1 Linear Classifiers

1.1 Binary vs. Multi-Class Classification

In a binary classification problem we have inputs $\mathbf{x} \in \mathbb{R}^d$ each with a corresponding class $y \in \{-1, +1\}$ and our aim is to find a prediction function $f : \mathbb{R}^d \to \mathbb{R}$ such that for new data $\mathbf{x}^*$ we predict the class -1 if $f(\mathbf{x}^*) < 0$ and class $+1$ if $f(\mathbf{x}^*) > 0$. We call $f(\mathbf{x}) = 0$ the *decision boundary*. Note that if we write $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + w_0$ then the perpendicular distance between $\mathbf{x}$ and $f(\mathbf{x}) = 0$ is given by $\frac{f(\mathbf{x})}{||\mathbf{w}||}$.

We can think of the binary classifier $f(\mathbf{x})$ as splitting $\mathbb{R}^d$ into two distinct regions, these being $R_- := \{\mathbf{x} \in \mathbb{R}^d | f(\mathbf{x}) < 0\}$ and $R_+ := \{\mathbf{x} \in \mathbb{R}^d | f(\mathbf{x}) > 0\}$, however, this geometry gets more complicated when we consider multi-class classification where $y \in \{1, ..., K\} = [K]$.

One approach is to construct $K-1$ different binary classifiers, each of which classifies a class $k \in [K]$ against the rest of the classes $[K] \backslash k$ (the Kth class can be classified through deduction from the other $K - 1$ classifiers), however, this can lead to regions in which classification is indeterminate[1]. Another approach might be to create $\binom{K}{2}$ different binary classifiers (one for each pair of distinct classes) and classify a new data point by the majority vote of these, although this can also lead to indeterminate regions in which the majority vote leads to ties. A better system for multi-class classification would be to fit a function $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^K$ and, for an input $\mathbf{x}^*$, predict the class $k^* = \text{argmax}_k \, f^{(k)}(\mathbf{x}^*)$. We will see this later in the portfolio.

1.2 Least Squares Classifier

We might consider a least squares approach to binary classification, in which we aim to find our prediction function $f(\mathbf{x}; \mathbf{w}_{LS}) = \langle \mathbf{w}_{LS}, \boldsymbol{\phi}(\mathbf{x}) \rangle$ for some feature transform $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^b$ via $\mathbf{w}_{LS} := \text{argmin}_{\mathbf{w}} \sum_{i \in D} (y_i - f(\mathbf{x}_i; \mathbf{w}))^2$ where $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is our dataset and we restrict $y_i \in \{-1, +1\}$. The benefit of using a feature transform $\boldsymbol{\phi}$ here is that even if the data isn't linearly separable in $\mathbb{R}^d$, it may be separable in the feature space created by $\boldsymbol{\phi}$.

We can extend this LS approach to multi-class classification if we use *one-hot encoding*: replace each $y_i = k$ with $\mathbf{t}_i \in \mathbb{R}^K$ such that $\mathbf{t}_i^{(k)} = 1$ and $\mathbf{t}_i^{(j)} = 0 \; \forall j \neq k$. Then our weights instead come as a matrix $\mathbf{W} \in \mathbb{R}^{(d+1) \times K}$, and $\mathbf{W}_{LS} := \text{argmin}_{\mathbf{W}} \sum_{i \in D} ||\mathbf{t}_i - \mathbf{f}(\mathbf{x}_i; \mathbf{W})||^2$ where $\mathbf{f}(\mathbf{x}; \mathbf{W}) = \mathbf{W}^T \tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}} := [\mathbf{x}^T, 1]^T$.

However, the LS classifier has a major drawback in that data points which are correctly classified but far away from the decision boundary (we might think of these points as being very easy to classify) impose a large penalty on the loss function and therefore have a large influence on the final decision boundary[2]. We might also want to avoid LS classification since, unlike LS regression, it doesn't have a probabilistic interpretation—we'll consider probabilistic approaches to classification later.

1.3 Fisher Discriminant Analysis

We can think of the inner product $\langle \mathbf{w}, \mathbf{x} \rangle$ as 'embedding' $\mathbf{x}$ onto the line following the direction of $\mathbf{w}$ from the origin. With this in mind, we might prefer choosing a $\mathbf{w}$ that separates the $\mathbf{x}$ of different classes more distinctly (i.e. further away from each other) in this embedding, whilst embedding inputs $\mathbf{x}$ of the same class close together.

The embedded center of all n_k data points $\mathbf{x}$ of class k is $\hat{\mu}_k = \frac{1}{n_k} \sum_{i \in D: y_i = k} \mathbf{w}^T \mathbf{x}_i$, and so we will define the *within-class scatterness* of a class k as $s_{w,k} = \sum_{i \in D: y_i = k} (\mathbf{w}^T \mathbf{x}_i - \hat{\mu}_k)^2$.

[1] As an example, note that with $K = 3$ and $d = 2$, $K - 1 = 2$ (nonparallel) decision boundaries would split $\mathbb{R}^2$ into four regions: one for each class k and one left over with two possible classifications.

[2] Consider two data points $\mathbf{x}_1$ and $\mathbf{x}_2$ both of class $+1$ with $f(\mathbf{x}_1) = -1$ and $f(\mathbf{x}_2) = 5$. The squared error of the incorrectly classified $\mathbf{x}_1$ is 4 whereas $\mathbf{x}_2$, which is correctly classified, has a larger squared error of 16.

Similarly we define the embedded center of the entire dataset as $\hat{\mu} = \frac{1}{n}\sum_{i\in D}\mathbf{w}^T\mathbf{x}_i$ and the *between-class scatterness* for class k as $s_{b,k} = n_k(\hat{\mu}_k - \hat{\mu})$ (note the multiplication by n_k to scale $s_{b,k}$ alongside $s_{w,k}$).

From our previous discussion on what a 'good' embedding onto $\mathbf{w}$ would be, we clearly would like to maximise $s_{b,k}$ (so different classes are embedded further apart) and minimise $s_{w,k}$ (so data points in the same class are embedded close together) for all k, i.e. we want to find $\mathbf{w}$ that maximises $\sum_{k\in[K]} s_{b,k}/\sum_{k\in[K]} s_{w,k}$. If $K = 2$ the optimal $\mathbf{w}$ is given by $\mathbf{w}_{FDA} = \mathbf{S}_{\mathbf{w}}^{-1}(\mu_1 - \mu_2)$ where $\mathbf{S}_{\mathbf{w}} := \sum_{k=1}^{K}\mathbf{S}_k$, $\mathbf{S}_k$ is the sample covariance of class k times n_k, and μ_k is the mean of (non-embedded) data points of class k.

It is important to note that the analysis here cannot be used to predict classes directly ($f(\mathbf{x};\mathbf{w}_{FDA}) = \langle\mathbf{w}_{FDA},\mathbf{x}\rangle > 0$ does not necessarily mean $\mathbf{x}$ is predicted to a particular class) since FDA doesn't try to maximize classification accuracy, it instead helps us to identify useful embeddings of our dataset (from which classification might be easier).

2 Probabilistic Classification

In a probabilistic approach to classification we need to infer $p(y|\mathbf{x})$ as this will allow us to predict the class $\hat{y} := \text{argmin}_{y_0}\mathbb{E}_{p(y|\mathbf{x})}[L(y,y_0)|\mathbf{x}]$ for an input $\mathbf{x}$, given some loss function L that measures the importance of misclassification based on the classes involved. A discriminative approach would require us to infer $p(y|\mathbf{x})$ directly, however, first we'll look at the problem from a generative point of view in which we can infer $p(\mathbf{x}|y)$ and use the fact that $p(y|\mathbf{x}) \propto p(\mathbf{x}|y)p(y)$.

2.1 Generative Classifiers

2.1.1 Continuous Input

If $\mathbf{x}$ is continuous, we can use the model $p(\mathbf{x}|y = k,\mathbf{w}) = \mathcal{N}_{\mathbf{x}}(\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$. If we assume that each $\mathbf{x}$ is i.i.d and that each class has identical covariance (so $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}\ \forall k \in [K]$), then the likelihood over our dataset D is $p(D|\mathbf{w}) = \prod_{i\in D}p(\mathbf{x}_i|y_i;\mathbf{w})p(y_i) = \prod_{i\in D}\mathcal{N}_{\mathbf{x}_i}(\boldsymbol{\mu}_{y_i},\boldsymbol{\Sigma})p(y_i)$ Estimating $p(y_i = k)$ by n_k/n we can then obtain the MLEs:

$$\hat{\boldsymbol{\mu}}_1,...,\hat{\boldsymbol{\mu}}_k,\hat{\boldsymbol{\Sigma}} := \underset{\boldsymbol{\mu}_1,...,\boldsymbol{\mu}_k,\boldsymbol{\Sigma}}{\text{argmax}}\sum_{i\in D}\log\mathcal{N}_{\mathbf{x}_i}(\boldsymbol{\mu}_{y_i},\boldsymbol{\Sigma})p(y_i)$$

which lead to $\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k}\sum_{i\in D:y_i=k}\mathbf{x}_i$ and $\hat{\boldsymbol{\Sigma}} := \sum_{k\in[K]}\frac{n_k}{n}\frac{1}{n_k}\sum_{i\in D,y_i=k}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T$ (note that this is the weighted mean of $\frac{1}{n_k}\sum_{i\in D,y_i=k}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T$, i.e. the covariance MLE of each class). This gives us the decision boundary

$$\{\mathbf{x}|p(y = k|\mathbf{x};\hat{\mathbf{w}}) = p(y = k'|\mathbf{x};\hat{\mathbf{w}})\}_{\forall k\neq k'} = \left\{\mathbf{x}|\frac{p(\mathbf{x}|y = k;\hat{\mathbf{w}})p(y = k)}{p(\mathbf{x}|y = k';\hat{\mathbf{w}})p(y = k')} = 1\right\}_{\forall k\neq k'}$$

which we show is piecewise-linear in Appendix A. We can do this same analysis without assuming each class has identical covariance, estimating each class covariance individually, however, this will not (in general) lead to a linear decision boundary.

2.1.2 Discrete Input: Naïve Bayes

We now consider an example where $\mathbf{x}$ is discrete, in particular, we work with $\mathbf{x}_i = [x_i^{(1)},...,x_i^{(d)}]^T$ where $x_i^{(j)} \in \mathcal{N}_0$ is the frequency of the jth word from our dictionary (of length d) in the ith document[3]—the task is to classify each document into one of K classes $y \in [K]$.

[3]This is known as the "bag of words" representation.

Assuming $x_i^{(1)}, ..., x_i^{(d)}$ follows a multinomial distribution and each of the $\mathbf{x}_i$ are i.i.d. (this is the 'naïve' assumption), then we should predict class $\hat{y} := \text{argmax}_y\, p(\mathbf{x}|y)p(y)$ for an input $\mathbf{x}$. In the Naïve Bayes algorithm we use the estimates $p(y = k) = n_k/n$ and $p(\mathbf{x}|y) \propto \prod_{j\in[d]} \beta(j|y)^{x^{(j)}}$ where $\beta(j|y)$ is the probability that word j occurs in a document of class k, and is approximated by

$$\beta(j|y = k) \approx \frac{\sum_{i\in D:y_i=k} x_i^{(j)}}{\sum_{i\in D:y_i=k} \sum_{l\in[d]} x_i^{(l)}}.$$

(This is the frequency of word j in all documents of class k divided by the total number of words in all documents of class k.) By calculating the MLE of multinomial parameters (see Appendix B) we can see that Naïve Bayes is essentially an application of these MLEs alongside the i.i.d. assumption in order to predict a classification label.

2.2 Discriminative Classifiers: Logistic Regression

If we want to infer $p(y|\mathbf{x})$ we can use Bayes' rule to see that $p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{\sum_{y'} p(\mathbf{x}|y')p(y')}$. Furthermore if $y \in \{-1, +1\}$ (and assuming $p(\mathbf{x}|y)p(y) > 0\ \forall \mathbf{x}, y$) then this becomes

$$p(y = 1|\mathbf{x}) = \frac{p(\mathbf{x}|y = 1)p(y = 1)}{p(\mathbf{x}|y' = 1)p(y' = 1) + p(\mathbf{x}|y' = -1)p(y' = -1)} = \frac{1}{1 + \frac{p(\mathbf{x}|y=-1)p(y=-1)}{p(\mathbf{x}|y=1)p(y=1)}}.$$

Thus we can proceed in a discriminative approach by modelling the density ratio $\frac{p(\mathbf{x}|y=-1)}{p(\mathbf{x}|y=1)}$ (whereas generative learning models class density $p(\mathbf{x}|y)$). As such, we can construct a model $f(\mathbf{x}; \mathbf{w}) = \langle \mathbf{w}', \mathbf{x} \rangle + w_0 := \log \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x}|y=-1)p(y=-1)}$ so that $p(y = 1|\mathbf{x}; \mathbf{w}) := \sigma(f(\mathbf{x}; \mathbf{w}))$ where $\sigma(t) = \frac{1}{1+\exp(-t)}$ is the sigmoid function[4]. Similarly note that $p(y = -1|\mathbf{x}; \mathbf{w}) := \sigma(-f(\mathbf{x}; \mathbf{w}))$ so we can generalise the model as $p(y|\mathbf{x}; \mathbf{w}) := \sigma(y \cdot f(\mathbf{x}; \mathbf{w}))$.

Assuming that each data point in D is i.i.d. we can construct the likelihood $\prod_{i\in D} p(y_i|\mathbf{x}_i; \mathbf{w})$ and thus find the MLE for $p(y|\mathbf{x}; \mathbf{w})$ as

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

(which has no closed form solution and so has to be solved numerically). This procedure is known as logistic regression (despite not being a regression) and is a very popular and robust classification technique, not being affected by correctly classified outliers like LS classification since doesn't care about squared distances—instead we're using σ which is bounded in $[-1, 1]$—and also since it allows us to fit with feature transforms $\boldsymbol{\phi}$ (using $f(\mathbf{x}; \mathbf{w}) = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}) \rangle$), leading to very flexible decision boundaries[5]. It is also fairly simple to perform logistic regression in a multi-class context, as discussed in Appendix C.

We can also estimate $\mathbf{w}$ with a MAP estimator (assuming priors on $\mathbf{w}$):

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

or use a full probabilistic approach (though this integral cannot be calculated in closed form):

$$p(y|\mathbf{x}) = \int p(y|\mathbf{x}; \mathbf{w})p(\mathbf{w}|D)d\mathbf{w} \propto \int \sigma(y_i \cdot f(\mathbf{x}_i; \mathbf{w}) \prod_{i\in D} \sigma(y_i \cdot f(\mathbf{x}_i; \mathbf{w})p(\mathbf{w})d\mathbf{w}.$$

[4] Note that $\sigma(f(\mathbf{x}; \mathbf{w}))$ is a "generalised linear model"—a linear function wrapped in a non-linear transform.

[5] In general the decision function we obtain from binary logistic regression is $p(y|\mathbf{x}; \mathbf{w}) - c$ where $c \in (0, 1)$ is chosen based on the cost of misclassification.

3 Support Vector Machines

It is often possible to construct multiple decision boundaries that all correctly classify the training data, however, for a model to generalise well, it should choose the decision boundary that will be most likely to correctly classify the unseen test data. That is, we want a decision boundary with a thick error margin—the shortest distance between the decision boundary and a (correctly classified) data point—so that data points would only be misclassified if they are significantly far away from the rest of their class (in the training set). It is easy to verify that the thickness of the error margin between $f(\mathbf{x};\mathbf{w}) := \langle \mathbf{w}', \mathbf{x}\rangle + w_0 = 0$ and $f(\mathbf{x};\mathbf{w}) = 1$ or $f(\mathbf{x};\mathbf{w}) = -1$ is $1/||\mathbf{w}'||$, so to maximise the width of the error margin (in which there should be no training data) we must maximise $1/||\mathbf{w}'||$ (or equivalently, minimise $||\mathbf{w}'||^2$, which will be easier computationally) whilst making sure training data are still correctly classified (i.e. $\forall i \in D : f(\mathbf{x}_i;\mathbf{w})y_i \geq 1$ where $y_i \in \{-1, +1\}$). This is a constrained minimisation problem that will lead us to the **maximal margin classifier**.

In the case that the dataset is not linearly separable we introduce the **soft-margin classifier** by allowing some data points to be misclassified/inside the error margin, for which we add a positive 'compensation' ϵ_i. Thus the constrained optimisation problem becomes $\hat{\mathbf{w}}, \hat{\epsilon} = \text{argmin}_{\mathbf{w},\epsilon} ||\mathbf{w}'||^2 + \sum_i \epsilon_i$ subject to $\forall i : y_i f(\mathbf{x}_i;\mathbf{w}) + \epsilon_i \geq 1$, $\epsilon_i \geq 0$. Note that the solution for $\hat{\epsilon}$ will be sparse, since correctly classified inputs $\mathbf{x}_i$ will have $\epsilon_i = 0$, and also that this is a convex minimization problem, meaning every local minimum is a global minimum.

Unfortunately, this constrained optimisation problem can be difficult to solve, so we often instead use the **Lagrangian dual** problem of this 'primal' problem, which is an equivalent problem leading us to the same solutions but that will (hopefully) be easier to solve. To do this, we first construct the Lagrangian by subtracting[6] our constraints multiplied by Lagrangian multipliers $\lambda_i, \lambda_i' \geq 0$) from our objective function to obtain:

$$l(\boldsymbol{\lambda}) := \min_{\mathbf{w},\epsilon} ||\mathbf{w}'||^2 + \sum_i \left(\epsilon_i - \lambda_i[y_i(\langle \mathbf{w}', \mathbf{x}_i\rangle + w_0) + \epsilon_i - 1] - \lambda_i'\epsilon_i\right).$$

From here we can obtain the optimality condition (derived in Appendix D):

$$\mathbf{w}' = \frac{1}{2}\sum_{i=1}^{n} \lambda_i y_i \mathbf{x}_i, \quad \sum_{i=1}^{n} \lambda_i y_i = 0, \quad \forall i \ \lambda_i + \lambda_i' = 1, \tag{1}$$

thus we may write the Lagrangian as $l(\boldsymbol{\lambda}) = -\frac{\tilde{\boldsymbol{\lambda}}^T \mathbf{X}^T \mathbf{X} \tilde{\boldsymbol{\lambda}}}{4} + \langle \boldsymbol{\lambda}, \mathbf{1}\rangle$ where $\mathbf{X} := [\mathbf{x}_1, ..., \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ and $\tilde{\boldsymbol{\lambda}} := [\lambda_1 \cdot y_1, ..., \lambda_n \cdot y_n]^T$ (see Appendix D). Thus the optimal $\mathbf{w}'$ can be found via 1 after solving the dual problem $\max_{\boldsymbol{\lambda}} -\frac{\tilde{\boldsymbol{\lambda}}^T \mathbf{X}^T \mathbf{X} \tilde{\boldsymbol{\lambda}}}{4} + \langle \boldsymbol{\lambda}, \mathbf{1}\rangle$ subject to $0 \leq \lambda_i \leq 1$, $\sum_{i=1} \lambda_i y_i = 0$.

Our data is now only used within an inner product $\mathbf{X}^T\mathbf{X} =: \mathbf{K}$ where $K^{(i,j)} = \langle \mathbf{x}_i, \mathbf{x}_j\rangle$, so we can improve model flexibility by instead using a kernel function k, leading to the decision function $f(\mathbf{x};\mathbf{w}) := \langle \mathbf{x}, \frac{1}{2}\sum_{i=1}^n \lambda_i y_i \mathbf{x}_i\rangle + w_0 = w_0 + \frac{1}{2}\sum_{i=1}^n \lambda_i y_i \mathbf{x}^T\mathbf{x}_i = w_0 + \frac{1}{2}\sum_{i=1}^n \lambda_i y_i k(\mathbf{x}, \mathbf{x}_i)$.

Note that we will always have data points lying on our decision boundary, otherwise the boundary wouldn't be optimal. We call these points *support vectors*, and they have the property that $\epsilon_i = 0$ and $y_i(\langle \mathbf{w}', \mathbf{x}_i\rangle + w_0) = 1$, so to find w_0 we often take the mean w_0 value found from multiple support vectors.

Both the primal and dual problem are computationally expensive (as constrained optimisation problems often are), however, the dual is quadratic in $\boldsymbol{\lambda} \in \mathbb{R}^n$ (and has simpler constraints) whereas the primal is quadratic in $\mathbf{w} \in \mathbb{R}^{d+1}$, so the dual will be slow for large n and the primal slow for large d. SVMs are very powerful classifiers, however, they lack a probabilistic interpretation and extending them to multi-class classification is non-trivial.

(Further discussion on SVMs can be found in Appendix E).

[6] We subtract rather than add since our constraints are of the form $g(\mathbf{w}, \epsilon) \geq 0$ and not $g(\mathbf{w}, \epsilon) \leq 0$.

A Piecewise-Linear Decision Boundary For Shared Covariance Continuous Input

Here we will show that the decision boundary

$$\{\mathbf{x}|p(y=k|\mathbf{x};\hat{\mathbf{w}}) = p(y=k'|\mathbf{x};\hat{\mathbf{w}})\}_{\forall k \neq k'}$$

is piecewise linear if we assume the data $\mathbf{x}_i$ are independent and that the classes have follow a shared-covariance MVN model, i.e. $p(\mathbf{x}|y=k,\mathbf{w}) = \mathcal{N}_{\mathbf{x}}(\boldsymbol{\mu}_k, \Sigma)\ \forall k \in [K]$.

Proof. Let $k, k' \in [K]$ be distinct. It will suffice to show that the boundary

$$\{\mathbf{x}|p(y=k|\mathbf{x};\hat{\mathbf{w}}) = p(y=k'|\mathbf{x};\hat{\mathbf{w}})\}$$

is linear. To this end let us first apply Bayes' rule along with the assumption on the priors that $p(y=k) = \frac{n_k}{n}$ and $p(y=k') = \frac{n_{k'}}{n}$ to obtain

$$\begin{aligned}
p(y=k|\mathbf{x};\hat{\mathbf{w}}) &= p(y=k'|\mathbf{x};\hat{\mathbf{w}}) \\
\frac{p(\mathbf{x}|y=k;\hat{\mathbf{w}})p(y=k)}{p(\mathbf{x}|\hat{\mathbf{w}})} &= \frac{p(\mathbf{x}|y=k';\hat{\mathbf{w}})p(y=k')}{p(\mathbf{x}|\hat{\mathbf{w}})} \\
p(\mathbf{x}|y=k;\hat{\mathbf{w}})p(y=k) &= p(\mathbf{x}|y=k';\hat{\mathbf{w}})p(y=k') \\
\mathcal{N}_{\mathbf{x}}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})\frac{n_k}{n} &= \mathcal{N}_{\mathbf{x}}(\boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma})\frac{n_{k'}}{n}.
\end{aligned}$$

Taking the natural log of both sides we then see that

$$(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) + \log n_k = (\mathbf{x}-\boldsymbol{\mu}_{k'})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{k'}) + \log n_k.$$

From which we can see that both the left- and right-hand-side are quadratic in $\mathbf{x}$ with a quadratic term of $\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}$. These quadratic terms cancels out, leaving a linear equation in $\mathbf{x}$, hence this boundary is linear. □

B Multinomial Parameter MLE

Suppose that $x^{(1)}, ..., x^{(d)}$ follow a multinomial distribution such that $\sum_{i=1}^d x^{(i)} = n$ and each $x^{(k)}$ occurs independently with probability p_k such that $\sum_{i=1}^d p_i = 1$. First we will prove that the MLE of our parameters $p_1, ..., p_d$ are given by $\hat{p}_i = \frac{x^{(i)}}{n}$.

Proof. This multinomial distribution has the mass function $n!\prod_{i=1}^n \frac{{p_i}^{x^{(i)}}}{x^{(i)}!}$ leading to the log-likelihood function

$$\log n! + \sum_{i=1}^d x^{(i)} \log p_i - \sum_{i=1}^d \log x^{(i)}!.$$

To find our MLE for each p_i we want to maximise this expression, but we also have the constraint $\sum_{i=1}^d p_i = 1$, so we construct the Lagrangian:

$$\log n! + \sum_{i=1}^d x^{(i)} \log p_i - \sum_{i=1}^d \log x^{(i)}! + \lambda(1 - \sum_{i=1}^d p_i)$$

with the Lagrangian multiplier $\lambda \geq 0$. We then find the optimal parameters $\hat{p}_i$ by differentiating this with respect to p_i and setting it equal to zero, in which case we obtain:

$$\frac{x^{(i)}}{\hat{p}_i} - \lambda = 0 \implies \hat{p}_i = \frac{x^{(i)}}{\lambda}.$$

From our constraint, we know that summing this last expression over $i \in [d]$ equals 1, hence we arrive at our MLEs as follows:

$$\sum_{i=1}^{d} \frac{x^{(i)}}{\lambda} = 1 \implies \lambda = \sum_{i=1}^{d} x^{(i)} = n \implies \hat{p}_i = \frac{x^{(i)}}{n}.$$

That is, the MLE for the probability of p_i is simply the number of trials that resulted in i divided by the total number trials, n. □

From this we can see that the Naïve Bayes method is an application of these MLEs, since it uses the estimates $p(y = k) = n_k/n$ and $p(\mathbf{x}|y) \propto \prod_{j\in[d]} \beta(j|y)^{x^{(j)}}$ where

$$\beta(j|y = k) \approx \frac{\sum_{i\in D:y_i=k} x_i^{(j)}}{\sum_{i\in D:y_i=k} \sum_{l\in[d]} x_i^{(l)}}.$$

Using the 'bag of words' representation, the first of these estimates the prior probability of a document being of class k by the number of class-k documents (n_k) divided by the total number of documents (n) in the dataset. Similarly, we estimate $\beta(j|y)$, that is, the probability that that word j occurs in a document of class k, by dividing the frequency of word j in all class-k documents by the total number of words in all class-k documents. In the calculation of $p(\mathbf{x}|y)$ we are simply using the fact that each of the trials in a multinomial distribution is independent, so we can multiply each of the $\beta(j|y)^{x^{(j)}}$ together over all words in the dictionary to obtain a value proportional to the probability that an input $\mathbf{x}$ is of class y.

C Multi-Class Logistic Regression

Extending this problem to the multi-class context, we can obtain a similar result to before by noting again that $p(y = k|\mathbf{x}) \propto p(\mathbf{x}|y = k)p(y = k)$, so for each class $k \in [K]$ we can create a model

$$f^{(k)}(\mathbf{x}; \mathbf{w}_k) = \langle \mathbf{w}_k, \tilde{\mathbf{x}} \rangle := \log p(\mathbf{x}|y = k)p(y = k)$$

where $\tilde{\mathbf{x}} = [\mathbf{x}^T, 1]^T$. We can combine these K models together with the one-hot encoding from section 1.2, so that $\mathbf{f}(\mathbf{x}; \mathbf{W}) = \mathbf{W}^T\tilde{\mathbf{x}}$ with $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_K] \in \mathbb{R}^{(d+1)\times K}$. Then we obtain very similar results as in the $K = 2$ case since

$$p(y = k|\mathbf{x}; \mathbf{w}) = \frac{p(\mathbf{x}|y = k)p(y = k)}{\sum_{k'} p(\mathbf{x}|y = k')p(y = k')} = \frac{\exp(f^{(k)}(\mathbf{x}; \mathbf{w}_k))}{\sum_{k'} \exp(f^{(k')}(\mathbf{x}; \mathbf{w}_{k'}))}$$

which leads to the MLE

$$\mathbf{w}_{MLE} = \underset{\mathbf{w}}{\operatorname{argmax}} \log \prod_{i\in D} p(y|\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmax}} \sum_{i\in D} \log \sigma(\mathbf{f}(\mathbf{x}_i; \mathbf{w}), \mathbf{t}_i)$$

where (since for data point $i \in D$ we have $\langle \mathbf{f}, \mathbf{t_i} \rangle = f^{(y_i)}(\mathbf{x}_i; \mathbf{w}_{y_i})$)

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

D Deriving the Soft-Margin Dual Problem

Here we show that with the Lagrangian of the soft-margin primal problem (with Lagrangian multipliers $\lambda_i, \lambda_i' \geq 0$):

$$l(\boldsymbol{\lambda}) := \min_{\mathbf{w},\epsilon} ||\mathbf{w}'||^2 + \sum_i \left(\epsilon_i - \lambda_i[y_i(\langle \mathbf{w}', \mathbf{x}_i\rangle + w_0) + \epsilon_i - 1] - \lambda_i'\epsilon_i\right). \tag{*}$$

we can obtain the optimality conditions

$$\mathbf{w}' = \frac{1}{2}\sum_{i=1}^{n} \lambda_i y_i \mathbf{x}_i, \quad \sum_{i=1}^{n} \lambda_i y_i = 0, \quad \forall i \ \lambda_i + \lambda_i' = 1, \tag{1}$$

and hence arrive at the soft-margin dual problem given by

$$\max_{\boldsymbol{\lambda}} -\frac{\tilde{\boldsymbol{\lambda}}^T \mathbf{X}^T \mathbf{X} \tilde{\boldsymbol{\lambda}}}{4} + \langle \boldsymbol{\lambda}, \mathbf{1}\rangle$$

subject to

$$0 \leq \lambda_i \leq 1, \quad \sum_{i=1} \lambda_i y_i = 0,$$

where $\tilde{\boldsymbol{\lambda}} := [\lambda_1 \cdot y_1, ..., \lambda_n \cdot y_n]^T$ and $\mathbf{X} := [\mathbf{x}_1, ..., \mathbf{x}_n] \in \mathbb{R}^{d\times n}$.

Proof. Since, as previously mentioned, the primal problem is a convex optimisation problem, the KKT conditions are sufficient, that is, if we find $(\hat{\mathbf{w}}, \hat{\boldsymbol{\lambda}})$ that satisfy the conditions:

$$\nabla_{\mathbf{w}} l(\hat{\boldsymbol{\lambda}}) = 0 \tag{2}$$

$$g(\hat{\mathbf{w}}) \leq 0 \tag{3}$$

$$\forall i : \hat{\lambda}_i, \hat{\lambda}_i' \geq 0 \tag{4}$$

$$\hat{\boldsymbol{\lambda}} \cdot g(\hat{\mathbf{w}}) = 0 \tag{5}$$

(where we are using $g(\mathbf{w})$ to represent our primal constraints), then $(\hat{\mathbf{w}}, \hat{\boldsymbol{\lambda}})$ is an optimal solution.

To find the optimal solution let us differentiate * with respect to $\mathbf{w}'$ and set it equal to zero (this will also mean we satisfy 2) we obtain the first of our optimality conditions:

$$2\mathbf{w}' - \sum_i \lambda_i y_i \mathbf{x}_i = 0 \implies \mathbf{w}' = \frac{1}{2}\sum_i \lambda_i y_i \mathbf{x}_i.$$

Similarly differentiating * w.r.t. ϵ_i and $f(\mathbf{x}_i, \mathbf{w}) := \langle \mathbf{w}', \mathbf{x}_i\rangle + w_0$ before setting these equal to zero we obtain the other two optimality conditions:

$$1 - \lambda_i - \lambda_i' = 0 \implies \lambda_i + \lambda_i' = 1$$

$$\sum_i \lambda_i y_i = 0, \tag{6}$$

the latter of which corresponds to 5 (note that 3 and 4 are also satisfied by hypothesis), and which allows us to rewrite the former simply as

$$\lambda_i \in [0, 1]. \tag{7}$$

Thus we have derived the desired optimality conditions.

Now we move on to rewrite * to form our dual problem's objective function. To do this, first observe that

$$||\mathbf{w}'||^2 = \langle \frac{1}{2}\sum_i \lambda_i y_i \mathbf{x}_i, \frac{1}{2}\sum_i \lambda_i y_i \mathbf{x}_i, \rangle = \frac{1}{4}\sum_j \sum_i \lambda_i y_i \lambda_j y_j \mathbf{x}_i^T \mathbf{x}_j = \frac{\tilde{\boldsymbol{\lambda}}^T \mathbf{X}^T \mathbf{X} \tilde{\boldsymbol{\lambda}}}{4}.$$

Furthermore, see that

$$\sum_i \epsilon_i - \sum_i \lambda_i \epsilon_i - \sum_i \lambda_i' \epsilon_i = \sum_i (1 - \lambda_i - \lambda_i') = 0$$

and

$$\begin{aligned} -\sum_i \lambda_i y_i (\langle \mathbf{w}', \mathbf{x}_i \rangle + w_0) &= -\sum_i (\lambda_i y_i \mathbf{w}'^T \mathbf{x}) - \sum_i (\lambda_i y_i w_0) \\ &= -\sum_i (\lambda_i y_i (\frac{1}{2}\sum_i \lambda_i y_i \mathbf{x}_i)\mathbf{x}) - w_0 \sum_i (\lambda_i y_i) \\ &= -\frac{1}{2}\sum_i \sum_j \lambda_i y_i \lambda_j y_j \mathbf{x}_i^T \mathbf{x}_j - 0 \\ &= -\frac{\tilde{\boldsymbol{\lambda}}^T \mathbf{X}^T \mathbf{X} \tilde{\boldsymbol{\lambda}}}{2}. \end{aligned}$$

Hence the Lagrangian * can be rewritten as

$$||\mathbf{w}'||^2 + \sum_i \left(\epsilon_i - \lambda_i [y_i (\langle \mathbf{w}', \mathbf{x}_i \rangle + w_0) + \epsilon_i - 1] - \lambda_i' \epsilon_i\right) = -\frac{\tilde{\boldsymbol{\lambda}}^T \mathbf{X}^T \mathbf{X} \tilde{\boldsymbol{\lambda}}}{4} + \sum_i \lambda_i.$$

Finally then we arrive at the soft-margin dual problem (in which we *maximise* the objective function w.r.t. λ) using the above result and the constraints given by 6 and 7:

$$\max_{\boldsymbol{\lambda}} -\frac{\tilde{\boldsymbol{\lambda}}^T \mathbf{X}^T \mathbf{X} \tilde{\boldsymbol{\lambda}}}{4} + \langle \boldsymbol{\lambda}, \mathbf{1} \rangle$$

subject to

$$0 \leq \lambda_i \leq 1, \quad \sum_{i=1} \lambda_i y_i = 0,$$

□

E Further Discussion of SVMs

Recall our soft-margin primal problem:

$$\hat{\mathbf{w}}, \hat{\epsilon} = \underset{\mathbf{w}, \epsilon}{\operatorname{argmin}} ||\mathbf{w}'||^2 + \sum_i \epsilon_i \quad \text{subject to} \quad \forall i : y_i f(\mathbf{x}_i; \mathbf{w}) + \epsilon_i \geq 1, \;\; \epsilon_i \geq 0.$$

Note that by the definition of our 'compensation' variables ϵ_i, we may rewrite the objective function as

$$||\mathbf{w}'||^2 + \sum_i \begin{cases} 0 & \text{if } y_i f(\mathbf{x}_i; \mathbf{w}) \geq 1 \\ 1 - y_i f(\mathbf{x}_i; \mathbf{w}) & \text{otherwise.} \end{cases}$$

If we define $L(t) = \max(0, 1 - y_i f(\mathbf{x}_i; \mathbf{w}))$, this can be simplified to

$$||\mathbf{w}'||^2 + \sum_i L(y_i f(\mathbf{x}_i; \mathbf{w}))$$

which can be seen as a regularised loss function, much like those we have seen in previous portfolios. Indeed we can, for example, instead use a sigmoid loss $L_\sigma(t) = -\sigma(t) = -\frac{1}{1-\exp(-t)}$, leading to a form of regularised logisitic regression.

Our particular choice of $L(t) = \max(0, 1 - y_i f(\mathbf{x}_i; \mathbf{w}))$ here helps us to see why SVMs manage to avoid the least-squares pitfall of punishing data points that are 'too easy' to classify. For any correctly classified data point $\mathbf{x}_i$ outside of the error margin (i.e. $y_i f(\mathbf{x}_i; \mathbf{w}) > 1$), the objective function incurs no loss on $\mathbf{x}_i$, whereas a least squares loss will incur some positive loss on any $\mathbf{x_i}$ where $y_i \neq f(\mathbf{x}_i; \mathbf{w})$ (even if $\mathbf{x}_i$ is correctly classified).

If we further investigate the use of the least-squares loss function in this context, we see that the constrained optimisation problem

$$\min_{\mathbf{w}} \sum_i (y_i - f(\mathbf{x}_i; \mathbf{w}))^2 \quad \text{subject to} \quad ||\mathbf{w}'||^2 \leq C$$

can be solved using Lagrangian multipliers, in which case we obtain the Lagrangian function

$$\begin{aligned} l(\lambda) &= \sum_i (y_i - f(\mathbf{x}_i; \mathbf{w}))^2 + \lambda(||\mathbf{w}'||^2 - C) \\ &= \sum_i ((y_i - w_0) - \mathbf{w}'^T \mathbf{x}_i)^2 + \lambda(||\mathbf{w}'||^2 - C) \end{aligned}$$

where $\lambda \geq 0$. This is the same as the loss function in regularised least squares regression with a couple of extra terms (due to our inclusion of w_0). Hence when we differentiate this with respect to $\mathbf{w}'$ (and set the derivative equal to 0) to find the optimal $\hat{\mathbf{w}}'$ we obtain:

$$\hat{\mathbf{w}}' = (\mathbf{X}\mathbf{X}^T + \lambda \mathbf{I})^{-1} \mathbf{X} \tilde{\mathbf{y}}^T$$

where $\tilde{\mathbf{y}} \in \mathbb{R}^n$ has entries $\tilde{y}^{(j)} = y_j - w_0$. Similarly, differentiating with respect to w_0 and setting equal to zero we obtain

$$\begin{aligned} 0 &= \sum_i (2\hat{w}_0 - 2y_i + 2\mathbf{w}'^T \mathbf{x}_i) \\ n\hat{w}_0 &= \sum_i (y_i - \mathbf{w}'^T \mathbf{x}_i) \\ \implies \hat{w}_0 &= \frac{1}{n} \sum_i (y_i - \mathbf{w}'^T \mathbf{x}_i). \end{aligned}$$