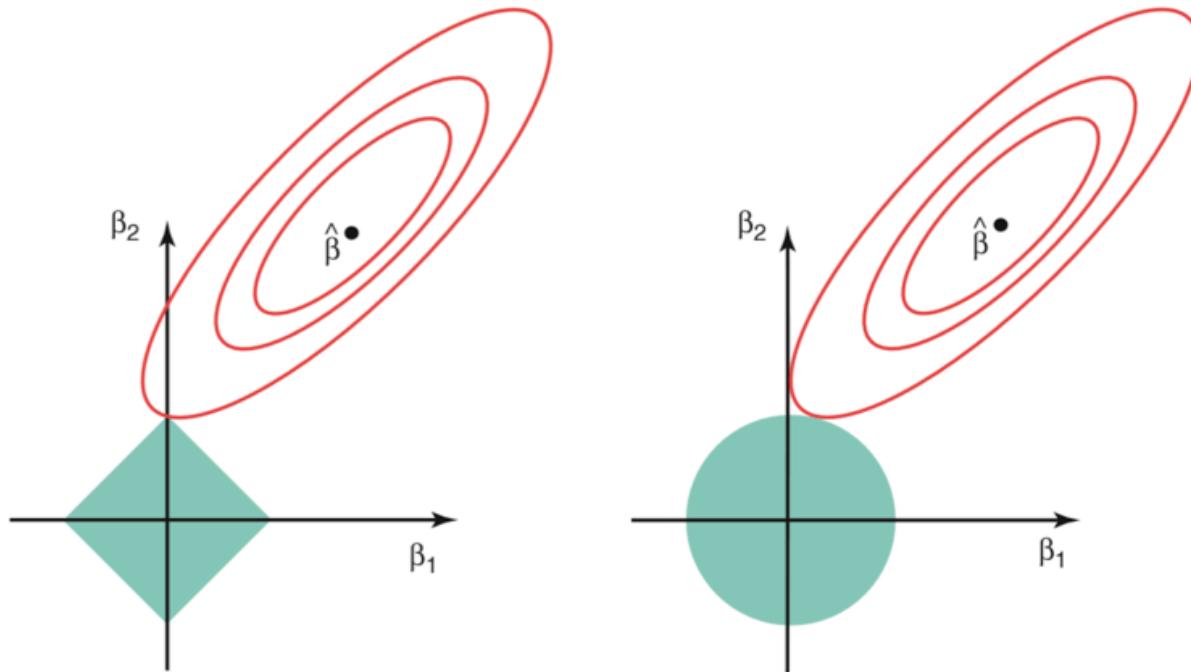
L2 Regularization

- 使得参数的值比较小

$$\sum_{i=1}^n \left(y_i - \sum_{j=1}^p x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^p \beta_j^2$$



L1 V. S. L2 Regularization



Elastic Net

- L1 和 L2 正则化各有千秋, 将他们结合起来就构成Elastic Net

Ridge, Lasso, ElasticNet

Ridge Regression

$$L_{\text{ridge}}(\hat{\beta}) = \sum_{i=1}^n \left(y_i - x_i' \hat{\beta} \right)^2 + \lambda \sum_{j=1}^m \hat{\beta}_j^2 = \| y - X \hat{\beta} \|^2 + \lambda \| \hat{\beta} \|^2$$

$$\hat{\beta}_{ridge} = (X'X + \lambda I)^{-1}(X'Y)$$

- As $\lambda \rightarrow 0$, $\hat{\beta}_{ridge} \rightarrow \hat{\beta}_{OLS}$;
- As $\lambda \rightarrow \infty$, $\hat{\beta}_{ridge} \rightarrow 0$.

Lasso Regression

$$L_{lasso}(\hat{\beta}) = \sum_{i=1}^n (y_i - x_i' \hat{\beta})^2 + \lambda \sum_{j=1}^m |\hat{\beta}_j|$$

Ridge V. S. Lasso

- 通常情况下，两者并不是其中一个一定比另一个更好。
- Lasso可以将一些系数设置为零，从而执行变量选择，而Ridge则不能。
- 两种方法都允许使用相关预测变量，但它们以不同方式解决多重共线性问题：
 - 在Ridge回归中，相关预测变量的系数相似；
 - 在Lasso中，其中一个相关预测因子具有较大的系数，而其余的（几乎）归零。
- 如果存在少量重要参数且其他参数接近于零，那么Lasso往往表现良好（当只有少数预测因子实际影响输出时）。
- 如果存在大约相同值的许多重要参数，那么Ridge运行良好。
- 但是，在实践中，我们不知道真正的参数值，所以前两点在某种程度上是理论上的。只需运行交叉验证即可为特定案例选择更适合的模型。

Elastic Net

$$L_{enet}(\hat{\beta}) = \frac{\sum_{i=1}^n (y_i - x_i' \hat{\beta})^2}{2n} + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^m \hat{\beta}_j^2 + \alpha \sum_{j=1}^m |\hat{\beta}_j| \right)$$

- 其中 α 控制了 ridge ($\alpha = 0$) 与 lasso ($\alpha = 1$) 的强度.

总结 Lasso, Ridge, ElasticNet

- 如果线性模型包含许多预测变量或者这些变量是相关的，则标准参数估计值具有较大的方差，从而使模型不可靠。
- 为了解决这个问题，你可以使用正则化：一种允许以引入一些偏差为代价来减少这种差异的技术。找到良好的偏差bias - 方差variance平衡可以最小化模型的总误差。
- 有三种流行的正则化技术，每种技术都旨在减小系数的大小：
- Ridge回归，惩罚平方系数之和（L2惩罚）。
- Lasso回归，它惩罚系数的绝对值之和（L1惩罚）。
- ElasticNet，是Ridge和Lasso的凸组合。
- 可以通过交叉验证来调整相应惩罚项的大小，以找到模型的最佳拟合。

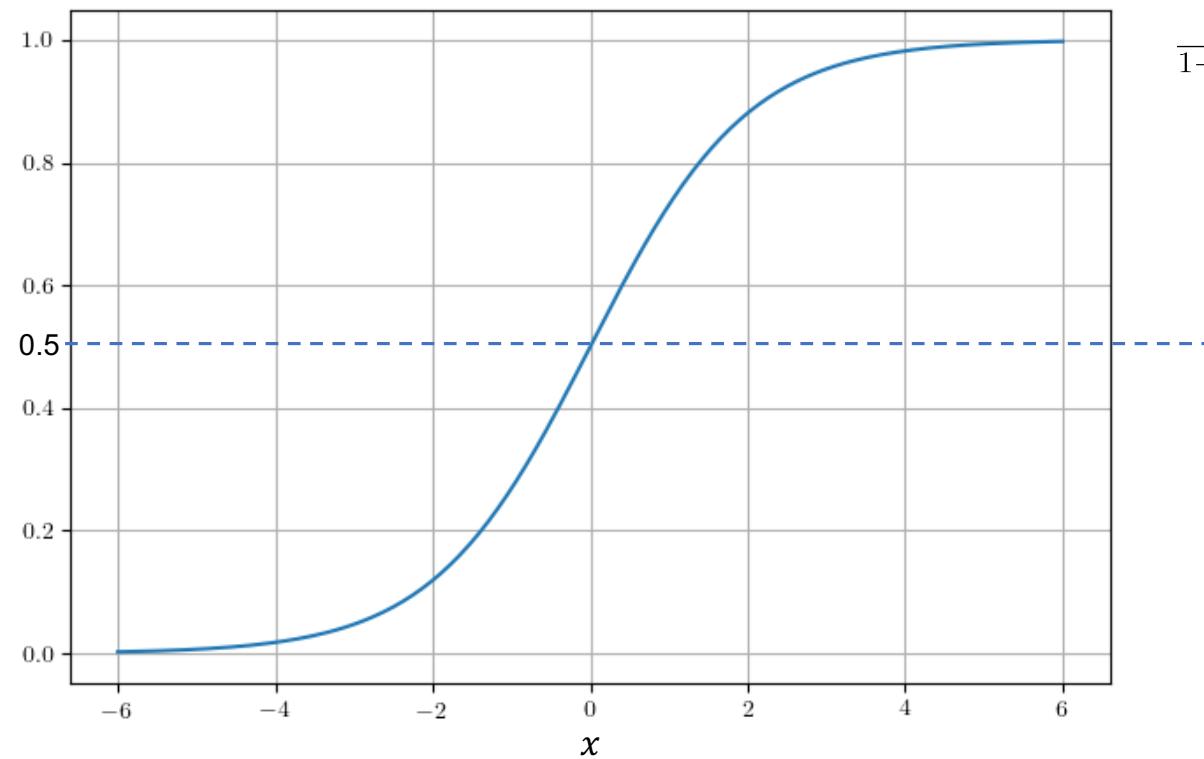
逻辑回归

逻辑函数

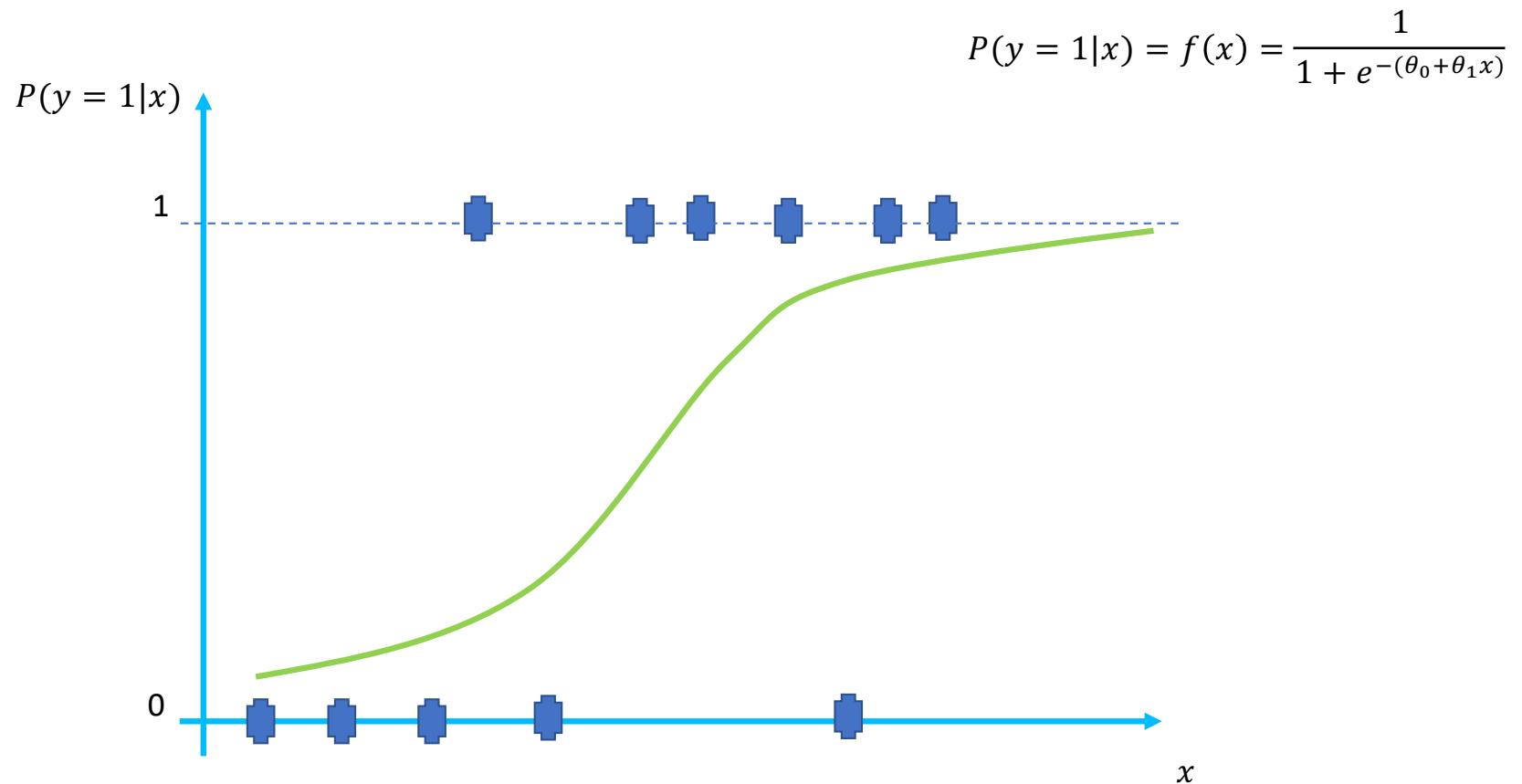
$$f(x) = \frac{1}{1 + e^{-x}}$$

$$\frac{1}{1+e^\infty} = \frac{1}{1+\infty} = 0$$

$$\frac{1}{1+e^{-\infty}} = \frac{1}{1+0} = 1$$

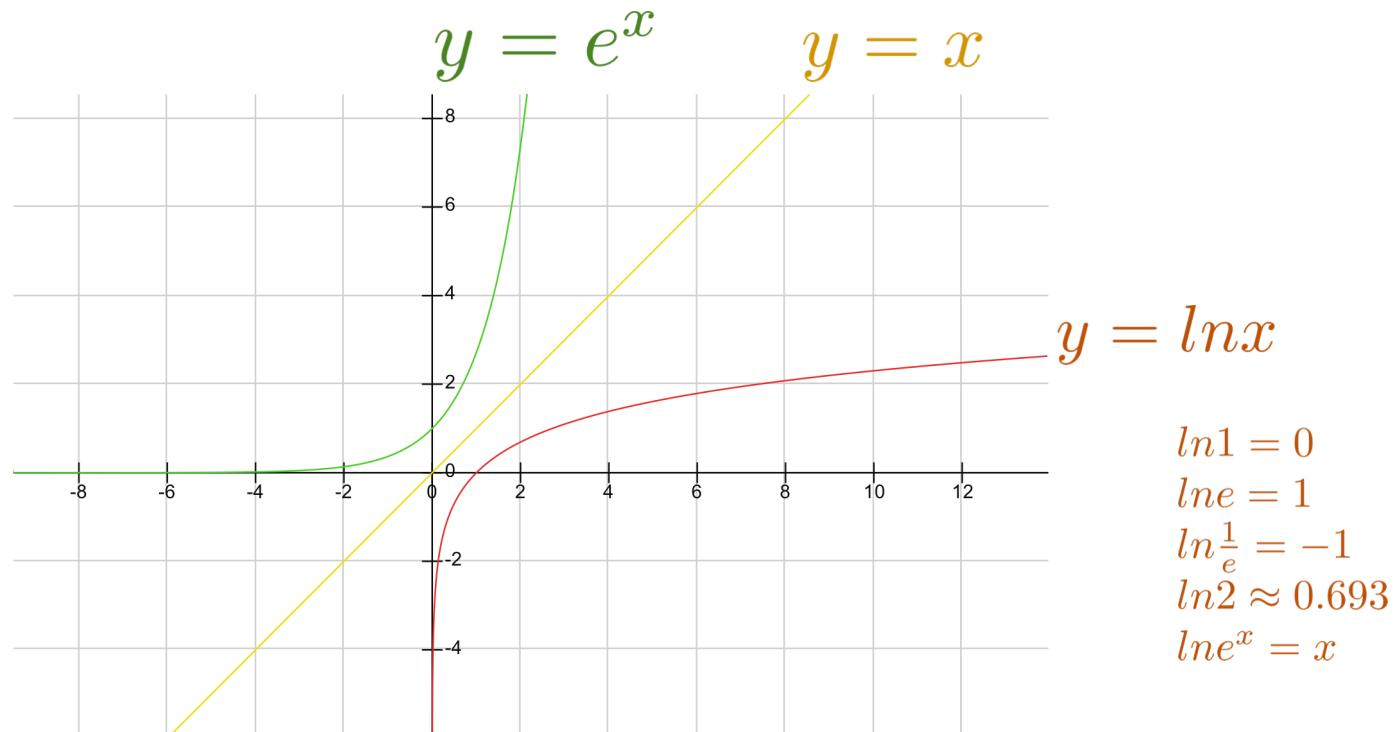


逻辑回归



指数与对数

$$\begin{aligned}e^0 &= 1 \\e^1 &= e \approx 2.72 \\e^{-1} &= \frac{1}{e} \\e^{0.693} &\approx 2 \\e^{\ln x} &= x\end{aligned}$$



$$\begin{aligned}\ln 1 &= 0 \\ \ln e &= 1 \\ \ln \frac{1}{e} &= -1 \\ \ln 2 &\approx 0.693 \\ \ln e^x &= x\end{aligned}$$

逻辑回归

- 解决二元 (0/1) 分类的问题

- $P(y = 1|x; \theta) = f(x; \theta) = \frac{1.0}{1.0 + e^{-\theta^T x}}$

- $\theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots$

- $\theta = [\theta_0, \theta_1, \theta_2, \theta_3, \dots]$

- $x = [1, x_1, x_2, x_3, \dots]$

- 当 $P(y = 1|x)$ 的值大于 0.5， 输出 1； 否则输出 0

逻辑回归知识点

| | |
|----------------|--|
| 类别1的概率 | $P = \frac{1}{1+e^{-(\theta^T x)}}$ |
| 类别0的概率 | $1 - P = \frac{1+e^{-\theta^T x} - 1}{1+e^{-\theta^T x}} = \frac{1}{1+e^{\theta^T x}}$ |
| 类别1与0概率比值 | $\frac{P}{1-P} = e^{\theta^T x}$ |
| 类别1与0概率比值的自然对数 | $\ln \frac{P}{1-P} = \theta^T x$ |

损失函数

$$P(Y = 1|x; \theta) = f(x; \theta) = \frac{1}{1 + e^{-\theta^\top x}}$$

$$J(\theta) = - \sum_{i=1}^N y^{(i)} \ln(P(Y = 1|X = x^{(i)}; \theta)) + (1 - y^{(i)}) \ln(1 - P(Y = 1|X = x^{(i)}; \theta))$$

$$\nabla_\theta J(\theta) = \sum_i x^{(i)} (f(x^{(i)}; \theta) - y^{(i)})$$

梯度下降法

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

$$\nabla_{\theta} J(\theta) = \sum_i x^{(i)} (f(x^{(i)}; \theta) - y^{(i)})$$

$$\theta = \theta - \alpha \nabla_{\theta} J(\theta) = \theta - \alpha \sum_i x^{(i)} (f(x^{(i)}; \theta) - y^{(i)})$$

新的值 旧的值 学习率 特征的值 已知类别的值
预测类别1的概率

系数的意义

$$P(Y = 1|x; \theta) = f(x; \theta) = \frac{1}{1 + e^{-\theta^T x}}$$

$$\text{概率比值} odds = \frac{p}{1-p} = e^{\theta^T x}$$

系数 θ_j 意味着，假设原来的odds为 λ_1 ，若对应的特征 x_j 增加1，假设新的odds为 λ_2 ，那么 $\frac{\lambda_2}{\lambda_1} \equiv e^{\theta_j}$

模型训练

| 年龄 (x_1) | 年收入(x_2)(万元为单位) | 是否买车 (1表示是, 0表示否) |
|--------------|---------------------|-------------------|
| 20 | 3 | 0 |
| 23 | 7 | 1 |
| 31 | 10 | 1 |
| 42 | 13 | 1 |
| 50 | 7 | 0 |
| 60 | 5 | 0 |



$$P(Y = 1|x; \theta) = \frac{1.0}{1.0 + e^{-\theta^T x}}$$

如何根据左边的训练数据得到系数 θ 的值：
 $\theta_0 = -0.04, \theta_1 = -0.20, \theta_2 = 0.92$

系数 $\theta_2 = 0.92$ 意味着，如果年收入增加1万，一个人买车和不买车的概率的比值与之前的比值相比较，增加 $e^{0.92} = 2.5$ 倍

系数 $\theta_1 = -0.20$ 意味着，如果年龄增加1岁，一个人买车和不买车的概率的比值与之前的比值比较降低 $e^{-0.20} = 0.82$ 倍

似然函数

$$P(Y = 1|x; \theta) = f(x; \theta) = \frac{1}{1 + e^{-\theta^\top x}}$$

$$P(Y = 0|x; \theta) = 1 - P(Y = 1|x; \theta)$$

$$L(\theta) = \prod_{i \in \{1, \dots, N\}, y^{(i)}=1} P(Y = 1|X = x^{(i)}; \theta) \cdot \prod_{i \in \{1, \dots, N\}, y^{(i)}=0} P(Y = 0|X = x^{(i)}; \theta)$$

$$L(\theta) = \prod_{i \in \{1, \dots, N\}, y^{(i)}=1} P(Y = 1|X = x^{(i)}; \theta) \cdot \prod_{i \in \{1, \dots, N\}, y^{(i)}=0} (1 - P(Y = 1|X = x^{(i)}; \theta))$$

$$J(\theta) = -\ln L(\theta) = -\sum_{i=1}^N y^{(i)} \ln(P(Y = 1|X = x^{(i)}; \theta)) + (1 - y^{(i)}) \ln(1 - P(Y = 1|X = x^{(i)}; \theta))$$

$$\nabla_\theta J(\theta) = \sum_i x^{(i)} (f(x^{(i)}; \theta) - y^{(i)})$$

似然函数

$$p(\mathbf{y}^{(i)} = 1 | \mathbf{X}^{(i)}; \mathbf{w}) = h_{\mathbf{w}}(\mathbf{X}^{(i)}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{X}^{(i)}}} = \frac{1}{1 + e^{-(\mathbf{w}_0 \mathbf{X}_0^{(i)} + \mathbf{w}_1 \mathbf{X}_1^{(i)} + \mathbf{w}_2 \mathbf{X}_2^{(i)} + \dots)}} \quad \mathbf{X}_0^{(i)} \equiv 1$$

$$l(\mathbf{w}) = \sum_{i=1}^m \mathbf{y}^{(i)} \log h_{\mathbf{w}}(\mathbf{X}^{(i)}) + (1 - \mathbf{y}^{(i)}) \log (1 - h_{\mathbf{w}}(\mathbf{X}^{(i)}))$$

$$\frac{\partial l(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^m (\mathbf{y}^{(i)} - h_{\mathbf{w}}(\mathbf{X}^{(i)})) \mathbf{X}_j^{(i)}$$

似然函数

$$p(\mathbf{y}^{(i)} = 1 | \mathbf{X}^{(i)}; \mathbf{w}) = h_{\mathbf{w}}(\mathbf{X}^{(i)}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{X}^{(i)}}} = \frac{1}{1 + e^{-(\mathbf{w}_0 \mathbf{X}_0^{(i)} + \mathbf{w}_1 \mathbf{X}_1^{(i)} + \mathbf{w}_2 \mathbf{X}_2^{(i)} + \dots)}}$$

$$\begin{aligned} \mathbf{L}(\mathbf{w}) &= p(\mathbf{y} | \mathbf{X}; \mathbf{w}) = \prod_{i=1}^m p(\mathbf{y}^{(i)} | \mathbf{X}^{(i)}; \mathbf{w}) = \prod_{i=1}^m p(\mathbf{y}^{(i)} = 1 | \mathbf{X}^{(i)}; \mathbf{w})^{y^{(i)}} p(\mathbf{y}^{(i)} = 0 | \mathbf{X}^{(i)}; \mathbf{w})^{1-y^{(i)}} \\ &= \prod_{i=1}^m p(\mathbf{y}^{(i)} = 1 | \mathbf{X}^{(i)}; \mathbf{w})^{y^{(i)}} (1 - p(\mathbf{y}^{(i)} = 1 | \mathbf{X}^{(i)}; \mathbf{w}))^{1-y^{(i)}} = \prod_{i=1}^m (h_{\mathbf{w}}(\mathbf{X}^{(i)}))^{y^{(i)}} (1 - h_{\mathbf{w}}(\mathbf{X}^{(i)}))^{1-y^{(i)}} \end{aligned}$$

$$l(\mathbf{w}) = \log \mathbf{L}(\mathbf{w}) = \sum_{i=1}^m y^{(i)} \log h_{\mathbf{w}}(\mathbf{X}^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\mathbf{w}}(\mathbf{X}^{(i)}))$$

$$\log AB = \log A + \log B; \quad \log A^B = B \log A$$

似然函数，对权值求偏导

$$l(\mathbf{w}) = \sum_{i=1}^m \mathbf{y}^{(i)} \log h_{\mathbf{w}}(\mathbf{X}^{(i)}) + (1 - \mathbf{y}^{(i)}) \log (1 - h_{\mathbf{w}}(\mathbf{X}^{(i)}))$$

$$h(z) = \frac{1}{1 + e^{-z}}$$

$$\frac{\partial h(z)}{\partial z} = h(z)(1 - h(z))$$

$$h_{\mathbf{w}}(\mathbf{X}^{(i)}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{X}^{(i)}}}$$

$$\begin{aligned}\frac{\partial l(\mathbf{w})}{\partial \mathbf{w}_j} &= \sum_{i=1}^m \left(\mathbf{y}^{(i)} \frac{1}{h_{\mathbf{w}}(\mathbf{X}^{(i)})} + (1 - \mathbf{y}^{(i)}) \frac{1}{1 - h_{\mathbf{w}}(\mathbf{X}^{(i)})} \right) \frac{\partial h_{\mathbf{w}}(\mathbf{X}^{(i)})}{\partial \mathbf{w}_j} \\ &= \sum_{i=1}^m \left(\mathbf{y}^{(i)} \frac{1}{h_{\mathbf{w}}(\mathbf{X}^{(i)})} + (1 - \mathbf{y}^{(i)}) \frac{1}{1 - h_{\mathbf{w}}(\mathbf{X}^{(i)})} \right) h_{\mathbf{w}}(\mathbf{X}^{(i)}) (1 - h_{\mathbf{w}}(\mathbf{X}^{(i)})) \frac{\partial \mathbf{w}^T \mathbf{X}^{(i)}}{\partial \mathbf{w}_j} \\ &= \sum_{i=1}^m \left(\mathbf{y}^{(i)} (1 - h_{\mathbf{w}}(\mathbf{X}^{(i)})) + (1 - \mathbf{y}^{(i)}) h_{\mathbf{w}}(\mathbf{X}^{(i)}) \right) \mathbf{X}_j^{(i)} \\ &= \sum_{i=1}^m (\mathbf{y}^{(i)} - h_{\mathbf{w}}(\mathbf{X}^{(i)})) \mathbf{X}_j^{(i)}\end{aligned}$$

证明逻辑回归的损失函数是convex的

- $J(\mathbf{w}) = \sum_{i=1}^m \mathbf{y}^{(i)} [-\log h_{\mathbf{w}}(\mathbf{X}^{(i)})] + (1 - \mathbf{y}^{(i)}) \left[-\log (1 - h_{\mathbf{w}}(\mathbf{X}^{(i)})) \right]$
- $h_{\mathbf{w}}(\mathbf{X}^{(i)}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{X}^{(i)}}}$
- 证明:
- $-\log h_{\mathbf{w}}(\mathbf{x}) = -\log \left(\frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}} \right) = \log \left(1 + e^{-\mathbf{w}^T \mathbf{x}} \right)$
- $\nabla_{\mathbf{w}} [-\log h_{\mathbf{w}}(\mathbf{x})] = \nabla_{\mathbf{w}} \left[\log \left(1 + e^{-\mathbf{w}^T \mathbf{x}} \right) \right] = \left(\frac{0 - e^{-\mathbf{w}^T \mathbf{x}}}{1 + e^{-\mathbf{w}^T \mathbf{x}}} \right) \mathbf{x} = \left(\frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}} - 1 \right) \mathbf{x} = [h_{\mathbf{w}}(\mathbf{x}) - 1] \mathbf{x}$
- Hessian:
- $\nabla_{\mathbf{w}}^2 [-\log h_{\mathbf{w}}(\mathbf{x})] = \nabla_{\mathbf{w}} (\nabla_{\mathbf{w}} [-\log h_{\mathbf{w}}(\mathbf{x})]) = \nabla_{\mathbf{w}} ([h_{\mathbf{w}}(\mathbf{x}) - 1] \mathbf{x}) = h_{\mathbf{w}}(\mathbf{x}) [1 - h_{\mathbf{w}}(\mathbf{x})] \mathbf{x} \mathbf{x}^T$
- Hessian is positive semi-definite(半正定):
- $\forall \mathbf{z}: \mathbf{z}^T \nabla_{\mathbf{w}}^2 [-\log h_{\mathbf{w}}(\mathbf{x})] \mathbf{z} = \mathbf{z}^T \{h_{\mathbf{w}}(\mathbf{x})[1 - h_{\mathbf{w}}(\mathbf{x})] \mathbf{x} \mathbf{x}^T\} \mathbf{z} = h_{\mathbf{w}}(\mathbf{x})[1 - h_{\mathbf{w}}(\mathbf{x})] \mathbf{z}^T \mathbf{x} \mathbf{x}^T \mathbf{z} = h_{\mathbf{w}}(\mathbf{x})[1 - h_{\mathbf{w}}(\mathbf{x})] (\mathbf{x}^T \mathbf{z})^2 \geq 0$

证明逻辑回归的损失函数是convex的(续)

- $J(\mathbf{w}) = \sum_{i=1}^m \mathbf{y}^{(i)} [-\log h_{\mathbf{w}}(\mathbf{X}^{(i)})] + (1 - \mathbf{y}^{(i)}) [-\log (1 - h_{\mathbf{w}}(\mathbf{X}^{(i)}))]$
- $h_{\mathbf{w}}(\mathbf{X}^{(i)}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{X}^{(i)}}}$
- 证明:
 - $-\log(1 - h_{\mathbf{w}}(\mathbf{x})) = -\log\left(1 - \frac{1}{1+e^{-\mathbf{w}^T \mathbf{x}}}\right) = -\log\left(\frac{e^{-\mathbf{w}^T \mathbf{x}}}{1+e^{-\mathbf{w}^T \mathbf{x}}}\right) = \mathbf{w}^T \mathbf{x} + \log\left(1 + e^{-\mathbf{w}^T \mathbf{x}}\right)$
 - $\nabla_{\mathbf{w}}(-\log(1 - h_{\mathbf{w}}(\mathbf{x}))) = \nabla_{\mathbf{w}}\left[\mathbf{w}^T \mathbf{x} + \log\left(1 + e^{-\mathbf{w}^T \mathbf{x}}\right)\right] = \mathbf{x} + \nabla_{\mathbf{w}}\left[\log\left(1 + e^{-\mathbf{w}^T \mathbf{x}}\right)\right]$
- Hessian:
 - $\nabla_{\mathbf{w}}^2[-\log(1 - h_{\mathbf{w}}(\mathbf{x}))] = \nabla_{\mathbf{w}}\left(\nabla_{\mathbf{w}}(-\log(1 - h_{\mathbf{w}}(\mathbf{x})))\right) = \nabla_{\mathbf{w}}\left(\mathbf{x} + \nabla_{\mathbf{w}}\left[\log\left(1 + e^{-\mathbf{w}^T \mathbf{x}}\right)\right]\right) = \nabla_{\mathbf{w}}\left(\nabla_{\mathbf{w}}\left[\log\left(1 + e^{-\mathbf{w}^T \mathbf{x}}\right)\right]\right) = \nabla_{\mathbf{w}}^2[-\log h_{\mathbf{w}}(\mathbf{x})]$ (已经证明其为半正定)
 - $[-\log h_{\mathbf{w}}(\mathbf{x})]$ 和 $[-\log(1 - h_{\mathbf{w}}(\mathbf{x}))]$ 分别都半正定, 他们的线性组合也是正定的

多元分类

$$h_{\mathbf{w}}(x) = \begin{bmatrix} P(y=1|x; \mathbf{w}) \\ P(y=2|x; \mathbf{w}) \\ \vdots \\ P(y=K|x; \mathbf{w}) \end{bmatrix} = \frac{1}{\sum_{j=1}^K \exp(w^{(j)\top} x)} \begin{bmatrix} \exp(w^{(1)\top} x) \\ \exp(w^{(2)\top} x) \\ \vdots \\ \exp(w^{(K)\top} x) \end{bmatrix}$$

$$\mathbf{w} = \left[\begin{array}{c|c|c|c} w^{(1)} & w^{(2)} & \dots & w^{(K)} \\ \hline | & | & | & | \end{array} \right]$$

多元分类:似然函数及其导数

$$l(\mathbf{w}) = \sum_{i=1}^n \sum_{k=1}^K 1\left\{y^{(i)} = k\right\} \log \frac{\exp(w^{(k)\top} x^{(i)})}{\sum_{j=1}^K \exp(w^{(j)\top} x^{(i)})}$$

$$\nabla_{w^{(k)}} l(\mathbf{w}) = \sum_{i=1}^n \left[x^{(i)} \left(1\{y^{(i)} = k\} - P(y^{(i)} = k | x^{(i)}; w) \right) \right]$$

$$P(y^{(i)} = k | x^{(i)}; \mathbf{w}) = \frac{\exp(w^{(k)\top} x^{(i)})}{\sum_{j=1}^K \exp(w^{(j)\top} x^{(i)})}$$

Softmax with cross-entropy loss 求导 (1)

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^N e^{a_k}}$$

当 i=j,

$$\begin{aligned}\frac{\partial \frac{e^{a_i}}{\sum_{k=1}^N e^{a_k}}}{\partial a_j} &= \frac{e^{a_i} \sum_{k=1}^N e^{a_k} - e^{a_j} e^{a_i}}{\left(\sum_{k=1}^N e^{a_k}\right)^2} \\ &= \frac{e^{a_i} \left(\sum_{k=1}^N e^{a_k} - e^{a_j}\right)}{\left(\sum_{k=1}^N e^{a_k}\right)^2} \\ &= \frac{e^{a_j}}{\sum_{k=1}^N e^{a_k}} \times \frac{\left(\sum_{k=1}^N e^{a_k} - e^{a_j}\right)}{\sum_{k=1}^N e^{a_k}} \\ &= p_i(1 - p_j)\end{aligned}$$

Softmax with cross-entropy loss 求导 (2)

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^N e^{a_k}}$$

当 $i \neq j$,

$$\begin{aligned}\frac{\partial \frac{e^{a_i}}{\sum_{k=1}^N e^{a_k}}}{\partial a_j} &= \frac{0 - e^{a_j} e^{a_i}}{\left(\sum_{k=1}^N e^{a_k}\right)^2} \\ &= \frac{-e^{a_j}}{\sum_{k=1}^N e^{a_k}} \times \frac{e^{a_i}}{\sum_{k=1}^N e^{a_k}} \\ &= -p_j \cdot p_i\end{aligned}$$

综合而言:

$$\frac{\partial p_i}{\partial a_j} = \begin{cases} p_i(1 - p_j) & \text{if } i = j \\ -p_j \cdot p_i & \text{if } i \neq j \end{cases}$$

定义:

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

则:

$$\frac{\partial p_i}{\partial a_j} = p_i(\delta_{ij} - p_j)$$

Softmax with cross-entropy loss 求导 (3)

$$\begin{aligned} L &= - \sum_i y_i \log(p_i) \\ \frac{\partial L}{\partial o_i} &= - \sum_k y_k \frac{\partial \log(p_k)}{\partial o_i} \\ &= - \sum_k y_k \frac{\partial \log(p_k)}{\partial p_k} \times \frac{\partial p_k}{\partial o_i} \\ &= - \sum_k y_k \frac{1}{p_k} \times \frac{\partial p_k}{\partial o_i} \end{aligned}$$

$$\begin{aligned} \frac{\partial L}{\partial o_i} &= -y_i(1 - p_i) - \sum_{k \neq i} y_k \frac{1}{p_k} (-p_k \cdot p_i) \\ &= -y_i(1 - p_i) + \sum_{k \neq i} y_k \cdot p_i \\ &= -y_i + y_i p_i + \sum_{k \neq i} y_k \cdot p_i \\ &= p_i \left(y_i + \sum_{k \neq i} y_k \right) - y_i \end{aligned}$$

Softmax with cross-entropy loss 求导 (4)

$$L = - \sum_i y_i \log(p_i)$$

$$\begin{aligned}\frac{\partial L}{\partial o_i} &= -y_i(1 - p_i) - \sum_{k \neq i} y_k \frac{1}{p_k} (-p_k \cdot p_i) \\ &= -y_i(1 - p_i) + \sum_{k \neq i} y_k \cdot p_i \\ &= -y_i + y_i p_i + \sum_{k \neq i} y_k \cdot p_i \\ &= p_i \left(y_i + \sum_{k \neq i} y_k \right) - y_i\end{aligned}$$

已知 y 是独热编码的向量, 所以 $\sum_k y_k = 1$,
即 $y_i + \sum_{k \neq i} y_k = 1$, 于是

$$\begin{aligned}\frac{\partial L}{\partial o_i} &= p_i - y_i \\ \frac{\partial L}{\partial w_k} &= \sum_{i=1}^K \frac{\partial L}{\partial o_i} \frac{\partial o_i}{\partial w_k} = \frac{\partial L}{\partial o_k} \frac{\partial o_k}{\partial w_k} + \sum_{i \neq k} \frac{\partial L}{\partial o_i} \frac{\partial o_i}{\partial w_k} \\ &= (p_k - y_k)x + 0 \\ &= (p_k - y_k)x\end{aligned}$$

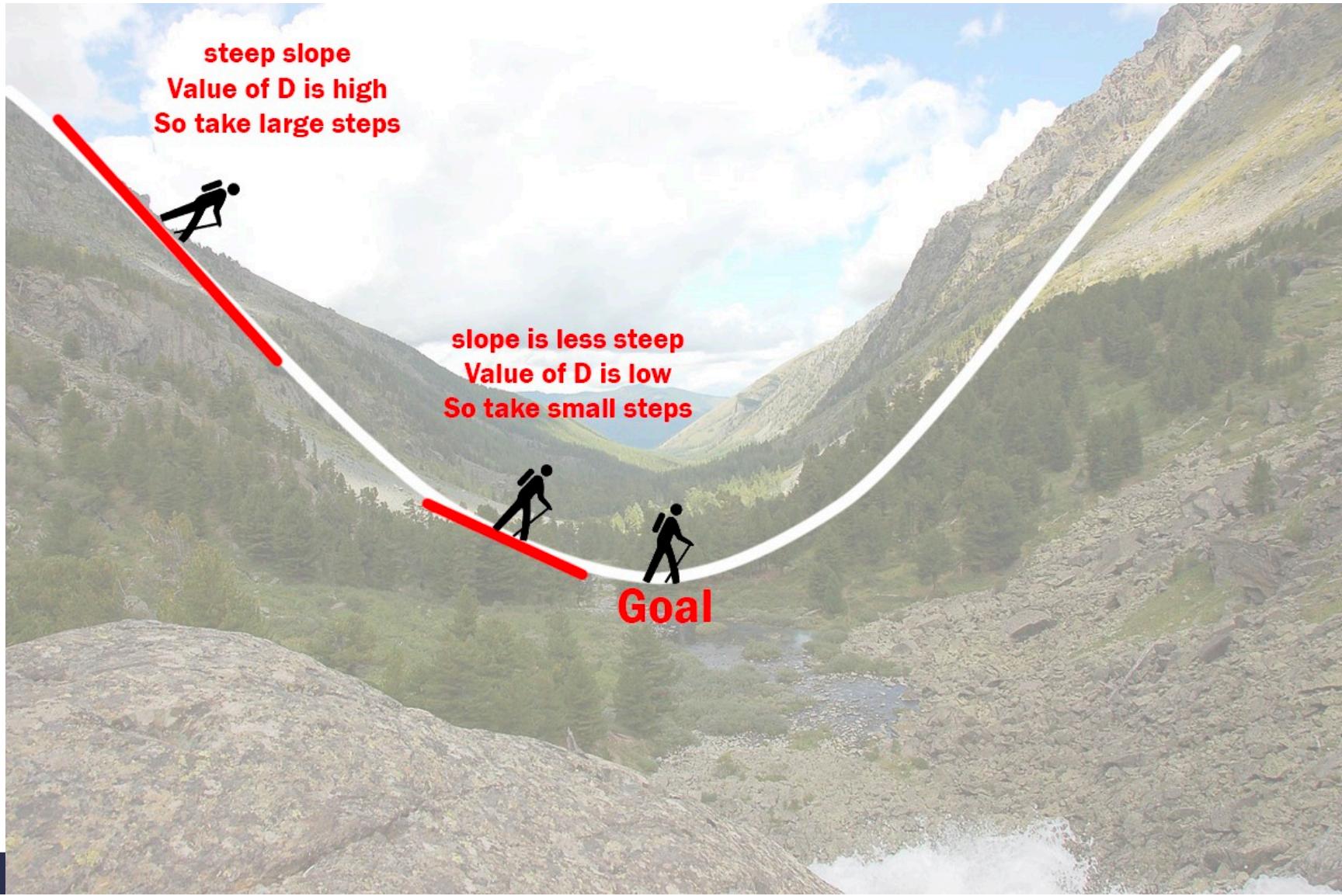
梯度下降算法

梯度下降算法

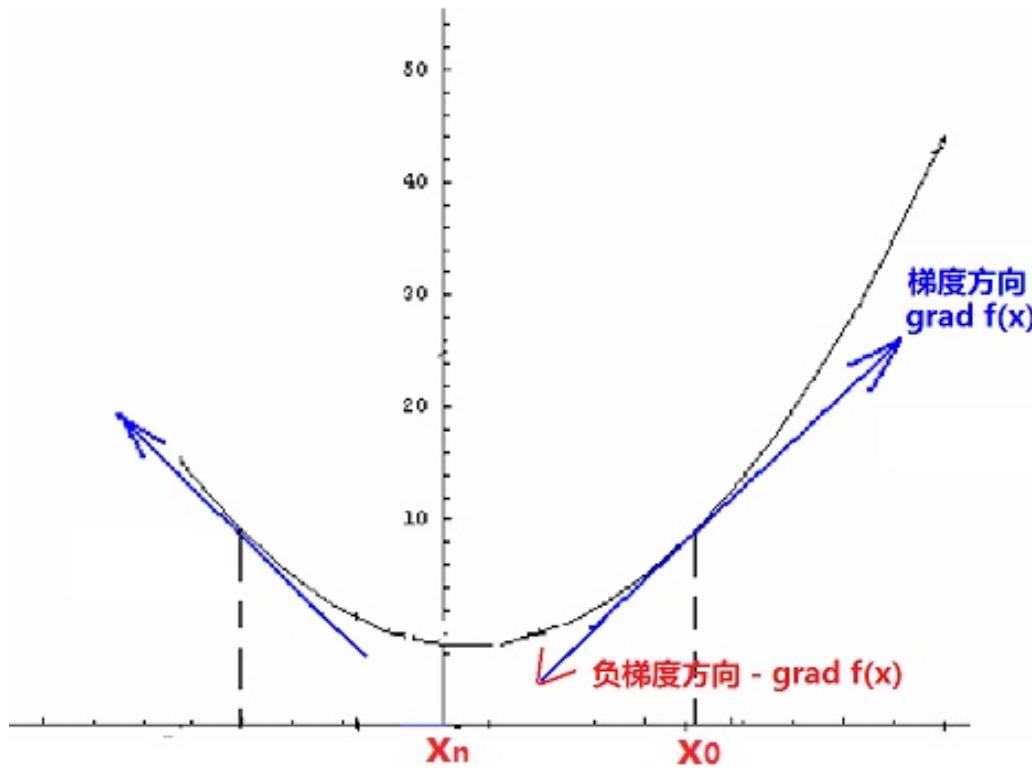
- 梯度就是函数的导数方向
- 梯度下降法是求解无约束多元函数极值最常用的数值方法，很多机器学习的常用算法都是以它作为算法框架，进行改进而导出更为复杂的优化方法。
- 在求解目标函数 $f(x)$ 的最小值时，如果目标函数是无约束的凸函数，就可以使用梯度下降法。

$$\min f(x)$$

根据导数的定义，函数 $f(x)$ 的导函数就是目标函数在 x 上的变化率。在多元的情况下，目标函数 $f(x, y, z)$ 在某点的梯度 $\text{grad}f(x, y, z) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$ 是一个由各个分量的偏导数构成的向量，负梯度方向是 $f(x, y, z)$ 减小最快的方向。



梯度下降算法(续)



如上图所示，当需要求 $f(x)$ 的最小值时（机器学习中的 $f(x)$ 一般就是损失函数，而我们的目标就是希望损失函数最小化），我们就可以先任意选取一个函数的初始点 x_0 （三维情况就是 (x_0, y_0, z_0) ），让其沿着途中红色箭头（负梯度方向）走，依次到 x_0, x_1, \dots, x_n （迭代n次）这样可最快达到极小值点。

梯度下降算法(续)

梯度下降方法基于以下的观察：如果实值函数 $F(\mathbf{x})$ 在点 \mathbf{a} 处可微且有定义，那么函数 $F(\mathbf{x})$ 在 \mathbf{a} 点沿着梯度相反的方向 $-\nabla F(\mathbf{a})$ 下降最快。

因而，如果

$$\mathbf{b} = \mathbf{a} - \gamma \nabla F(\mathbf{a})$$

对于 $\gamma > 0$ 为一个够小数值时成立，那么 $F(\mathbf{a}) \geq F(\mathbf{b})$ 。

考虑到这一点，我们可以从函数 F 的局部极小值的初始估计 \mathbf{x}_0 出发，并考虑如下序列 $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots$ 使得

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), n \geq 0.$$

因此可得到

$$F(\mathbf{x}_0) \geq F(\mathbf{x}_1) \geq F(\mathbf{x}_2) \geq \dots,$$

如果顺利的话序列 (\mathbf{x}_n) 收敛到期望的极值。注意每次迭代步长 γ 可以改变。

批梯度下降法

- 批量梯度下降法 (Batch Gradient Descent , 简称BGD) 是梯度下降法最原始的形式，它的具体思路是在更新每一参数时都使用所有的样本来进行更新，也就是方程中的m表示样本的所有个数。

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

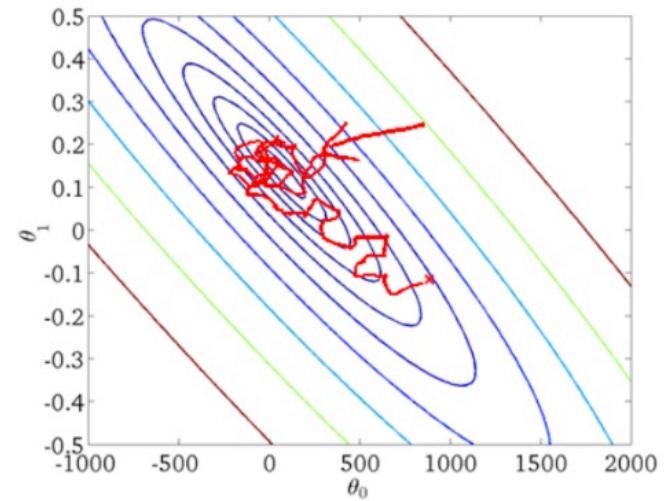
- 优点：全局最优解；易于并行实现；
- 缺点：当样本数目很多时，训练过程会很慢。

随机梯度下降法SGD

- 由于批梯度下降每跟新一个参数的时候，要用到所有的样本数，所以训练速度会随着样本数量的增加而变得非常缓慢。随机梯度下降正是为了解决这个办法而提出的。
- 它的具体思路是在更新每一参数时都使用一个样本来进行更新，也就是以上批处理方程中的 m 等于1。每一次跟新参数都用一个随机选择的样本，更新很多次。如果样本量很大的情况（例如几十万），那么可能只用其中几万条或者几千条的样本，就已经将theta迭代到最优解了，对比上面的批量梯度下降，迭代一次需要用到十几万训练样本，一次迭代不可能最优，如果迭代10次的话就需要遍历训练样本10次，这种跟新方式计算复杂度太高。
- 但是，SGD伴随的一个问题是噪音较BGD要多，使得SGD并不是每次迭代都向着整体最优化方向。
- **优点：训练速度快；**
- **缺点：准确度下降，并不是全局最优；不易于并行实现。**
- 从迭代的次数上来看，SGD迭代的次数较多，在解空间的搜索过程看起来很盲目。

随机梯度下降法SGD(续)

- 随机梯度下降是通过每个样本来迭代更新一次，对比上面的批量梯度下降，迭代一次需要用到所有训练样本（往往如今真实问题训练数据都是非常巨大），一次迭代不可能最优，如果迭代10次的话就需要遍历训练样本10次。但是，SGD伴随的一个问题是噪音较BGD要多，使得SGD并不是每次迭代都向着整体最优化方向。



小批量梯度下降法

- 我们从上面两种梯度下降法可以看出，其各自均有优缺点，那么能不能在两种方法的性能之间取得一个折衷呢？即，算法的训练过程比较快，而且也要保证最终参数训练的准确率，而这正是小批量梯度下降法（Mini-batch Gradient Descent，简称MBGD）的初衷。
- 小批量梯度下降法（Mini-batch Gradient Descent，简称MBGD）：它的具体思路是在更新每一参数时都使用一部分样本来进行更新，也就是批处理方程中的 m 的值大于1小于所有样本的数量。为了克服上面两种方法的缺点，又同时兼顾两种方法的优点。
- 三种方法使用的情况：
 - 如果样本量比较小，采用批量梯度下降算法。如果样本太大，或者在线算法，使用随机梯度下降算法。在实际的一般情况下，采用小批量梯度下降算法。

THANKS

贪心学院讲师：袁源



贪心科技 让每个人享受个性化教育服务