# CS 181 Spring 2017 Section 3 Notes
# (Bayesian Linear Regression, Classification)

## 1 Bayesian Linear Regression

### 1.1 Parameter Distributions

Let $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$. Consider the generative model

$$y_i \sim \mathcal{N}(\mathbf{w}^\top \mathbf{x}_i, \beta^{-1}) \tag{1}$$

The likelihood of the data has the form:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \beta^{-1}\mathbf{I}) \tag{2}$$

Put conjugate prior on the weights as (assume precision $\beta^{-1}$ known):

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0) \tag{3}$$

We want a posterior distribution on $\mathbf{w}$ by using Bayes's Theorem, which states that:

$$p(\mathbf{w}|D) \propto p(D|\mathbf{w})p(\mathbf{w}) \tag{4}$$

It turns out (see practice question) that our posterior after $n$ examples is also Gaussian:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_n, \mathbf{S}_n) \tag{5}$$

where

$$\mathbf{S}_n = \left(\mathbf{S}_0^{-1} + \beta\mathbf{X}^\top\mathbf{X}\right)^{-1} \tag{6}$$

$$\mathbf{m}_n = \mathbf{S}_n(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\mathbf{X}^\top\mathbf{y}) \tag{7}$$

### 1.2 Posterior Predictive Distributions

We don't just want to find the value of $\mathbf{w}$– we want to be able to predict $y$ for new values of the input data. Let $\mathbf{x}$ denote one such new data point.

For now, we assume that $p(\mathbf{w}) \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$. Marginalizing over $\mathbf{w}$, we have that

$$p(y|\mathbf{x}, D) = \int_{\mathbf{w}} p(y|\mathbf{x}, \mathbf{w})p(\mathbf{w}|D)d\mathbf{w} \tag{8}$$

$$= \int_{\mathbf{w}} \mathcal{N}(y|\mathbf{w}^\top\mathbf{x}, \beta^{-1})\mathcal{N}(\mathbf{w}|\mathbf{m}_n, \mathbf{S}_n)d\mathbf{w} \tag{9}$$

Since each of the terms on the right hand side follows a normal distribution, we can use some math (see lecture 5, slide 33) to find that

$$p(y|\mathbf{x}, D) = \mathcal{N}(y|\mathbf{x}^\top\mathbf{m}_n, \mathbf{x}^\top\mathbf{S}_n\mathbf{x} + \beta^{-1}) \tag{10}$$

## 2 Practice Questions

1. **Posterior Weight Distribution By Completing the Square (Bishop 3.7)**

We know from (3.10) in Bishop that the likelihood can be written as

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \mathbf{w}) &= \prod_{i=1}^{n} \mathcal{N}(y_i|\mathbf{w}^\top \mathbf{x}_i, \beta^{-1}) \\ &\propto \exp\left(-\frac{\beta}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w})\right) \end{aligned}$$

where precision $\beta = \frac{1}{\sigma^2}$ and in the second line above we have ignored the Gaussian normalization constants. By completing the square, show that with a prior distribution on $\mathbf{w}$ given by $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)$, the posterior distribution $p(\mathbf{w}|D)$ is given by

$$p(\mathbf{w}|D) = \mathcal{N}(\mathbf{w}|\mathbf{m}_n, \mathbf{S}_n)$$

where

$$\begin{aligned} \mathbf{m}_n &= \mathbf{S}_n(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\mathbf{X}^\top\mathbf{y}) \\ \mathbf{S}_n &= \left(\mathbf{S}_0^{-1} + \beta\mathbf{X}^\top\mathbf{X}\right)^{-1} \end{aligned}$$

2. **Bayesian Updates in Linear Regression (Bishop 3.8)**

Suppose we have the standard Bayesian linear regression model and we have already observed $n$ data points, so the posterior distribution is

$$p(\mathbf{w}|D) = \mathcal{N}(\mathbf{w}|\mathbf{m}_n, \mathbf{S}_n)$$

where

$$\mathbf{m}_n = \mathbf{S}_n(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\mathbf{X}^\top\mathbf{y})$$
$$\mathbf{S}_n = \left(\mathbf{S}_0^{-1} + \beta\mathbf{X}^\top\mathbf{X}\right)^{-1}$$

Suppose we observe a new data point $(\mathbf{x}_{n+1}, y_{n+1})$. Show that the resulting posterior distribution is of the same form with $\mathbf{m}_{n+1}$ and $\mathbf{S}_{n+1}$.

# 3 Classification

The goal in classification is to take an input vector $\mathbf{x}$ and assign it to one of $c$ discrete classes, $C_k$, where $k = 1, \ldots, c$. The input space is thus divided into **decision regions** whose boundaries are called **decision boundaries or surfaces**.

## 3.1 Binary Linear Classification

A discriminant function is one that directly assigns each vector $\mathbf{x}$ to a specific class. We first assume two classes, i.e. our responses are binary and $c = 2$. Linear classification seeks to divide the 2 classes by a linear separator in the feature space– if $m = 2$ the separator is a line; if $m = 3$ the separator is a plane; for general $m$ the separator is a $(m-1)$-dimensional hyperplane.

The simplest representation of a linear discriminant function is obtained by taking a linear function of the input vector as such:

$$h(\mathbf{x}; \mathbf{w}, w_0) = \mathbf{w}^\top \mathbf{x} + w_0$$

The corresponding decision boundary is defined by the relation $h(\mathbf{x}; \mathbf{w}, w_0) = 0$, which corresponds to the $(m-1)$-dimensional hyperplane within the $d$-dimensional input space.

The corresponding classifier will predict $\hat{y} = 1$ if $h(\mathbf{x}; \mathbf{w}, w_0) > 0$, and predict $\hat{y} = -1$ otherwise.

Weight vector $\mathbf{w}$ is orthogonal to every vector lying within the decision surface, and so $\mathbf{w}$ determines the orientation of the decision boundary. Mathematically, consider a vector $\mathbf{x}_1 - \mathbf{x}_2$ that lies on the boundary. We have:

$$\mathbf{w}^\top(\mathbf{x}_1 - \mathbf{x}_2) = -w_0 - (-w_0) = 0$$

From this, we see that $\mathbf{w}$ is orthogonal to a vector that lies on the decision boundary (since the inner product is zero).

Moreover, $w_0$ (called the bias or threshold), determines the location of the decision boundary. In particular, we can ask for the normalized distance from a point $\mathbf{x}$ on the boundary to the origin. By definition, the normalized distance from the origin to a point $\mathbf{x}$ on the boundary is $\frac{\mathbf{w}^\top \mathbf{x}}{\|\mathbf{w}\|}$, where we consider the inner product between $\mathbf{w}$ (the orthogonal vector to the plane) and the point $\mathbf{x}$. We divide by the norm $\|\mathbf{w}\|$, where $\|\mathbf{w}\|^2 = \mathbf{w}^\top \mathbf{w}$, to get the normalized distance. Noting that $h(\mathbf{x}; \mathbf{w}, w_0) = \mathbf{w}^\top \mathbf{x} + w_0 = 0$, since it is on the decision boundary, we substitute $\mathbf{w}^\top \mathbf{x} = -w_0$ to get

$$\frac{\mathbf{w}^\top \mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|},$$

and see that $w_0$ together with $\|\mathbf{w}\|$ define the location of the decision boundary.
The input space can also be transformed through a basis technique, so that

$$h(\mathbf{x}; \mathbf{w}, w_0) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) + w_0,$$

and this can help with linear separability.

### 3.2 Perceptron Algorithm

An important way to train a linear discriminant model is via the perceptron algorithm.

This works for a two-class model. Rather than a $0/1$ error function (or sum-squared error), the perceptron algorithm adopts an alternative error function known as the rectified linear activation activation (ReLU), given by:

$$f_{relu}(z) = \begin{cases} z & z > 0 \\ 0 & o.w. \end{cases} = \max\{0, z\}$$

For a binary classification problem with classes 1 and -1, note that if $h(\mathbf{x}; \mathbf{w}, w_0) > 0$ (and thus $\hat{y} = 1$), then there is a classification error if the correct label is -1. Similarly (and ignoring when a point is right on the boundary) if $h(\mathbf{x}; \mathbf{w}, w_0) < 0$ (and thus $\hat{y} = -1$), then there is a classification error if the correct label is 1. For this reason, when the product $-h(\mathbf{x}_i; \mathbf{w}, w_0)y_i < 0$ then there is a classification error.

The perception loss function is defined as:

$$\begin{aligned} \mathcal{L}(\mathbf{w}) &= \sum_{i=1}^{n} f_{relu}(-h(\mathbf{x}_i; \mathbf{w}, w_0)y_i) \\ &= -\sum_{i=1:y_i \neq \hat{y}_i}^{n} (\mathbf{w}^\top \mathbf{x}_i + w_0)y_i \end{aligned}$$

The first term takes the sum over all training examples of the ReLU function applied to $-h(\mathbf{x}_i; \mathbf{w}, w_0)y_i$. In particular, when there is a misclassified example, then this value is positive and it counts as a loss. Equivalently, we can simply write this as the negated sum over all misclassified examples of $h(\mathbf{x}_i; \mathbf{w}, w_0)y_i$.

This loss function has a gradient that is easier to work with than if we had used a $0/1$ error function, and we can now apply stochastic gradient descent. This keeps doing a gradient update on weights for one of the currently misclassifed examples. The change in weight vector from step $\tau$ to $\tau + 1$ is given by the following iteration on an incorrect example:

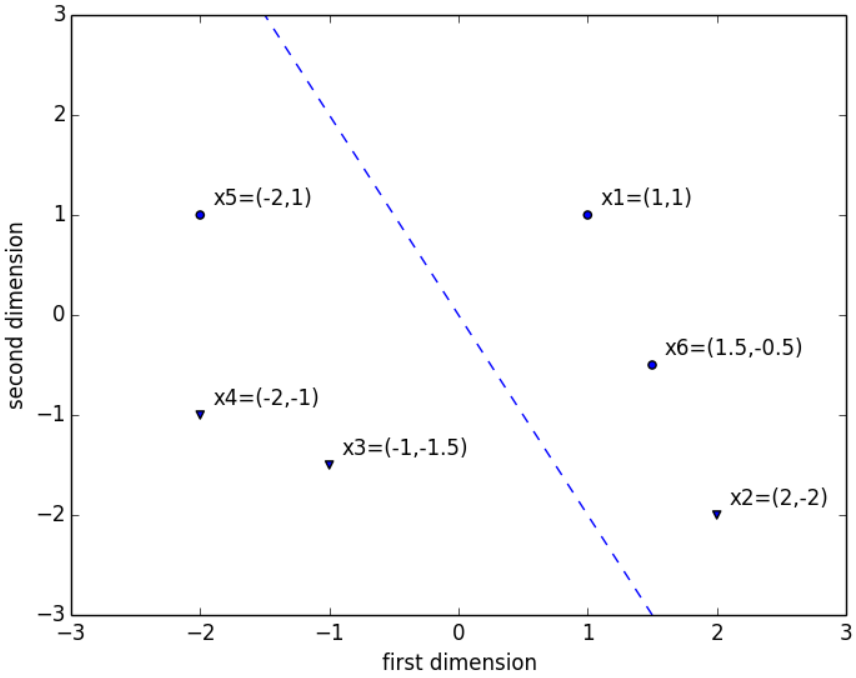
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \frac{\partial}{\partial \mathbf{w}} \mathcal{L}^{(i)}(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta y_i \mathbf{x}_i,$$

where $\eta$ is the learning rate parameter. Note that as the weight vector evolves during training, the set of examples that are misclassified will also change.

### 3.3 Example of Perceptron Algorithm

Let the initial values be $\eta = 0.2$[†], $\mathbf{w} = \binom{w_1}{w_2} = \binom{1}{0.5}$, $w_0 = 0$. The data $\{\mathbf{x}_i\}_{i=1}^n, \mathbf{x}_i \in \mathbf{R}^2$ and initial separation boundary are illustrated below. We choose the circles to belong to class 1 where (with $y_i = 1$) and the triangles to belong to class 2 (with $y_i = -1$).

[†]For the perceptron, $\eta = 1$ almost all the time, but we set it differently for the sake of the exercise




*Left as an exercise for the students.*