Notes

STA314H1 - Fall 2020

Ziyue Yang

November 28, 2020

Contents

1 Week 5

1.1 Norms

1.1.1 Induced Norm of a Matrix

Question 1.1. If we have a vector x and a matrix A, how big is Ax compared to x?

Suppose that we find a constant $C_{A,q}$ such that

$$\|Ax\|_q \leq C_{A,q}\|x\|_q, \tag{1.1}$$

then the **smallest such constant** would be a good indication of how much multiplying by A changes the length of a vector.

- This lets us define the **induced norm** of a matrix

Definition 1.1 (The Induced Norm of a Matrix)**.** The induced norm of a matrix A is defined as

$$\|A\|_q = \sup_x \frac{\|Ax\|_q}{\|x\|_q}. \tag{1.2}$$

- NB: This works when A is rectangular.

1.1.2 2-Norm of a Matrix

We hold a special place for 2-norms, as well as the induced 2-norm of a matrix.

Definition 1.2 (The Induced 2-Norm of a Matrix)**.**

$$\|A\|_2^2 = \sup_x \frac{\|Ax\|_2^2}{\|x\|_2^2} = \sup_x \frac{x^T A^T Ax}{x^T x}. \tag{1.3}$$

Note that this is the **largest eigenvalue of** $A^T A$. Hence the 2-norm of a matrix A is the square root of the largest eigenvalue of $A^T A$.

1.1.3 Orthogonal Invariance

It turns out that both the matrix and vector 2-norms are invariant to multiplication by an orthogonal matrix. Note that this is NOT true for other norms.

Example 1.1. Suppose U is a $p \times p$ orthogonal matrix, then

$$\|Ux\|_2^2 = x^T Y^T Yx = x^T x = \|x\|_2^2. \tag{1.4}$$

Suppose A is an $n \times p$ matrix, V is an $n \times n$ orthogonal matrix. Then

$$\|U^T AVx\|_2^2 = \|U^T A\tilde{x}\|_2^2 = \tilde{x}^T A^T UU^T A\tilde{x} = \tilde{x}^T A^T A\tilde{x}, \tag{1.5}$$

where $\tilde{x} = Vx, \|\tilde{x}\|_2^2 = \|x\|_2$ and so

$$\|U^T AV\|_2 = \|A\|_2. \tag{1.6}$$

1.2 The Singular Value Decomposition (of Non-Symmetic and Rectangular Matrices)

Remark 1.1. We know that $\|X\|_2$ is the square root of the largest eigenvalue of X^TX, hence at some point , whenever we see one of the inner product matrices, we should recall the **PCA**.

The SVD is a good way to understand exactly how PCA is working in terms of the feature matrix X.

Theorem 1.1 (The Singular Value Decomposition)**.** Assume that $p < n$.

Let X be an $n \times p$ matrix. Then there exists an orthogonal $p \times p$ matrix V (i.e. $V^TV = VV^T = I$) and an orthogonal $n \times n$ matrix U such that

$$U^TXV = D, \tag{1.7}$$

where $D = \text{diag}(\sigma_1, \ldots, \sigma_p)$, and $\sigma_1 \geq \cdots \geq \sigma_p \geq 0$.

Proof. Omitted for now. □

1.2.1 The SVD and Principal Components

Remark 1.2. So what good is an SVD?

The SVD is like the *eigendecomposition* of a symmetic matrix, except it is defined for **all** matrices.

We can express the SVD of an $n \times p$ matrix X in several equivalent ways:

1. As a singular tuple (σ, u, v) that satisfies $Xv = \sigma u$ and $X^Tu = \sigma v$.
2. As a matrix decomposition $X = UDV^T$, where V is a $p \times p$ orthogonal matrix, and U is a $n \times n$ orthogonal matrix.
3. As a way of representing the matrix as a sum

$$X = \sum_{j=1}^{p} \sigma_j u_j v_j^T. \tag{1.8}$$

Remark 1.3 (The SVD and Principal Components)**.** Recall that the factor loadings are the eigenvectors of X^TX.

If $X = UDV^T$, then $X^TX = V^TDU^TUDV^T = VD^2V^T$.

- The V in the SVD is exactly the matrix of factor loadings.
- The eigenvalues of X^TX are the squares of the singular values.

Note that the score vectors were defined as $t_j = Xv_j$, and We can use one of the representations of singular vectors to see that

$$t_j = Xv_j = \sigma_j u_j. \tag{1.9}$$

1.2.2 SVD and Principlal Component Regression

Remark 1.4. The SVD makes it easy to solve the normal equations.

Recall that

$$\hat{\beta} = (X^T X)^{-1} X^T y \tag{1.10}$$
$$= VD^{-2}V^T VDU^T y \tag{1.11}$$
$$= VD^{-1}U^T y \tag{1.12}$$
$$= \sum_{j=1}^{p} \frac{u_j^T y}{\sigma_j} v_j. \tag{1.13}$$

PCR just snipes off the small eigenvectors:

$$\hat{\beta}_{\text{pcr}} = V_k D_k^{-2} V_k^T VDU^T = \sum_{j=1}^{k} \frac{u_j^T y}{\sigma_j} v_j. \tag{1.14}$$

1.3 Ridge Regression

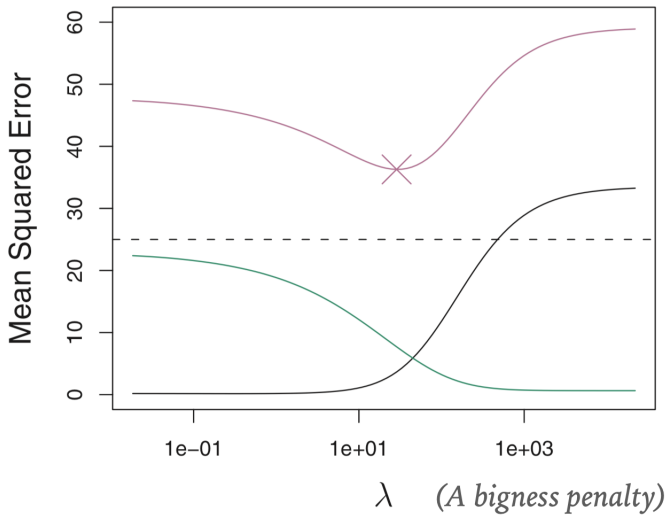
1.3.1 Overfitting Discussion

As we add more features to our regression, our training error will decrease, as our test error will go down for a while, then will increase again. A sign that something has gone wrong is that we have wildly varying regression coefficients, reasons include

- (Almost) co-linearity of features;
- Big feature weights are needed to fit data perfectly,

which can lead to out-of-sample *bias*.

What if we force the coefficients to be small?



The idea is: *Don't let things get to big.*

1.3.2 Ridge Regression Objective Function

Instead of solving our standard least-squares problem, we should use the **ridge regression** objective function.

Definition 1.3 (Ridge Regression Objective Function)**.**

$$\min_{\beta_0,\beta} \sum_{i=1}^{n} (y_i - \beta_0 - X\beta)^2 + \lambda \|\beta\|_2^2. \tag{1.15}$$

This means that we need to **balance** the goodness of fit with the requirements that the β_j aren'r too big. The parameter λ controls the balance.

If $\lambda = 0$, we recover LS;

If $\lambda \to \infty$, then all the coefficeints are zero.

Side note about the intercept...

If we want to include an intercept term, it's importatnt to make sure the features are **centred**, in which we can do quick maths to show that the first row of the normal equation is

$$1^T \begin{pmatrix} 1 & X \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta \end{pmatrix} = 1^T y. \tag{1.16}$$

If X is centred (such that $1^T X = 0$), then it follows that

$$\beta_0 = \frac{1}{n} \sum_{i=1}^{n} y_i \tag{1.17}$$

Points to y_i: don't shirnk.

What does the solution look like?

Assuming we cetre the predictors, we already know what the value of β_0 is. Now assume that the data has the mean of zero. Notice that the objective function (1.15) is *smooth* and quadratic, so we can minimize it as before.

The components of the gradient are

$$\frac{\partial}{\partial \beta_k} \left[\sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right] = 2 \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} \beta_j x_{ij} \right) x_{ik} + 2\lambda \beta_k \tag{1.18}$$

$$= 2X^T y - 2X^T X\beta - \lambda\beta \tag{1.19}$$

$$= 0 \qquad \text{when } (X^T X + \lambda I)\beta = X^T y. \tag{1.20}$$

1.3.3 Ridge Regression Estimates

By performing a **singular value decomposition** of X, we obtain

$$X = UDV^T, \tag{1.21}$$

here

U and V have orthonormal columns,

D is diagonal

This is related to the eigendecomposition by

$$X^T X = VDU^T UDV^T = VD^2V^T, \tag{1.22}$$

plugging which into the ridge regression equations (?ref here), we get

$$(X^T X + \lambda I)\beta = X^T y \tag{1.23}$$
$$V(D^2 + \lambda I)V^T \beta = VDU^T y \tag{1.24}$$
$$V^T \beta = (D^2 + \lambda I)^{-1} DU^T y \tag{1.25}$$
$$\beta = V(D^2 + \lambda I)^{-1} DU^T y. \tag{1.26}$$

What does the fit look like?

Comparison of β_{ridge} and β_{LS}:

$$\beta_{bridge} = \sum_{j=1}^{p} \frac{\sigma_j}{\sigma_j^2 + \lambda} v_j u_j^T y \tag{1.27}$$

$$\beta_{LS} = \sum_{j=1}^{p} \frac{1}{\sigma_j} v_j u_j^T y \tag{1.28}$$

Hence the ridge regression shrinks each coefficient by a factor of

$$\frac{\sigma_i^2}{\sigma_i^2 + \lambda}. \tag{1.29}$$

> Note that the shrinkage is NOT uniform.
>
> Refers to *Week 5, Part 3: Ridge Regression.*

2 Week 6 (Quiz 1)

3 Week 7

3.1 Introduction to Lasso

Remark 3.1. Ridge regression stabilizes the least-squares estimates by shrinking low-variance directions, which makes it like a *softer* version of **principal component regression**.

> Can we use penalized regression to make a softer version of variable selection? Yes. But we need to use a different penalty.

OMITTED TO SAVE TIME

3.2 Coordinate Descent

Algorithm 3.1 (Coordinate Descent)**.** For the Lasso,

1. Start with $\beta = \beta_0$
2. Repeat until convergence: for each $j = 1 \dots p$,

Update the j^{th} coordinate of

$$\beta_j = \text{sign}(X_{:j}^T r_j) \left(|X_j^T r_j| - \lambda \right)_+ \tag{3.1}$$

where

$$r_j = y - \sum_{\substack{i=1 \\ k \neq j}}^{p} \beta_k X_{:k} \tag{3.2}$$

4 Week 8/9

4.1 Logistic Regression

4.1.1 Classification Boundaries

From the Bayes' classifier we know that the best decision boundary is $Pr(y = 1 \mid x) = 0.5$.

Taking logs, we observe that

$$\log(Pr(y = 1 \mid x)) = -\log 2 \tag{4.1}$$
$$\log(1 - Pr(y = 1 \mid x)) = -\log 2 \tag{4.2}$$

implying that the decision boundary is

$$\implies \log\left(\frac{Pr(y = 1 \mid x)}{1 - Pr(y = 1 \mid x)}\right) = \beta_0 + x^T\beta = 0. \tag{4.3}$$

- This is a linear boundary.

4.1.2 Separation and Remedies

Separation *occasionally/almost* happens with real data, especially when there are a lot of predictors.

An option to prevent this is to add a penalty, which prevents us from being able to increase the value of β forever.

L1-penalized logistic regression/logistic lasso minimizes

$$-\frac{1}{n}\sum_{i=1}^{n}(y_i \log(p(\beta, x_i)) + (1 - y)\log(1 - p(\beta, x_i))) + \lambda\|\beta\|_1 \tag{4.4}$$

L2-penalized logistic regression minimizes

$$-\frac{1}{n}\sum_{i=1}^{n}(y_i \log(p(\beta, x_i)) + (1 - y)\log(1 - p(\beta, x_i))) + \lambda\|\beta\|_2^2 \tag{4.5}$$

4.2 Fitting Logistic Regression

Recall that in logistic regression, we estimate our parameters by maximizing the log-likelihood

$$L(\beta) = \sum_{i=1}^{n} (y_i \log(p(x_i)) + (1 - y_i) \log(1 - p(x_i))) \tag{4.6}$$

$$= \sum_{i=1}^{n} \left[y_i(\beta_0 + x_i^T \beta) - \log\left(1 + \exp(\beta_0 + x_i^T \beta)\right) \right] \tag{4.7}$$

In fact, this minimizes the *negative log-likelihood*.

Unlike linear regression, there is NO closed form for (β_0, β), which makes it challenging to find the solution.

How to find the solution?

Notice that the log likelihood for logistic regression is convex, hence it has unique global minimum.

4.2.1 Newton's Method

Theorem 4.1 (Newton's Method)**.** Suppose we want to minimize $f(x)$. Starting at point x_0, we know that by Taylor's Theorem, for any x near x_0,

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(x_0)(x - x_0)^2 \tag{4.8}$$

$$= f(x_0) - x_0 f'(x_0) + \frac{1}{2} x_0^2 + x(f'(x_0) - x_0 f''(x_0)) + \frac{1}{2} f''(x_0) x^2 \tag{4.9}$$

This *quadratic approximation* has an **explicit minimum**:

$$\frac{x_0 f''(x_0) - f'(x_0)}{f''(x_0)} = x_0 - \frac{f'(x_0)}{f''(x_0)} \tag{4.10}$$

Theorem 4.2 (Newton's Method in 1D)**.** Newton's method continues by using the minimum of the quadratic approximation as the new starting point, forming a new approximation, and moving to the minimum.

The updates are

$$x_{m+1} = x_m - \frac{f'(x_m)}{f''(x_m)}, \tag{4.11}$$

which will converge if the starting point is *close enough* to the true minimum.

There are procedures that can be useful to improve convergence, but we will omit it for now.

Theorem 4.3 (Newton's Method in Higher Dimensions)**.** We need to build a *quadratic approximation* to the multivariate function $f(x)$.

To update the vector x, we use the gradient ∇f and the *Hessian*, a matrix H of all possible second order derivatives, where:

$$H_{ij}[f](x) = \frac{\partial^2 f}{\partial x_i \partial x_j}(x). \tag{4.12}$$

The quadratic approximation around x_k now looks like:

$$f(x) \approx f(x_m) + (x - x_m)^T \nabla f(x_m) + \frac{1}{2}(x - x_m)^T H[f](x_m)(x - x_m) \tag{4.13}$$

We can find the minimum of the quadratic approximation using the *same techniques* (reference?) as we used to minimize the least-squares risk to obtain

$$x_{m+1} = x_m - H[f](x_m)^{-1} \nabla f(x_m)??? \tag{4.14}$$

Now we can apply all these to logistic regression. We will need to compute derivatives of

$$L(\beta_0, \beta) = \sum_{i=1}^{n} \left[y_i(\beta_0 + x_i^T \beta) - \log(1 + \exp(\beta_0 + x_i^T \beta)) \right] \tag{4.15}$$

But it gets complicated if we compute all of the first and second derivatives. To makes things seasier, we *suck* the intercept into β, and set $x_{i0} = 1$ and write the likelihood as

$$L(\beta) = \sum_{i=1}^{n} \left[y_i x_i^T \beta - \log(1 + \exp(x_i^T \beta)) \right] \tag{4.16}$$

$$\implies \frac{\partial L}{\partial \beta_j} = \sum_{i=1}^{n} \left(y_i x_{ij} - \frac{x_{ij} \exp(x_i^T \beta)}{1 + \exp(x_i^T \beta)} \right) \tag{4.17}$$

So the linear part of the quadratic expansion around $\beta^{(m)}$ by multiplying each of these with β_j and adding to get

$$\sum_{i=1}^{n} \left(y_i x_i^T \beta - \frac{x_i^T \exp(x_i^T \beta^{(m)})}{1 + \exp(x_i^T \beta^{(m)})} \right) + \text{constant}(\beta^{(m)}) \tag{4.18}$$

Differentiating (4.17) again we obtain

$$\frac{\partial^2 L}{\partial \beta_k \partial \beta_l} = -\sum_{i=1}^{n} \frac{x_{ik} x_{il} \exp(x_i^T \beta)}{(1 + \exp(x_i^T \beta))^2}, \tag{4.19}$$

then for $b = \beta$ or $b = \beta^{(i)}$,

$$\beta^T H[L](\beta^{(i)}) b = -\sum_{i=1}^{n} \beta^T x_i x_i^T b \frac{\exp(\beta_0^{(i)} + x_i^T \beta^{(i)})}{(1 + \exp(\beta_0^{(i)}) + x_i^T \beta^{(i)})^2} \tag{4.20}$$

To deal with the logistic transformation, we have

$$\frac{p(x_i)}{1-p(x_i)} = \exp(x_i^T\beta) \tag{4.21}$$

implying

$$p(x_i) = \frac{\exp(x_i^T)}{1+\exp(x_i^T\beta)} \tag{4.22}$$

$$1-p(x_i) = \frac{1}{1+\exp(x_i^T\beta)} \tag{4.23}$$

Hence we can write

$$\frac{\exp(x_i^T\beta^{(m)})}{(1+\exp(x_i^T\beta^{(m)}))^2} = p^{(m)}(x_i)(1-p^{(m)}(x_i)) \tag{4.24}$$

Finally, we can make our **quadratic approximation**:

$$L(\beta) \approx \sum_{i=1}^{n}(y_i x_i^T\beta - p^{(m)}(x_i)x_i^T\beta) \tag{4.25}$$

$$+\sum_{i=1}^{n} p^{(m)}(x_i)(1-p^{(m)}(x_i))(\beta^{(m)})^T x_i x_i^T\beta \tag{4.26}$$

$$-\frac{1}{2}\sum_{i=1}^{n} p^{(m)}(x_i)(1-p^{(m)}(x_i))\beta^T x_i x_i^T\beta + \text{constant}(\beta^{(m)}) \tag{4.27}$$

Ignoring the constants we get

$$\frac{1}{2}\sum_{i=1}^{n} p^{(m)}(x_i)(1-p^{(m)}(x_i))\left((x_i^T\beta)^2 - 2\left[\frac{y_i - p^{(m)}(x_i)}{p^m(x_i)(1-p^{(m)}(x_i))}\right](x_i^T\beta)\right) \tag{4.28}$$

For $i = 1 \ldots n$, set $w_i^{(m)} := p^{(m)}(x_i)(1-p^{(m)}(x_i))$, $z_i^{(m)} := \frac{y_i - p^{(m)}(x_i)}{p^m(x_i)(1-p^{(m)}(x_i))}$, we can re-arrange quadratic approximation (4.28) as

$$-\frac{1}{2}\sum_{i=1}^{n} w_i^{(m)}\left((x_i^T\beta)^2 - 2z_i^{(m)}(x_i^T\beta)\right) \tag{4.29}$$

Now we can **complete the square** without moving the minimum to get the final quadratic approximation:

$$L(\beta) \approx -\frac{1}{2}\sum_{i=1}^{n} w_i^{(m)}(z_i^{(m)} - x_i^T\beta)^2 + \text{constant}(y_i, x_i, b^{(m)}), \tag{4.30}$$

turning it into the form of a **weighted least squares** problem

$$\beta^{(m+1)} = \arg\min_{\beta} \sum_{i=1}^{n} w_i^{(m)} (z_i^{(m)} - x_i^T \beta)^2 \tag{4.31}$$

which gives the algorithm its other name: **Iteratively Reweighted Least Squares (IRLS)**.

4.2.2 IRLS and Penalties

Suppose we want to solve a *penalized* logistic regression with penalty $P(\beta)$:

$$\min_{\beta_0,\beta} (-L(\beta_0, \beta) + \lambda P(\beta)) \tag{4.32}$$

As in (4.25), we make a quadratic approximation to the negative-log likelihood term, and minimize

$$(\beta_0^{(m+1)}, \beta^{(m+1)}) = \arg\min_{\beta_0,\beta} \sum_{i=1}^{n} w_i^{(m)} (z_i^{(m)} - \beta_0 - x_i^T \beta^2) + \lambda P(\beta) \tag{4.33}$$

Note that we've separated out the intercept, since we **never** want to penalize the intercept.