STA414/2104: Practice Problems 2

January, 2018

These practice problems are not for credit. Students may complete independently, in groups, or however they like. Treat these as representative of what might be on the tests.

Questions involving coding may be completed in any language. If you would like help regarding your language of choice from the course team, use R or Python. There will not be any *code*-related questions on the tests, but you will be asked about comuptational algorithms.

1. *Gradient Descent*: Prove the statement that I made in lecture: let $\mathbf{x} \in \mathbb{R}^p$. Then $\mathbf{x}^T\mathbf{y}$ is maximized over vectors $\mathbf{y} \in \mathbb{R}^p$ when $\mathbf{y} = a\mathbf{x}$ for some $a \in \mathbb{R}$.

2. *Gradient Descent*: Consider the code example discussed in class. The prediction function is

$$y(\mathbf{w}, x) = \frac{w_1 + w_2 \sin(2\pi x)}{1 + w_3 x}$$

   and the loss function is

$$L(\mathbf{w}|\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} \big(y(\mathbf{w}, x_i) - t_i\big)^2$$

   where $\mathbf{t}$ represents the vector of observed targets, and $\mathbf{x}$ represents the vector of observed features.

   (a) Derive the gradient and Hessian of $L(\mathbf{w}|\mathbf{x})$, with respect to $\mathbf{w}$.

   (b) Implement them and re-run the example, playing around with the step size and starting values. Do you see how much work it took to get get Newton's method to converge to something sensible?

   (c) Modify the code to give you *Stochastic Gradient Descent*. Try this with different mini-batch sizes and starting values, to get a feel for how it works- particularly the stability of the algorithm with respect to these *hyper-parameters*.

3. *Automatic Differentiation*: Do the homework from the lecture slides. Then, let $\mathbf{x} = (x_1, x_2, x_3)$, $f(\mathbf{x}) = 2x_2 + (x_3 x_1)^2$.

   (a) Derive the gradient and Hessian of f.

   (b) Code these up in Python (or `R`, but we didn't cover using the `madness` package so it's probably easier to do it in Python), and evaluate them for $\mathbf{x} = (1, 2, 3)$.

   (c) Use `autograd` to get the gradient and Hessian of f, and verify that you get the same answer.

4. *Maximum Likelihood*: Let $X_i \sim N(\mu, \sigma)$, the *univariate* normal distribution. Find the maximum likelihood estimator for (μ, σ).

5. *Exponential Families*: Consider the exponential family form as given in the last few lecture slides of lecture 2:

$$L(\eta|\mathbf{X}) = \left(\prod_{i=1}^{n} h(\mathbf{x})\right) g(\eta)^n \exp\left(\eta^T \sum_{i=1}^{n} u(\mathbf{x}_i)\right)$$

   (a) Write down the log-likelihood, which is just the log of the above expression.

   (b) Show that the univariate Gaussian distribution can be written in this form, and give explicit expressions for η, $u(x)$, $h(x)$ and $g(\eta)$ in terms of x, μ, σ. This question is done for you in the slides, but do it again yourself.

   (c) Recall for the univariate Gaussian that $E(x) = \mu$ and $E(x^2) = \sigma^2 + \mu^2$. Verify this is true using the formula for the exponential family,

$$-\nabla \log g(\eta) = E(u(x))$$

   Note how much easier that was than integrating the density function to find the moments.

   (d) Now, repeat the above for the *multivariate* Gaussian: find the exponential family form, giving explicit expressions for the η, $u(x)$, $h(x)$ and $g(\eta)$ in terms of $\mathbf{x}, \boldsymbol{\mu}, \Sigma$.

   (e) Find $E(\mathbf{x})$ and $E(\mathbf{x}\mathbf{x}^T)$. Again, use the exponential family identity from (c). After, write down the equivalent integral that you just solved, and pat yourself on the back.

   (f) Find the maximum likelihood estimators for η, and then use that to find the maximum likelihood estimators for $\boldsymbol{\mu}$ and Σ. Use the identity at the very end of the lecture slides,

$$-\nabla \log g(\hat{\eta}) = \frac{1}{n} \sum_{i=1}^{n} u(\mathbf{x}_i)$$

   to make your life way easier.

The fact that maximum likelihood estimation for the exponential family works out so nicely is what lets us build *generalized linear models*, i.e. regression models for any type of exponential family response (continuous, binary, count, etc).