CS 475 Machine Learning: Lecture 3
Notes for Regression

Prof. Mark Dredze

1 Maximum Likelihood for Gaussians

Let's begin by looking at Maximum Likelihood for Gaussians. We are given a dataset that contains points that we believe have been sampled from a Gaussian distribution. Our set of points is given by $\mathbf{X} = \{\mathbf{x}_i\}$, where $\mathbf{x}_i \in \mathcal{R}^M$. We would like to estimate the parameters of this Gaussian: μ and $\mathbf{\Sigma}$. We want to measure the likelihood that these points (our data $\mathcal{D}$) were generated by the parameters μ and $\mathbf{\Sigma}$.

Using an M-dimensional Gaussian the likelihood of a single instance is

$$p(\mathbf{x}|\mu, \mathbf{\Sigma}) = \frac{1}{\sqrt{(2\pi)^M}} \frac{1}{\sqrt{|\mathbf{\Sigma}|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right\}$$

where $|\mathbf{\Sigma}|$ is the determinant of $\mathbf{\Sigma}$. The likelihood of our data is the product of the likelihood of each individual example.

$$p(\mathcal{D}|\mu, \mathbf{\Sigma}) = \prod_{i=1}^{N} \frac{1}{\sqrt{(2\pi)^M}} \frac{1}{\sqrt{|\mathbf{\Sigma}|}} \exp\left\{-\frac{1}{2}(\mathbf{x}_i-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}_i-\mu)\right\}$$

To simplify the function, we take the log of the likelihood.

$$\mathcal{L} = \log p(\mathcal{D}|\mu, \mathbf{\Sigma}) = -\frac{NM}{2}\log(2\pi) - \frac{N}{2}\log|\mathbf{\Sigma}| - \frac{1}{2}\sum_{i=1}^{N}(\mathbf{x}_i-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}_i-\mu)$$

We now have a function of variables $\mathbf{\Sigma}$ and μ. We can find the maximum of this function, ie. the point of highest likelihood, by taking the derivative of the function with respect to each of the variables. We then set the derivatives to 0 and solve for the variable.

Since only the last term depends on μ so we can get:

$$\frac{\partial \mathcal{L}}{\partial \mu} = \sum_{i=1}^{N} \mathbf{\Sigma}^{-1}(\mathbf{x}_i - \mu) = 0$$

Moving the μ terms to the right hand side and multiplying by $\mathbf{\Sigma}$, we get a solution for μ

$$\mu_{ML} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$$

This is the maximum likelihood solution for the variable μ.

The same approach for $\mathbf{\Sigma}$ is much more involved. It is illustrative to try the 1-dimensional case, where $\mathbf{\Sigma}$ is actually a 1×1 matrix (ie. a scalar) which we indicate by σ. Using the 1-dimesional Gaussian we get:

$$\sigma^2_{ML} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \mu_{ML})(\mathbf{x}_i - \mu_{ML})^T$$

What are the expectations of these parameters, meaning what values do we expect to get for these parameters?

$$\mathbb{E}[\mu_{ML}] = \mu$$
$$\mathbb{E}[\mathbf{\Sigma}_{ML}] = \frac{N-1}{N}\mathbf{\Sigma}$$

Notice that the expectation of μ is correct, it is equal to the true expectation of the Gaussian. However, the expectation of $\mathbf{\Sigma}$ is biased, it underestimates the true parameter by $\frac{N-1}{N}$. Maximum likelihood estimation thinks that there is less variance than truly present in the data, meaning that it over-fits the observed data. Over-fitting means that we have high variance (in the bias vs. variance sense): we are too trusting of our observed data. In contrast, we believe there is lower variance (not in the bias vs. variance sense) for the Gaussian.

When $N \to \infty$, we have infinite data, then this isn't a problem. However, with limited data, this becomes an issue.

If the estimator for μ is unbiased, why is estimator for $\mathbf{\Sigma}$ biased? The estimator for the variance is based on the sample mean ($\mathbf{X}$), and not the true mean of the distribution. That means that sometimes the mean is too low, and sometimes the mean is too high, when compared to the actual true mean of the distribution. When we compute μ_{ML} these biases cancel each other out (the "too high" estimates balance out the "too low" estimates.) However, variance squares the sample mean, which means that even when its under-estimates (misses the true mean by a negative number), these numbers become positive (in the square.) When all our estimates are off by a positive number, the positive and negative (relative) estimates no longer are canceled, so we end up over-estimating the distribution's mean.[1]

Note that we could modify the maximum likelihood estimator for variance to overcome this problem:

$$\mathbf{\Sigma} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}_i - \mu_{ML})(\mathbf{x}_i - \mu_{ML})^T$$

2 Maximum Likelihood for Least Squares Regression

In regression, we are given a set of examples $\mathbf{x}_i$, where $\mathbf{x}_i \in \mathcal{R}^M$ and $y_i \in \mathcal{R}$. We assume that these examples are generated by a linear function $f_{\mathbf{w}}(\mathbf{x})$ and are then permuted by Gaussian noise.

Let's start by writing the probability of the data $\mathcal{D}$. We replace the mean of the Gaussian with our prediction function $\mathbf{w}^T \cdot \mathbf{x}_i$. Recall that $\mathbf{w}$ is a $n \times 1$ vector and x is a $n \times 1$ vector, so the dot production of our prediction is a real number. Since the parameters $\mathbf{w}$ are the weights given to each feature, we call $\mathbf{w}$ the weight vector.

We replace y with our prediction function $\mathbf{w}^T \cdot \mathbf{x}_i$ and write the log likelihood of all of our regression data:

$$\log p(\mathcal{D}|\mathbf{w}, \mathbf{\Sigma}) = \log \prod_{i=1}^{N} \mathcal{N}(\mathbf{w}^T \cdot \mathbf{x}_i, \mathbf{\Sigma})$$

We can distribute the log, which breaks the products into sums. For simplicity, we switch to an alternate form of the Gaussian, which replaces $\mathbf{\Sigma}$ with β^{-1}, where β is the precision

[1] See a nice explanation here: `http://stats.stackexchange.com/questions/136673/how-to-understand-that-mle-of-variance-is-biased-in-a-gaussian-distribution`.

(inverse variance) of the Gaussian:

$$\log p(\mathcal{D}|\mathbf{w}, \mathbf{\Sigma}) = \frac{N}{2} \log \beta^{-1} - \frac{N}{2} \log(2\pi) - \frac{\beta}{2} \sum_{i=1}^{N} \{y_i - \mathbf{w}^T \cdot \mathbf{x}_i\}^2$$

To maximize the likelihood we take the gradient with respect to each parameter in $\mathbf{w}$; we write the jth position as w^j.

For simplicity, let's start by assuming that $\mathbf{x}_i$ is of length two. We'll exclude the bias term b (in $y = \mathbf{w}\mathbf{x} + b$) in this presentation, though it is easy to add.

There is only a single term that depends on $\mathbf{w}$ in the likelihood, which means the likelihood is proportional to the function $\mathcal{L}(\mathbf{w})$:

$$\log p(\mathcal{D}|\mathbf{w}, \mathbf{\Sigma}) \propto \mathcal{L}(\mathbf{w}) = -\frac{\beta}{2} \sum_{i=1}^{n} (y_i - w^0 x_i^0 - w^1 x_i^1)^2$$

Here we have expanded the weight vector into each of its components. Taking the gradient with respect to w^1:

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial w^1} = -\beta \sum_{i=1}^{n} (y_i - w^0 x_i^0 - w^1 x_i^1)(-x_i^1)$$

We can set this to 0 and solve for w^1 in terms of w^0.

We can do the same step for w^0 to obtain.

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial w^0} = -\beta \sum_{i=1}^{n} (y_i - w^0 x_i^0 - w^1 x_i^1)(-x_i^0)$$

We now have two equations and two variables, so we can now solve for w^1 and w^0.

Returning to the general case, we have a weight vector $\mathbf{w}$ of length M. Rather than writing the solution for each w^j separately, let's write them together. As before, the gradient of the log likelihood function with respect to $\mathbf{w}$ only depends on the last term. We can rearrange terms and drop β to yield:

$$\nabla \log p(\mathcal{D}|\mathbf{w}, \beta) = \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i^T$$

The right hand side evaluations to a vector of values, which is the solution for $\mathbf{w}$. By setting the gradient equal to the 0 vector, we distribute $\mathbf{x}_i^T$ and move in the summation:

$$0 = \sum_{i=1}^{n} y_i \mathbf{x}_i^T - \mathbf{w}^T \left(\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T \right)$$

Rearranging terms to solve for w:

$$\sum_{i=1}^{n} y_i \mathbf{x}_i^T = \mathbf{w}^T \left(\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T \right)$$

Notice that each sum is over a matrix, so we have matrix addition.

To move the matrix resulting from the summation over $\mathbf{x}$, we take the inverse and solve for $\mathbf{w}$ to obtain.

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$$

where $\mathbf{X}$ is a matrix where each row is a vector $\mathbf{x}_i$.

This solution expresses the maximum likelihood solution for $\mathbf{w}$ in terms of our training data $\mathbf{X}$ and $\mathbf{Y}$.

2.1 Loss function

In lecture we decided to use the sum of squares loss function:

$$\sum_{i=1}^{n}(y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$$

We considered two methods for selecting $\mathbf{w}$.

1. Select the $\mathbf{w}$ that maximizes the likelihood of the observed data.
2. Select the $\mathbf{w}$ that minimizes the error on the observed data.

We chose to pursue the first and developed the maximum likelihood solution. Let us now consider the second approach.

Consider again the log-likelihood:

$$\log p(\mathcal{D}|\mathbf{w}, \mathbf{\Sigma}) = \frac{N}{2}\log\beta^{-1} - \frac{N}{2}\log(2\pi) - \frac{\beta}{2}\sum_{i=1}^{N}\{y_i - \mathbf{w}^T \cdot \mathbf{x}_i\}^2$$

As we mentioned before, only the last term depends on our choice of $\mathbf{w}$. Look carefully at this last term and our loss function. You will see they are the same. When we are maximizing the log-likelihood we are in effect minimizing the error function. Therefore, least squares linear regression is maximum likelihood estimation for Gaussians!

3 Regularized Least Squares

Forcing the most likely parameters for our data may cause us to favor parameters that do not generalize on test data. Following Occam's razor, we want to favor simpler models over complex models. Therefore, we want to both minimize the error but enforce a simple choice of parameters.

We can achieve this by rewriting our objective as a combination of two terms.

$$E_{\mathcal{D}}(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

The first term $E_{\mathcal{D}}(\mathbf{w})$ is the error of $\mathbf{w}$ on our data $\mathcal{D}$. The second term is a measure of error of $\mathbf{w}$ on itself, ie. some intrinsic property of $\mathbf{w}$. We call the second term a regularizer, since it imposes some regular constraint over the weight vector. λ indicates how important each term is. We want to minimize both the error on the data and the complexity of $\mathbf{w}$.

We again choose the sum of squares error:

$$E_{\mathcal{D}}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\{y_i - \mathbf{w}^T \cdot \mathbf{x}_i\}^2$$

To penalize $\mathbf{w}$ we take a similar approach and compute its sum of squares of the weight vector elements:

$$E_W(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

We combine these into a single objective.

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\{y_i - \mathbf{w}^T \cdot \mathbf{x}_i\}^2 + \lambda\frac{1}{2}\mathbf{w}^T\mathbf{w}$$

This function is quadratic in $\mathbf{w}$ and so we can find its solution in closed form thanks to the quadratic formula. If we proceed as before and solve for $\mathbf{w}$ we get:

$$\mathbf{w} = \left(\lambda\mathbf{I} + \mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}$$

$\mathbf{I}$ is the identity matrix, the all 0 matrix whose diagonal elements are all 1. Notice we are just increasing the values of $\mathbf{X}^T\mathbf{X}$ where λ indicates by how much. When λ is 0 we have regular least squares. When λ increases, $\mathbf{w}$ gets smaller since λ is in the inverse.

3.1 Generalized Regularization

We can write a more general form for regularization:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\{y_i - \mathbf{w}^T \cdot \mathbf{x}_i\}^2 + \lambda\frac{1}{2}\sum_{j=1}^{M}|w^j|^q$$

Here we specify the regularizer by setting q. Notice that $q = 2$ is the quadratic regularizer. This is typically called $L2$ regularization.

Another common choice is $q = 1$, which is called $L1$ regularization. Regularized least squares with an $L1$ regularizer is called *Lasso*. Lasso has the property that for large λ the number of 0 terms in $\mathbf{w}$ increases leading to a sparse solution.

Sparse solutions are often desirable. In many applications, we assume that despite the large number of input features, only a few are actually predictive of the label. A sparse $\mathbf{w}$ enforces this assumption. Additionally, we may want to deploy our learned model on a resource limited device, such as a mobile phone. While we could learn a solution using many of the features, we'd prefer a model that used only a few features, reducing the cost of extracting these features and representing a large model.