CS 6362 Machine Learning, Fall 2017: Homework 2

Venkataramana Nagarajan

Question 1:

(a) (1)

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

$$\binom{n}{x} e^{xlnp + (n-x)ln(1-p)}$$

$$\binom{n}{x} e^{x[lnp - ln(1-p)] + nln(1-p)}$$

$$\binom{n}{x} e^{xln\frac{p}{1-p} + nln(1-p)}$$

$$h(x) = \binom{n}{x}, \eta(\theta) = ln\frac{p}{1-p}, T(x) = x$$

$$A(\theta) = -nln(1-p)$$

(2)

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

$$\frac{1}{x!} e^{xln\lambda - \lambda}$$

$$h(x) = 1/x!, \eta(\theta) = ln\lambda, T(x) = x, A(\theta) = \lambda$$

(3)

$$f(X = x; \mu) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(X-\mu)^2}{2\sigma^2}}$$

$$= \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2} - \frac{\mu^2}{2\sigma^2} + \frac{2x\mu}{2\sigma^2} - \frac{log\sigma}{2}}$$

$$= \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} e^{(x)(\frac{\mu}{\sigma^2}) - \frac{\mu^2}{\sigma^2} + \frac{log\sigma}{2}}$$

$$h(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

$$T(x) = x, \eta(\theta) = \frac{\mu}{\sigma^2}, A(\theta) = \frac{\mu^2}{2\sigma^2} + \frac{log\sigma}{2}$$

(b) From 1(a)(2) we know that $\eta(\theta) = \log \lambda$

$$f(Y = y; \lambda) = \frac{\lambda^y e^{-\lambda}}{y!}$$

Let

$$X_i = X_{1i}, X_{2i}, ...X_{ni}$$

Mean λ_i is

$$Expectation[y_i|X_1, X_2, ...X_n] = e^{\alpha_0 + \alpha_1 x_1 + ... + \alpha_n x_n}$$

$$log\lambda_i = \alpha_0 + \alpha_1 x_1 + ... + \alpha_n x_n$$

Likelihood function will be

$$L(y, W, \lambda) = \prod_{i=1}^{n} \frac{e^{\lambda_i} \lambda_i^{y_i}}{y_i!}$$

Log likelihood would be

$$l(y, w, \lambda) = \sum_{i=1}^{n} [-\lambda_i + y_1 log(\lambda_i) - log(y_i!)]$$

(c) GLMs have non-linear transformation and yet are called linear because the final predictions are always a sum of product of the input values and their associated weights. The way weights are calculated is where the method can become non-linear. For example if you consider a 2-d plane then the weights that are required would be the slope and the intercept but as the number of weights increase the dimensions increase which makes in non-linear to calculate the actual value of weights.

Question 2: According to conditional probability
P(Y,Xd1,Xd2,....,Xd) = P(X1,X2,...,Xd—Y)P(Y)
as all the features are iid.

P(Y,X1,X2,....,Xd) = P(X1—Y)P(X2—Y)...P(X(d-1)—Y)P(Xd—Y)P(Y)
As we know that Xd is unknown we have to iterate and sum over all possible values of Xd.

$$\sum_{X_d} P(X_1|Y)P(X_2|Y)...P(X_d|Y)P(Y)$$

$$P(X_1|Y)P(X_2|Y)...\sum_{X_d} P(X_d|Y)P(Y)$$

The total sum of prob will be 1.

$$= P(X_1|Y)P(X_2|Y)...P(Y)$$

$$P(Y, X_1, X_2, ..., X_{d-1})$$

Question 3: By adding λ smoothing we try to weigh in features which are not present in the training set by adding λ to their count. This makes sure the even unseen features have some probability during test time. As given in the equation we can see that this λ is added both in the numerator and the denominator. So we can say that a high λ will lead to a high bias and small variance because when λ is sufficiently large we will get a large constant bias. Also vice-versa when λ is small we get small bias and large variance.

Question 4:

(a) The computational complexity of computing the gradient at each iteration would be O(kn)

(b) As we increase k, the computation in each iteration increases but we converge faster to the local minimum i.e the path that we take to get to the local minimum becomes more and more straighter instead of being zigzag. As k decreases , the computation in each iteration becomes lighter but we converge slower to the local minimum i.e the path we take to get to the local minimum becomes zigzag and longer.

(c) An example of an algorithm that uses stochastic gradient would be a Perceptron Classifier. We see that in most classifiers we process the whole batch of data first and then try to minimize the cost function based on the whole training set. But in Perceptron classifier it updates the weight incrementally after it encounters each individual training sample. This is nothing but

stochastic gradient descent. We use it so as to converge faster because we update weights after each training sample.