Statistical Methods: Portfolio 3

By Henry Bourne

1 Linear Classifiers

In the portfolio 1 we saw how to conduct binary classification, now we will look at how to conduct **multi-class classification**. This is where we have an input $\boldsymbol{x} \in \mathbb{R}^d$ and an output $y \in \{1, ..., K\}$.

The geometry of the problem in this case is more complicated than we had with binary classification, we can no longer simply check the sign of a single $f(\boldsymbol{x})$ to classify. We could try introducing multiple functions. Let's say we have 3 classes, we could try introducing another function so now we have f_1 and f_2, we could then perform classification by checking the signs of both f_1 and f_2 and have this dictate the class. However, this can get confusing as for example in this case we have 4 possible outcomes, $\{+, -\}^2$, for the signs of our f's, however, we only have 3 classes. What about if we introduce pairwise binary classifiers, ie. we have $f_{i,j}$ classifies a point as either i or j and we have such a function for any pair of classes, then we classify a point as the majority vote given by the binary classifiers. The problem here is that all the classifiers may disagree in which case there is no majority vote.

Rather than relying on the sign of a function f to make predictions lets instead fit a vector valued function $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^K$, where K is the number of classes. Given an input $\boldsymbol{x}$ our prediction is $\hat{k} = \text{argmax}_k \, f^{(k)}(\boldsymbol{x})$. Note that this no longer has a simple geometric interpretation anymore.

1.1 Least Squares Classifier

We will first define the least squares classifier in the binary classification case and after extend it to the multi-class case:

Definition 1.1 ***Least Squares Binary Classifier:***
We first perform LS on the data, ie. find:

$$\boldsymbol{w}_{LS} := \underset{\boldsymbol{w}}{\text{argmin}} \sum_{i \in D} [y_i - f(\boldsymbol{x}_i; \boldsymbol{w})]^2 \tag{1}$$

Then we can find the predicted label $\hat{y} := sign(f(\boldsymbol{x}_i; \boldsymbol{w}_{LS}))$

Note that we can also use a feature transform for f as well, which would allow us to fit more complex classifiers, as data not separable in the original space may be separable in the feature space. In the multi-class case:

Definition 1.2 ***Multi-class LS classification:***
We use a ***one-hot encoding*** *which is where we replace* $y_i = k$ *in our data with* $\boldsymbol{t}_i \in \{0, 1\}^K$ *where all entries are 0 bar* $t_i^{(k)} = 1$. *Then we have:*

$$\boldsymbol{W}_{LS} := \underset{\boldsymbol{W}}{\text{argmin}} \sum_{i \in D} ||\boldsymbol{t}_i - f(\boldsymbol{x}_i; \boldsymbol{W})||^2 \tag{2}$$

where $\boldsymbol{W} \in \mathbb{R}^{(d+1) \times K}$, $f(\boldsymbol{x}; \boldsymbol{W}) = \boldsymbol{W}^T \tilde{\boldsymbol{x}}$ *and* $\tilde{\boldsymbol{x}} := [\boldsymbol{x}^T, 1]^T$.
Then our prediction $\hat{y} = \text{argmax}_k \, f(\boldsymbol{x}; \boldsymbol{W})^{(k)} = \text{argmax}_k (\boldsymbol{w}_{LS}^{(k)})^T \tilde{\boldsymbol{x}}$, *where* $\boldsymbol{w}^{(k)}$ *is the k-th column of* $\boldsymbol{W}$.

Although this method can work the square loss tends not to make sense in a classification task, as a point far away from the boundary (fit without that point) can dramatically affect our decision boundary, even if it is correctly classified by the decision boundary (fit without that point). Also, unlike with LS regression, LS classification lacks a probabilistic interpretation.

1.2 Fisher Discriminant Analysis (FDA)

Note that taking the inner product $\langle \boldsymbol{w}, \boldsymbol{x} \rangle$ embeds $\boldsymbol{x}$ onto a one-dimensional line along the $\boldsymbol{w}$ direction. We can say $\boldsymbol{w}$ gives a good embedding if the $\boldsymbol{x}$ it embeds are close together in the embedding if they are from the same class but far apart if they are from different classes. We can define these two properties we want from our embedding as:

Definition 1.3 ***Within-class Scatterness:***
*For embedding $\boldsymbol{w}^T\boldsymbol{x}$, the **embedding centre** for class k is:*

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i, y_i = k} \boldsymbol{w}^T \boldsymbol{x}_i \tag{3}$$

then the within-class scatterness of class k is:

$$s_{\boldsymbol{w},k} := \sum_{i, y_i = k} (\boldsymbol{w}^T \boldsymbol{x}_i - \hat{\mu}_k)^2 \tag{4}$$

Definition 1.4 ***Between-class Scatterness:***
*The **embedded dataset centre** is:*

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{w}^T \boldsymbol{x}_i \tag{5}$$

then teh between-class scatterness is:

$$s_{b,k} = n_k (\hat{\mu}_k - \hat{\mu})^2 \tag{6}$$

note the n_k is needed to make $s_{b,k}$ the same scale as $s_{w,k}$.

Then ideally we would like to maximize the between-class scatterness and minimize the within-class scatterness for all the classes, which is what the following does:

Definition 1.5 ***Fisher Discriminant Analysis (FDA):***

$$\max_{\boldsymbol{w}} [\sum_k s_{b,k} / \sum_k s_{\boldsymbol{w},k}] \tag{7}$$

if $K = 2$ then this has a simple solution: $\boldsymbol{w} := \boldsymbol{S}_{\boldsymbol{w}}^{-1}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)$, where $\boldsymbol{S}_{\boldsymbol{w}} := \sum_{k=1}^{K} \boldsymbol{S}_k$ and $\boldsymbol{S}_k$ is the sample covariance matrix of class k times n_k. However, note that the FDA does not learn a decision function f, the $\boldsymbol{w}_{FDA}$ obtained cannot be used directly by the prediction function to make a prediction. This is because (eg. in the binary case) $f(\boldsymbol{x}; \boldsymbol{w}_{FDA}) > 0$ does not mean that $\boldsymbol{x}$ is predicted as the positive class. The FDA also doesn't care about classification accuracy, ie. minimizing the FP or FN rate.

1.3 Probabilistic (Generative) Classifiers

How do we put a classification problem under a probabilistic framework? Let's say we would like to **minimize the expected loss:** $\hat{y} := \text{argmin}_{y_0} \mathbb{E}_{p(y|\boldsymbol{x})}[L(y, y_0)|\boldsymbol{x}]$. To do this we need $p(y|\boldsymbol{x})$. The **discriminative approach** would be to infer $p(y|\boldsymbol{x})$ directly. Here, we will look at the **generative approach** which is where we note that $p(y|\boldsymbol{x}) \propto p(\boldsymbol{x}|y)p(y)$ and we infer $p(\boldsymbol{x}|y)$ [1].

1.3.1 Continuous input

If $\boldsymbol{x}$ is a continuous variable then the Multi-Variate Normal (MVN) is a natural choice for the model of $p(\boldsymbol{x}|y)$, ie. we have $p(\boldsymbol{x}|y = k; \boldsymbol{w}) := N_{\boldsymbol{x}}(\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})$ (assuming iid data and that all classes have same covariance matrix). We can write the likelihood of the parameters over the data, D, as: $p(D|\boldsymbol{w}) = \prod_{i \in D} p(\boldsymbol{x}_i, y_i|\boldsymbol{w}) = \prod_{i \in D} p(\boldsymbol{x}_i|y_i; \boldsymbol{w})p(y_i) = \prod_{i \in D} N_{\boldsymbol{x}_i}(\boldsymbol{\mu}_{y_i}; \boldsymbol{\Sigma})p(y_i)$. Therefore our MLE's are: $\hat{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K, \boldsymbol{\Sigma}} := \text{argmax}_{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K, \hat{\boldsymbol{\Sigma}}} \sum_{i \in D} \log[N_{\boldsymbol{x}_i}(\boldsymbol{\mu}_{y_i}; \boldsymbol{\Sigma})p(y_i)]$. If we plug in our estimates for $p(y_i = k)$ which are $\frac{n_k}{n}$, then we can find the MLE for $\hat{\boldsymbol{\mu}}_k := \frac{1}{n_k} \sum_{i \in D, y_i = k} \boldsymbol{x}_i$. We can then plug in our $\hat{\boldsymbol{\mu}}_k$'s to work out: $\hat{\boldsymbol{\Sigma}} := \sum_{k=1,...,K} \frac{n_k}{n} \frac{1}{n_k} \sum_{i \in D, y_i = k} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^T$,

[1] $p(y)$ is just the probability of the class being class y

where we notice that in red is the MLE of the covariance of individual classes. Our prediction is $\hat{y} := \text{argmax}_y\, p(y|\boldsymbol{x}; \hat{\boldsymbol{w}}) \propto p(\boldsymbol{x}|y; \hat{\boldsymbol{w}})p(y)$. We can also prove that when using the shared covariance matrix MVN model, the decision boundary is piecewise-linear (Section A.1).

What if we assume that for each class k there are different covariance matrices? then the MLE reduces to estimating individual $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$. Note that in this case the decision boundary is no longer linear.

1.3.2 Discrete input

Here we will look at the case where $\boldsymbol{x}$ is discrete. Let $\boldsymbol{x} := [x^{(1)}, ..., x^{(d)}]^T$ and assume $x^{(1)}, ..., x^{(d)}$ follow a multinomial distribution, where each $x^{(i)}$ is a quantity for a feature. Using Bayes rule we have that $p(y = k|\boldsymbol{x}) = (p(\boldsymbol{x}|y = k) \cdot p(y = k))/p(\boldsymbol{x})$. We now introduce our "naive assumptions" which are that all the features in $\boldsymbol{x}$ are mutually independent, conditional on $y = k$. Using these assumptions we can write $p(y = k|\boldsymbol{x}) \propto p(y = k) \prod_{i=1}^{n} p(x_i|y = k)$ [2]. Therefore $p(\boldsymbol{x} = \boldsymbol{x}_0|y = k) \propto \prod_{i=1,...,d} \beta(i|y = k)^{x_0^{(i)}}$, for some $\boldsymbol{x}_0$ where $\beta(i|y = k)$ is the probability feature i occurs in class k. It is easy to estimate this quantity:

$$\beta(i|y = k) \approx \frac{\sum_{j \in D, y_j = k} x_j^{(i)}}{\sum_{j \in D, y_j = k} \sum_{l=1}^{d} x_j^{(l)}} \tag{8}$$

Then our prediction is $\hat{y} := \text{argmax}_y\, p(\boldsymbol{x} = \boldsymbol{x}_0|y) \cdot p(y)$ where as before $p(y = k)$ can be estimated as $\frac{n_k}{n}$ and note that $\beta(i|y = k)$ can be estimated by counting. This is called **Naive Bayes classification"**.

2 Discriminative Classifiers

Before we looked at Generative Classifiers, now we will look at discriminative classifiers. **Discriminative classifiers** aim to infer $p(y|\boldsymbol{x})$ given dataset D. The process normally looks like so:

1. Make a model assumption $p(y|\boldsymbol{x}; \boldsymbol{w})$.
2. Construct the likelihood function $p(D|\boldsymbol{w})$.
3. Estimate the parameters (eg. using MLE, MAP, full probabilistic approach, etc.).

The first question we ask is: what model should we use? Bayes rule says:

$$p(y|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y)p(y)}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|y)p(y)}{\sum_{y'} p(\boldsymbol{x}, y')} = \frac{p(\boldsymbol{x}|y)p(y)}{\sum_{y'} p(\boldsymbol{x}|y')p(y')} \tag{9}$$

Notice that we are now representing $p(y|\boldsymbol{x})$ using $p(\boldsymbol{x}|y)$. Suppose $y \in \{-1, 1\}$ then $p(y = 1|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=1)p(y=1)}{p(\boldsymbol{x}|y'=1)p(y'=1)+p(\boldsymbol{x}|y'=-1)p(y'=-1)}$. Assume $p(\boldsymbol{x}|y)p(y) > 0, \forall \boldsymbol{x}, y$ then this equals $1/(1 + \frac{p(\boldsymbol{x}|y=-1)p(y=-1)}{p(\boldsymbol{x}|y=1)p(y=1)})$, ie. we can rewrite $p(y|\boldsymbol{x})$ using the ratio $\frac{p(\boldsymbol{x}|y=-1)p(y=-1)}{p(\boldsymbol{x}|y=1)p(y=1)}$. So in a discriminative learning we model the **density ratio** $\frac{p(\boldsymbol{x}|y=-1)}{p(\boldsymbol{x}|y=1)}$ as opposed to the class density like in generative learning [3]. We will model the log of the inverse of the density ratio (as $f(\boldsymbol{x}; \boldsymbol{w})$) and so in our expression above we can replace the density ratio with $\exp(-f(\boldsymbol{x}; \boldsymbol{w}))$.

The **sigmoid function** is denoted $\sigma(t) := \frac{1}{1+\exp(-t)}$. Then note that $p(y = 1|\boldsymbol{x}; \boldsymbol{w}) = \frac{1}{1+\exp(-f(\boldsymbol{x};\boldsymbol{w}))} = \sigma(f(\boldsymbol{x}; \boldsymbol{w}))$, also note that when $y = -1$ our density ratio is inverse and so we no longer need the minus in front of f, therefore, in general $p(y|\boldsymbol{x}; \boldsymbol{w}) = \sigma(f(\boldsymbol{x}; \boldsymbol{w}) \cdot y)$.

Now, assuming the data, D, is iid., we would like to find the MLE for $p(y|\boldsymbol{x}; \boldsymbol{w})$:

$$\boldsymbol{w}_{MLE} = \underset{\boldsymbol{w}}{\text{argmax}} \log \prod_{i \in D} p(y_i|\boldsymbol{x}_i; \boldsymbol{w}) = \underset{\boldsymbol{w}}{\text{argmax}} \sum_{i \in D} \log p(y_i|\boldsymbol{x}_i; \boldsymbol{w}) = \underset{\boldsymbol{w}}{\text{argmax}} \sum_{i \in D} \log \sigma(f(\boldsymbol{x}_i; \boldsymbol{w}) \cdot y_i) \tag{10}$$

[2] We get this by discarding the $p(\boldsymbol{x})$, which makes it proportional and then using our assumption on $p(\boldsymbol{x}|y = k)$

[3] As a side note: clearly modelling the density ratio requires a lot fewer assumptions on your class densities, so models on class densities will model the density ratio, however models on the density ratio will not necessarily model the class density

This procedure is called **Logistic Regression** (even though it is not a regression!). Note that logistic regression is robust to outliers, unlike the LS classifier, this is because the sigmoid function tapers their affect. As before with the LS classifier we can also fit non-linear classifiers using a feature transform.

Apart from logistic regression there are other methods for estimating $p(y|\boldsymbol{x};\boldsymbol{w})$. For example assuming a prior on $\boldsymbol{w}$ we could use MAP:

$$\boldsymbol{w}_{MAP} = \underset{\boldsymbol{w}}{\text{argmax}} \sum_{i \in D} \log(\sigma(f(\boldsymbol{x}_i;\boldsymbol{w}) \cdot y_i) p(\boldsymbol{w})) = \underset{\boldsymbol{w}}{\text{argmax}} \sum_{i \in D} \log(\sigma(f(\boldsymbol{x}_i;w) \cdot y_i)) + \log(p(\boldsymbol{w})) \quad (11)$$

We can also use the full probabilistic approach [4]:

$$p(y|\boldsymbol{x}) = \int p(y|\boldsymbol{x};\boldsymbol{w}) p(\boldsymbol{w}|D) d\boldsymbol{w} \propto \int p(y|\boldsymbol{x};\boldsymbol{w}) p(D|\boldsymbol{w}) p(\boldsymbol{w}) d\boldsymbol{w} \quad (12)$$

$$\propto \int \sigma(f(\boldsymbol{x}_i;\boldsymbol{w}) \cdot y_i) \prod_{i \in D} \sigma(f(\boldsymbol{x}_i;\boldsymbol{w}) \cdot y_i) p(\boldsymbol{w}) d\boldsymbol{w} \quad (13)$$

What about in the multi-class scenario? we can extend logistic regression to the multi-class scenario, where the density ratio is now:

$$p(y = i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y = i)p(y = i)}{\sum_k p(\boldsymbol{x}|y = k)p(y = k)} \quad (14)$$

We expand further on this in conjunction with one-hot encoding in question B.2.3. Note that unlike in LS, logistic regression does not have a closed form solution and therefore must employ numerical methods to find $\boldsymbol{w}_{MLE}$.

3 Support Vector Machines

In binary classification it might be that there are many decision boundaries we could draw that would correctly classify all the training data, this leaves us asking: what is the "optimal" decision boundary in binary classification? For our decision boundary to be generalizable we want to minimize the error on unseen datasets rather than the training set. We would like a decision boundary that will correctly classify a small perturbation of one of our training data points whilst still correctly classifying all of our training data, ie. we want to maximize the gap (the **margin**) between the decision boundary and the lines given by $f(\boldsymbol{x};\boldsymbol{w}) = 1$ and $f(\boldsymbol{x};\boldsymbol{w}) = -1$, whilst keeping data points on the correct side of the margin.

If we are using decision boundary $f(\boldsymbol{x};\boldsymbol{w}) := \langle \boldsymbol{w}_1, \boldsymbol{x} \rangle + w_0$ then we can calculate the thickness of our margin as $\frac{1}{||\boldsymbol{w}_1||}$. Then we can formulate the following constrained minimization problem: minimize $||\boldsymbol{w}_1||^2$ subject to $\forall i, y_i \cdot f(\boldsymbol{x}_i;\boldsymbol{w}) \geq 1$.

However, in many cases the dataset is not separable and so it is not possible to satisfy our constraint. We can relax these constraints which is what we do in the **soft-margin classifier**: $\min_{\boldsymbol{w},\boldsymbol{\epsilon}} ||\boldsymbol{w}_1||^2 + \sum_i \epsilon^{(i)}$ subject to $\forall i,\ y_i(\langle \boldsymbol{w}_1, \boldsymbol{x}_i \rangle + w_0) + \epsilon^{(i)} \geq 1,\ \epsilon^{(i)} \geq 0$. It turns out that this classifier is actually a convex minimization problem and so every local minimum is a global minimum.

We can use the lagrangian dual (discussed in section A.2) to turn the minimization problem of the soft-margin classifier into: $\max_{\boldsymbol{\lambda}} -\frac{\tilde{\boldsymbol{\lambda}}^T \boldsymbol{X}^T \boldsymbol{X} \tilde{\boldsymbol{\lambda}}}{4} + \langle \boldsymbol{\lambda}, \boldsymbol{1} \rangle$ subject to $0 \leq \lambda_i \leq 1$, $\sum_i \lambda_i y_i = 0$. Note that the constraints here are simpler and that its quadratic wrt. $\boldsymbol{\lambda} \in \mathbb{R}^n$ as opposed to quadratic wrt. $\boldsymbol{w} \in \mathbb{R}^{d+1}$ in our previous optimization problem. Hence, the lagrangian dual is slow when n is large and our previous problem is slow when d is large, also note that it is possible to use kernel methods in solving the lagrangian!

Notice that the border of the (optimal) margins will always pass through some data points, these points can be thought of as resisting the expansion of the margin, hence are called **support vectors**. Hence the name of this method: "Support Vector Machines" (SVM). However, the SVM has some limitations, for example it is important to note the SVM is not a probabilistic classifier as it lacks a probabilistic interpretation. Also note that the computational cost of SVM is high as it involves solving a constrained optimization problem (as opposed to unconstrained). Finally multi-class SVM is non-trivial as SVM is motivated by the geometry of binary classification.

[4] Note that unlike regression using MVN models, we cannot calculate this integral in closed form

Appendices

A Proofs

A.1 We prove the following from Section 1.3.1

Prove that when using the shared covariance matrix MVN model, the decision boundary is piecewise-linear:
The decision boundary for a class k' is given by the following set:

$$\{\boldsymbol{x}|p(y=k|\boldsymbol{x};\hat{\boldsymbol{w}}) = p(y=k'|\boldsymbol{x};\hat{\boldsymbol{w}})\}, \quad \forall k \neq k' \tag{15}$$

We can rewrite this set as follows:

$$\left\{\boldsymbol{x}\middle|\frac{p(\boldsymbol{x}|y=k;\hat{\boldsymbol{w}})p(y=k)}{p(\boldsymbol{x}|y=k';\hat{\boldsymbol{w}})p(y=k')} = 1\right\} \tag{16}$$

Which taking logs can be rewritten as:

$$\{\boldsymbol{x}|\log[p(\boldsymbol{x}|y=k;\hat{\boldsymbol{w}})p(y=k)] - \log[p(\boldsymbol{x}|y=k';\hat{\boldsymbol{w}})p(y=k')] = 0\} \tag{17}$$

$$= \{\boldsymbol{x}|\log[N_{\boldsymbol{x}}(\hat{\boldsymbol{\mu}}_k, \boldsymbol{\Sigma})] + \log[p(y=k)] - \log[N_{\boldsymbol{x}}(\hat{\boldsymbol{\mu}}_{k'}, \boldsymbol{\Sigma})] - \log[p(y=k')] = 0\} \tag{18}$$

$$= \{\boldsymbol{x}|(\Sigma^{-1}\boldsymbol{\mu}_k)^T\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log(p(y=k)) \tag{19}$$

$$- [(\Sigma^{-1}\boldsymbol{\mu}_{k'})^T\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_{k'}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_{k'} + \log(p(y=k'))] = 0\} \tag{20}$$

We notice that the condition above simply says that two linear equations of $\boldsymbol{x}$ are equal to each other. The $\boldsymbol{x}$ that satisfy this must also therefore be described by a linear relationship.

A.2 Obtain the stated optimization problem in section 3 by finding the Lagrangian dual:

First we will define what the lagrangian dual is:

Definition A.1 ***Lagrangian Dual:***
A technique that transforms a constrained problem into a less constrained problem. For a constrained problem:

$$\min_{\theta} f(\theta), \ \textit{subject to}\ g_i(\theta) \leq 0, \forall i \tag{21}$$

We can construct the lagrangian:

$$mathcalL(\boldsymbol{\lambda}) := \min_{\theta} f(\theta) + \sum_i \lambda_i g_i(\theta) \tag{22}$$

where $\lambda_i \geq 0$ are called ***lagrangian multipliers***

We now will write the lagrangian dual for the soft-margin classifier:

$$\mathcal{L}(\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}}) := \min_{\boldsymbol{w},\boldsymbol{\epsilon}} ||\boldsymbol{w}_1||^2 + \sum_i \epsilon_i - \sum_i \lambda_i y_i(\langle \boldsymbol{w}_1, \boldsymbol{x}_i\rangle + w_0) + \lambda_i\epsilon_i - \lambda_i - \sum_i \lambda_i'\epsilon_i \tag{23}$$

with optimality conditions,

$$\boldsymbol{w}' = \frac{\sum_i \lambda_i y_i \boldsymbol{x}_i}{2}, \ \lambda_i + \lambda_i' = 1, \ \sum_i \lambda_i y_i = 0 \tag{24}$$

We will now rewrite this, first note that,

$$||\boldsymbol{w}_1||^2 = \frac{||\sum_i \lambda_i y_i \boldsymbol{x}_i||}{4} \tag{25}$$

And that,

$$-\sum_i \lambda_i y_i \langle \boldsymbol{w}_1, \boldsymbol{x}_i\rangle = -\frac{\langle \sum_i \lambda_i y_i \boldsymbol{x}_i, \sum_i \lambda_i y_i \boldsymbol{x}_i\rangle}{2} \tag{26}$$

and that eq. (25) + eq. (26) is,

$$-\frac{\langle \sum_i \lambda_i y_i \boldsymbol{x}_i, \sum_i \lambda_i y_i \boldsymbol{x}_i \rangle}{4} = -\frac{\tilde{\boldsymbol{\lambda}}^T \boldsymbol{X}^T \boldsymbol{X} \tilde{\boldsymbol{\lambda}}}{4} \tag{27}$$

where $\tilde{\boldsymbol{\lambda}} = [\lambda_1 y_1 \cdots \lambda_n y_n]^T$. Also note that

$$\sum_i \epsilon_i - \sum_i \lambda_i \epsilon_i - \sum_i \lambda_i' \epsilon_i \tag{28}$$

$$= \sum_i \epsilon_i (1 - \lambda_i - \lambda_i') \tag{29}$$

$$= 0 \tag{30}$$

finally note,

$$-\sum_i \lambda_i y_i w_0 = -w_0 \sum_i \lambda_i y_i = 0 \tag{31}$$

Note that $\mathcal{L}(\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}}) =$ eq. (25) + eq. (26) + eq. (28) + eq. (31) $+ \sum_i \lambda_i$, therefore,

$$\mathcal{L}(\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}}) := -\frac{\tilde{\boldsymbol{\lambda}}^T \boldsymbol{X}^T \boldsymbol{X} \tilde{\boldsymbol{\lambda}}}{4} + \sum_i \lambda_i \tag{32}$$

So our lagrangian problem is:

$$\max_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}) := \max_{\boldsymbol{\lambda}} -\frac{\tilde{\boldsymbol{\lambda}} \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{\lambda}}{4} + \langle \tilde{\boldsymbol{\lambda}}, \mathbf{1} \rangle \tag{33}$$

Under constraints:

$$0 \leq \lambda_i \leq 1, \sum_i \lambda_i y_i = 0 \tag{34}$$

where $0 \leq \lambda_i \leq 1$ comes from the fact that,

$$\forall_i : \lambda_i \geq 0, \lambda_i' \geq 0, \text{ and } \lambda_i + \lambda_i' = 1 \tag{35}$$

B Homeworks

B.1 For Section 1

Question B.1.1 ***Proof in section A.1***

Question B.1.2 ***Derive the MLE for the parameters of the multinomial distribution:***
Let $\boldsymbol{p} = [p_1, ..., p_d]^T$, where p_i denotes the probability of event i occurring and d is the number of mutually exclusive events. Then we can write out the likelihood:

$$p(D|\boldsymbol{p}) = \sum_{j=1}^{n} \frac{k!}{x_j^{(1)} \cdots x_j^{(d)}} \prod_{i=1}^{d} p_i^{x_j^{(i)}} = k! \sum_{j=1}^{n} \prod_{i=1}^{d} \frac{p_i^{x_j^{(i)}}}{x_j^{(i)}!} \tag{36}$$

Now taking the log we get:

$$log(p(D|\boldsymbol{p})) = log(k!) + \sum_{j=1}^{n} \sum_{i=1}^{d} x_j^{(i)} log(p_i) - log(x_j^{(i)}!) \tag{37}$$

We must have that $\sum_{i=1}^{d} p_i = 1$, the lagrangian with this constraint is:

$$\mathcal{L}(\boldsymbol{p}, \lambda) = log(p(D|\boldsymbol{p})) + \lambda(1 - \sum_{i=1}^{d} p_i) \tag{38}$$

To find the MLE we now differentiate the lagrangian wrt. p_i:

$$\frac{\partial}{\partial p_i} \mathcal{L}(\boldsymbol{p}, \lambda) = \frac{x_j^{(i)}}{\sum_{j=1}^{n} p_i} - \lambda \tag{39}$$

Setting this equal to zero and solving we get:

$$p_i = \sum_{j=1}^{n} \frac{x_j^{(i)}}{\lambda} \tag{40}$$

Now using our constraint we solve for λ:

$$\sum_{i=1}^{d} p_i = \sum_{i=1}^{d} \sum_{j=1}^{n} \frac{x_j^{(i)}}{\lambda} \tag{41}$$

$$\Rightarrow 1 = \frac{1}{\lambda} \sum_{i=1}^{d} \sum_{j=1}^{n} x_j^{(i)} \tag{42}$$

$$\Rightarrow \lambda = \sum_{j=1}^{n} \sum_{i=1}^{d} x_j^{(i)} \tag{43}$$

Therefore our MLE of the parameters is:

$$\hat{p_i} = \frac{\sum_{j=1}^{n} x_j^{(i)}}{\sum_{j=1}^{n} \sum_{i=1}^{d} x_j^{(i)}} \tag{44}$$

Question B.1.3 ***Explain the Naive Bayes classifier using a maximum likelihood framework:***
Lets say we would like to model a classifier using the multinomial distribution as our model. When choosing to use the multinomial distribution as our model, we "naivly" assume that all the features are mutually independent. We would like to formulate a prediction:

$$\hat{y} := p(y|\boldsymbol{x}; \boldsymbol{p}) \tag{45}$$

Using Bayes rule we have that:

$$p(y|D,\boldsymbol{p}) = \frac{p(D|\boldsymbol{p})p(y)}{p(D)} \tag{46}$$

$$\propto p(y) \cdot p(D|\boldsymbol{p}) \tag{47}$$

Using the previous question we can find using the MLE a $\hat{\boldsymbol{p}}$ that maximizes $p(D|\boldsymbol{p})$, then $p(y)$ can simply be calculated by finding the proportion of the time in the dataset where y equals a certain value. This then allows us to formulate a prediction, that has the maximum likelihood, for the classification of a given input $\boldsymbol{x}$ as long as our "naive" assumptions are true.

B.2 For Section 2

Question B.2.1 ***What are the decision functions given by a binary logistic regression?*** *One of them is $p(y|\boldsymbol{x};\boldsymbol{w}) - \frac{1}{2}$ the other is $\frac{1}{2} - p(y|\boldsymbol{x};\boldsymbol{w})$.*

Question B.2.2 ***Prove that if*** $p(\boldsymbol{x}|y=1)$ ***and*** $p(\boldsymbol{x}|y=-1)$ ***are MVN with shared covariance matrix $\boldsymbol{\Sigma}$ but different means $\boldsymbol{\mu}_+, \boldsymbol{\mu}_-$:***

Sub-Question B.2.2.1 $\exists \boldsymbol{w}^*$ ***such that*** $p(y|\boldsymbol{x}) = \sigma((\langle \boldsymbol{x}, \boldsymbol{w}_1^* \rangle + w_0^*) \cdot y)$***:***
We have that,

$$p(y|\boldsymbol{x};\boldsymbol{w}) = \sigma(f(\boldsymbol{x};\boldsymbol{w}) \cdot y) \tag{48}$$

and that,

$$f(\boldsymbol{x};\boldsymbol{w}) = \log\left(\frac{p(\boldsymbol{x}|y=1)}{p(\boldsymbol{x}|y=-1)}\right) \tag{49}$$

Note that the log ratio of these densities makes up the binary decision boundary for this classification problem. Also recall that in section 1.3.1 we claimed that if $p(\boldsymbol{x}|y=k;\boldsymbol{w}) = N_x(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$ for all classes k (for some covariance matrix $\boldsymbol{\Sigma}$), then the decision boundaries are piecewise linear, ie. in the binary case $\exists \boldsymbol{w}^*$ such that:

$$f(\boldsymbol{x},\boldsymbol{w}) = \langle \boldsymbol{x}, \boldsymbol{w}_1^* \rangle + w_0^* \tag{50}$$

hence,

$$p(y|\boldsymbol{x}) = \sigma((\langle \boldsymbol{x}, \boldsymbol{w}_1^* \rangle + w_0^*) \cdot y) \tag{51}$$

Sub-Question B.2.2.2 Find $\boldsymbol{w}^*$:

$$f(\boldsymbol{x};\boldsymbol{w}) = \log\left(\frac{p(\boldsymbol{x}|y=1)}{p(\boldsymbol{x}|y=-1)}\right) \tag{52}$$

$$= \log\left(\frac{N_{\boldsymbol{x}}(\boldsymbol{\mu}_+, \boldsymbol{\Sigma})}{N_{\boldsymbol{x}}(\boldsymbol{\mu}_-, \boldsymbol{\Sigma})}\right) \tag{53}$$

$$= \log\left(\frac{(2\pi)^{-d/2} det(\boldsymbol{\Sigma})^{-1/2} \exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_+)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_+))}{(2\pi)^{-d/2} det(\boldsymbol{\Sigma})^{-1/2} \exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_-)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_-))}\right) \tag{54}$$

$$= -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_+)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_+) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_-)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_-) \tag{55}$$

$$= (\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_+)^T\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_+^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_+ - (\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_-)^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{\mu}_-^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_- \tag{56}$$

$$= ((\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_+)^T - (\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_-)^T)\boldsymbol{x} + (\frac{1}{2}\boldsymbol{\mu}_-^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_- - \frac{1}{2}\boldsymbol{\mu}_+^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_+) \tag{57}$$

Hence we have found $\boldsymbol{w}^*$.

Question B.2.3 ***What is the probabilistic interpretation of f?:***
First, continuing from the end of section 2 we will describe using multi-class logistic regression with one-hot encoding. Note:

$$f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{W}^T\tilde{\boldsymbol{x}},\ \boldsymbol{w} \in \mathbb{R}^{d\times K},\ \tilde{\boldsymbol{x}} := \begin{bmatrix}\boldsymbol{x}\\ \mathbf{1}\end{bmatrix} \tag{58}$$

For $y_i \in \{1, ..., K\}$ we let $\boldsymbol{t}_i \in \mathbb{R}^K$ be the one-hot encoding for it, further let:

$$\sigma(f, t) := \frac{\exp\langle f, t\rangle}{\sum_k \exp f^{(k)}} \tag{59}$$

Then:

$$\boldsymbol{w}_{MLE} = \underset{\boldsymbol{w}}{\operatorname{argmax}} \sum_{i \in D} \log \sigma(f(\boldsymbol{x}_i; \boldsymbol{w}), \boldsymbol{t}_i) \tag{60}$$

In this scenario whats's the probabilistic interpretation of f? like from which we derived the model of f in the binary logistic regression case in section 2. Recall that in discriminative classifiers we are trying to infer $p(y|\boldsymbol{x}; \boldsymbol{w})$ and that:

$$p(y|\boldsymbol{x}; \boldsymbol{w}) = \frac{p(\boldsymbol{x}|y; \boldsymbol{w})p(y)}{\sum_{y'} p(\boldsymbol{x}|y'; \boldsymbol{w})p(y')} \tag{61}$$

Let $i \in \{1, ..., K\}$, consider $p(y = i|\boldsymbol{x}; \boldsymbol{w})$,

$$p(y = i|\boldsymbol{x}; \boldsymbol{w}) = \frac{p(\boldsymbol{x}|y = i; \boldsymbol{w})p(y = i)}{\sum_{y'} p(\boldsymbol{x}|y'; \boldsymbol{w})p(y')} \tag{62}$$

$$= \frac{1}{1 + \frac{\sum_{y', y' \neq i} p(\boldsymbol{x}|y'; \boldsymbol{w})p(y')}{p(\boldsymbol{x}|y=i; \boldsymbol{w})p(y=i)}} \tag{63}$$

Like in the binary case we can focus on modelling the density ratio, so the probabilistic interpretation of f is:

$$f(\boldsymbol{x}; \boldsymbol{w})^{(i)} = \log \left(\frac{p(\boldsymbol{x}|y = i; \boldsymbol{w})p(y = i)}{\sum_{y', y' \neq i} p(\boldsymbol{x}|y'; \boldsymbol{w})p(y')} \right) \tag{64}$$

We further note that if $\hat{y} := \operatorname{argmax}_y p(y|\boldsymbol{x}; \boldsymbol{w})$, then our predictions will correspond to the multi-class decision rule in definition 1.2, as:

$$\underset{y}{\operatorname{argmax}}\, p(y|\boldsymbol{x}; \boldsymbol{w}) = \underset{y}{\operatorname{argmax}}\, f(\boldsymbol{x}; \boldsymbol{w})^{(y)} \tag{65}$$