i.i.d: independent and identically distributed

1 Math Basics

Eigenvectors: $Ax = \lambda x$

A matrix $A \in \mathcal{R}^n$ has eigenvectors if A is square and not singular ($\det(A) \neq 0$).

$$A = A^\top \implies \lambda_i \text{ is real} \qquad \text{rank}(A) = n \implies \lambda_i \neq 0 \quad \forall i$$

$$x^\top Ax > 0 \implies \lambda_i > 0 \qquad x^\top Ax \geq 0 \implies \lambda_i \geq 0 \quad \forall i$$

Positive definite: $x^\top Ax > 0 \quad \forall x \neq 0$

Positive semidefinite: $x^\top Ax \geq 0 \quad \forall x \neq 0$

Jacoobi-Matrix: $\mathbf{J}_f(\mathbf{x}) = \frac{df(\mathbf{x})}{d\mathbf{x}} = \left[\frac{df_i(\mathbf{x})}{dx_j}\right]_{i=1\ldots m; j=1\ldots n}$

$g(x) \circ f(x) \implies \mathbf{J}_{g \circ f}(\mathbf{x}) = \mathbf{J}_g(f(\mathbf{x})) \cdot \mathbf{J}_f(\mathbf{x})$

Hessian-Matrix: $\mathbf{H}_f(\mathbf{x}) = \frac{d^2 f(\mathbf{x})}{d\mathbf{x}^2} = \left[\frac{\partial^2 f_i(\mathbf{x})}{\partial x_j \partial x_k}\right]_{i=1\ldots m; j,k=1\ldots n}$

1.1 Statistics

Normal Distribution: $f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

Bernoulli Distribution: $f_X(x) = p^x(1-p)^{1-x} \; x \in \{0,1\}$

Maximum likelihood estimation

$$L(x;\theta) = \prod_{i=1}^{N} f_{X_i}(x_i;\theta) \qquad l(x;\theta) = \sum_{i=1}^{N} \log f_{X_i}(x_i;\theta)$$

1.2 Convexity

1.3 Non-Linear Optimization

2 Kernel Trick

$\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a **kernel function** if the following two conditions are fulfilled (with an arbitrary function $f \in L_2(\mathcal{X})$):

Positive definite: $\int_{\mathcal{X}\times\mathcal{X}} f(\mathbf{x})\kappa(\mathbf{x},\mathbf{y})f(\mathbf{y})dxdy \geq 0$

Symmetry: $\kappa(\mathbf{x},\mathbf{y}) = \kappa(\mathbf{y},\mathbf{x})$

2.1 Common Kernels ($a, c, d \geq 0$ and $\sigma > 0$)

Linear Kernel: $\kappa(\mathbf{x},\mathbf{y}) = \mathbf{x}^T\mathbf{y} + c$

Polynomial Kernel: $\kappa(\mathbf{x},\mathbf{y}) = (a\mathbf{x}^T\mathbf{y} + c)^d$

Gaussian Kernel: $\kappa(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2}\right)$

Exponential Kernel: $\kappa(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{||\mathbf{x}-\mathbf{y}||}{2\sigma^2}\right)$

2.2 Properties

Given the kernels κ_1 and κ_2, $c > 0$ and an arbitrary function f the following combinations are valid kernels:

$$c\kappa_1(\mathbf{x}) \qquad c + \kappa_1(\mathbf{x}) \qquad f(\mathbf{x})f(\mathbf{y})$$

$$\kappa_1(\mathbf{x})\kappa_2(\mathbf{x}) \qquad \kappa_1(\mathbf{x}) + \kappa_2(\mathbf{x})$$

Mercer's Theorem: Let κ be a kernel, then there exists functions ϕ_i and a $\lambda_i \geq 0$ such that:

$$\kappa(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x})\phi_i(\mathbf{y})$$

3 Unsupervised Learning

3.1 K-Means

$$\mathcal{F} = \{f : \mathbb{R}^p \to \{\mathbf{c}_1, \ldots, \mathbf{c}_k\} \subset \mathbb{R}^p\} \quad L(\mathbf{X}, f(\mathbf{X})) = ||\mathbf{X} - \mathbf{f}(\mathbf{X})||^2$$

Where $\mathbb{R}^p$ is the feature space. $\mathbf{c}_i$ is the center of cluster i.

Because f maps to a finite set, the volume of the set distribution is zero.

Algorithms: Lloyd's algorithm, MaxQueen's algorithm

3.2 Principle Component Analysis

$$\mathcal{F} = \left\{f : \mathbb{R}^p \to \mathbb{R}^k \subset \mathbb{R}^p\right\} \qquad L(\mathbf{X}, f(\mathbf{X})) = ||\mathbf{X} - \mathbf{U}_k\mathbf{U}_k^\top\mathbf{X}||_2^2$$

Where $\mathbb{R}^p$ is the feature space. And $f(\mathbf{X}) = \mathbf{U}_k^\top\mathbf{X}$ is a (orthogonal) projection on to the subspace $\mathbb{R}^k$.

Principle components: $\mathbf{S} = \mathbf{U}_k^\top\mathbf{X} = \mathbf{\Sigma}_k\mathbf{V}_k^\top$

3.2.1 Kernel PCA

4 Supervised Learning

Expected Prediction Error: $\text{EPE(f)} = \text{E}\left[L(Y, f(X))\right]$

$$f(x) = \text{argmin}_{f\in\mathcal{F}}\,\text{EPE}(f) \approx \text{argmin}_{f\in\mathcal{F}}\,\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) = \hat{f}(x)$$

Quadratic Loss: $L_2(Y, f(X)) = (Y - f(X))^2$

Absolute Loss (l_1-loss): $L_1(Y, f(X)) = |Y - f(X)|$

$$f_2(x) = \text{E}_{Y|X=x}[Y] \qquad f_1(x) = \text{median}_{Y|X=x}[Y]$$

Generalization error: $G_N(f) = \text{EPE}(f) - \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i))$

4.1 Linear regression

$$\mathcal{F} = \left\{f(\mathbf{x}) = \theta_0 + \sum_{k=1}^{p}\theta_k x_k \,\middle|\, \theta_k \in \mathbb{R}, p \in \mathbb{N}\right\}$$

Where x_k is the kth element of the vector $\mathbf{x}$

4.2 K-nearest Neighbors

$$\hat{f}_k(x) = \frac{1}{k}\sum_{i\in N_k(x)} y_i \quad \forall i \text{ s.t. } x_i \in N_k(x)$$

Where $N_k(x)$ is the set of the k nearest neighbors of x

$$\lim_{N,k\to\infty,\,\frac{k}{N}\to 0} \hat{f}_k(x) = \text{E}[Y|X=x]$$

4.3 Logistic regression

$$\mathcal{F} = \{f(\mathbf{x}) = \mathbf{w}^\top\mathbf{x} + b \mid \mathbf{w} \in \mathbb{R}^p, b \in \mathbb{R}\}$$

$$L_{0,1}(Y, f(X)) = \begin{cases} 1 & \text{if } Yf(X) \leq 0 \\ 0 & \text{otherwise} \end{cases} \quad l(Y, f(X)) = \log\left(1 + e^{-Yf(X)}\right)$$

f could be an arbitrary function, but usually f is chosen as defined above.

$$Pr(Y = y|\mathbf{x}) = \exp(-l(y, f(\mathbf{x}))) = \frac{1}{1 + e^{-yf(\mathbf{x})}} = \sigma(yf(\mathbf{x}))$$

Regularization: $\tilde{l}(y, f(\mathbf{x})) = l(y, f(\mathbf{x})) + \lambda\left(||\mathbf{w}||^2 + b^2\right)$

4.4 Feedforward Neural Network

$$f : \mathbb{R}^p \to \mathbb{R}^r \qquad x \mapsto \sigma_l \circ \varphi_{\mathbf{w}_l} \circ \cdots \circ \sigma_1 \circ \varphi_{\mathbf{w}_1}(x)$$

$$\varphi_{\mathbf{w}} : \mathbb{R}^p \to \mathbb{R}^m \quad \mathbf{x} \mapsto \mathbf{W}\mathbf{x} \qquad \sigma : \mathbf{x} \mapsto [\sigma(x_1), \ldots, \sigma(x_m)]^\top$$

Where m is the number of neurons in the layer and p is the number of input features/neurons from the previous layer.

Rectified Linear Unit (ReLU): $\sigma(x) = \max(0, x)$

Update rule: $\mathbf{W}_j \leftarrow \mathbf{W}_j - \alpha\pi_j^{-1}\left(\frac{d}{d\mathbf{W}_j}L(\mathbf{y}_i, f(\mathbf{x}_i))\right)^\top$

Softmax: $\mathbf{x} \mapsto \left(\sum_{\mathbf{x}} \exp(x_i)\right)^{-1} [\exp(x_1), \ldots, \exp(x_C)]^\top$

Cross entropy: $H(p,q) = -\sum_{i=1}^{n} p_i \log q_i$; $L(f(\mathbf{x}), \mathbf{y}_c) = -log(f(\mathbf{x})_c)$

Where c is the correct class in a multiclass classification problem with C classes

4.5 Support Vector machine