

Machine Learning - P24

线性判别分析，对于训练集 $T = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$
其中输出空间 Y 为二分类标记 $\{0, 1\}$

输入空间 $X \subseteq \mathbb{R}^D$, 即 \vec{x} 为 D 维列向量

训练集中 $Y=1$ 共出现 N_1 次, $Y=0$ 共出现 N_0 次

取 D 维列向量 \vec{w} , 并令 $y = \vec{w}^\top \vec{x}$

则此时 y 不再是二分标记 $\{0, 1\}$

而是表示 \vec{x} 在 \vec{w} 上的投影到原点的距离

目标是投影的类内方差最小, 而类间方差最大

首先有每个类的期望, 有 $\bar{\mu}_i = \frac{1}{N_i} \sum_{x, y=i} \vec{x}$

则期望在 \vec{w} 上的投影 $\tilde{\mu}_i = \frac{1}{N_i} \sum_{x, y=i} y = \frac{1}{N_i} \sum_{x, y=i} \vec{w}^\top \vec{x} = \vec{w}^\top \bar{\mu}_i$

即投影后的均值等于均值的投影

于是类间方差可以表示为 $(\tilde{\mu}_0 - \tilde{\mu}_1)^2$

再取每个类的散列值 (scatter), 对类的投影求散列值 $\tilde{s}_i^2 = \sum_{y=i} (y - \tilde{\mu}_i)^2$

于是类内方差可以表示为 $\sum_i \tilde{s}_i^2$

对于二分类, 则有 $\tilde{s}_0^2 + \tilde{s}_1^2$

于是可以取度量函数, 并使函数值最大化

$$J(\vec{w}) = (\tilde{\mu}_0 - \tilde{\mu}_1)^2 / (\tilde{s}_0^2 + \tilde{s}_1^2)$$

$$\text{又对于 } \tilde{s}_i^2 = \sum_{y=i} (y - \tilde{\mu}_i)^2 = \sum_{y=i} (\vec{w}^\top \vec{x} - \vec{w}^\top \bar{\mu}_i)^2$$

$$= \sum_{y=i} \vec{w}^\top (\vec{x} - \bar{\mu}_i) (\vec{x} - \bar{\mu}_i)^\top \vec{w}$$

取散列矩阵 (scatter matrix) $S_i = \sum_{y=i} (\vec{x} - \bar{\mu}_i) (\vec{x} - \bar{\mu}_i)^\top$

则有 $\tilde{s}_i^2 = \vec{w}^\top S_i \vec{w}$

并有类内散列矩阵 (within-class scatter matrix), $S_w = \sum_i S_i$

$$\text{对于 } (\tilde{\mu}_0 - \tilde{\mu}_1)^2 = (\vec{w}^\top \bar{\mu}_0 - \vec{w}^\top \bar{\mu}_1)^2 = \vec{w}^\top (\bar{\mu}_0 - \bar{\mu}_1) (\bar{\mu}_0 - \bar{\mu}_1)^\top \vec{w}$$

则有类间散列矩阵 (between-class scatter matrix), $S_B = (\bar{\mu}_0 - \bar{\mu}_1) (\bar{\mu}_0 - \bar{\mu}_1)^\top$

于是有度量函数 $J(\vec{w}) = \vec{w}^\top S_B \vec{w} / \vec{w}^\top S_w \vec{w}$

也称为广义瑞利商 (generalized Rayleigh quotient)

为了防止 \vec{w} 扩大任意倍数使得 $J(\vec{w})$ 最大化

首先对分子进行归一化, 即令 $\vec{w}^\top S_w \vec{w} = 1$

于是有拉格朗日函数 $L(\vec{w}) = \vec{w}^\top S_B \vec{w} - \lambda (\vec{w}^\top S_w \vec{w} - 1)$

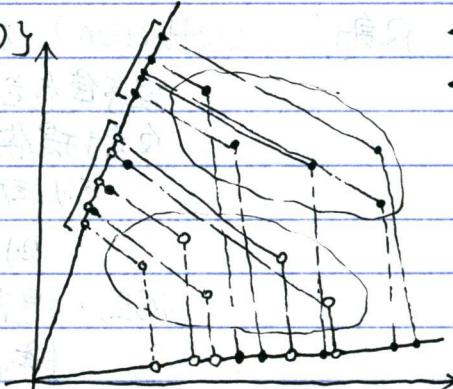
$$L'(\vec{w}) = 2S_B \vec{w} - 2\lambda S_w \vec{w} = 0$$

(Fisher linear discrimination) 即 $S_w^{-1} S_B \vec{w} = \lambda \vec{w}$, 即 \vec{w} 为 $S_w^{-1} S_B$ 的特征向量

$$\text{又 } S_B \vec{w} = (\bar{\mu}_0 - \bar{\mu}_1) (\bar{\mu}_0 - \bar{\mu}_1)^\top \vec{w} = (\bar{\mu}_0 - \bar{\mu}_1) \cdot \lambda_w, \text{ 其中 } \lambda_w \text{ 为特征值}$$

$$\text{于是 } S_w^{-1} S_B \vec{w} = S_w^{-1} (\bar{\mu}_0 - \bar{\mu}_1) \lambda_w = \lambda \vec{w}.$$

由于扩大缩小 \vec{w} 不影响结果, 于是有 $\vec{w} = S_w^{-1} (\bar{\mu}_0 - \bar{\mu}_1)$



Machine

Learning - P25

线性回归 (linear regression), 对于 D 维列向量 $\vec{x} = (x_1, x_2, \dots, x_D)^T$

有权重向量 (weight vector) $\vec{w} = (w_0, w_1, \dots, w_D)^T$

残差 (residual error) $\epsilon \sim N(0, \sigma^2)$

则有 $y(\vec{x}) = \vec{w}^T \vec{x} + \epsilon = \sum_{j=1}^D w_j x_j + \epsilon$

称为线性回归模型

于是有 条件概率 $P(y|\vec{x}, \theta) = N(y|\mu(\vec{x}), \sigma^2)$

由于 $\vec{w}^T \vec{x}$ 为两个 D 维向量的标量积 (scalar product / inner product)

则有 $\mu(\vec{x}) = \vec{w}^T \vec{x}$, $\sigma^2(\vec{x}) = \sigma^2$

即 模型参数 $\theta = (\vec{w}, \sigma^2)$

当将 \vec{x} 替换为 non-linear function of the input $\phi(\vec{x})$

即有 $\phi(\vec{x}) = (x^0, x^1, x^2, \dots, x^D) = (1, x_1, x_2^2, \dots, x_D)$

则取 $\vec{w} = (w_0, w_1, w_2, \dots, w_D)$

于是有 $\vec{w}^T \phi(\vec{x}) = w_0 + w_1 x_1 + w_2 x_2^2 + \dots + w_D x_D$

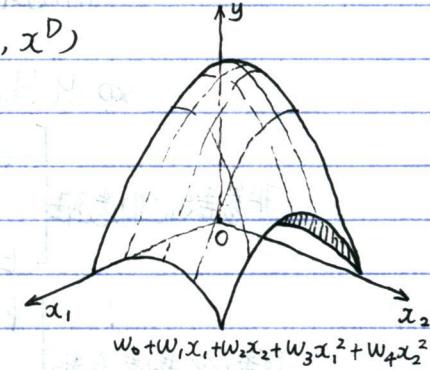
其中 w_0 称为 intercept or bias term

这个过程称为 basis function expansion

于是有 $P(y|\vec{x}, \theta) = N(y|\vec{w}^T \phi(\vec{x}), \sigma^2)$

其中 $y(\vec{x}) = \vec{w}^T \phi(\vec{x}) + \epsilon$

称为多项式回归 (polynomial regression) 模型



也可以将 $\phi(\vec{x})$ 扩展为 多于一个输入, 如 $\phi(\vec{x}) = (1, x_1, x_2, x_1^2, x_2^2)$

于是有 $E(y|\vec{x}) = \vec{w}^T \vec{x} = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2$

线性回归的

对于线性回归模型, 有训练集 $T = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

则有参数估计 $\hat{\theta}_{MLE} = \operatorname{argmax}_{\theta} P(T|\theta)$

如果训练集 T 的样本服从独立同分布

则有对数似然函数

$$\begin{aligned} L(\theta) &= \log P(T|\theta) = \log \prod_{i=1}^N p(y_i|\vec{x}_i, \theta) = \sum_{i=1}^N \log p(y_i|\vec{x}_i, \theta) \\ &= \sum_{i=1}^N \log \left[\frac{1}{(2\pi)^{D/2}} \exp \left(-\frac{1}{2\sigma^2} (y_i - \vec{w}^T \vec{x}_i)^2 \right) \right] = \frac{-1}{2\sigma^2} \text{RSS}(\vec{w}) - \frac{N}{2} \log (2\pi\sigma^2) \end{aligned}$$

其中 $\text{RSS}(\vec{w}) = \sum_{i=1}^N (y_i - \vec{w}^T \vec{x}_i)^2$ 称为残差平方和 (residual sum of squares)

也称误差平方和 (sum of squared errors, SSE)

而 SSE/N 称为 均方误差 (mean squared error, MSE)

Machine

Learning - P26

线性回归的
极大似然估计

对于训练集 $T = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

其中输入空间为 D 维实数列向量 $\vec{x} \in \mathbb{R}^D$

输出 $y(\vec{x}) = \vec{w}^\top \vec{x} + \varepsilon$

其中权重向量为 D 维实数列向量 $\vec{w} = (w_1, w_2, \dots, w_D)^\top$

残差为服从正态分布的随机变量 $\varepsilon \sim N(0, \sigma^2)$

有模型参数 $\theta = (\vec{w}, \sigma^2)$

有对数似然函数 $L(\theta) = \frac{-1}{2\sigma^2} \text{RSS}(\vec{w}) - \frac{N}{2} \ln(2\pi\sigma^2)$

其中残差平方和 $\text{RSS}(\vec{w}) = \sum_{i=1}^N (y_i - \vec{w}^\top \vec{x}_i)^2$

又有 $\varepsilon_i = y_i - \vec{w}^\top \vec{x}_i$, 则取 N 维列向量 $\vec{\varepsilon} = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_N)^\top$

则有 $\text{RSS}(\vec{w}) = \sum_{i=1}^N \varepsilon_i^2 = \|\vec{\varepsilon}\|_2^2$

即 square of l_2 norm of vector of residual error

相比于最大化对数似然函数 $L(\theta) = \sum_{i=1}^N \ln p(y_i | \vec{x}_i, \theta)$

等价于最小化函数 $NLL(\theta) = -\sum_{i=1}^N \ln p(y_i | \vec{x}_i, \theta)$

称为负对数似然函数 (negative log-likelihood, NLL)

注意到在对数似然函数 $L(\theta)$ 中 $\text{RSS}(\vec{w})$ 的系数为负数, $-\frac{N}{2} \ln(2\pi\sigma^2)$ 与 \vec{w} 无关

于是最大化 $L(\theta)$ 的过程中实际上需要最小化 $\text{RSS}(\vec{w})$

即最小化 $\|\vec{\varepsilon}\|_2^2$ (least squares)

又 $\varepsilon_i = y_i - \vec{w}^\top \vec{x}_i = y_i - \vec{x}_i^\top \vec{w}$ 其中 $i=1, 2, \dots, N$

则取 N 维列向量 $\vec{\varepsilon} = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_N)^\top$

N 维列向量 $\vec{y} = (y_1, y_2, \dots, y_N)^\top$ 其中 $j=1, 2, \dots, D$

$N \times D$ 矩阵 $X = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N)^\top$ 即 $X_{ij} = \vec{x}_i^{(j)}$, 其中 $i=1, 2, \dots, N$

于是 $X\vec{w} = (\vec{x}_1^\top \vec{w}, \vec{x}_2^\top \vec{w}, \dots, \vec{x}_N^\top \vec{w})^\top$ 为 N 维列向量

即有 $\vec{\varepsilon} = \vec{y} - X\vec{w}$, 即 $\|\vec{\varepsilon}\|_2^2 = (\vec{y} - X\vec{w})^\top (\vec{y} - X\vec{w})$

再取负对数似然函数 $NLL(\vec{w}) = \frac{1}{2} (\vec{y} - X\vec{w})^\top (\vec{y} - X\vec{w})$

与 $\vec{y}^\top X\vec{w}$ 为相同的括号值

由于 $\vec{y}^\top \vec{y}$ 与 \vec{w} 无关, 且 $\vec{w}^\top X^\top \vec{y} = \frac{1}{2} \vec{w}^\top (X^\top X) \vec{w} - \vec{w}^\top (X^\top \vec{y})$

其中 $X^\top X = \sum_{i=1}^N \vec{x}_i \vec{x}_i^\top = \begin{bmatrix} \vec{x}_1^{(1)\top} & \cdots & \vec{x}_1^{(D)\top} \\ \vdots & \ddots & \vdots \\ \vec{x}_N^{(1)\top} & \cdots & \vec{x}_N^{(D)\top} \end{bmatrix}$

$X^\top \vec{y} = \sum_{i=1}^N \vec{x}_i^\top y_i = \sum_{i=1}^N \begin{bmatrix} \vec{x}_i^{(1)\top} \\ \vdots \\ \vec{x}_i^{(D)\top} \end{bmatrix} y_i$

于是此时的梯度 (gradient) $g(\vec{w}) = (X^\top X) \vec{w} - X^\top \vec{y}$

当 $g(\vec{w})$ 为零向量时, 有 $(X^\top X)^{-1} X^\top \vec{y} = \hat{\vec{w}}_{OLS}$

称为普通最小二乘解 (ordinary least squares, OLS solution)

Machine

Learning - P27

线性回归

对于训练集 $T = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

其中输入空间为 D 维实数列向量 $\vec{x} \in \mathbb{R}^D$

$$\text{输出 } y(\vec{x}) = \vec{w}^T \vec{x} + \epsilon$$

其中权重向量为 D 维实数向量 $\vec{w} = (w_1, w_2, \dots, w_D)^T$

残差为服从正态分布的随机变量 $\epsilon \sim N(0, \sigma^2)$

$$\text{有对数似然函数 } L(\vec{w}, \sigma^2) = -\frac{1}{2\sigma^2} \text{RSS}(\vec{w}) - \frac{N}{2} \ln(2\pi\sigma^2)$$

$$\text{其中残差平方和 } \text{RSS}(\vec{w}) = \sum_{i=1}^N (y_i - \vec{w}^T \vec{x}_i)^2$$

$$\text{有 } \hat{\vec{w}}_{OLS} = (\vec{x}^T \vec{x})^{-1} \vec{x}^T \vec{y}$$

$$\text{其中 } N \times D \text{ 矩阵 } \vec{x} = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N)^T, N \text{ 维列向量 } \vec{y} = (y_1, y_2, \dots, y_N)^T$$

注意在矩阵 $\vec{x} = \begin{bmatrix} x_{11} & \dots & x_{1D} \\ \vdots & \ddots & \vdots \\ x_{N1} & \dots & x_{ND} \end{bmatrix}$ 中的列定义了一个 D 维线性子空间

column define linear subspace of dimensionality D
embedded in N dimensions

令 $\tilde{\vec{x}}_j$ 表示矩阵 \vec{x} 的第 j 列，有 $\tilde{\vec{x}}_j \in \mathbb{R}^N$

$$\text{即有 } \tilde{\vec{x}}_j = (\vec{x}_{1j}, \vec{x}_{2j}, \dots, \vec{x}_{Nj})^T$$

于是有矩阵 \vec{x} 的线性生成空间 (linear span)

$$\text{span}(\vec{x}) = \text{span}(\tilde{\vec{x}}_1, \dots, \tilde{\vec{x}}_D) = \{w_1 \tilde{\vec{x}}_1 + w_2 \tilde{\vec{x}}_2 + \dots + w_D \tilde{\vec{x}}_D \mid w_1, \dots, w_D \in \mathbb{R}\}$$

则问题转换为希望得到一个 N 维列向量 $\hat{\vec{y}} \in \mathbb{R}^N$ 且 $\hat{\vec{y}} \in \text{span}(\vec{x})$

使得 $\hat{\vec{y}}$ 与已知的输出样本向量 \vec{y} 距离尽可能近

$$\text{即 } \underset{\hat{\vec{y}} \in \text{span}(\vec{x})}{\text{argmin}} \|\hat{\vec{y}} - \vec{y}\|_2$$

为了最小化残差的范数 (norm of residual)

希望残差向量与矩阵 \vec{x} 的每一列都是正交的

residual vector orthogonal to every column of \vec{x}

$$\text{即 } \forall 1 \leq j \leq D \quad \tilde{\vec{x}}_j^T (\vec{y} - \hat{\vec{y}}) = 0$$

又矩阵 $\vec{x} = (\tilde{\vec{x}}_1, \tilde{\vec{x}}_2, \dots, \tilde{\vec{x}}_D)$

$$\text{向量 } \hat{\vec{y}} = w_1 \tilde{\vec{x}}_1 + w_2 \tilde{\vec{x}}_2 + \dots + w_D \tilde{\vec{x}}_D = \vec{x} \vec{w}$$

$$\text{于是 } \vec{x}^T (\vec{y} - \hat{\vec{y}}) = (\tilde{\vec{x}}_1, \tilde{\vec{x}}_2, \dots, \tilde{\vec{x}}_D)^T \vec{x} (\vec{y} - \hat{\vec{y}})$$

$$= (\tilde{\vec{x}}_1^T (\vec{y} - \hat{\vec{y}}), \tilde{\vec{x}}_2^T (\vec{y} - \hat{\vec{y}}), \dots, \tilde{\vec{x}}_D^T (\vec{y} - \hat{\vec{y}}))^T$$

$$= \vec{0}, D \text{ 维列向量}$$

$$\text{即 } \vec{x}^T (\vec{y} - \hat{\vec{y}}) = \vec{x}^T (\vec{y} - \vec{x} \vec{w}) = \vec{0}, \quad \hat{\vec{w}} = (\vec{x}^T \vec{x})^{-1} \vec{x}^T \vec{y}$$

于是 $\hat{\vec{y}}$ 有 \vec{y} 的投影值 (projected value)

$$\hat{\vec{y}} = \vec{x} \hat{\vec{w}} = \vec{x} (\vec{x}^T \vec{x})^{-1} \vec{x}^T \vec{y}$$

称为向量 \vec{y} 在 column space of \vec{x} 上的正交投影 (orthogonal projection)

称投影矩阵 $P = \vec{x} (\vec{x}^T \vec{x})^{-1} \vec{x}^T$ 为 hat matrix

Machine

Learning - P28

线性回归

对于训练集 $T = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

其中输入空间为 D 维实数列向量 $\vec{x} \in \mathbb{R}^D$, 轮出 $y(\vec{x}) = \vec{w}^\top \vec{x} + \varepsilon$

权重向量为 D 维实数列向量 $\vec{w} = (w_1, w_2, \dots, w_D)^\top$

残差为服从正态分布的随机变量 $\varepsilon \sim N(0, \sigma^2)$

\vec{w} 的参数估计 $\hat{\vec{w}} = (\vec{x}^\top \vec{x})^{-1} \vec{x}^\top \vec{y}$

其中 $N \times D$ 矩阵 $\vec{X} = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N)^\top$, N 维列向量 $\vec{y} = (y_1, \dots, y_N)^\top$

则考虑对参数 σ^2 的估计

$$\text{有对数似然函数 } L(\sigma^2) = \sum_{i=1}^N \ln \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left[-\frac{(y_i - \hat{w}^\top \vec{x}_i)^2}{2\sigma^2} \right]$$

$$= -\frac{N}{2} \ln 2\pi - \frac{N}{2} \ln \sigma^2 - \frac{1}{2\sigma^2} \sum_{i=1}^N (y_i - \hat{w}^\top \vec{x}_i)^2$$

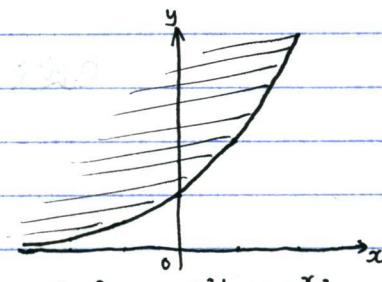
取一阶导数 $L'(\sigma^2) = 0$

$$\text{则 } -\frac{N}{2} \cdot \frac{2\sigma}{\sigma^2} - \frac{-2}{2\sigma^3} \sum_{i=1}^N (y_i - \hat{w}^\top \vec{x}_i)^2 = 0$$

$$\frac{N}{\sigma} = \frac{1}{\sigma^3} \sum_{i=1}^N (y_i - \hat{w}^\top \vec{x}_i)^2$$

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{w}^\top \vec{x}_i)^2$$

于是有参数估计 $\hat{\sigma}^2_{MLE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{w}^\top \vec{x}_i)^2$



凸性

(convexity), 对于集合 S , 如果对于任意 $\theta_1, \theta_2 \in S$ 都满足

$$\forall \lambda \in [0, 1] \quad \lambda \theta_1 + (1-\lambda) \theta_2 \in S$$

注意集合 S 中的元素 θ 可以是标量值, 也可以是向量值

在二维的形式中, 集合 S 可以表示为平面上点的集合构成的图形

注意这个图形是凸的(convex)并不蕴含图形是闭合的

由此定义函数 $f(\theta)$ 的凸性

令 $f(\theta)$ 的 epigraph 为集合 $\{(\theta, y) \in D \times \mathbb{R} \mid y \geq f(\theta)\}$

如果 $f(\theta)$ 的 epigraph 为一个凸的集合

则称 $f(\theta)$ 函数为凸函数 (convex function)

或者等价地定义为

$$\forall \theta_1, \theta_2 \in D \quad \forall \lambda \in [0, 1] \quad f[\lambda \theta_1 + (1-\lambda) \theta_2] \leq \lambda f(\theta_1) + (1-\lambda) f(\theta_2)$$

扩展到称函数 $f(\theta)$ 为严格凸的 (strictly convex)

如果 $\forall \theta_1, \theta_2 \in D \quad \forall \lambda \in (0, 1) \quad f[\lambda \theta_1 + (1-\lambda) \theta_2] < \lambda f(\theta_1) + (1-\lambda) f(\theta_2)$

对于具有二阶连续偏导数的多元函数 $f(\vec{\theta})$, 其中 $\vec{\theta} = (\theta_1, \dots, \theta_D)$

则其 Hessian 矩阵 $\begin{bmatrix} \frac{\partial^2}{\partial \theta_1^2} & \cdots & \frac{\partial^2}{\partial \theta_1 \partial \theta_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial \theta_D \partial \theta_1} & \cdots & \frac{\partial^2}{\partial \theta_D^2} \end{bmatrix}$

$f(\vec{\theta})$ 为正定矩阵 (positive definite)

对于 $n \times n$ 对称矩阵 A , 矩阵 A 为正定的 (positive definite)

如果对于任意 n 维非零向量 $\vec{v} \in \mathbb{R}^n$, 有 $\vec{v}^\top A \vec{v} > 0$

Machine

Learning - P29

多输出线性回归，对于训练集 $T = \{(\vec{x}_1, \vec{y}_1), (\vec{x}_2, \vec{y}_2), \dots, (\vec{x}_N, \vec{y}_N)\}$

其中输入空间为 D 维实数列向量集合 $X \subseteq \mathbb{R}^D$

输出空间为 M 维实数列向量集合 $Y \subseteq \mathbb{R}^M$

即具有 M 个相互独立的输出值

且有 $y_j(\vec{x}) = \vec{w}_j^\top \vec{x} + \epsilon_j$

其中 \vec{w}_j 为输出 j 的权重向量 D 维实数列向量 $(w_{j1}, w_{j2}, \dots, w_{jD})^\top$

输出 j 的残差服从正态分布 $\epsilon_j \sim N(0, \sigma_j^2)$

$N \times D$ 矩阵 $X = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N)^\top$

$N \times M$ 矩阵 $Y = (\vec{y}_1, \vec{y}_2, \dots, \vec{y}_N)^\top$

$D \times M$ 矩阵 $W = (\vec{w}_1, \vec{w}_2, \dots, \vec{w}_M)$

则有 $P(Y|X, W) = \prod_{j=1}^M N(y_j | \vec{w}_j^\top \vec{x}, \sigma_j^2)$

于是有对数似然函数

$$\begin{aligned} L(\theta) &= \ln P(T|\theta) = \ln \prod_{i=1}^N P(y_i | \vec{x}_i, \theta) = \sum_{i=1}^N \ln P(y_i | \vec{x}_i, \theta) \\ &= \sum_{i=1}^N \ln \prod_{j=1}^M N(y_{ij} | \vec{w}_j^\top \vec{x}_i, \sigma_j^2) \\ &= \sum_{i=1}^N \sum_{j=1}^M \ln N(y_{ij} | \vec{w}_j^\top \vec{x}_i, \sigma_j^2) \\ &= \sum_{j=1}^M \sum_{i=1}^N \ln N(y_{ij} | \vec{w}_j^\top \vec{x}_i, \sigma_j^2) \end{aligned}$$

即可以对于每个输出的权重向量 \vec{w}_j 进行独立的参数估计

取 N 维向量 \tilde{y}_j 表示矩阵 Y 的第 j 列

也表示所有输出中第 j 个结果所构成的向量

于是有 $\hat{w}_{j, OLS} = (X^\top X)^{-1} X^\top \tilde{y}_j$

则有对权重矩阵的参数估计

$$\begin{aligned} \hat{W} &= (\hat{w}_{1, OLS}, \hat{w}_{2, OLS}, \dots, \hat{w}_{M, OLS}) \\ &= ((X^\top X)^{-1} X^\top \tilde{y}_1, (X^\top X)^{-1} X^\top \tilde{y}_2, \dots, (X^\top X)^{-1} X^\top \tilde{y}_M) \\ &= (X^\top X)^{-1} X^\top (\tilde{y}_1, \tilde{y}_2, \dots, \tilde{y}_M) \\ &= (X^\top X)^{-1} X^\top Y \end{aligned}$$

于是有矩阵 Y 的投影值 (projected value)

$$\hat{Y} = X \hat{W} = X (X^\top X)^{-1} X^\top Y$$

可知对于多输出线性回归有相同的投影矩阵 (hat matrix) $P = X (X^\top X)^{-1} X^\top$

类似地对于输出 j 的参数 σ_j^2 也可以进行独立的参数估计

$$\hat{\sigma}_j^2 MLE = \frac{1}{N} \sum_{i=1}^N (\tilde{y}_j^{(i)} - X \hat{w}_j^{(i)})^2 = \frac{1}{N} \| \tilde{y}_j - X \hat{w}_j \|_2^2$$

$$= \frac{1}{N} (\tilde{y}_j - X \hat{w}_j)^\top (\tilde{y}_j - X \hat{w}_j)$$

Machine

Learning - P30

岭回归

(ridge regression), 对于训练集 $T = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

其中输入空间为 D 维实数列向量 $\vec{x} \in \mathbb{R}^D$, 输出 $y(\vec{x}) = \vec{w}^T \vec{x} + \varepsilon$

权重向量为 D 维实数列向量 $\vec{w} = (w_1, w_2, \dots, w_D)^T$

残差为服从正态分布的随机变量 $\varepsilon \sim N(0, \sigma^2)$

有对数似然函数 $L(\theta) = \frac{-1}{2\sigma^2} \sum_{i=1}^N (y_i - \vec{w}^T \vec{x}_i)^2 - \frac{N}{2} \ln(2\pi\sigma^2)$

注意到当 D 过大时, 对于参数 \vec{w} 的 OLS 估计

由于提高了拟合精度, 则会将噪音也进行拟合, 从而出现过拟合 (overfitting)

而在这个过程中 $\|\vec{w}\|$ 会变得非常大

通过将又微小的变化放大, 通过正负项叠加尽量将各个样本拟合

为了避免这种情况, 可以认为权重向量 \vec{w} 中的系数服从独立同分布的正态分布

即有 $P(\vec{w}) = \prod_{j=1}^D N(w_j | 0, \sigma^2)$

其中方差 σ^2 (系数 $1/\sigma^2$) 用于控制先验概率的强度

于是有权重向量 \vec{w} 的最大后验分布

$$\operatorname{argmax}_{\vec{w}} \ln P(\vec{w}|T) = \operatorname{argmax}_{\vec{w}} \ln [P(T|\vec{w}) P(\vec{w})]$$

$$= \operatorname{argmax}_{\vec{w}} \ln [\prod_{i=1}^N N(y_i | \vec{w}^T \vec{x}_i, \sigma^2) \prod_{j=1}^D N(w_j | 0, \sigma^2)]$$

$$= \operatorname{argmax}_{\vec{w}} \left[\sum_{i=1}^N \left[\ln N(y_i | \vec{w}^T \vec{x}_i, \sigma^2) + \sum_{j=1}^D \ln N(w_j | 0, \sigma^2) \right] \right]$$

$$= \operatorname{argmax}_{\vec{w}} \left[\sum_{i=1}^N \ln \left[\frac{1}{(2\pi)^{1/2} \sigma} \exp \left(\frac{(y_i - \vec{w}^T \vec{x}_i)^2}{-2\sigma^2} \right) \right] + \sum_{j=1}^D \ln \left[\frac{1}{(2\pi)^{1/2} \sigma} \exp \left(\frac{w_j^2}{-2\sigma^2} \right) \right] \right]$$

$$= \operatorname{argmax}_{\vec{w}} \left[\sum_{i=1}^N \left(-\frac{(y_i - \vec{w}^T \vec{x}_i)^2}{2\sigma^2} \right) + \sum_{j=1}^D \left(-\frac{w_j^2}{2\sigma^2} \right) \right]$$

$$= \operatorname{argmin}_{\vec{w}} \left[\sum_{i=1}^N (y_i - \vec{w}^T \vec{x}_i)^2 / 2\sigma^2 + \sum_{j=1}^D w_j^2 / 2\sigma^2 \right]$$

$$= \operatorname{argmin}_{\vec{w}} \left[\sum_{i=1}^N (y_i - \vec{w}^T \vec{x}_i)^2 + \frac{\sigma^2}{2} \|\vec{w}\|_2^2 \right]$$

σ^2, σ^2 均为与 \vec{w} 无关
的常数

再取 $N \times D$ 矩阵 $X = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N)^T$, N 维列向量 $\vec{y} = (y_1, y_2, \dots, y_N)^T$

$$= \operatorname{argmin}_{\vec{w}} \|\vec{y} - X\vec{w}\|_2^2 + \frac{\sigma^2}{2} \|\vec{w}\|_2^2$$

$$= \operatorname{argmin}_{\vec{w}} \|\vec{y} - X\vec{w}\|_2^2 + \lambda \vec{w}^T \vec{w}$$

取参数 $\lambda = \frac{\sigma^2}{2}$

再取 \vec{w} 的导数, 令导数为 0

则有 $2X^T(X\vec{w} - \vec{y}) + \lambda \vec{w} = 0$

$$(X^T X + \lambda I_D) \vec{w} = X^T \vec{y}$$

$$\text{于是有 } \hat{\vec{w}}_{\text{ridge}} = (X^T X + \lambda I_D)^{-1} X^T \vec{y}$$

这个方法可以直观地描述为对于参数 \vec{w} 过大进行惩罚 (penalty)

于是岭回归 (ridge regression) 也称为 penalized least square

加入惩罚项入 $\|\vec{w}\|_2^2$ 的过程称为正则化 (regularization)

由于惩罚项是 \vec{w} 的 l_2 范数, 于是也称为 l_2 regularization / weight decay

当 $\lambda \rightarrow 0$ 时, 与原来的目标函数接近 (逼近)

而当 λ 过大时, 惩罚项影响太大, 则最小化结果为 $\vec{w} = \vec{0}$, 出现拟合不足 (underfitting)

Machine

Learning - P31

机器学习 - 第三章

套索回归

(Lasso regression), 对于训练集 $T = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

其中输入空间为 D 维实数列向量 $\vec{x} \in \mathbb{R}^D$, 输出 $y(\vec{x}) = \vec{w}^\top \vec{x} + \epsilon$

权重向量为 D 维实数列向量 $\vec{w} = (w_1, w_2, \dots, w_D)^\top$

残差为服从正态分布的随机变量 $\epsilon \sim N(0, \sigma^2)$

则有线性回归参数 \vec{w} 的目标函数 $J(\vec{w}) = \|\vec{y} - \vec{x}\vec{w}\|_2^2$

其中 $N \times D$ 矩阵 $X = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N)^\top$, N 维向量 $\vec{y} = (y_1, y_2, \dots, y_N)^\top$

如果认为权重向量 \vec{w} 中的系数 w_j 服从独立同分布的正态分布

即有 $w_j \sim N(0, \gamma^2)$

则有一个包含权重向量 \vec{w} 的 L_2 范数的新目标函数

$$J(\vec{w}) = \|\vec{y} - \vec{x}\vec{w}\|_2^2 + \alpha \|\vec{w}\|_2^2, \text{ 其中 } \alpha \propto \sigma^2 / \gamma^2$$

而如果认为权重向量 \vec{w} 中的系数 w_j 服从 IID 的拉普拉斯分布 (Laplace distribution)

即 $w_j \sim La(0, \beta)$, 即 $P(\vec{w}) = \prod_{j=1}^D \frac{1}{2\beta} \exp\left[-\frac{|w_j|}{\beta}\right]$

是权重向量的最大后验分布

$$\begin{aligned} \operatorname{argmax}_{\vec{w}} P(\vec{w} | T) &= \operatorname{argmax}_{\vec{w}} \ln \left[\prod_{i=1}^N N(y_i; \vec{w}^\top \vec{x}_i, \sigma^2) \prod_{j=1}^D La(w_j; 0, \beta) \right] \\ &= \operatorname{argmax}_{\vec{w}} \left[\sum_{i=1}^N \ln \left[\frac{1}{c_2 \pi \sigma^2} \exp\left[-\frac{(y_i - \vec{w}^\top \vec{x}_i)^2}{2\sigma^2}\right] \right] + \sum_{j=1}^D \ln \left[\frac{1}{2\beta} \exp\left[-\frac{|w_j|}{\beta}\right] \right] \right] \\ &= \operatorname{argmin}_{\vec{w}} \sum_{i=1}^N (y_i - \vec{w}^\top \vec{x}_i)^2 + \frac{2\beta^2}{\sigma^2} \sum_{j=1}^D |w_j| \\ &= \operatorname{argmin}_{\vec{w}} \|\vec{y} - \vec{x}\vec{w}\|_2^2 + \frac{2\beta^2}{\sigma^2} \|\vec{w}\|_1 \end{aligned}$$

于是有目标函数 $J(\vec{w}) = \|\vec{y} - \vec{x}\vec{w}\|_2^2 + \alpha \|\vec{w}\|_1$, 其中 $\alpha \propto \sigma^2 / \beta$

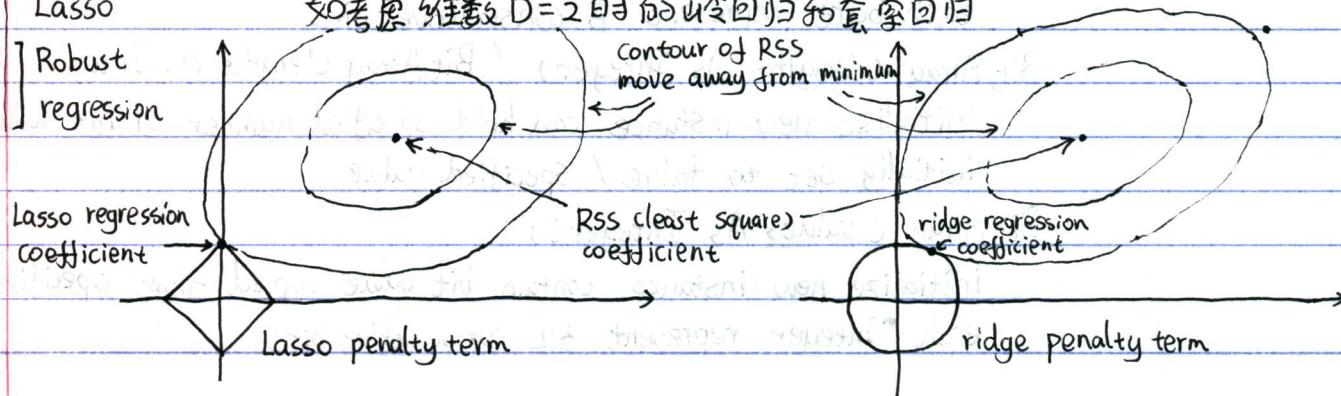
称为套索回归 (Lasso regression)

与岭回归相似地对权重向量过大进行惩罚

但惩罚项是权重向量 \vec{w} 的 L_1 范数 $\|\vec{w}\|_1$

系数 α 同样决定先验分布的强度

likelihood	prior	name	note
Gaussian	Uniform	Least Square	但是在套索回归的系数 α 增大的过程中
Gaussian	Gaussian	Ridge	会有越来越多的 \vec{w} 中的系数的最优化为 0
Gaussian	Laplace	Lasso	如考虑维数 $D=2$ 时的岭回归和套索回归
Laplace	Uniform	Robust regression	Contour of RSS move away from minimum
Student	Uniform		RSS (least square) coefficient



Machine

Learning - P32

逻辑回归 (logistic regression), 对于训练集 $T = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

其中输入空间为 D 维实数列向量 $\vec{x} \in \mathbb{R}^D$, 轮出空间为二分类标记 $y_i \in \{0, 1\}$

则有权重向量为 D 维实数列向量 $\vec{w} = (w_1, w_2, \dots, w_D)^T$

二分类模型 (binary classification model)

$$P(y| \vec{x}, \vec{w}) = \text{Ber}(y | \sigma(\vec{w}^T \vec{x}))$$

其中 $\sigma(\vec{w}^T \vec{x})$ 为 sigmoid function

$$\text{有 } \sigma(x) = \frac{1}{1+e^{-x}}, x \in \mathbb{R}$$

$\sigma(x)$ 的值域为 $(0, 1)$

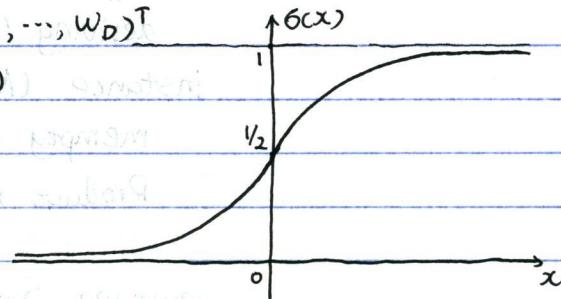
$$\text{于是有 } P(y=1 | \vec{x}, \vec{w}) = \sigma(\vec{w}^T \vec{x}) = \frac{1}{1+e^{-\vec{w}^T \vec{x}}}$$

对于任意输入/输出 (\vec{x}_i, y_i)

$$\text{有 } \sigma(\vec{w}^T \vec{x}_i)^{y_i} \cdot (1 - \sigma(\vec{w}^T \vec{x}_i))^{1-y_i} = P(y_i | \vec{x}_i)$$

即当 $y_i = 1$ 时, $P(\vec{x}_i, y_i) = \sigma(\vec{w}^T \vec{x}_i)$

当 $y_i = 0$ 时, $P(\vec{x}_i, y_i) = 1 - \sigma(\vec{w}^T \vec{x}_i)$



对于参数权重向量 \vec{w} 进行极大似然估计

$$\text{有 } \underset{\vec{w}}{\operatorname{argmax}} P(T | \vec{w}) = \underset{\vec{w}}{\operatorname{argmax}} \prod_{i=1}^N P(y_i | \vec{x}_i, \vec{w})$$

$$= \underset{\vec{w}}{\operatorname{argmax}} \ln \prod_{i=1}^N \sigma(\vec{w}^T \vec{x}_i)^{y_i} (1 - \sigma(\vec{w}^T \vec{x}_i))^{1-y_i}$$

$$= \underset{\vec{w}}{\operatorname{argmax}} \sum_{i=1}^N \ln [\sigma(\vec{w}^T \vec{x}_i)^{y_i} (1 - \sigma(\vec{w}^T \vec{x}_i))^{1-y_i}]$$

于是有负对数似然函数

$$NLL(\vec{w}) = -\frac{1}{N} \sum_{i=1}^N [y_i \ln \sigma(\vec{w}^T \vec{x}_i) + (1-y_i) \ln (1 - \sigma(\vec{w}^T \vec{x}_i))]$$

$$\text{又 } \sigma'(x) = (\frac{1}{1+e^{-x}})' = (\frac{-1}{(1+e^{-x})^2} \cdot (1+e^{-x})')' = \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} = \sigma(x)(1-\sigma(x))$$

于是对负对数似然求导则有

$$\frac{d}{d\vec{w}} NLL(\vec{w}) = -\frac{1}{N} \sum_{i=1}^N [y_i \ln \sigma(\vec{w}^T \vec{x}_i) + (1-y_i) \ln (1 - \sigma(\vec{w}^T \vec{x}_i))]'$$

$$= -\frac{1}{N} \sum_{i=1}^N [y_i / \sigma(\vec{w}^T \vec{x}_i) + (1-y_i) / (1 - \sigma(\vec{w}^T \vec{x}_i))] \sigma'(\vec{w}^T \vec{x}_i)$$

$$= -\frac{1}{N} \sum_{i=1}^N [y_i / \sigma(\vec{w}^T \vec{x}_i) - (1-y_i) / (1 - \sigma(\vec{w}^T \vec{x}_i))] \sigma(\vec{w}^T \vec{x}_i) (1 - \sigma(\vec{w}^T \vec{x}_i)) \vec{x}_i$$

$$= -\frac{1}{N} \sum_{i=1}^N [y_i (1 - \sigma(\vec{w}^T \vec{x}_i)) - (1-y_i) \sigma(\vec{w}^T \vec{x}_i)] \vec{x}_i$$

$$= -\frac{1}{N} \sum_{i=1}^N (6(\vec{w}^T \vec{x}_i) - y_i) \vec{x}_i$$

取 $N \times D$ 矩阵 $X = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N)^T$, N 维列向量 $\vec{y} = (y_1, y_2, \dots, y_N)^T$

$$N$$
 维列向量 $\vec{\mu} = (\mu_1, \mu_2, \dots, \mu_N)^T = (6(\vec{w}^T \vec{x}_1), 6(\vec{w}^T \vec{x}_2), \dots, 6(\vec{w}^T \vec{x}_N))^T$

$$\text{则有梯度 (gradient) 函数 } g(\vec{w}) = \frac{1}{N} \sum_{i=1}^N (6(\vec{w}^T \vec{x}_i) - y_i) \vec{x}_i = \frac{1}{N} X^T (\vec{\mu} - \vec{y})$$

特别注意: 由于 $6(\vec{w}^T \vec{x})$ 是非线性函数, 所以无法直接令导数为 0 进行求解

只能采取其他方式尽可能逼近最优化的 $\hat{\vec{w}}$

Machine

Learning - P33

逻辑回归 (logistic regression), 对于训练集 $T = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$
其中输入空间为 D 维实数列向量 $\vec{x} \in \mathbb{R}^D$, 轮出空间为二分类标记 $y_i \in \{0, 1\}$
则有权重向量为 D 维实数列向量 $\vec{w} = (w_1, w_2, \dots, w_D)^T$

$$\text{于是有 } P(y=1|\vec{x}, \vec{w}) = G(\vec{w}^T \vec{x}) = \frac{1}{1+e^{-\vec{w}^T \vec{x}}}$$

$$\text{有负对数似然函数 } NLL(\vec{w}) = -\frac{1}{N} \sum_{i=1}^N [y_i \ln G(\vec{w}^T \vec{x}_i) + (1-y_i) \ln (1-G(\vec{w}^T \vec{x}_i))]$$

$$\text{梯度函数 } g(\vec{w}) = \frac{1}{N} \sum_{i=1}^N (G(\vec{w}^T \vec{x}_i) - y_i) \vec{x}_i = \frac{1}{N} \vec{x}^T (\vec{\mu} - \vec{y})$$

$$\text{其中 } N \times D \text{ 矩阵 } \vec{X} = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N)^T, N \text{ 维向量 } \vec{y} = (y_1, y_2, \dots, y_N)^T$$

$$\text{且梯度向量 } g(\vec{w}) \text{ 只为 } N \text{ 维列向量 } \vec{\mu} = (\mu_1, \mu_2, \dots, \mu_N)^T = (G(\vec{w}^T \vec{x}_1), G(\vec{w}^T \vec{x}_2), \dots, G(\vec{w}^T \vec{x}_N))^T$$

由于 $y_i \in \{0, 1\}$, 于是考虑训练集中 $y=0$ 与 $y=1$ 的样本数分别为 N_0, N_1

$$\text{则 } NLL(\vec{w}) = -\frac{1}{N} \sum_{i=1}^N [y_i \ln G(\vec{w}^T \vec{x}_i) + (1-y_i) \ln (1-G(\vec{w}^T \vec{x}_i))]$$

由于 y_i 与 $(1-y_i)$ 的值可以看作 $y_i=1$ 与 $y_i=0$ 的概率

$$P(y_i=1) = y_i, P(y_i=0) = 1-y_i$$

$$\text{于是 } NLL(\vec{w}) = -\frac{1}{N} \sum_{i=1}^N P(y_i=1) \ln G(\vec{w}^T \vec{x}_i) - \frac{1}{N} \sum_{i=1}^N P(y_i=0) \ln (1-G(\vec{w}^T \vec{x}_i))$$

则负对数似然函数实质上可以看作

真实概率分布 $P(y_i)$ 与模型分布 $G(\vec{w}^T \vec{x}_i)$ 的交叉熵

于是 $NLL(\vec{w})$ 也可以作为 logistic regression 的损失函数 (cost function)

考虑用二分类标记 $\{-1, +1\}$ 代替 $\{0, 1\}$

$$\text{于是有 } P(y=-1|\vec{x}, \vec{w}) = 1 - G(\vec{w}^T \vec{x}) = \frac{e^{-\vec{w}^T \vec{x}}}{1+e^{-\vec{w}^T \vec{x}}}$$

$$= 1/(1+e^{\vec{w}^T \vec{x}}) = G(-\vec{w}^T \vec{x})$$

$$\text{则可以统一概率为 } P(y_i|\vec{x}_i, \vec{w}) = G(y_i \vec{w}^T \vec{x})$$

$$\text{于是有负对数似然函数 } NLL(\vec{w}) = \frac{1}{N} \sum_{i=1}^N \ln G(y_i \vec{w}^T \vec{x})$$

$$= \frac{1}{N} \sum_{i=1}^N \ln (1+e^{-y_i \vec{w}^T \vec{x}})$$

$$\text{于是有梯度函数 } g(\vec{w}) = \frac{d}{d\vec{w}} \cdot \sum_{i=1}^N \ln (1+e^{-y_i \vec{w}^T \vec{x}})$$

$$= \frac{1}{N} \sum_{i=1}^N \frac{1}{1+e^{-y_i \vec{w}^T \vec{x}}} \cdot \frac{d}{d\vec{w}} (1+e^{-y_i \vec{w}^T \vec{x}})$$

$$= \frac{1}{N} \sum_{i=1}^N e^{-y_i \vec{w}^T \vec{x}} / (1+e^{-y_i \vec{w}^T \vec{x}}) \cdot \frac{d}{d\vec{w}} (-y_i \vec{w}^T \vec{x})$$

$$= \frac{1}{N} \sum_{i=1}^N G(-y_i \vec{w}^T \vec{x}) \cdot (-y_i \vec{x})$$

可以看作是 $-y_i \vec{x}_i$ 的线性加权, 且要求为 0

如果是线性可分的, 则只要 $G(-y_i \vec{w}^T \vec{x})$ 为 0 即可

但根据 $G(-y_i \vec{w}^T \vec{x})$ 的性质, 相当于要求 $-y_i \vec{w}^T \vec{x} \ll 0$, 即 $y_i \vec{w}^T \vec{x} \gg 0$

实际上不可行, 于是还是需要其他途径得到最优化的权重向量 \vec{w}

Machine

Learning - P34

逻辑回归

对于训练集 $T = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

其中输入空间为 D 维实数列向量 $\vec{x} \in \mathbb{R}^D$, 轮出空间为二分类标记 $y_i \in \{0, 1\}$

有权重向量为 D 维实数列向量 $\vec{w} = (w_1, w_2, \dots, w_D)^T$

有负对数似然函数 $NLL(\vec{w}) = -\frac{1}{N} \sum_{i=1}^N [y_i \ln(\sigma(\vec{w}^T \vec{x}_i)) + (1-y_i) \ln(1-\sigma(\vec{w}^T \vec{x}_i))]$

梯度函数 $g(\vec{w}) = \frac{1}{N} \sum_{i=1}^N (\sigma(\vec{w}^T \vec{x}_i) - y_i) \vec{x}_i = \frac{1}{N} \vec{x}^T (\vec{\mu} - \vec{y})$

其中 $N \times D$ 矩阵 $\vec{X} = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N)^T$, N 维向量 $\vec{y} = (y_1, y_2, \dots, y_N)^T$

(非满秩) $\vec{\mu} = (\mu_1, \mu_2, \dots, \mu_N)^T = (\sigma(\vec{w}^T \vec{x}_1), \sigma(\vec{w}^T \vec{x}_2), \dots, \sigma(\vec{w}^T \vec{x}_N))^T$

梯度下降 (gradient descent), 也称 steepest descent

当数据维度较高或者导数比较复杂时, 很难通过令导数为0的方法进行求解

考虑目标函数 $J(\theta)$ 为一个凸函数, 则可以通过以下方法:

1. 对于给定的参数初值 θ , 有梯度函数 $\frac{\partial J(\theta)}{\partial \theta}$

2. 令梯度负方向更新参数的值 $\theta := \theta - \alpha \frac{\partial J(\theta)}{\partial \theta}$

其中 α 为学习率 (learning rate), step size

重复 1 和 2 步骤直至 θ 的值收敛

在逻辑回归中, 则有 $\vec{\theta}_{k+1} = \vec{\theta}_k - \eta_k \vec{g}_k$, 其中 η_k 为学习率

stable method for picking step size, guaranteed to converge to local optimum

称为 global convergence (全局收敛)

Taylor's theorem: $f(\vec{\theta} + \eta \vec{d}) \approx f(\vec{\theta}) + \eta \vec{g}^T \vec{d}$, \vec{d} : descent direction

pick $\arg\min_{\eta \geq 0} \phi(\eta) = f(\vec{\theta}_k + \eta \vec{d}_k)$

称为 line minimization / line search

zig-zag behavior: characteristic of steepest descent path

necessary condition for optimum: $\phi'(\eta) = 0$

$\phi'(\eta) = \vec{d}^T \vec{g}$, $\vec{g} = f(\vec{\theta} + \eta \vec{d})$ (gradient at the end of step)

于是或者 $\vec{g} = \vec{0}$, 即找到了一个驻点 (stationary point)

或者 $\vec{g} \perp \vec{d}$, 即局部梯度与 search direction 垂直

momentum term: reduce effect of zig-zag

$\vec{\theta}_{k+1} = \vec{\theta}_k - \eta_k \vec{g}_k + \mu_k (\vec{\theta}_k - \vec{\theta}_{k-1})$

参数 $0 \leq \mu_k \leq 1$ 控制 momentum term 的重要性

称为 heavy ball method

