Statistical Machine Learning: Asssignment 3

Joris van Vugt, s4279859

November 25, 2016

Code snippets are in Julia

Exercise 1 – Bayesian linear regression

1. The predictive distrubtion after observing these two data points is

$$p(t|x, \boldsymbol{x}, \boldsymbol{t}) = \mathcal{N}(t|m(x), s^2(x)).$$

Using

$$\begin{aligned}\boldsymbol{S}_N^{-1} &= \begin{pmatrix}\alpha & 0\\ 0 & \alpha\end{pmatrix} + N\beta \begin{pmatrix}1 & \bar{\mu}_x\\ \bar{\mu}_x & \bar{\mu}_{xx}\end{pmatrix}\\ &= \begin{pmatrix}2 & 0\\ 0 & 2\end{pmatrix} + 20 \begin{pmatrix}1 & 0.5\\ 0.5 & 0.26\end{pmatrix}\\ &= \begin{pmatrix}22 & 10\\ 10 & 7.2\end{pmatrix}\end{aligned}$$

the mean and standard deviation of the predictive distribution for each x are given by

$$m(x) = N\beta \begin{pmatrix}1 & x\end{pmatrix} \boldsymbol{S}_N \begin{pmatrix}\bar{\mu}_t\\ \bar{\mu}_{xt}\end{pmatrix}$$

$$s^2(x) = \beta^{-1} + \begin{pmatrix}1 & x\end{pmatrix} \boldsymbol{S}_N \begin{pmatrix}1\\ x\end{pmatrix}$$

2. Using the formulas listed above, we can come up with the following code

```
function calc_pred_dist(x, xs, t, alpha, beta)
    N = length(xs)
    S_N = ([alpha 0; 0 alpha] + N*beta*[1 mean(xs); mean(xs) mean(xs .* xs)])^-1
    mx = N*beta*[1 x]*S_N*[mean(t); mean(xs.*t);]
    s2x = beta^-1 + [1 x]*S_N*[1; x]
    mx[1], s2x[1]
end
```

See Figure 1. The main difference with Figure 3.8 from Bishop is the shape of the area of the standard deviation. This is due to the choice in basis functions. In Bishop, gaussian basis functions were used, in contrast with the constant and linear basis functions used here.

3. The probability distribution over the weights is given by

$$p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{x}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_N, \boldsymbol{S}_N)$$
$$\boldsymbol{m}_N = \beta \boldsymbol{S}_N \boldsymbol{\phi}^T \boldsymbol{t} = N\beta \boldsymbol{S}_N \begin{pmatrix} \bar{\mu}_t \\ \bar{\mu}_x t \end{pmatrix}$$

and can be translated to the following procedure.

```
function p_w(x, t, alpha, beta)
    N = length(x)
    S = ([alpha 0; 0 alpha] + N*beta*[1 mean(x); mean(x) mean(x.*x)])^-1
    m = N*beta*S*[mean(t); mean(x.*t)]
    m, S
end
```

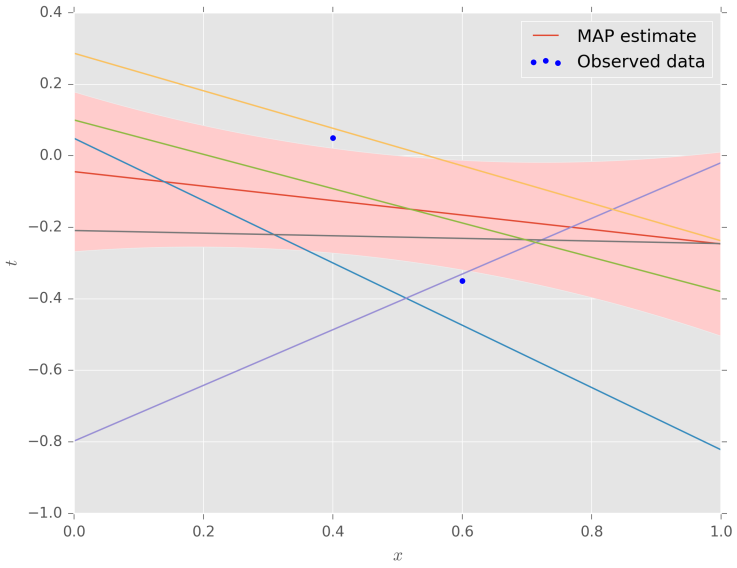

Figure 1: The blue points indicate the observed data points. The red line is the mean of the predictive distribution (i.e. the MAP estimate). The pink area indicates one standard deviation above and below the mean. The 5 other lines are functions sampled according to $p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{x})$. See Figure 2

Functions can be sampled with the following code

```
function f(x, w)
    [1 x] * w
end

m_N, S_N = p_w(x, t, alpha, beta)
weights_dist = MvNormal(m_N, S_N)
for _ in 1:5
    w = rand(weights_dist)
    y = [f(v, w) for v in vs]
    plot(vs, y)
end
```

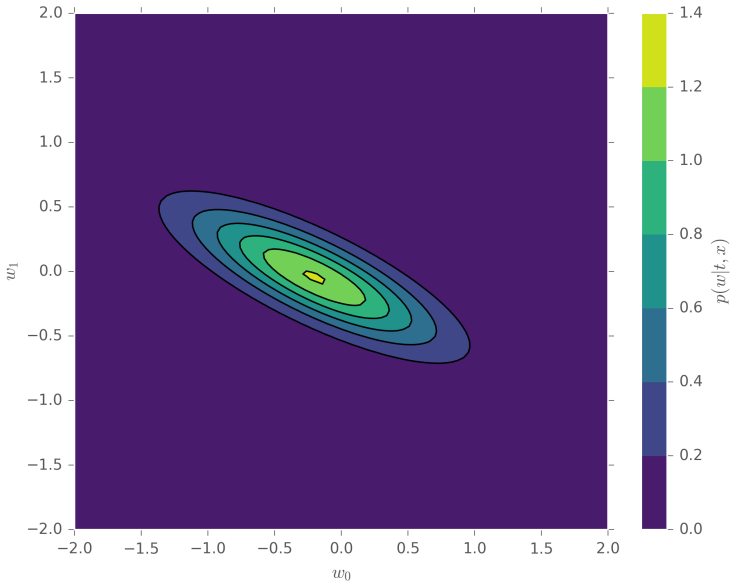

Figure 2: The probability distribution over the weights after observing two datapoints. Assuming the observed data points were not influenced by any noise, the weights should be $w_1 = \frac{-0.35-0.05}{0.6-0.4} = -2$ and $t = w_0 + w_1 x \Rightarrow w_0 = t - w_1 x = 0.05 + 2 * 0.4 = 0.85$

Exercise 2 – Logistic Regression

Part 1 – The IRLS algorithm

1. The expression for minimizing $f(x) = \sin(x)$ using the Newton-Raphson method is

$$\begin{aligned} x^{(n+1)} &= x^{(n)} - \frac{f'(x^{(n)})}{f''(x^{(n)})} \\ &= x^{(n)} + \frac{\cos(x^{(n)})}{\sin(x^{(n)})}. \end{aligned}$$

Which translates to the following code:

```
function nr_sin(x)
    x + cos(x)/sin(x)
end
```

For $x^{(0)} = 1$, the algorithm converges to the maximum in 3 steps. For $x^{(0)} = -1$, the algorithm coverges to a minimum in 3 steps.

2. The Newton-Raphson update rule for logistic regression is

$$\begin{aligned} \boldsymbol{w}^{(\text{new})} &= (\boldsymbol{\Phi}^T \boldsymbol{R} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{R} \mathbf{z} \\ \mathbf{z} &= \boldsymbol{\Phi} \boldsymbol{w}^{(\text{old})} - \boldsymbol{R}^{-1}(\mathbf{y} - \mathbf{t}). \end{aligned}$$

From this, we can come up with the following Julia implementation

```
function logistic_regression(phi, w, t, iterations=5)
    N = length(t)
    for _ in 1:iterations
        y = reshape(1. ./ (1+exp(-w' * phi)), N)
        R = diagm(y .* (1-y))
        z = phi' * w - inv(R) * (y - t)
        w = inv(phi*R*phi') * phi * R * z
    end
    w
end
```

After 4 iterations, the weights are approximately $[9.8, -21.7] \approx \hat{\boldsymbol{w}}^T$. The decision boundary can be found by solving the following equation for ϕ

$$
\begin{aligned}
p(C_1|\boldsymbol{\phi}) &= \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{\phi}}} \\
0.5 &= \frac{1}{1+e^{-[9.8,-21.7]^T[1,\phi]}} \\
0.5 &= \frac{1}{1+e^{-9.8+21.7\phi}} \\
2 &= 1+e^{-9.8+21.7\phi} \\
1 &= e^{-9.8+21.7\phi} \\
\ln(1) &= -9.8+21.7\phi \\
0 &= -9.8+21.7\phi \\
21.7\phi &= 9.8 \\
\phi &= \frac{9.8}{21.7} \approx 0.45
\end{aligned}
$$

Part 2 – Two-class classification using logistic regression

1. We can generate a plot with the following code:

```
data = readdlm("a010_irlsdata.txt")
X = data[:, 1:2]
C = data[:, 3]

scatter(X[C .== 0, 1], X[C .== 0, 2], c="b", label=L"$C=0$")
scatter(X[C .== 1, 1], X[C .== 1, 2], c="r", label=L"$C=1$")
```

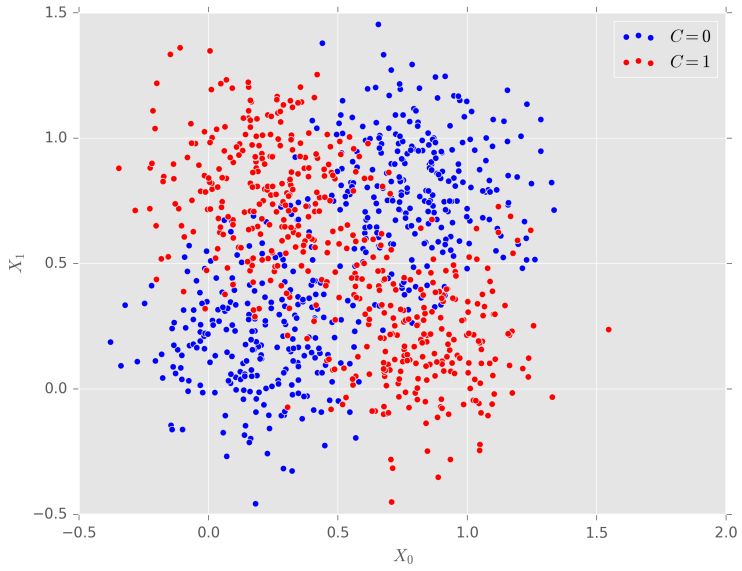

Figure 3: A scatter plot of the two classes

I think logistic regression will have difficulties classifying this data. There is a large area where the both classes are represented equally. However, there are also some areas which strictly belong to a single class.

2. Before optimization, the class probabilities are

$$\begin{aligned} p(C_1|\boldsymbol{\phi}) &= \frac{1}{1+e^{[0,0,0]^T\boldsymbol{\phi}}} \\ &= \frac{1}{1+e^0} = \frac{1}{2}. \end{aligned}$$

I.e., both classes are equiprobable.

3. The cross-entropy before optimization is 693.15. After optimization, the cross entropy decreases only slightly to 692.97. Figure 4 shows that the model is never certain of a class, as the probabilities are always around 0.5. Because only linear basis functions are used, the decision boundary is also linear. However, the data is not linearly seperable.

```
function cross_entropy(y, t)
    -sum(t.*log(y) + (1-t).*log(1-y))
end
```

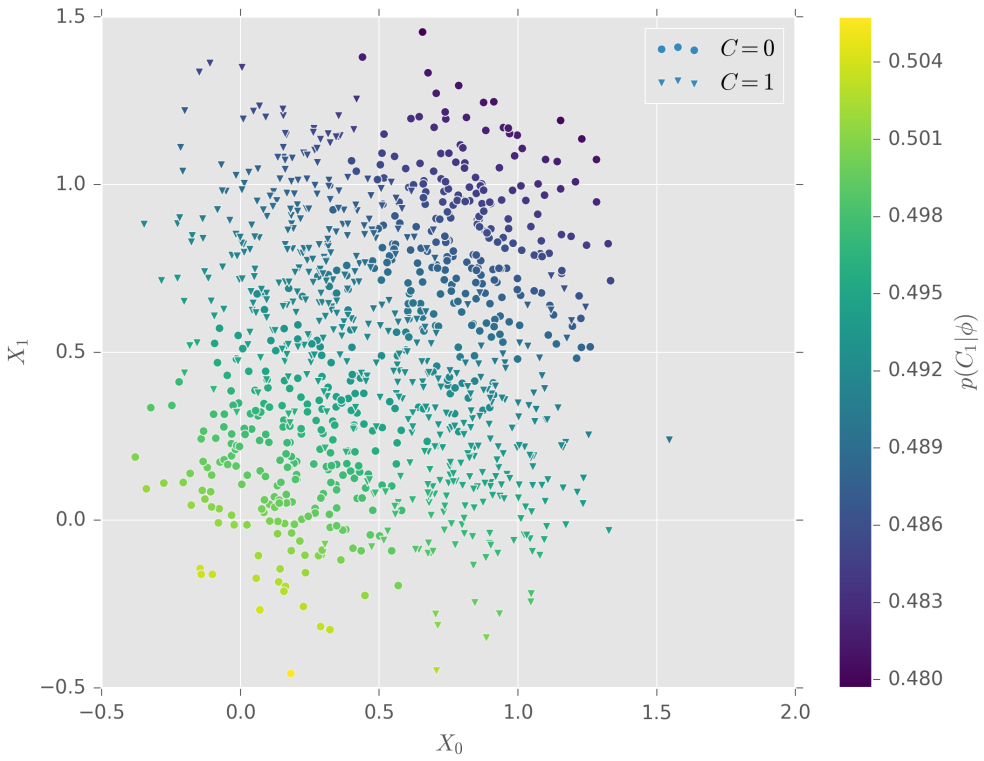

Figure 4: Class probabilities $P(C = 1|X_n)$ after optimization.

4. I think logistic regression will be a lot more succesful in this case. Both gaussians have the same covariance matrix, so the decision boundary will be linear. Both classes can be seperated fairly well in this case.

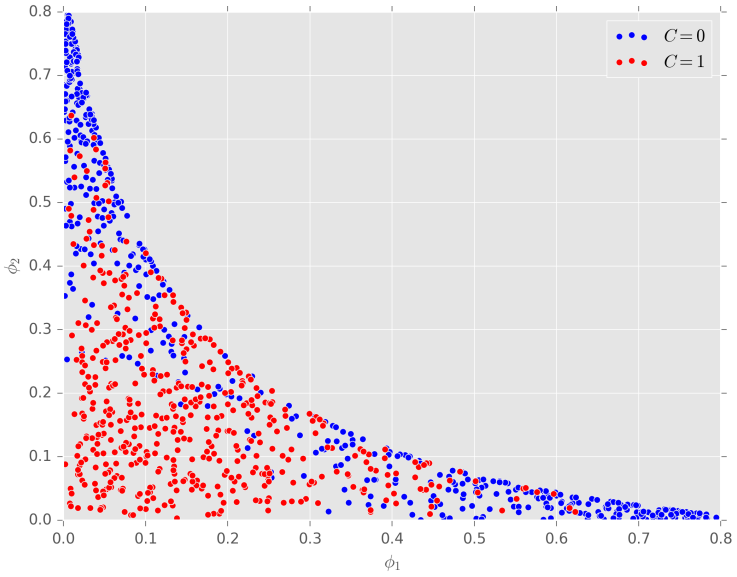

Figure 5: A scatter plot of the data in the feature domain using two Gaussian basis functions.

The following code was used to create the feature matrix

```
sigma = diagm([0.2, 0.2])
gauss_1 = MvNormal([0, 0], sigma)
gauss_2 = MvNormal([1, 1], sigma)
phi = zeros(3, N)
for i in 1:N
    phi[1, i] = 1
    phi[2, i] = pdf(gauss_1, X[i, :])
    phi[3, i] = pdf(gauss_2, X[i, :])
end
```

5. After 5 iterations, the cross-entropy is 346.50. This approach works a lot better. Only the middle region still has a lot of uncertainty. All points in the outer edges are classified correctly. Figure 7 shows that the model has indeed found a linear decision boundary in the feature domain.

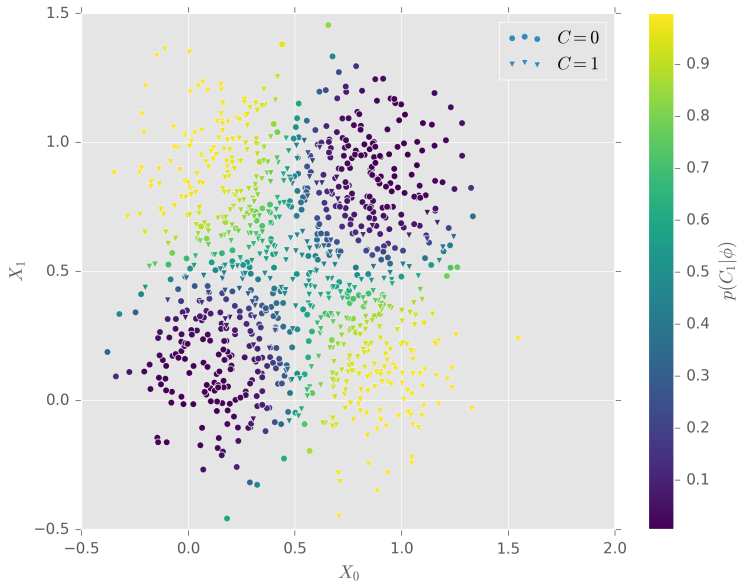

Figure 6: Class probabilities $P(C = 1|X_n)$ after optimization using Gaussian basis functions.

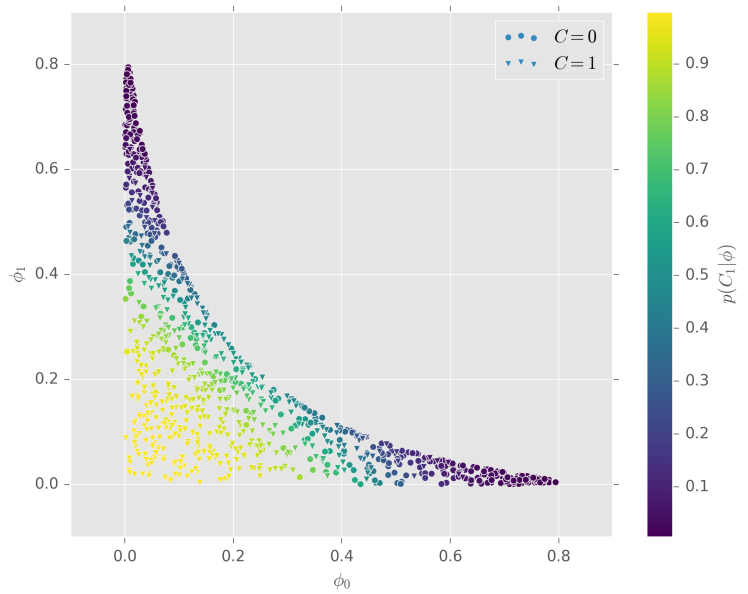

Figure 7: Class probabilities $P(C = 1|X_n)$ in the feature domain after optimization using Gaussian basis functions.