CS 6362 Machine Learning, Fall 2017: Homework 2

Jiani Li

Question 1:

(a) (1)

$$f(X=x;p)=\binom{n}{x}p^x(1-p)^{(n-x)}, x\in\{0,1,\dots,n\}$$

$$\begin{aligned}\log(f(X=x;p)) &= \log(\binom{n}{x}p^x(1-p)^{(n-x)})\\ &= \log(\binom{n}{x}) + \log(p^x) + \log((1-p)^{(n-x)})\\ &= \log(\binom{n}{x}) + x\log p + (n-x)\log(1-p)\end{aligned}$$

$$\begin{aligned}f(X=x;p) &= exp\{\log(\binom{n}{x}) + x\log p + (n-x)\log(1-p)\}\\ &= exp\{\log(\binom{n}{x}) + x\log p + (n-x)\log(1-p)\}\\ &= \binom{n}{x}exp\{x\log\frac{p}{1-p} + n\log(1-p)\}\end{aligned}$$

Here, $h(x)=\binom{n}{x}, \eta(p)=\log\frac{p}{1-p}, T(x)=x, A(p)=-n\log(1-p)$, and satisfies

$$f(X=x;p)=h(x)e^{\eta(p)T(x)-A(p)}$$

Therefore, binomial distribution belongs to the exponential family.

(2)

$$f(X=x;\lambda)=\frac{\lambda^x e^{-\lambda}}{x!}$$

$$\log(f(X=x;\lambda)) = x\log\lambda - \lambda - \log x!$$

$$\begin{aligned}f(X=x;\lambda) &= exp\{x\log\lambda - \lambda - \log x!\}\\ &= \frac{1}{x!}exp\{x\log\lambda - \lambda\}\}\end{aligned}$$

Here, $h(x)=\frac{1}{x!}, \eta(\lambda)=\log\lambda, T(x)=x, A(\lambda)=\lambda$, and satisfies

$$f(X=x;\lambda)=h(x)e^{\eta(\lambda)T(x)-A(\lambda)}$$

Therefore, Poisson distribution belongs to the exponential family.

(3)

$$\begin{aligned} f(X=x;\mu) &= \frac{1}{\sqrt{2\pi}\sigma}exp\{-\frac{(x-\mu)^2}{2\sigma^2}\} \\ &= \frac{1}{\sqrt{2\pi}\sigma}exp\{-\frac{x^2}{2\sigma^2}+\frac{\mu x}{\sigma^2}-\frac{\mu^2}{2\sigma^2}\} \\ &= \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{x^2}{2\sigma^2}}exp\{\frac{\mu}{\sigma^2}x-\frac{\mu^2}{2\sigma^2}\} \end{aligned}$$

Here, $h(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{x^2}{2\sigma^2}}, \eta(\mu)=\frac{\mu}{\sigma^2}, T(x)=x, A(\mu)=\frac{\mu^2}{2\sigma^2}$, and satisfies

$$f(X=x;\mu)=h(x)e^{\eta(\mu)T(x)-A(\mu)}$$

Therefore, Gaussian distribution belongs to the exponential family.

(b) Log-likelihood function for the Poisson regression:

$$\log(f(x;\lambda))=\sum_{i=1}^{n}x_i\log\lambda-n\lambda-\sum_{i=1}^{n}\log x_i!$$

(c) Although the log-odds-ratio transformation is non-linear, the model is linear with respect to the log-likelihood function (You can just sum up the log-likelihood function to get the output). And the it is linear because the relationship of features is linear, it is feature1 * parameter1 + feature2 * parameter2 + ... You do not do non-linear computation of features (e.g. product of features).

Question 2:

$$\begin{aligned} P(Y,X_1,X_2,\ldots,X_{d-1}) &= P(Y)P(X_1,X_2,\ldots,X_{d-1}|Y) \\ &= \sum_{X_d}P(Y)P(X_1,X_2,\ldots,X_d|Y) \\ &= \sum_{X_d}P(Y)P(X_1|Y)P(X_2|Y)\ldots P(X_d|Y) \\ &= P(Y)(\sum_{X_d}P(X_d|Y))P(X_1|Y)P(X_2|Y)\ldots P(X_{d-1}|Y) \\ &= P(Y)P(X_1|Y)P(X_2|Y)\ldots P(X_{d-1}|Y) \end{aligned}$$

Question 3: By increasing λ, the bias will be high and variance will be low. Consider the case when $\lambda\to\infty$, the probability of all features will be uniform. Thus, the bias will be high, but variance is low.

By decreasing λ, the bias will be low and variance will be high. Consider the case when $\lambda\to 0$, the bias is low, but the variance is high, meaning changes in training set will have a large impact on the decision.

Question 4:

(a) The gradient is obtained by:

for $j = 1 : n$

$$\{\theta_j := \theta_j - \sum_{i=1}^{k}(y^{(i) - h_\theta(x^{(i)})})x_j^{(i)}\}$$

There is n features, and for each feature, the complexity is k. Therefore, the computational complexity of computing gradient at each iteration is: kn.

(b) Advantages: Increasing k will improve the convergence rate. It will decreases the times of iteration. And it will get the global optimal solution.

Disadvantages: Increasing k will enlarge the computational complexity of computing gradient at each iteration. Each iteration will be slower.

The trade-off by increasing/decreasing k: Increasing k decreases the times of iteration but increase the time of each iteration. Increasing k can get the global optimal solution but it is slower. Decreasing k will be faster but it will get the local optimum solution.

(c) Perceptron Algorithm uses stochastic gradient descent. The loss function of Perceptron algorithm is:

$$J(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}\max(0, y_i\boldsymbol{w}\boldsymbol{x}_i)$$

$$J_i(\boldsymbol{w}) = \max(0, y_i\boldsymbol{w}\boldsymbol{x}_i)$$

$$\frac{\partial J_i}{\partial \boldsymbol{w}} = \begin{cases} 0, \quad \text{if} \quad y_i\boldsymbol{w}\boldsymbol{x}_i \geq 0 \\ y_i\boldsymbol{x_i}, \quad \text{otherwise} \end{cases}$$

The update

$$\boldsymbol{w'} = \boldsymbol{w} + \eta\frac{\partial J_i}{\partial \boldsymbol{w}}$$

Therefore, it uses gradient descent.