Discussion 5 **Alex Francis**
CS 189 Email: afrancis@berkeley.edu
02/21/2017 OH: Wednesday 10:00am - 12:00pm (283E Soda)

1 Weighted Least Squares

In our traditional least squares scenario, we minimize the least squares error, or:

$$L(\beta) = \sum_{i=1}^{n} (y_i - \beta^T \mathbf{x}_i)^2$$

Note: $\beta \in \mathbb{R}^d$ is just a different notation for $\mathbf{w}$.

A generalization of this scenario is one where we minimize a sum of weighted errors, where some training points may have more weight than others. One might imagine scenarios where the weights correspond to sensor error probability or how much you trust a user's account. Given some weight vector, $[c_1, c_2, \ldots, c_n]^T$,

$$L(\beta) = \sum_{i=1}^{n} c_i (y_i - \beta^T \mathbf{x}_i)^2$$

Find the value of β that minimizes the weighted least-squares error.

Hint: first vectorize the least squares error (express as a matrix expression).

Solution. Taking the hint in the problem statement, we first notice that we can construct a diagonal matrix of weights C such that

$$L(\beta) = \|C^{1/2}(Y - X\beta)\|^2$$

Where $C^{1/2}$ is the diagonal matrix of the square roots of the values in C, the diagonal matrix of the vector above. This can be rewritten as,

$$L(\beta) = (Y - X\beta)^\top C(Y - X\beta)$$

The rest is standard gradient computation.

$$\begin{aligned}
\nabla_\beta L(\beta) &= 0 \\
\nabla_\beta \Big((Y - X\beta)^\top C(Y - X\beta) \Big) &= 0 \\
\nabla_\beta \Big((Y^\top - \beta^\top X^\top)(CY - CX\beta) \Big) &= 0 \\
\nabla_\beta \Big(Y^\top CY - Y^\top CX\beta - \beta^\top X^\top CY + \beta^\top X^\top CX\beta \Big) &= 0 \\
-Y^\top CX - X^\top CY + 2X^\top CX\beta &= 0 \\
2X^\top CX\beta &= 2X^\top CY \\
\beta &= (X^\top CX)^{-1} X^\top CY
\end{aligned}$$

Note that instead of using the expansion approach, we could have used the chain rule. This approach is generally more complicated, and requires vector-by-vector differentiation, which has

thus far not been covered in the course. I've attached some details on it below, in case you're interested.

Definition 2.1. (**Vector-by-Vector Differentiation**). The derivative of a vector function (a vector whose components are functions), $y = \begin{bmatrix} y_1 \dots y_m \end{bmatrix}^\top$ with respect to an input vector $x = \begin{bmatrix} x_1 \dots x_n \end{bmatrix}^\top$ is written,

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

Lemma 2.2.

$$\nabla_x (Ax) = A$$

Proof. Apply Definition 2.1. □

Lemma 2.3.

$$\nabla_x (x^\top A) = A$$

Proof. Apply Definition 2.1. □

2 Discriminant Analysis (Spring 2016 Midterm)

Let's derive the decision boundary when one class is Gaussian, and the other class is exponential. Our feature space is one-dimensional ($d = 1$).

We have two classes, named N for normal and E for exponential. For the former class ($Y = N$), the prior probability is $\pi_N = P(Y = N) = \frac{\sqrt{2\pi}}{1 + \sqrt{2\pi}}$ and the class conditional $P(X \mid Y = N)$ has the normal distribution $\mathcal{N}(0, \sigma^2)$. For the latter, the prior probability is, $\pi_E = P(Y = E) = \frac{1}{1 + \sqrt{2\pi}}$ and the class conditional has the exponential distribution.

$$P(X = x \mid Y = E) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Write an equation in x for the decision boundary (ignore all $x < 0$). Use the 0-1 loss function. Simplify the equation until it is quadratic in x. (Ignore the fact that 0 might not also be a point in the decision boundary). **Show your work**, starting from the posterior probability.

Solution. Students should have the intuition that, given 0-1 loss, the decision boundary is the point where,

$$P(Y = N \mid X = x) = P(Y = E \mid X = x)$$

This was derived in an earlier lecture, where we were minimizing *expected loss*, or "risk." Then, we have,

$$\frac{\mathbb{P}(X = x \mid Y = N)P(Y = N)}{P(X = x)} = \frac{P(X = x \mid Y = E)P(Y = E)}{P(X = x)}$$

$$\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)\left(\frac{\sqrt{2\pi}}{1+\sqrt{2\pi}}\right) = \lambda \exp(-\lambda x)\left(\frac{1}{1+\sqrt{2\pi}}\right)$$

$$\frac{1}{\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) = \lambda \exp(-\lambda x)$$

Taking logs on both sides,

$$\log P(Y = N \mid X = x) = \log P(Y = E \mid X = x)$$

$$-\log(\sigma) - \frac{x^2}{2\sigma^2} = \log(\lambda) - \lambda x$$

$$\frac{x^2}{2\sigma^2} - \lambda x + \log(\sigma) + \log(\lambda) = 0$$

The final line is the correct decision boundary.

3 Lecture on Bayesian Parameter Estimation

Last week, I lied to you and told you that there were three forms of estimation that are commonly used. Oops. Another form of parameter estimation is called *Bayesian Parameter Estimation* and has much the same flavor as *Maximum Likelihood Estimation* (and is, in fact, equivalent in the case in which the prior is the uniform distribution — more on this later). The idea behind Bayesian parameter estimation is the following: instead of maximizing the conditional probability of the data given the parameter, let's maximum the conditional probability of the parameter given the data. Of course, this is equivalent to maximizing the *posterior* probability. This method is, therefore, sometimes called *Maximum A Posteriori Estimation* (MAPE). Recall, denoting as the parameter vector θ, and denoting the data points $x_1, \ldots, x_n$ (this is standard notation),

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

$$\mathbb{P}(\theta \mid x_1, \ldots, x_n) = \frac{\mathbb{P}(x_1, \ldots, x_n \mid \theta) \times \mathbb{P}(\theta)}{\mathbb{P}(x_1, \ldots, x_n)}$$

The nature of the mechanic of the optimization is sometimes quite similar, and sometimes quite different, here. One way to solve for the parameter vector is to do the exact same thing we did with MLE's: take log, compute derivatives, set equal to zero. Another way is to compute the *distributional form* of the resulting posterior probability, then use the *mode* (the peak of the distribution) or *mean* (expected value) of the resulting distribution as an optimal parameter vector.[1]

Claim 2.1. MAPE and MLE are the same when the prior is uniform.

Proof. Say θ is a single random variable. A similar result exists for a vector of random variables. When the prior is uniform,

$$P(\theta) = \frac{1}{b-a}$$

[1] See *Mathematical Statistics and Data Analysis* by Rice.

Where b and a are the upper and lower limits of the interval upon which the random variable is defined, respectively. Then, we would like to maximize,

$$\mathbb{P}(\theta \mid x_1, \ldots, x_n) = \frac{\mathbb{P}(x_1, \ldots, x_n \mid \theta)}{(b-a)\mathbb{P}(x_1, \ldots, x_n)}$$

Since the bottom is some normalizing constant that does not depend on θ, maximizing the posterior for θ is the same as maximizing the likelihood for θ. □

The uniform distribution is a "non-committal" and "utilitarian" prior. In words, choosing this as a prior means "letting the data do the talking." Intuitively, this method is more likely to overfit, so if we can choose a meaningful prior, the parameter vector will be less dramatically altered by the data.

Out of Scope. Since the second approach for optimization requires more advanced knowledge of probability, we will not discuss it in this course (this will be skipped in discussion). However, to get a sense of how this works, I have included an interesting, and relatively simple, example.

Fitting a Poisson Distribution. Recall that the Poisson probability mass function is,

$$f_{X_i|\Lambda}(x_i \mid \lambda) = \frac{\lambda^{x_i} e^{-\lambda}}{x_i!}$$

Using the methods of Bayesian Parameter Estimation, compute λ for arbitrary an arbitrary vector of data $\langle x_1, \ldots, x_n \rangle$, assuming that the prior distribution $\sim$ Gamma(v, α) where the formula for the gamma density is,

$$f(x) = \frac{v^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-vx}$$

Solution. Using the omniscient relationship,

$$\begin{aligned}
\text{posterior} \;&\propto\; \text{likelihood} \;\times\; \text{prior} \\
\mathbb{P}(\lambda \mid x_1, \ldots, x_n) &= \frac{\mathbb{P}(x_1, \ldots, x_n \mid \lambda) \times \mathbb{P}(\lambda)}{\mathbb{P}(x_1, \ldots, x_n)} \\
&= \frac{\mathbb{P}(\lambda) \prod_{i=1}^n \mathbb{P}(x_i \mid \lambda)}{\mathbb{P}(x_1, \ldots, x_n)} \\
&= \frac{\left(\frac{v^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-v\lambda}\right) \left(\lambda^{\sum_{i=1}^n x_i} e^{-n\lambda}\right)}{\int_0^\infty \left(\frac{v^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-v\lambda}\right) \left(\lambda^{\sum_{i=1}^n x_i} e^{-n\lambda}\right)} \\
&= \frac{\lambda^{\alpha-1+\sum_{i=1}^n x_i} e^{-\lambda(n+v)}}{\int_0^\infty \lambda^{\alpha-1+\sum_{i=1}^n x_i} e^{-\lambda(n+v)}}
\end{aligned}$$

Since the denominator is just a constant factor that makes the density integrate to 1, we conclude that the posterior probability $\sim$ Gamma$(v' = n + v, \alpha' = \alpha + \sum_{i=1}^n x_i)$.[2] Now, we have a distributional form for the parameter λ. Two good guesses for λ would be the mode (the

[2]When the posterior has the same for as the prior, the prior is said to be a *conjugate prior*. Another famous example, ubiquitous in Data Science applications, is the *Beta* distribution.

highest probability value) and the mean. Both are valid, but we'll use the latter, since it's easy to compute. It is known that, for a gamma random variable X,

$$\mathbb{E}[X] = \frac{\alpha}{v}$$

Therefore, a Bayesian parameter estimate for λ is,

$$\lambda = \frac{\alpha'}{v'} = \frac{\alpha + \sum_{i=1}^{n} x_i}{n + v}$$

---

4 Ridge Regression with Laplace Prior

As we discussed in lecture, linear regression can be expressed as a model of the form $P(y|\mathbf{x}, \sigma^2) \sim \mathcal{N}(\mathbf{w^T x}, \sigma^2)$.

Next week we will talk about regularization, which are methods to prevent overfitting (you have been using regularization throughout the class even if we haven't explained it yet). The reason that maximum likelihood estimation (MLE) can overfit is that it is picking the parameters that are the best for modeling the training data: but if the data is noisy, such parameters can result in complex functions. Here we will dive into one way to prevent overfitting: L1 regularization, also known as LASSO. The way we prevent MLE from overfitting is we assume a prior distribution on parameters $\mathbf{w}$. Let us assume the prior is a Laplace distribution, $w_j \sim Laplace(0, t)$, i.e. $P(w_j) = \frac{1}{2t} e^{-|w_j|/t}$ and $P(\mathbf{w}) = \prod_{j=1}^{d} P(w_j) = (\frac{1}{2t})^d \cdot e^{-\frac{\sum |w_j|}{t}}$.

When you assume a non-uniform prior, this is called maximum a posteriori (MAP) estimation. Show that finding the MAP estimate for $\mathbf{w}$ is equivalent to minimizing the following:

$$L(\mathbf{w}) = \sum_{i=1}^{n} (y_i - \mathbf{w^T x}_i)^2 + \lambda \|\mathbf{w}\|_1$$

And find the constant λ.

Solution. Start with the important formula.

$$\begin{aligned}
\text{posterior} &\propto \text{likelihood} \times \text{prior} \\
\mathbb{P}(\mathrm{w} \mid y_1, \ldots, y_n) &\propto \mathbb{P}(y_1, \ldots y_n \mid \mathbf{w}) \times \mathbb{P}(\mathbf{w}) \\
&= \mathbb{P}(\mathbf{w}) \prod_{i=1}^{n} \mathbb{P}(y_i \mid \mathbf{w}) \\
&= \left(\frac{1}{2t}\right)^d \exp\left(-\frac{\sum_j |w_j|}{t}\right) \prod_{i=1}^{n} \left(\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y_i - \mathbf{w}^\top \mathbf{x}_i}{\sigma}\right)\right)\right)
\end{aligned}$$

Since the logarithm function is monotonically increasing, it is sufficient to maximize the log.

$$\begin{aligned}
\log \mathbb{P}(\mathrm{w} \mid y_1, \ldots, y_n) &= \log\left(\left(\frac{1}{2t}\right)^d \exp\left(-\frac{\sum_j |w_j|}{t}\right) \prod_{i=1}^{n} \left(\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y_i - \mathbf{w}^\top \mathbf{x}_i}{\sigma}\right)\right)\right)\right) \\
&= -d\log(2t) - \frac{1}{t}\sum_j |w_j| - \frac{n}{2}\log(2\pi) - n\log(\sigma) - \frac{1}{2\sigma^2}\sum_{i=1}^{n} (y_i - \mathbf{w}^\top \mathbf{x}_i)^2
\end{aligned}$$

The only relevant terms are the ones with $\mathbf{w}$. Therefore, we would like to maximize,

$$-\frac{1}{t}\sum_j |w_j| - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2$$

Equivalently, we would like to minimize

$$\frac{1}{t}\sum_j |w_j| + \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2$$

Which is the same as minimizing (by multiplying both sides by $2\sigma^2$).

$$\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \frac{2\sigma^2}{t}\|\mathbf{w}\|_1$$

This completes the proof, with

$$\lambda = \frac{2\sigma^2}{t}$$

$$\begin{aligned}
\mathbb{P}\{\lambda \mid X_1 = x_1, \ldots, X_n = x_n\} &\propto \mathbb{P}\{X_1 = x_1, \ldots, X_n = x_n \mid \lambda\}\mathbb{P}\{\lambda\} \\
&= \prod_{i=1}^{n} \mathbb{P}\{X_i = x_i \mid \lambda\}\mathbb{P}\{\lambda\} \\
&= \begin{cases} \left(\prod_{i=1}^{n} \lambda e^{-\lambda x_i}\right) \times 1 & \lambda \in [0,1] \\ 0 & \text{o/w} \end{cases} \\
&= \begin{cases} \left(\lambda^n \exp\left(-\lambda \sum_{i=1}^{n} x_i\right)\right) & \lambda \in [0,1] \\ 0 & \text{o/w} \end{cases} \\
\log(\mathbb{P}\{\lambda \mid X_1 = x_1, \ldots, X_n = x_n\}) &= \begin{cases} n\log(\lambda) - \lambda \sum_{i=1}^{n} x_i & \lambda \in [0,1] \\ 0 & \text{o/w} \end{cases}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial \lambda}\left(\log(\mathbb{P}\{\lambda \mid X_1 = x_1, \ldots, X_n = x_n\})\right) &= \frac{n}{\lambda} - \sum_{i=1}^{n} x_i = 0 \\
\lambda &= \frac{n}{\sum_{i=1}^{n} x_i} = \frac{1}{\bar{x}}
\end{aligned}$$

5 The Social Network

Social networks are excellent examples of undirected graphs in the real world. Let $G = (V, E)$ be a graph in which the vertices V are members of a social media website and the edges E represent connections between two individuals in a social network. In this problem, we will attempt to cluster users into representative "sub-networks," so that we can identify the social "groups" in which our users participate.

Lets use the techniques of spectral graph clustering to learn the sub-networks. One way to model the network is to let the edge weights $w_{ij} = \#$ of mutual friend between user i and user j. Assume for now that we simply have the adjacency matrix for the matrix M, where $M_{ij} \in \{0, 1\}$, and is 1 if and only if there is an edge connecting node i to node j.

1. (3 points) State the Laplacian matrix, L, in terms of the entries in matrix M.

   Solution. Notice that the number of mutual connections between individuals in the network, (i, j) can be modeled as the sum,

   $$w_{ij} = \sum_{k=1}^{|V|} M_{ik} M_{jk} = \langle M_i, M_j \rangle$$

   Where M_i and M_j are the ith and jth rows of the matrix, respectively. Then, the Laplacian is,

   $$L_{ij} = \begin{cases} -\langle M_i, M_j \rangle & i \neq j \\ \sum_{k \neq i} \langle M_i, M_k \rangle & i = j \end{cases}$$

   (a) $Mass(G) \left(\text{Cut}(G_1, G_2) \left(\frac{1}{\text{Mass}(G_1)} + \frac{1}{\text{Mass}(G_2)} \right) \right)$ (b) $\frac{\text{Cut}(G_1, G_2)\text{Mass}(G)}{\text{Mass}(G_1) + \text{Mass}(G_2)}$ (c) $\frac{\text{Cut}(G_1, G_2)}{\text{Mass}(G_1)\text{Mass}(G_2}$

   Show that finding the eigendecomposition of the generalized eigensystem $Lv = \lambda Mv$ is the same as finding the eigendecomposition of the matrix A.