2016 Spring Notes

Yue Yu

February 25, 2016

Contents

1　各种相关背景知识

1.1　概率论

1.1.1　条件概率期望:

$$E(X) = E(E(X|Y)) \tag{1}$$

1.1.2　Correlation coefficient(相关系数)

$$\begin{aligned}
\rho_{X,Y} &= \frac{Cov(X,Y)}{\sigma_X \sigma_Y} && (2)\\
&= \frac{E(X-E(X))(Y-E(Y))}{\sigma_X \sigma_Y} && (3)\\
&= \frac{E(XY)-E(X)E(Y)}{\sigma_X \sigma_Y} && (4)
\end{aligned}$$

1.1.3　协方差矩阵

$$\begin{aligned}
\mathrm{X} &= [X_1, \ldots, X_n]^T && (5)\\
\Sigma &= \mathrm{E}\Big[(\mathrm{X}-\mathrm{E}(\mathrm{X}))(\mathrm{X}-\mathrm{E}(\mathrm{X}))^T\Big] && (6)\\
\Sigma_{i,j} &= Cov(X_i, X_j) && (7)\\
&= \mathrm{E}\Big[(X_i-\mu_i)(X_j-\mu_j)\Big] && (8)
\end{aligned}$$

1.1.4　Multivariate normal distribution

- 多维高斯联合分布:

$$f_X(x_1, \ldots, x_n) = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}}\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$$

- 当为二维高斯分布时 (ρ 为相关系数)

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}}\exp\Big(-\frac{1}{2(1-\rho)^2}\Big[\frac{(x-\mu_x)}{\sigma_x^2}+\frac{(y-\mu_y)}{\sigma_y^2}-\frac{2\rho(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y}\Big]\Big)$$

  条件概率服从:

$$P(X_1\big|X_2 = x_2) \sim N(\mu_1 + \frac{\sigma_1}{\sigma_2}\rho(x_2-\mu_2), (1-\rho^2)\sigma_1^2)$$

- Product of multivariate gaussian

$$\begin{aligned}
N_X(\mu_a, \Sigma_a)\cdot N_X(\mu_b, \Sigma_b) &= z_c N_X(\mu_c, \Sigma_c)\\
\Sigma_c &= (\Sigma_a^{-1}+\Sigma_b^{-1})^{-1}\\
\mu_c &= \Sigma_c(\Sigma_a^{-1}\mu_a + \Sigma_b^{-1}\mu_b)\\
z_c &= |2\pi(\Sigma_a+\Sigma_b)|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\mu_a-\mu_b)^T(\Sigma_a+\Sigma_b)^{-1}(\mu_a-\mu_b)\right)
\end{aligned}$$

1.1.5 Laplace Distribution:

- 概率密度函数

$$f(x|\mu, b) = \frac{1}{2b}\exp\big(-\frac{|x-\mu|}{b}\big)$$

- 期望

$$\mu$$

- 方差

$$2b^2$$

1.2 线性代数

- Trace

$$\mathrm{Tr}(\mathbf{ABC}) = \mathrm{Tr}(\mathbf{BCA}) = \mathrm{Tr}(\mathbf{CAB})$$

1.3 矩阵求导法则

- 雅各比矩阵
  假设某函数从 $\mathbb{R}^n$ 映射到 $\mathbb{R}^m$

$$J_F(x_1, \dots, x_n) = \frac{\partial(y_1, \dots, y_m)}{\partial(x_1, \dots, x_n)} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix} \tag{9}$$

- Chain Rules:
  Suppose $\mathbf{y} = f(\mathbf{u}), \mathbf{u} = g(\mathbf{x})$

$$\frac{\partial(y_1, \dots, y_m)}{\partial(x_1, \dots, x_n)} = \frac{\partial(y_1, \dots, y_m)}{\partial(u_1, \dots, u_k)}\frac{\partial(u_1, \dots, u_k)}{\partial(x_1, \dots, x_n)} \tag{10}$$

- 对向量 (Vector Function) 求导 (定义列向量对标量求导得到的是行向量)：

$$\begin{aligned}
\frac{\partial \mathbf{u^T A v}}{\partial \mathbf{x}} &= \frac{\partial \mathbf{u}}{\partial \mathbf{x}}\mathbf{Av} + \frac{\partial \mathbf{v}}{\partial \mathbf{x}}\mathbf{A^T u} && (11)\\
\frac{\partial(\mathbf{u}(\mathbf{x}) + \mathbf{v}(\mathbf{x}))}{\partial \mathbf{x}} &= \frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} + \frac{\partial \mathbf{v}(\mathbf{x})}{\partial \mathbf{x}} && (12)\\
\frac{\partial \mathbf{Ax}}{\partial \mathbf{x}} &= \mathbf{A^T} && (13)\\
\frac{\partial \mathbf{a}}{\partial \mathbf{x}} &= \mathbf{0} && (14)\\
\frac{\partial \mathbf{x^T A b}}{\partial \mathbf{x}} &= \mathbf{Ab} && (15)\\
\frac{\partial \mathbf{x^T A x}}{\partial \mathbf{x}} &= (\mathbf{A} + \mathbf{A^T})\mathbf{x} && (16)\\
&= \mathbf{2Ax} \qquad \text{如果 A 为对称阵} && (17)\\
& && (18)
\end{aligned}$$

- 矩阵行列式对矩阵求导：

$$\begin{aligned}
\frac{\partial |\mathbf{X}|}{\partial \mathbf{X}} &= |\mathbf{X}|(\mathbf{X^{-1}})^\mathbf{T} && (19)\\
\frac{\partial \ln |\mathbf{X}|}{\partial \mathbf{X}} &= (\mathbf{X^{-1}})^\mathbf{T} && (20)
\end{aligned}$$

- 矩阵求导

$$\begin{aligned}
\frac{\partial \mathbf{a^T X b}}{\partial \mathbf{X}} &= \mathbf{ab^T} && (21)\\
\frac{\partial \mathbf{a^T X^T b}}{\partial \mathbf{X}} &= \mathbf{ba^T} && (22)\\
\partial \mathbf{X^T} &= (\partial \mathbf{X})^\mathbf{T} && (23)\\
\partial(\mathrm{Tr}(\mathbf{X})) &= \mathrm{Tr}(\partial \mathbf{X}) && (24)\\
& && (25)
\end{aligned}$$

2 Information Theory

2.1 Differential Entropy

2.1.1 常见 Differential Entropy

- *Uniform distribution(from 0 to a)*
$$h(X) = \log(a)$$
- *Normal Distribution*
$$X \sim N(0, \sigma^2)$$
$$h(X) = \frac{1}{2}\log 2\pi e\sigma^2$$
- *Multivariate Normal Distribution*
$$N_n \sim (\mu, K)$$
μ is mean and K is covariance matrix
$$h(X_1, \dots, X_n) = \frac{1}{2}\log(2\pi e)^n|K|$$

2.1.2 Properties

- $h(X + c) = h(X)$
- $h(aX) = h(X) + \log|a|$
- $h(AX) = h(X) + \log\big|\det(A)\big|$

3 Machine Learning

3.1 MAXIMUM LIKELIHOOD ESTIMATION (MLE)

3.1.1 GAUSSIAN MLE

$$\hat{\mu}_{ML} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\hat{\Sigma}_{ML} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu}_{ML})(x_i - \hat{\mu}_{ML})^T$$

$$\mathbb{E}(\hat{\mu}_{ML}) = \mu$$

$$\mathbb{E}(\hat{\Sigma}_{ML}) = \frac{m-1}{m}\Sigma$$

So the mean is unbiased and covariance is biased.

3.2 Linear Regression

3.2.1 Least Squares

Usually, for linear regression (and classification) we include an intercept term w_0 that doesn't interact with any element in the vector x. It will be convenient to attach a 1 to the first dimension of each vector x_i.

$$x_i = \begin{bmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{id} \end{bmatrix}, \qquad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & \dots & x_{1d} \\ 1 & x_{21} & \dots & x_{2d} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \dots & x_{nd} \end{bmatrix}$$

这种情况下，得到的解为:

$$w_{\text{ML}} = (X^T X)^{-1} X^T y$$

预测新的点为:

$$y_{\text{new}} = x_{\text{new}}^T w_{\text{ML}}$$

3.3 Classification

3.3.1 Bayes Classifier

The Bayes classifier has the smallest prediction error of all classifiers.
假设 (X, Y) 独立同分布，那么 optimal classifier is:

$$f(x) = \arg\max_{y\in\mathcal{Y}} P(Y = y|X = x) \tag{26}$$

Using Bayes rule and ignoring $P(X = x)$ we equivalently have:

$$f(x) = \arg\max_{y\in\mathcal{Y}} P(Y = y) \times P(X = x|Y = y) \tag{27}$$

其中 $P(Y = y)$ 叫做 class prior，$P(X = x|Y = y)$ 叫做 data likelihood given class.

3.3.2 THE PERCEPTRON ALGORITHM

- Suppose there is a linear classifier with zero training error:

$$y_i = \text{sign}(x_i^T w)$$

Then the data is linearly 'separable'

- By using the linear classifier $y = \text{sign}(x^T w)$ the Perceptron seeks to minimize

$$\mathcal{L} = -\sum_{i=1}^{n}(y_i \cdot x_i^T w)\mathbb{1}\{y_i \neq \text{sign}(x_i^T w)\}$$

Because $y \in \{-1, +1\}$,

$$y_i \cdot x_i^T w \quad \text{is} \quad \begin{cases} > 0 & y_i = x_i^T w \\ < 0 & y_i \neq x_i^T w \end{cases}$$

By minimizing $\mathcal{L}$ we're trying to always predict the correct label.

- $\nabla_w \mathcal{L} = 0$ 没有办法直接解出来，所以使用 gradient descent 的方法迭代求解 ($\mathcal{M}_t$ 表示在第 t 步被分错类的数据下标的集合)：

$$\nabla_w \mathcal{L} = \sum_{i \in \mathcal{M}_t} -y_i x_i \tag{28}$$

$$w' \leftarrow w - \eta \nabla_w \mathcal{L} \tag{29}$$

$$\mathcal{L}(w') < \mathcal{L}(w) \tag{30}$$

- Perceptron 算法的一些问题：
When the data is separable, there are an infinite number of hyperplanes.
This algorithm. doesn't take "quality"into consideration. It converges to the first one it finds.
When the data isn't separable, the algorithm doesn't converge. The hyperplane of w is always moving around.

3.3.3 LOGISTIC REGRESSION

- The distance of x to a hyperplane $x^T w + w_0$ is:

$$\left| \frac{x^T w}{||w||_2} + \frac{w_0}{||w||_2} \right|$$

- 鉴于 hyperplane 与 log odds 有很多相似的性质，我们可以直接用 hyperplane 来表示 log odds(注意，与前面 Bayes 得到的公式不同的是，前面是通过先验概率和 class prior 算出来的 w,w_0，这里我们对于这些先验概率没有约束 Discriminative classifier)：

$$\ln \frac{P(y = +1|x)}{P(y = -1|x)} = x^T w + w_0$$

$$P(y=+1|x) = \frac{\exp(x^T w + w_0)}{1+\exp(x^T w + w_0)} = \sigma(x^T w + w_0)$$

其中 σ 叫做 *sigmoid function*, in this case $x^T w + w_0$ is the *link function* for the log odds.

- 与 linear regression 类似，我们可以通过给 x 加入一维全为 1 的数据，将 w_0 并入到 w；

$$w \leftarrow \begin{bmatrix} w_0 \\ w \end{bmatrix}, \quad x \leftarrow \begin{bmatrix} 1 \\ x \end{bmatrix}$$

$$x^T w + w_0 \leftarrow x^T w$$

- Data likelihood
$y_1, \ldots, y_n$ 的联合分布就是把每个的概率乘起来 (默认 $P(Y=y_i|X=x_i)$ 独立同分布)，对联合分布取对数的到目标函数 $\mathcal{L}$，算法的目标是找到参数 w 使得目标函数值最大 (实际上就是 Maximum likelihood)

$$\mathcal{L} = \sum_{i=1}^{n} \ln \sigma_i(y_i \cdot w)$$

$$w_{ML} = \arg\max_{w} \mathcal{L}$$

- 与 Perceptron 算法类似，使用梯度进行极值的计算 (其中 $\sigma_i(y_i \cdot w)$ 表示 $P(Y=y_i|X=x_i)$)

$$w^{(t+1)} = w^t + \eta \nabla_w \mathcal{L}, \qquad \nabla_w \mathcal{L} = \sum_{i=1}^{n} \{1 - \sigma_i(y_i \cdot w)\} y_i x_i$$

有时为了避免“over fitting” 使用下述公式:

$$\mathcal{L} = \sum_{i=1}^{n} \ln \sigma_i(y_i \cdot w) - \lambda w^T w$$

- 上述算法叫做 *steepest ascent*, 还有一种 Newton 算法利用的是 $\mathcal{L}$ 的二阶导数, 参见 lecture_9 第 14 页

- Laplace approximation for logistic regression 参见 lecture_9 第 24 页

3.4 Kernels

3.4.1 Defination

A kernel $K(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a symmetric function defined as follows:
For any set of data $x_1, \ldots, x_n \in \mathbb{R}^d$, the $n \times n$ matrix K with $K_{i,j} = K(x_i, x_j)$ and there is a mapping $\phi : \mathbb{R}^d \to \mathbb{R}^D$ such that $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

3.4.2 Gaussian Kernel

Also called *radial basis function* (RBF),

$$K(x,x') = a\exp(-\frac{1}{b}||x-x'||^2)$$

3.4.3 产生新 Kernel

假设 K_1, K_2 为两个 kernel，可以利用现有的 kernel 组成新的 kernel K：

$$\begin{aligned} K(x,x') &= K_1(x,x') + K_2(x,x') && (31) \\ K(x,x') &= K_1(x,x')K_2(x,x') && (32) \\ K(x,x') &= \exp K_1(x,x') && (33) \end{aligned}$$

3.4.4 Kernelized Perceptron

- For Perceptron classfier,
  $$w = \sum_{i\in\mathcal{M}} y_i x_i$$
  where $\mathcal{M}$ is *sequentially constructed* set of misclassified examples.
  $$y_0 = \text{sign}(x_0^T w) = \text{sign}(\sum_{i\in\mathcal{M}} y_i x_0^T x_i)$$
- Using feature expansions, we can rewrite the function into the form below:
  $$y_0 = \text{sign}(\phi(x_0)^T w) = \text{sign}\big(\sum_{i\in\mathcal{M}} y_i \phi(x_0)^T \phi(x_i)\big)$$
- Use the kernel replace the $\phi(x_0)^T\phi(x_i)$
  $$y_0 = \text{sign}\big(\sum_{i\in\mathcal{M}} y_i K(x_0, x_i)\big)$$
- Learning the kernelized Perceptron
  和普通的 Perceptron 基本一样，只不过每次更新 $\mathcal{M}$ 是用 kernelized 的公式错判的点，具体可以看 lecture 10 第 15 页

3.4.5 Kernel k-NN

lecture 10 第 16 页

$$y_0 = \frac{1}{Z}\text{sign}\big(\sum_{i=1}^{n} y_i e^{-\frac{1}{b}||x_0-x_i||^2}\big)$$

其中 $Z = \sum_{i=1}^{n} e^{-\frac{1}{b}||x_0-x_i||^2}$

3.4.6 Gaussian Processes

具体参考 lecture 10 第 19 页以后

- Defination
  $f(x)$ is a Gaussian process and $y(x)$ is the noise-added process where $y = (y_1, \ldots, y_n)^T$ and K is $n \times n$ matrix with $K_{ij} = K(x_i, x_j)$:

  $$y|f \sim N(f, \sigma^2 I),\ f \sim N(0, K) \Leftrightarrow y \sim N(0, \sigma^2 I + K)$$

  可以这样理解： *$f(x)$ is the GP and $y(x)$ equals $f(x)$ plus i.i.d. noise*

- Predictions with Gaussian Processes
  Given measured data $\mathcal{D}_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, the distribution of $y(x)$ can be calculated at any *new* x to make predictions.
  $K(x, \mathcal{D}_n) = [K(x, x_1), \ldots, K(x, x_n)]$ and K_n is the $n \times n$ kernel matrix restricted points in $\mathcal{D}_n$

  $$\begin{aligned} y(x)|\mathcal{D}_n &\sim N(\mu(x), \Sigma(x)) && (34) \\ \mu(x) &= K(x, \mathcal{D}_n) K_n^{-1} y && (35) \\ \Sigma(x) &= \sigma^2 + K(x, x) - K(x, \mathcal{D}_n) K_n^{-1} K(x, \mathcal{D}_n)^T && (36) \end{aligned}$$

  For the posterior of $f(x)$ instead of $y(x)$, just remove σ^2