Week 2.5-3 Document

aniruddh.kishore@iiitb.ac.in

14 September 2021

Part I
Multivariate Gaussians

1 Introduction to the Multivariate Gaussian

The functional form of the multivariate Gaussian is pretty intimidating, but can be demystified if we consider the bivariate case. The multivariate Gaussian:

$$N(\vec{\mu}, \Sigma) \sim \frac{\exp(-\frac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu}))}{\sqrt{(2\pi)^k |\Sigma|}}$$

Where $\vec{x}$ is k-dimensional, Σ is the covariance matrix and $\vec{\mu}$ is the mean vector.

That's a lot of things that don't seem to make much sense yet.

- $\vec{x}$ is just the input we give, like how x was with the univariate case. Here, we consider it to be k-dimensional.
- $\vec{\mu}$ is the mean of the multivariate Gaussian – k-dimensional.
- Σ[1] ($k \times k$ dim) not only tells us what the spread is in a particular axis, but also what the linear dependence is between the various axes – remember that the whole point of this can of worms is because the features need not be linearly independent. Note that Σ is a positive, semi-definite symmetric matrix[2] (a fact that we will use later).

The **bivariate** normal distribution is:

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{1}{2(1-\rho^2)}\left[\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2} - 2\rho(\frac{x-\mu_x}{\sigma_x})(\frac{y-\mu_y}{\sigma_y})\right]\right)$$

Here, we have five parameters:

[1] `https://en.wikipedia.org/wiki/Covariance_matrix` is a good reference.

[2] The main properties: The determinant is ≥ 0 , and $v^t A v \geq 0$ for all real v. See Wikipedia for more.

μ_x and μ_y as the means in x and y, σ_x and σ_y are the variances in those directions (a positive quantity) and ρ is the correlation coefficient – a measure of the linear dependence between x and y (a quantity between -1 and 1). You can explore an interactive contour plot of the bivariate normal at `https://www.desmos.com/calculator/8mckqwfrq8`, and the 3D plot of the bivariate at `http://socr.ucla.edu/htmls/HTML5/BivariateNormal/`. Since those plots are available, I omit all diagrams here.

The equation for the bivariate normal is perfectly equivalent to the earlier definition of the multivariate normal with

$$\vec{\mu} = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}$$

and

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}$$

2 Back to Maximum Likelihood Estimation

In the last write-up, before we applied the Naive condition, we were at:

$$\mathcal{L}_X(\theta) = \prod_{c_j} \prod_{y_i = c_j} P(X[i,:] | y_i = c_j)$$

Now, we do not apply the Naive condition, and instead, we consider that for each class, we have one multivariate gaussian which encompasses all the features. We have a Σ and a $\vec{\mu}$ for each class.
Substituting those Gaussians, we have[3]:

$$\mathcal{L}_X(\theta) = \prod_{c_j} \prod_{y_i = c_j} \frac{\exp(-\frac{1}{2}(X[i,:] - \vec{\mu}_{c_j})^T \Sigma_{c_j}^{-1} (X[i,:] - \vec{\mu}_{c_j}))}{\sqrt{(2\pi)^k |\Sigma_{c_j}|}}$$

Despite the way it looks, the numerator and denominator both evaluate to positive scalars. We take the logarithm, as before:

$$\log(\mathcal{L}_X(\theta)) = \sum_{c_j} \sum_{y_i = c_j} \left(-\frac{1}{2} \log|\Sigma_{c_j}| - \frac{1}{2}(X[i,:] - \vec{\mu}_{c_j})^T \Sigma_{c_j}^{-1} (X[i,:] - \vec{\mu}_{c_j}) \right) + \text{terms involving } \pi$$

[4]

The logarithm comes from the $\sqrt{\Sigma}$ term in the bottom, and the rest comes from the exp term.

[3]Here, I'm considering $X[i,:]$ to be the column vector composed of the ith row of the matrix X. Don't get confused!

[4]I hope you can differentiate between $\sum$, the summation operator and Σ, the covariance matrix, by looking at the context.

Now, here, we have a pickle. There's no more tricks we can use to avoid solving this. [5]

First, let's simplify the expression a bit further.

$$\log(\mathcal{L}_X(\theta)) = \sum_{c_j}\left(-\frac{1}{2}n_{c_j}\log|\Sigma_{c_j}| - \frac{1}{2}\sum_{y_i=c_j}\left(X[i:,]^T\Sigma_{c_j}^{-1}X[i,:] + \vec{\mu}_{c_j}^T\Sigma_{c_j}^{-1}\vec{\mu}_{c_j} - 2X[:,i]^T\Sigma_{c_j}^{-1}\vec{\mu}_{c_j}\right)\right)$$

All I've done is that I've applied the inner $\sum$ to the log term, and I've expanded the transpose-inverse product into the individual terms. Here, we use the fact that $a^TXb = b^TXa$ if X is symmetric and if the resulting a^TXb and b^TXa are scalars.
Now, putting all the summations in the right place, we get:

$$\log(\mathcal{L}_X(\theta)) = \sum_{c_j}\left(-\frac{1}{2}n_{c_j}\log|\Sigma_{c_j}| - \frac{1}{2}n_{c_j}\vec{\mu}_{c_j}^T\Sigma_{c_j}^{-1}\vec{\mu}_{c_j} - \frac{1}{2}\sum_{y_i=c_j}\Big(X[i:,]^T\Sigma_{c_j}^{-1}X[i,:]\Big) + \Big(\sum_{y_i=c_j}X[:,i]^T\Big)\Sigma_{c_j}^{-1}\vec{\mu}_{c_j}\right)$$

2.1 Solving for μ

Now, we need a bit of matrix calculus. There is a very useful reading that I use as a reference `http://cs229.stanford.edu/section/cs229-linalg.pdf`. **Only refer to pages 20 to 23 for now – and definitely avoid section 4.5.**. We will need these identities for Linear Regression, Logistic Regression and for Neural Networks as well. The identities we need for now are:

$$\frac{\partial(\vec{b}^T\vec{x})}{\partial\vec{x}} = \vec{b}$$

and

$$\frac{\partial(\vec{x}^TA\vec{x})}{\partial\vec{x}} = 2A\vec{x}$$

where A is a symmetric matrix.

Applying this to the log-likelihood, we get:

$$\frac{\partial\log(\mathcal{L}_X(\theta))}{\partial\vec{\mu}_{c_j}} = -n_{c_j}\Sigma_{c_j}^{-1}\vec{\mu}_{c_j} + \Sigma_{c_j}^{-1}\Big(\sum_{y_i=c_j}X[:,i]\Big)$$

Taking Σ^{-1} common,

$$\frac{\partial\log(\mathcal{L}_X(\theta))}{\partial\vec{\mu}_{c_j}} = \Sigma_{c_j}^{-1}\left(\Big(\sum_{y_i=c_j}X[:,i]\Big) - n_{c_j}\vec{\mu}_{c_j}\right) = 0$$

[5]In this doc, I'll just be solving it for μ and not for Σ – the latter is an extremely advanced topic due to the derivative-of-determinant term. In case you're interested, `https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/other-readings/chapter13.pdf` should help you out.

Now, this is an equation of the form $Ax = 0$, where we assume A to be constant and solve for x. Now, $Ax = 0$ has nonzero solutions only when A is **singular**. However, we have already considered Σ^{-1} here, and invertible matrices are never singular. Therefore, the only solution to $Ax = 0$ is $x = 0$. Therefore,

$$\Big(\sum_{y_i = c_j} X[:, i]\Big) - n_{c_j} \vec{\mu}_{c_j} = 0$$

Or, as we have so often seen before:

$$\vec{\mu}_{c_j} = \frac{1}{n_{c_j}} \Big(\sum_{y_i = c_j} X[:, i]\Big)$$

Which is the same, by-and-large, as the univariate case. The derivation for Σ is pretty involved, so we skip it. The final result is that:

$$\Sigma_{c_j} = \frac{1}{n_{c_j}} \sum_{y_i = c_j} \Bigg((X[i, :] - \vec{\mu}_{c_j})(X[i, :] - \vec{\mu}_{c_j})^T \Bigg)$$

Using these, you can perform the inference as desired – this ends our standalone discussion on Generative models.

3 Generative vs Discriminative models

So far, we have been referring to these models as **generative** models because, once we obtain the values for the parameters, we can actually generate new *synthetic* data samples from the model. The procedure for that is:

1. Sample a y from the frequency distribution for the classes – the n_{c_j}/n distribution.

2. Once you have the y, sample from the distribution/distributions for x, corresponding to that value of y.

There exists another class of models known as **discriminative** models – these models cannot be used to generate new points, because they do not really store information about the underlying distribution of the data. They do, however, store useful information about how to identify the class of a given data point, be it through a weight vector, a separating plane etc.
A useful fact about some of these discriminative models is that they make no assumption about the distribution of the data – and they can sometimes be very robust in classifying points that come outside the distribution of the training data.

Part II

Binary Classification Using Logistic Regression

4 The Logistic/Sigmoid Function

The sigmoid function is a very useful kind of ”squashing” function. An interactive sigmoid plot is given at `https://www.desmos.com/calculator/yvvhtxhiqv`. The functional form of the sigmoid is:

$$S(x) = \frac{1}{1 + e^{-x}}$$

The main properties of the sigmoid that you should notice are:

- It is defined, continuous and differentiable everywhere – which is good, from a gradient descent standpoint.
- It has a range of $(0, 1)$. As you approach $-\infty$, the sigmoid goes to 0, and at $+\infty$, it goes to 1.
- It is approximately linear in the middle region (around $x = 0$), and is approximately flat at the extremes.

These properties are very useful from the point of view of classification, which brings us to the logistic regression model.

5 Logistic Regression model definition

Given an $\vec{x}$, the logistic regression model performes the following:

1. Obtain a scalar that is a linear combination of the entries in x (like in linear regression).
2. Take the output of the previous step, and apply the sigmoid function to it.

Formally,

$$M(x) = \frac{1}{1 + e^{-w^T x}}$$

Where w is a weight vector and x is the input vector. **Note:** We assume that x has a ”dummy entry”, similar to the dummy column we had in closed-form linear regression, so that a bias (intercept) term can be encapsulated in the weights.

This model gives us a value between 0 and 1 – therefore, we can interpret $M(x)$ as $P(y = 1|x)$ (here, it is the probability mass, since y is from a discrete distribution of just 0 and 1).

6 Maximum Likelihood Estimation for Logistic Regression

6.1 Buildup

Here, we turn the MLE we did for Gaussians on its head. Instead of using Bayes' rule to obtain $P(y|x)$ in terms of $P(x|y)$, we approach the problem head-on – we directly estimate $P(y|x)$.

Therefore, we have:

$$\mathcal{L}_X(\theta) = \prod_{i=1}^{n} P(y = y_i|x_i)$$

Taking the logarithm,

$$\log(\mathcal{L}_X(\theta)) = \sum_{i=1}^{n} \log(P(y = y_i|x_i))$$

Now, to evaluate $\log(P(y = y_i|x_i))$, consider that:

- If $y_i = 1$, then $\log(P(y = y_i|x_i)) = \log(M(x_i))$.
- If $y_i = 0$, then $\log(P(y = y_i|x_i)) = \log(1 - M(x_i))$ – one minus the probability that y_i is 1.

But this if/else is a little clunky. With a little imagination, one can find that:

$$\log(P(y = y_i|x_i)) = y_i \log(M(x_i)) + (1 - y_i)\log(1 - M(x_i))$$

Stare at this equation for a little while. Plug in $y_i = 1$ and $y_i = 0$ and see for yourself that this definition is equivalent to the if/else definition above.
Plugging this into the likelihood, we get:

$$\log(\mathcal{L}_X(\theta)) = \sum_{i=1}^{n} \left(y_i \log(M(x_i)) + (1 - y_i)\log(1 - M(x_i)) \right)$$

7 The Binary Cross-Entropy Loss

We absolutely cannot obtain a closed-form solution for maximizing this likelihood function. What we *can* do, however, is to take the negative of the log likelihood, and run gradient descent on it.
We define a new cost function called the **binary cross-entropy loss**:

$$J(y, \hat{y}) = -y\log(\hat{y}) - (1 - y)\log(1 - \hat{y})$$

Where y is the true value and $\hat{y}$ is the predicted probability.

Convince yourself that, if $\hat{y}_i = M(x_i)$, then:

$$\sum_{i=1}^{N} J(y_i, M(x_i)) = -\log(\mathcal{L}_X(\theta))$$

Therefore, minimizing this loss function is equivalent to maximizing the likelihood as above.

Note: In case you think this part is fluff, it actually is. But a lot of the libraries and the literature refer to the binary cross-entropy loss, so it would be criminal to not refer to it here. We use the BCE loss later on for binary classification using neural networks.

8 The Gradient for Gradient Descent

$$\sum_{i=1}^{N} J(y_i, M(x_i)) = \sum_{i=1}^{n} \left(y_i \log(M(x_i)) + (1 - y_i)\log(1 - M(x_i)) \right)$$

$$= -\sum_{i=1}^{n} \left(y_i \log(\frac{1}{1 + e^{-w^T x_i}}) + (1 - y_i)\log(1 - \frac{1}{1 + e^{-w^T x_i}}) \right)$$

$$= -\sum_{i=1}^{n} \left(y_i \log(\frac{1}{1 + e^{-w^T x_i}}) + (1 - y_i)\log(\frac{1 + e^{-w^T x_i} - 1}{1 + e^{-w^T x_i}}) \right)$$

$$= -\sum_{i=1}^{n} \left(y_i \log(\frac{1}{1 + e^{-w^T x_i}}) + (1 - y_i)\log(\frac{e^{-w^T x_i}}{1 + e^{-w^T x_i}}) \right)$$

$$= -\sum_{i=1}^{n} \left(y_i \log(\frac{1}{1 + e^{-w^T x_i}}) + (1 - y_i)\log(\frac{1}{1 + e^{w^T x_i}}) \right)$$

$$= \sum_{i=1}^{n} \left(y_i \log(1 + e^{-w^T x_i}) + (1 - y_i)\log(1 + e^{w^T x_i}) \right)$$

Okay, let's take a breather. Above was just a little bit of reorganizing so that when we take the gradient, things become easier. Now comes the actual gradient calculation:

$$\frac{\partial}{\partial w} \sum_{i=1}^{N} J(y_i, M(x_i)) = \sum_{i=1}^{N} \left(y_i \frac{-x_i e^{-w^T x_i}}{1 + e^{-w^T x_i}} + (1 - y_i)\frac{x_i e^{w^T x_i}}{1 + e^{w^T x_i}} \right)$$

$$= \sum_{i=1}^{N} \left(y_i \frac{-x_i e^{-w^T x_i}}{1 + e^{-w^T x_i}} + (1 - y_i)\frac{x_i}{1 + e^{-w^T x_i}} \right)$$

$$= \sum_{i=1}^{N} \left(\frac{x_i - x_i y_i - x_i y_i e^{-w^T x_i}}{1 + e^{-w^T x_i}} \right)$$

$$= \sum_{i=1}^{N} \left(\frac{x_i - x_i y_i (1 + e^{-w^T x_i})}{1 + e^{-w^T x_i}} \right)$$

$$= \sum_{i=1}^{N} (x_i M(x_i) - x_i y_i)$$

$$= \sum_{i=1}^{N} (x_i (M(x_i) - y_i))$$

Some variants of this will have a $1/n$ stuck at the beginning of the expression, but the core idea is the same.

9 Summary and looking ahead

To recap, for **training** in Logistic Regression:

1. Formulate the model as

$$M(x) = \frac{1}{1 + e^{-w^T x}}$$

2. Minimize the binary cross-entropy loss with respect to w.

For **inference**:

1. Find $M(x_i)$ and interpret it as $P(y_i = 1|x_i)$.
2. If you're only interested in the class label, round $M(x_i)$ to the nearest integer – which will give you a 0 or 1 label.

Another thing you can explore is the Maximum Likelihood Estimation formulation of Linear Regression. Read up on it from various sources and give it a shot on your own!