STA414/2104: Practice Problems 1

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.

Questions involving material not directly covered in lecture (mostly the linear algreba questions) *may* be labelled as "optional". *Optional* means that a question of that nature won't appear on the tests. I put these questions on the assignment because I think they are important, but I recognize that we have a very wide range of backgrounds in this course, and it's not feasible to review everything.

1. *Covariance.* Let $\mathbf{x} = (X_1, \ldots, X_j)$ be a vector-valued random variable taking values in the vector space $\mathbb{R}^p$. Define the *covariance matrix* of $\mathbf{x}$ to be $\Sigma = E((\mathbf{x} - E(\mathbf{x})(\mathbf{x} - E(\mathbf{x}))^T)$, which satisfies $\Sigma_{ij} = Cov(X_i, X_j)$. E here means *expectation*, not *error*.
   (a) Show $Var(\mathbf{a}^T\mathbf{x}) = \mathbf{a}^T\Sigma\mathbf{a}$ for any fixed $\mathbf{a} \in \mathbb{R}^p$.
   (b) Prove that Σ is positive definite. You may assume that $\mathbf{x}$ is what we call a *regular* random variable, which just means that every element of $\mathbf{x}$ has variance that is strictly positive. This question is a one-liner if you use the *definition* of positive definite-ness, so really what I'm asking you to do is look up the definition of "positive-definite" and clearly state why a covariance matrix should be so, given part (a).

2. *Review of vector calculus*: This quesiton reviews basic results in vector calculus (basic doesn't mean easy, just "not complicated"). Note that if you're looking this stuff up on the internet, some sources write the gradient as a *row* vector, where others write it as a *column* vector. Here I use the *column* vector notation, but it really doesn't matter as long as you are clear in your answer.

   (a) The *gradient* ∇f of a multivariable function $f(\mathbf{x}) : \mathbb{R}^m \to \mathbb{R}$ is the $m-$dimensional vector whose i-th element equals the derivative of f with respect to $\mathbf{x}_i$:

   $$\nabla f = \left(\frac{\partial f}{\partial \mathbf{x}_1}, \ldots, \frac{\partial f}{\partial \mathbf{x}_m} \right)^T$$

   You can compute it element-wise. Show that for $f(x, y) = x^2 + y^2$, $\nabla f = (2x, 2y)^T$.

   (b) We don't want to do it element-wise every time. Show that

   (i) $\nabla x_1 = (1, 0, \ldots, 0)^T$

   (ii) $\nabla \mathbf{x}^T \mathbf{x} = 2\mathbf{x}$

   (iii) $\nabla \mathbf{a}^T \mathbf{x} = \mathbf{a}$

   (iv) $\nabla \mathbf{x}^T \mathbf{A} \mathbf{x} = 2\mathbf{A}\mathbf{x}$ for $\mathbf{A}$ symmetric and not depending on $\mathbf{x}$.

   where all differentiation is with respect to $\mathbf{x}$.

3. *Ridge Regression.* The ridge regression model in lecture 1 is an example of regularlization. For a dataset $(\mathbf{t}, \mathbf{X})$ consisting of targets $\mathbf{t} = (t_1, \ldots, t_N)$ and features $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)'$, we wish to build a model $y(\mathbf{x}_n, \mathbf{w})$ by minimizing

$$L(\mathbf{w}) = \frac{1}{2} (\mathbf{y} - \mathbf{t})' (\mathbf{y} - \mathbf{t}) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

with respect to $\mathbf{w}$, where $\mathbf{y} = \mathbf{X}\mathbf{w}$ is the vector of predicted targets and $\mathbf{X}$ contains a columnn of ones.

(a) Show that when $\lambda = 0$ (no regularization) the solution to the above is the *least squares* weights defined in lecture (slide above the ridge regression slide). Use the results about gradients from the previous question to take the gradient of $L(\mathbf{w})$, set it to zero, and solve.

(b) Show that the solution for $\lambda \in [0, \infty)$ is given by

$$\hat{\mathbf{w}} = \left(\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}'\mathbf{t}$$

where $\mathbf{I}$ is the identity matrix of appropriate size.

(c) *Optional*: Show that if $\mathbf{X}$ does not have full column rank, then the unique un-regularlized least squares solution does not exist.

(d) *Optional*: Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ have rank $q < p$. Show that the ridge regression solution you derived above *does* exist, i.e. show $rank\left(\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}\right) = p$.

4. *Ridge Regression: coding it up*

   (a) Implement the ridge regression problem above. Consider λ a fixed parameter. Assume all the features $\mathbf{X}$ are continuous.

   (b) Generate examples of the polynomial curve dataset; you can use the example `R` code below. Your curve may look different than mine because the values are randomly generated.

   (c) To fit your model with different values of M, literally just add columns to the design matrix that equal $x^2, x^3, \ldots$ etc. For fixed $\lambda = 0.1$, play around with a few values of M, and see if you can get an intuition for what the best value might be.

   (d) Pick $M = 14$ (assuming you used $n = 15$ like I did in the example code). Fit your model with $\lambda = 0$. What happens?

   (e) Fit your model for a few nonzero values of λ. You can start with $\lambda = 1$ and use a sequence $\lambda = 0.1, 0.01, 0.001, \ldots$ etc. What do you think the best value of λ is?

   (f) Compare your results to what you get using the `glmnet` package in `R`, using the below code. This is just example code and is not complete. The `glmnet` package chooses the grid of λ values using a very intelligent and efficient path algorithm that we might learn about later in the course. It then chooses the best λ via cross validation.

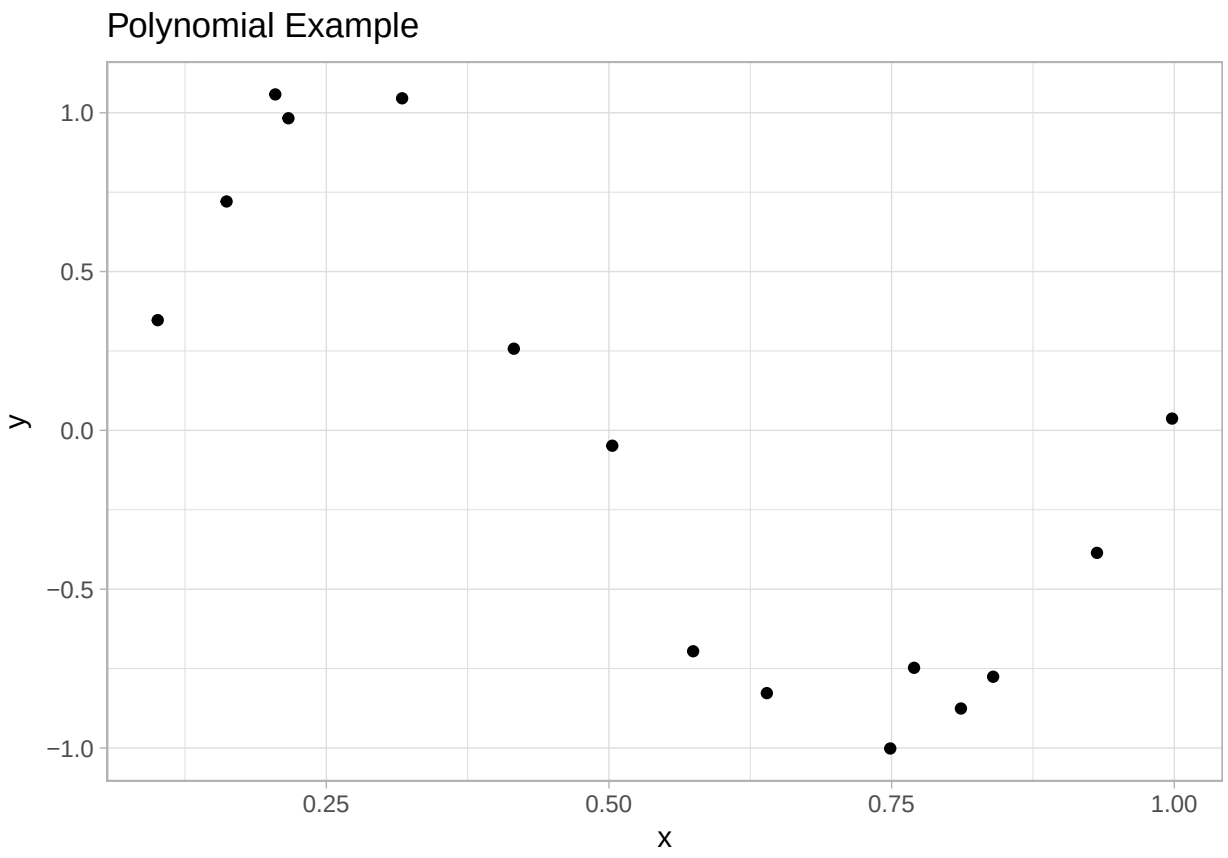
Example polynomial curve generation:

```
suppressWarnings({
  suppressMessages({
    library(dplyr)
    library(ggplot2)
  })
})
generate_polynomial <- function(n) {
  true_function <- function(x) sin(2*pi*x)
  x <- runif(n,0,1)

  data_frame(x = x,y = true_function(x) + rnorm(n,sd=0.2))
}

# Plot
generate_polynomial(15) %>%
  ggplot(aes(x=x,y=y)) +
  theme_light() +
  geom_point() +
```

```
  labs(title="Polynomial Example",
       x="x",
       y="y")
```



Example `glmnet` code:

```
# install.packages("glmnet")
dat <- generate_polynomial(15)
dat$bias <- rep(1,nrow(dat))
dat$x2 <- dat$x^2
dat$x3 <- dat$x^3
# ...and so on. Try and find a more elegant way of creating the 15 columns you need.

# glmnet takes matrices/vectors, not dataframes
glmnet_x <- as.matrix(dplyr::select(dat,-y))

mod <- glmnet::cv.glmnet(x=glmnet_x,
                         y=dat$y,
                         nfolds=3,
```

```
                        alpha=0) # Choosing alpha = 0 gives ridge regression


mod$lambda.min
```

```
## [1] 0.07881102
```

5. *Cross-validation.* Implement 3-fold cross-validation as described in lecture for your ridge regression model above, to pick λ. This means for a pre-specified grid of values for λ:
   (a) Randomly split the data into a number of subsets containing $N - 3$ observations
   (b) Fit the model to that subset
   (c) Report the squared prediction error on the 3 held-out observations
   (d) Repeat for each value of λ, and average the prediction errors for each
   (e) Pick the λ with the lowest cross-validated prediction error. Compare your results to those given by the `cv.glmnet` function in `R` above.

   Note: you might find that the values of λ picked by your procedure and by `cv.glmnet` are different. Run `cv.glmnet` a few times- do the results differ between runs? This might happen because the dataset is small, so try increasing n to 150, 1500, etc. and see if that helps.

6. *Elements of Statistical Learning, pg. 18.* Let the input variable X and output variable Y have joint distribution $P(X, Y)$, and suppose we seek a function $f(X)$ that approximates Y. In Elements of Statistical Learning, the authors suggest to use

$$f(x) = argmin_c E_{Y|X}((Y - c)^2 | X = x)$$

the conditional expected squared error in approximating Y with c given $X = x$.

(a) Prove their statement that this is solved by

$$f(x) = E(Y|X = x)$$

(b) Suppose we instead want to minimize the expected *absolute* deviation, $E_{Y|X}(|Y - c| \, | X = x)$. Show that the solution is given by the conditional median,

$$f(x) = Med(Y|X = x)$$

which satisfies $P(Y < f(x)|X = x) = P(Y > f(x)|X = x) = 1/2$.