# Supervised Learning: Coursework 2

Sage Bergerson, Enric Balaguer Rodon

January 3, 2024

## Contents

# 1 Exercise 1

## 1.1

We define $f\left(\bar{X}\right) = e^{\lambda \bar{X}}$. We apply Jensen's inequality: $f\left(\mathbb{E}\left[\bar{X}\right]\right) \leq \mathbb{E}\left[f\left(\bar{X}\right)\right]$.

$$e^{\lambda \mathbb{E}[\bar{X}]} \leq \mathbb{E}\left[e^{\lambda \bar{X}}\right] \tag{1}$$

$$\log\left(e^{\lambda \mathbb{E}[\bar{X}]}\right) \leq \log\left(\mathbb{E}\left[e^{\lambda \bar{X}}\right]\right) \tag{2}$$

$$\lambda \mathbb{E}\left[\bar{X}\right] \leq \log\left(\mathbb{E}\left[e^{\lambda \bar{X}}\right]\right) \tag{3}$$

$$\mathbb{E}\left[\bar{X}\right] \leq \frac{1}{\lambda} \log\left(\mathbb{E}\left[e^{\lambda \bar{X}}\right]\right) \tag{4}$$

## 1.2

We start by recognising the following:

$$\mathbb{E}\left[e^{\lambda \bar{X}}\right] \leq \sum_{i=1}^{m} \mathbb{E}\left[e^{\lambda X_i}\right] \tag{5}$$

We further acknowledge that such bound can be increased using Hoeffding's Lemma by recognising that with our provided random variable, $X - \mathbb{E}\left[X\right] = X$, given that $\mathbb{E}\left[X\right] = 0$.

$$\sum_{i=1}^{m} \mathbb{E}\left[e^{\lambda X_i}\right] = \sum_{i=1}^{m} \mathbb{E}\left[e^{\lambda (X_i - \mathbb{E}[X_i])}\right] \tag{6}$$

$$\leq \sum_{i=1}^{m} e^{\frac{\lambda^2 (b-a)^2}{8}} \tag{7}$$

We can now drop the sum, as no variable is being indexed by it.

$$\sum_{i=1}^{m} e^{\frac{\lambda^2 (b-a)^2}{8}} = m e^{\frac{\lambda^2 (b-a)^2}{8}} \tag{8}$$

We can now take the log of both sides of our inequality and multiply by $\frac{1}{\lambda}$.

$$\mathbb{E}\left[e^{\lambda \bar{X}}\right] \leq m e^{\frac{\lambda^2 (b-a)^2}{8}} \tag{9}$$

$$\frac{1}{\lambda} \log\left(\mathbb{E}\left[e^{\lambda \bar{X}}\right]\right) \leq \frac{1}{\lambda} \log(m) + \frac{\lambda (b-a)^2}{8} \tag{10}$$

## 1.3

So far, we have found:

$$\mathbb{E}\left[\bar{X}\right] \leq \frac{1}{\lambda} \log\left(\mathbb{E}\left[e^{\lambda \bar{X}}\right]\right) \leq \frac{1}{\lambda} \log(m) + \frac{\lambda (b-a)^2}{8} \tag{11}$$

We can minimise the highest bound by finding a $\lambda$ that minimises such expression. Thus, we proceed to find that expression's minima. Please note: $f(\lambda) = \frac{1}{\lambda} \log(m) + \frac{\lambda (b-a)^2}{8}$.

$$\frac{\partial}{\partial \lambda} f(\lambda) = \frac{-\log(m)}{\lambda^2} + \frac{(b-a)^2}{8} \tag{12}$$

WE set this expression equal to zero and find the extrema for $\lambda$.

$$\frac{\log(m)}{\lambda^2} + \frac{(b-a)^2}{8} = 0 \tag{13}$$

$$\frac{(b-a)^2}{8} = \frac{-\log(m)}{\lambda^2} \tag{14}$$

$$\lambda^2 = \frac{8\log(m)}{(b-a)^2} \tag{15}$$

$$\lambda = \frac{2\sqrt{2\log(m)}}{(b-a)} \tag{16}$$

To determine if this result is a minima or maxima, we differentiate the expression again, substitute the $\lambda$ we just obtained as the extrema and determine if the result is positive or negative.

$$\frac{\partial^2}{\partial^2\lambda} f(\lambda) = \frac{2\log(m)}{\lambda^3} \tag{17}$$

$$= \frac{2\log(m)}{\left(\frac{2\sqrt{2\log(m)}}{(b-a)}\right)^3} \tag{18}$$

$$= \frac{2\log(m)(b-a)^3}{8\log(m)^{\frac{3}{2}}} \tag{19}$$

$$= \frac{(b-a)^3}{8\sqrt{2\log(m)}} \tag{20}$$

We observe that $(b-a) > 0$ and $\log(m) > 0$ as long as $m > 1$. Assuming $m > 1$, then $\frac{\partial^2}{\partial^2\lambda} f(\lambda) > 0$. Thus, we can conclude $\lambda = \frac{2\sqrt{2\log(m)}}{(b-a)}$ minimises our upper bound. In equation 16, $\lambda$ could also be negative, as $\lambda = -\frac{2\sqrt{2\log(m)}}{(b-a)}$. However, when substituting this expression into the double derivative, it would yield a minimiser for when $m < 1$, which is not our case. Therefore, we ignore this case. Finally, we substitute in the expression of $\lambda$ that minimises $f(\lambda)$ to obtain:

$$\mathbb{E}\left[\bar{X}\right] \leq \frac{1}{\lambda}\log(m) + \frac{\lambda(b-a)^2}{8} \tag{21}$$

$$= \frac{\log(m)(b-a)}{2\sqrt{2\log(m)}} + \frac{(b-a)^2 2\sqrt{2\log(m)}}{8(b-a)} \tag{22}$$

$$= \frac{\sqrt{\log(m)}(b-a)}{2\sqrt{2}} + \frac{(b-a)\sqrt{\log(m)}}{2\sqrt{2}} \tag{23}$$

$$= \frac{(b-a)\sqrt{\log(m)}}{\sqrt{2}} \tag{24}$$

This gives the desired expression:

$$\mathbb{E}\left[\bar{X}\right] \leq \frac{(b-a)}{2}\sqrt{2\log(m)} \tag{25}$$

### 1.4

We start by using Jensen's inequality again, $f\left(\mathbb{E}\left[\bar{X}\right]\right) \leq \mathbb{E}\left[f\left(\bar{X}\right)\right]$, where $f(X) = e^{\lambda X}$, and in our case $\bar{X} = \max_{x \in \mathcal{S}} \frac{1}{n} \sum_{j=1}^{n} \sigma_j x_j$. This gives:

$$\exp\left(\lambda \mathbb{E}_\sigma \left[\max_{x \in \mathcal{S}} \frac{1}{n} \sum_{j=1}^{n} \sigma_j x_j\right]\right) \leq \mathbb{E}\left[\exp\left(\lambda \max_{x \in \mathcal{S}} \frac{1}{n} \sum_{j=1}^{n} \sigma_j x_j\right)\right] \tag{26}$$

$$= \mathbb{E}\left[\max_{x \in \mathcal{S}} \exp\left(\frac{\lambda}{n} \sum_{j=1}^{n} \sigma_j x_j\right)\right] \tag{27}$$

where we have pulled the $\max_{x \in \mathcal{S}}$ outside of the exponent on the last step. Now we can use Hoeffding's Lemma by treating $\sigma_j x_j$ as our random variables, $\sigma_j x_j = \{-x_j, x_j\}$. With such random variables, we have the same scenario as we had encountered in above subsection, $\mathbb{E}\left[\sigma_j x_j\right] = 0$, allowing us to apply Hoeffding's Lemma the same way. We start by recognising:

$$\mathbb{E}\left[\max_{x \in \mathcal{S}} \exp\left(\frac{\lambda}{n} \sum_{j=1}^{n} \sigma_j x_j\right)\right] \leq \sum_{x \in \mathcal{S}} \mathbb{E}\left[\exp\left(\frac{\lambda}{n} \sum_{j=1}^{n} \sigma_j x_j\right)\right]. \tag{28}$$

We now re-write the expression using exponent properties in order to be able to apply Hoeffding's Lemma:

$$\sum_{x \in \mathcal{S}} \mathbb{E}\left[\exp\left(\frac{\lambda}{n} \sum_{j=1}^{n} \sigma_j x_j\right)\right] = \sum_{x \in \mathcal{S}} \mathbb{E}\left[\prod_{j=1}^{n} \exp\left(\frac{\lambda}{n} \sigma_j x_j\right)\right] = \sum_{x \in \mathcal{S}} \prod_{j=1}^{n} \mathbb{E}\left[\exp\left(\frac{\lambda}{n} \sigma_j x_j\right)\right]. \tag{29}$$

We then apply Hoeffding's Lemma after recognising $(b - a) = 2x_j$ according to the Hoeffding's Lemma definition. We then proceed to increase the bound via equation 31, given that $|\mathcal{S}| = m$.

$$\sum_{x \in \mathcal{S}} \prod_{j=1}^{n} \mathbb{E}\left[\exp\left(\frac{\lambda}{n} \sigma_j x_j\right)\right] \leq \sum_{x \in \mathcal{S}} \prod_{j=1}^{n} \exp\left(\frac{\lambda^2}{n^2} \frac{2^2 x_j^2}{8}\right) \tag{30}$$

$$\leq |\mathcal{S}| \max_{x \in \mathcal{S}} \prod_{j=1}^{n} \exp\left(\frac{\lambda^2}{n^2} \frac{2^2 x_j^2}{8}\right) \tag{31}$$

$$= m \max_{x \in \mathcal{S}} \exp\left(\frac{\lambda^2}{2n^2} \sum_{j=1}^{n} x_j^2\right) \tag{32}$$

Combining terms across the expressions gives us:

$$\exp\left(\lambda \mathbb{E}_\sigma \left[\max_{x \in \mathcal{S}} \frac{1}{n} \sum_{j=1}^{n} \sigma_j x_j\right]\right) \leq m \max_{x \in \mathcal{S}} \exp\left(\frac{\lambda^2}{2n^2} \sum_{j=1}^{n} x_j^2\right) \tag{33}$$

which we can re-write by logging both sides as:

$$\mathbb{E}_\sigma \left[\max_{x \in \mathcal{S}} \frac{1}{n} \sum_{j=1}^{n} \sigma_j x_j\right] \leq \max_{x \in \mathcal{S}} \left(\frac{\log(m)}{\lambda} + \frac{\lambda \|\mathbf{x}\|_2^2}{2n^2}\right). \tag{34}$$

We can now minimise the upper bound by finding the appropriate $\lambda$:

$$\frac{\partial}{\partial\lambda}\left(\frac{\log(m)}{\lambda}+\frac{\lambda\|\mathbf{x}\|_2^2}{2n^2}\right)=0 \tag{35}$$

$$\frac{-\log(m)}{\lambda^2}+\frac{\|\mathbf{x}\|_2^2}{2n^2}=0 \tag{36}$$

$$\frac{\lambda^2\|\mathbf{x}\|_2^2}{2n^2}=\log(m) \tag{37}$$

$$\lambda=\sqrt{\frac{2n^2\log(m)}{\|\mathbf{x}\|_2^2}} \tag{38}$$

$$\lambda=\frac{n}{\|\mathbf{x}\|_2}\sqrt{2\log(m)}. \tag{39}$$

We substitute $\lambda$ back into the original to get expression:

$$\mathbb{E}_\sigma\left[\max_{x\in\mathcal{S}}\frac{1}{n}\sum_{j=1}^{n}\sigma_j x_j\right]\leq\max_{x\in\mathcal{S}}\left(\frac{\log(m)\|\mathbf{x}\|_2}{n\sqrt{2\log(m)}}+\frac{n\sqrt{2\log(m)\|\mathbf{x}\|_2^2}}{\|\mathbf{x}\|_2 2n^2}\right) \tag{40}$$

$$=\max_{x\in\mathcal{S}}\left(\frac{\|\mathbf{x}\|_2\sqrt{\log(m)}}{\sqrt{2}n}+\frac{\sqrt{\log(m)}\|\mathbf{x}\|_2}{\sqrt{2}n}\right) \tag{41}$$

$$=\max_{x\in\mathcal{S}}\left(\frac{\|\mathbf{x}\|_2\sqrt{2\log(m)}}{n}\right). \tag{42}$$

Simplifying this, we achieve the desired result:

$$\mathbb{E}_\sigma\left[\max_{x\in\mathcal{S}}\frac{1}{n}\sum_{j=1}^{n}\sigma_j x_j\right]\leq\max_{x\in\mathcal{S}}\|\mathbf{x}\|_2\frac{\sqrt{2\log(m)}}{n}. \tag{43}$$

### 1.5

Empirical Rademacher complexity is given by: $\mathcal{R}_\mathcal{S}(\mathcal{H})=\mathbb{E}_\sigma\left[\sup_{f\in\mathbb{H}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i f(x_i)\right]$. We follow the same process as in 1.4, starting with Jensen's Inequality, $f\left(\mathbb{E}\left[\bar{X}\right]\right)\leq\mathbb{E}\left[f\left(\bar{X}\right)\right]$, where $\bar{X}=\sup_{f\in\mathbb{H}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i f(x_i)$ and $f(X)=e^{\lambda X}$. We then pull out $\sup_{f\in\mathbb{H}}$ outside of the exponent in the last step:

$$\exp\left(\lambda\mathbb{E}_\sigma\left[\sup_{f\in\mathbb{H}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i f(x_i)\right]\right)\leq\mathbb{E}_\sigma\left[\exp\left(\lambda\sup_{f\in\mathbb{H}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i f(x_i)\right)\right] \tag{44}$$

$$=\mathbb{E}_\sigma\left[\sup_{f\in\mathbb{H}}\exp\left(\frac{\lambda}{n}\sum_{i=1}^{n}\sigma_i f(x_i)\right)\right]. \tag{45}$$

We already have

$$\mathbb{E}_\sigma\left[\sup_{f\in\mathbb{H}}\exp\left(\frac{\lambda}{n}\sum_{i=1}^{n}\sigma_i f(x_i)\right)\right]\leq\sum_{f\in\mathcal{H}}\mathbb{E}_\sigma\left[\exp\left(\frac{\lambda}{n}\sum_{i=1}^{n}\sigma_i f(x_i)\right)\right] \tag{46}$$

which we can re-write the second term of using exponent properties to obtain:

$$\sum_{f\in\mathcal{H}}\mathbb{E}_\sigma\left[\exp\left(\frac{\lambda}{n}\sum_{i=1}^{n}\sigma_i f(x_i)\right)\right]=\sum_{f\in\mathcal{H}}\mathbb{E}_\sigma\left[\prod_{i=1}^{n}\exp\left(\frac{\lambda}{n}\sigma_i f(x_i)\right)\right]=\sum_{f\in\mathcal{H}}\prod_{i=1}^{n}\mathbb{E}_\sigma\left[\exp\left(\frac{\lambda}{n}\sigma_i f(x_i)\right)\right]. \tag{47}$$

We can now apply Hoeffding's Lemma by recognising $\sigma_i f(x_i)$ as random variables, $\sigma_i f(x_i) = \{-f(x_i), f(x_i)\}$. Furthermore, $\mathbb{E}_\sigma[\sigma_i f(x_i)] = 0$ and $(b-a) = 2f(x_i)$, where $b$ and $a$ are the variables $X - \mathbb{E}_\sigma[X] \in [a,b]$. Thus:

$$\sum_{f\in\mathcal{H}}\prod_{i=1}^{n}\mathbb{E}_\sigma\left[\exp\left(\frac{\lambda}{n}\sigma_i f(x_i)\right)\right]\leq\sum_{f\in\mathcal{H}}\prod_{i=1}^{n}\exp\left(\frac{\lambda^2}{n^2}\frac{2^2 f(x_i)^2}{8}\right) \tag{48}$$

$$\sum_{f\in\mathcal{H}}\prod_{i=1}^{n}\exp\left(\frac{\lambda^2}{n^2}\frac{2^2 f(x_i)^2}{8}\right)\leq|\mathcal{H}|\sup_{f\in\mathbb{H}}\prod_{i=1}^{n}\exp\left(\frac{\lambda^2}{n^2}\frac{2^2 f(x_i)^2}{8}\right) \tag{49}$$

$$=|\mathcal{H}|\sup_{f\in\mathbb{H}}\exp\left(\frac{\lambda^2}{2n^2}\sum_{i=1}^{n}f(x_i)^2\right). \tag{50}$$

By combing terms and taking the log of both sides we get

$$\exp\left(\lambda\mathbb{E}_\sigma\left[\sup_{f\in\mathbb{H}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i f(x_i)\right]\right)\leq|\mathcal{H}|\sup_{f\in\mathbb{H}}\exp\left(\frac{\lambda^2}{2n^2}\sum_{i=1}^{n}f(x_i)^2\right) \tag{51}$$

$$\mathbb{E}_\sigma\left[\sup_{f\in\mathbb{H}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i f(x_i)\right]\leq\frac{\log(|\mathcal{H}|)}{\lambda}+\sup_{f\in\mathbb{H}}\frac{\lambda}{2n^2}\sum_{i=1}^{n}f(x_i)^2 \tag{52}$$

in which from now on, we will use $Z = \sup_{f\in\mathbb{H}}\sum_{i=1}^{n}f(x_i)^2$ for simplicity. Although we have already found an upper bound for the empirical Rademacher complexity where the cardinality of $\mathcal{H}$ appears logarithmically, we want to reduce this bound by finding the $\lambda$ that minimises the expression. Given that $f(\lambda) = \frac{\log(|\mathcal{H}|)}{\lambda} + \frac{\lambda Z}{2n^2}$

$$\frac{\partial}{\partial\lambda}f(\lambda)=\frac{-\log(|\mathcal{H}|)}{\lambda^2}+\frac{Z}{2n^2}. \tag{53}$$

Setting $\frac{\partial}{\partial\lambda}f(\lambda) = 0$ and solving for $\lambda$:

$$\frac{-\log(|\mathcal{H}|)}{\lambda^2}+\frac{Z}{2n^2}=0 \tag{54}$$

$$\lambda=\sqrt{\frac{2n^2\log(|\mathcal{H}|)}{Z}} \tag{55}$$

which wee substitute into $f(\lambda)$ to find:

$$\exp\left(\lambda\mathbb{E}_\sigma\left[\sup_{f\in\mathbb{H}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i f(x_i)\right]\right)\leq\frac{\log(|\mathcal{H}|)\sqrt{Z}}{\sqrt{\log(|\mathcal{H}|)}2n}+\frac{\log(|\mathcal{H}|)Z}{\sqrt{2}n} \tag{56}$$

$$=\frac{\sqrt{\log(|\mathcal{H}|)Z}}{\sqrt{2}n}+\frac{\sqrt{\log(|\mathcal{H}|)Z}}{\sqrt{2}n} \tag{57}$$

$$=\frac{\sqrt{2\log(|\mathcal{H}|)Z}}{n}. \tag{58}$$

This give us the final result of:

$$\exp\left(\lambda \mathbb{E}_\sigma \left[\sup_{f\in\mathbb{H}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i f(x_i)\right]\right) \leq \frac{\sqrt{2\log(|\mathcal{H}|)}}{n}\sqrt{\sup_{f\in\mathbb{H}} \sum_{i=1}^{n} f(x_i)^2}. \tag{59}$$

# 2 Exercise 2

## 2.1

Given that we are dealing with binary classification problems where $\mathcal{Y} = \{1, -1\}$, we can marginalise over $x$ and re-write $R(c)$ as:

$$\mathcal{R}(c) = \mathbb{P}_{(x,y)\sim\rho}(c(x) \neq y) = \left(\mathbb{P}_{(x,y)\sim\rho}\left(c(x) = 1, y = -1|x\right) + \mathbb{P}_{(x,y)\sim\rho}\left(c(x) = -1, y = 1|x\right)\right)\mathbb{P}_{\rho}(x) \tag{60}$$

which, using the 0-1 loss function, simplifies to:

$$\mathcal{R}(c) = \left(\mathbb{P}_{(x,y)\sim\rho}\left(y = -1|x\right)\mathbf{1}_{c(x)=1} + \mathbb{P}_{(x,y)\sim\rho}\left(y = 1|x\right)\mathbf{1}_{c(x)=-1}\right)\mathbb{P}_{\rho}(x). \tag{61}$$

Given that our input-output pair $(x, y)$ is sampled from distribution $\rho$ on $\mathcal{X} \times \mathcal{Y}$, we can further re-write the expression in terms of $\rho$ as

$$\mathcal{R}(c) = \int_{x\in\mathcal{X}} \left(\sum_{y\in\mathcal{Y}} \mathbf{1}_{c(x)\neq y}\rho(y|x)\right) d\rho_{\mathcal{X}}(x) \tag{62}$$

where the inner sum over $\mathcal{Y}$ space is point-wise over $\mathcal{X}$, or in other words $\forall x \in \mathcal{X}$. We can finally re-write it in integral form and marginalise over $x$ to acquire the joint probability:

$$\mathcal{R}(c) = \int_{x\in\mathcal{X}} \left(\int_{y\in\mathcal{Y}} \mathbf{1}_{c(x)\neq y} d\rho(y|x)\right) d\rho_{\mathcal{X}}(x) \tag{63}$$

$$= \int_{\mathcal{X}\times\mathcal{Y}} \mathbf{1}_{c(x)\neq y}\ d\rho(x, y). \tag{64}$$

## 2.2

### 2.2.1 Squared loss

For the squared loss, we can find $f_*$ that minimises $\varepsilon(f)$ by finding $f(x)$ that minimises the inner integral only. The inner integral treats $x$ as fixed, allowing us to re-write the inner integral as a sum over all discrete $y \in \mathcal{Y}$:

$$\int_{y\in\mathcal{Y}} \frac{\partial}{\partial f}(f(x) - y)^2 d\rho(y|x) = 0 \tag{65}$$

$$\sum_{y\in\mathcal{Y}} \frac{\partial}{\partial f}(f(x) - y)^2 \rho(y|x) = 0 \tag{66}$$

$$\sum_{y\in\mathcal{Y}} 2(f(x) - y)\rho(y|x) = 0 \tag{67}$$

$$\rho(y = 1|x)(f(x) - 1) + \rho(y = -1|x)(f(x) + 1) = 0 \tag{68}$$

Proceeding to solve for $f_*(x)$:

$$f(x)\left(\rho(y = 1|x) + \rho(y = -1|x)\right) = \rho(y = 1|x) - \rho(y = -1|x) \tag{69}$$

$$f(x) = \frac{\rho(y = 1|x) - \rho(y = -1|x}{\rho(y = 1|x) + \rho(y = -1|x)} \tag{70}$$

$$f_*(x) = \rho(y = 1|x) - \rho(y = -1|x) \tag{71}$$

where $\rho(y = 1|x) + \rho(y = -1|x) = 1$.

### 2.2.2 Exponential loss

Re-writing the inner integral as a sum over all discrete $y \in \mathcal{Y}$:

$$\int_{y\in\mathcal{Y}} \frac{\partial}{\partial f} e^{(-yf(x))} d\rho(y|x) = 0 \tag{72}$$

$$\sum_{y\in\mathcal{Y}} \frac{\partial}{\partial f} e^{(-yf(x))} \rho(y|x) = 0 \tag{73}$$

$$\sum_{y\in\mathcal{Y}} -ye^{-yf(x)} \rho(y|x) = 0 \tag{74}$$

$$-e^{-f(x)}\rho(y=1|x) + e^{f(x)}\rho(y=-1|x) = 0 \tag{75}$$

Proceeding to solve for $f_*(x)$:

$$e^{f(x)}\rho(y=-1|x) = e^{-f(x)}\rho(y=1|x) \tag{76}$$

$$e^{2f(x)} = \frac{\rho(y=1|x)}{\rho(y=-1|x)} \tag{77}$$

$$f_*(x) = \frac{1}{2}\log\left(\frac{\rho(y=1|x)}{\rho(y=-1|x)}\right) \tag{78}$$

### 2.2.3 Logistic loss

Re-writing the inner integral as a sum over all discrete $y \in \mathcal{Y}$:

$$\int_{y\in\mathcal{Y}} \frac{\partial}{\partial f} \log\left(1 + e^{-yf(x)}\right) d\rho(y|x) = 0 \tag{79}$$

$$\sum_{y\in\mathcal{Y}} \frac{\partial}{\partial f} \log\left(1 + e^{-yf(x)}\right) \rho(y|x) = 0 \tag{80}$$

$$\sum_{y\in\mathcal{Y}} -\frac{y}{e^{yf(x)}+1} \rho(y|x) = 0 \tag{81}$$

$$-\frac{\rho(y=1|x)}{e^{f(x)}+1} + \frac{\rho(y=-1|x)}{e^{-f(x)}+1} = 0 \tag{82}$$

Proceeding to solve for $f_*(x)$:

$$\frac{\rho(y=-1|x)}{e^{-f(x)}+1} = \frac{\rho(y=1|x)}{e^{f(x)}+1} \tag{83}$$

$$\frac{\rho(y=-1|x)e^{f(x)}}{e^{f(x)}+1} = \frac{\rho(y=1|x)}{e^{f(x)}+1} \tag{84}$$

$$e^{f(x)} = \frac{\rho(y=1|x)}{\rho(y=-1|x)} \tag{85}$$

$$f_*(x) = \log\left(\frac{\rho(y=1|x)}{\rho(y=-1|x)}\right) \tag{86}$$

### 2.2.4 Hinge loss

The $f(x)$ that minimises $\varepsilon(f)$ is the $f(x)$ that minimises the inner integral only. However, the hinge loss cannot be differentiated. Writing out the sum over $y \in \mathcal{Y}$ for the expression we get:

$$= \int_{y\in\mathcal{Y}} \max(0, 1 - yf(x))d\rho(y|x) \tag{87}$$

$$= \sum_{y\in\mathcal{Y}} \max(0, 1 - yf(x))\rho(y|x) \tag{88}$$

$$= \max(0, 1 - f(x))\rho(y = 1|x) + \max(0, 1 + f(x))\rho(y = -1|x). \tag{89}$$

We recognise three distinct values our expression can adopt depending on $f(x)$:

- $(1 + f(x))\rho(y = -1|x)$ if $f(x) \geq 1$
- $(1 - f(x))\rho(y = 1|x)$ if $f(x) \leq 1$
- $(1 + f(x))\rho(y = -1|x) + (1 - f(x))\rho(y = 1|x)$ if $-1 < f(x) < 1$

We proceed to find their respective gradients:

- $\rho(y = -1|x)$ if $f(x) \geq 1$
- $-\rho(y = 1|x)$ if $f(x) \leq 1$
- $\rho(y = -1|x)$ if $-1 < f(x) < 1$

Considering that $\rho(y|x) \geq 0$, the global minima is either when $f_*(x) = 1$ or $f_*(x) = -1$, depending on $\rho(y|x)$.

- $f_*(x) = 1$ if $\rho(y = 1|x) > \rho(y = -1|x)$
- $f_*(x) = -1$ if $\rho(y = -1|x) > \rho(y = 1|x)$

Thus, the minimiser for the Hinge loss is:

$$f_*(x) = \text{sign}\left(\rho(y = 1|x) - \rho(y = -1|x) = \begin{cases} +1 & \text{if } \rho(y = 1|x) \geq \rho(y = -1|x) \\ -1 & \text{if } \rho(y = 1|x) < \rho(y = -1|x) \end{cases}\right. \tag{90}$$

*Note:* When $\rho(y = 1|x) = \rho(y = -1|x)$, we face aleatoric uncertainty, in other words, irreducible uncertainty that is embedded within the measurement process. In our case, both labels are equally probable to be predicted. Therefore, equality can be given to either case and won't change $f_*(x)$. This also applies to the next subsection.

## 2.3

We can re-write $R(c)$ as we have done with $\varepsilon(f)$:

$$R(c) = \int_{x\in\mathcal{X}} \left( \int_{y\in\mathcal{Y}} \mathbf{1}_{c(x)\neq y} d\rho(y|x) \right) d\rho_{\mathcal{X}}(x) \tag{91}$$

By definition the limits over the integrals are the spaces expanded by $\rho$, $\mathcal{X} \times \mathcal{Y}$. By re-writing the integral to the expression we have above, we can treat the inner integral to be point-wise in $\mathcal{X}$, in other words $\forall x \in \mathcal{X}$. Thus, we are "counting" how many times our "predictor" $c(x)$

predicts the wrong label, given an $x$. $R(c)$ is then minimised when $c(x) = y$ for all $x \in \mathcal{X}$. Given that we know $\rho$ a priori, we know the distribution all $(x, y)$ are sampled from. Therefore, $c_*(x)$ should predict the label that has the highest probability given $x$ according to $\rho(y|x)$.

$$c_*(x) = \begin{cases} +1 & \text{if } \rho(y=1|x) \geq \rho(y=-1|x) \\ -1 & \text{if } \rho(y=1|x) < \rho(y=-1|x) \end{cases} \tag{92}$$

*Note:* When $\rho(y=1|x) = \rho(y=-1|x)$, we face aleatoric uncertainty, in other words, irreducible uncertainty that is embedded within the measurement process. In our case, both labels are equally probable to be predicted. Therefore, equality can be given to either case and won't change $c_*(x)$.

## 2.4

All of the surrogate frameworks are Fisher consistent. We find the Bayes decision rule for each loss and generalise our results at the end.

$R(c_{f_*})$ minimises $R(c_f)$. Given that $c_f(x) = d(f(x))$, we need to find the mapping $d$ that minimises misclassification error. We have found $f_*(x)$ in the above subsection, and proceed to define mappings for each provided loss. Ultimately, the mapping that will minimise misclassification error is one that returns a label $y = +1$ if $\rho(y=1|x) > \rho(y=-1|x)$ and $y = -1$ if $\rho(y=1|x) < \rho(y=-1|x)$.

### 2.4.1 Squared loss

Given that $f_*(x) = \rho(y=1|x) - \rho(y=-1|x)$ for squared loss, we examine both possibilities for $\rho$:

- If $\rho(y=1|x) \geq \rho(y=-1|x)$, $f_*(x) \geq 0$
- If $\rho(y=1|x) < \rho(y=-1|x)$, $f_*(x) < 0$

Thus, the mapping that minimises misclassification error given squared loss $f_*(x)$ is:

$$d(f_*(x)) = \begin{cases} +1 & \text{if } f_*(x) \geq 0 \\ -1 & \text{if } f_*(x) < 0 \end{cases} \tag{93}$$

### 2.4.2 Exponential loss

Given that $f_*(x) = \frac{1}{2}\log\left(\frac{\rho(y=1|x)}{\rho(y=-1|x)}\right)$ for exponential loss, we examine both possibilities for $\rho$:

- If $\rho(y=1|x) \geq \rho(y=-1|x)$, $f_*(x) \geq 0$
- If $\rho(y=1|x) < \rho(y=-1|x)$, $f_*(x) < 0$

Thus, the mapping that minimises misclassification error given exponential loss $f_*(x)$ is:

$$d(f_*(x)) = \begin{cases} +1 & \text{if } f_*(x) \geq 0 \\ -1 & \text{if } f_*(x) < 0 \end{cases} \tag{94}$$

### 2.4.3 Logistic loss

Given that $f_*(x) = \log\Big(\frac{\rho(y=1|x)}{\rho(y=-1|x)}\Big)$ for logistic loss, we examine both possibilities for $\rho$:

- If $\rho(y=1|x) \geq \rho(y=-1|x),\ f_*(x) \geq 0$
- If $\rho(y=1|x) < \rho(y=-1|x),\ f_*(x) < 0$

Thus, the mapping that minimises misclassification error given logistic loss $f_*(x)$ is:

$$d(f_*(x)) = \begin{cases} +1 & \text{if } f_*(x) \geq 0 \\ -1 & \text{if } f_*(x) < 0 \end{cases} \tag{95}$$

### 2.4.4 Hinge loss

The Hinge loss is already Fischer consistent as $f_*(x)$ already follows the expression in equation 92. In order to generalise the findings of this exercise, we set the mapping to be the sign function for $f_*(x)$, which is still Fisher consistent:

$$d(f_*(x)) = \begin{cases} +1 & \text{if } f_*(x) \geq 0 \\ -1 & \text{if } f_*(x) < 0 \end{cases} \tag{96}$$

### 2.4.5 Generalised mapping

After finding the mapping $d$ for each provided loss, we can generalise all of them by defining the mapping to be $d(f_*(x)) = \text{sign}(f_*(x))$:

$$d(f_*(x)) = \text{sign}(f_*(x)) = \begin{cases} +1 & \text{if } f_*(x) \geq 0 \\ -1 & \text{if } f_*(x) < 0 \end{cases} \tag{97}$$

## 2.5

### 2.5.1

We start by expanding $|R(\text{sign}(f(x))) - R(\text{sign}(f_*(x)))|$ in a similar way to 60:

$$\begin{aligned} &| \left(\mathbb{P}_{(x,y)\sim\rho}(y=-1|x)\mathbf{1}_{\text{sign}(f(x))=1} + \mathbb{P}_{(x,y)\sim\rho}(y=1|x)\mathbf{1}_{\text{sign}(f(x))=-1}\right)\mathbb{P}_\rho(x) \\ &- \left(\mathbb{P}_{(x,y)\sim\rho}(y=-1|x)\mathbf{1}_{\text{sign}f_*(x)=1} - \mathbb{P}_{(x,y)\sim\rho}(y=1|x)\mathbf{1}_{\text{sign}(f_*(x))=-1}\right)\mathbb{P}_\rho(x)| \end{aligned}$$

Since $|ab| = |a||b|$ we can recognise:

$$\begin{aligned} &|\mathbb{P}_{(x,y)\sim\rho}(y=-1|x)\mathbf{1}_{\text{sign}(f(x))=1} + \mathbb{P}_{(x,y)\sim\rho}(y=1|x)\mathbf{1}_{\text{sign}(f(x))=-1} \\ &- \mathbb{P}_{(x,y)\sim\rho}(y=-1|x)\mathbf{1}_{\text{sign}f_*(x)=1} - \mathbb{P}_{(x,y)\sim\rho}(y=1|x)\mathbf{1}_{\text{sign}(f_*(x))=-1}||\mathbb{P}_\rho(x)| \end{aligned} \tag{98}$$

In the above expression, $|R(\text{sign}(f(x))) - R(\text{sign}(f_*(x)))| = 0$ when $\text{sign}(f(x)) = \text{sign}(f_*(x))$. Thus, we focus on the cases where $\text{sign}(f(x)) \neq \text{sign}(f_*(x))$. Given that $\mathbb{P}_\rho(x) \geq 0$ and $|\mathbb{P}_\rho(x)| = \mathbb{P}_\rho(x)$, we can re-write the expression as:

$$|R(\text{sign}(f(x))) - R(\text{sign}(f_*(x)))| = \int_{x\in\mathcal{X}_f} \left| \left( \sum_{y\in\mathcal{Y}} \mathbf{1}_{\text{sign}(f(x))\neq y} - \mathbf{1}_{\text{sign}(f_*(x))\neq y} \right) \right| d\rho_\mathcal{X}(x) \tag{99}$$

$$\int_{x\in\mathcal{X}_f} \left|\rho(y=1|x)\left(\mathbf{1}_{\text{sign}(f(x))\neq 1} - \mathbf{1}_{\text{sign}(f_*(x))\neq 1}\right) + \rho(y=-1|x)\left(\mathbf{1}_{\text{sign}(f(x))\neq -1} - \mathbf{1}_{\text{sign}(f_*(x))\neq -1}\right)\right| d\rho_\mathcal{X}(x) \tag{100}$$

Reminding ourselves that $f_*(x) = \rho(y = 1|x) - \rho(y = -1|x)$, we can now study equation 100. Our integrand can only take two values:

$$\begin{cases} |-\rho(y = 1|x) + \rho(y = -1|x)| & \text{if } \text{sign}(f(x)) = y \text{ and } \text{sign}(f_*(x)) \neq y \\ |\rho(y = 1|x) - \rho(y = -1|x)| & \text{if } \text{sign}(f(x)) \neq y \text{ and } \text{sign}(f_*(x)) = y \end{cases} \tag{101}$$

In both cases, the integrand equals $|f_*(x)|$, allowing us to re-write the expression as desired:

$$|R(\text{sign}(f(x))) - R(\text{sign}(f_*(x)))| = \int_{x \in \mathcal{X}_f} |f_*(x)| d\rho_{\mathcal{X}}(x) \tag{102}$$

### 2.5.2

We come up with the first inequality qualitatevly as:

$$\int_{\mathcal{X}_f} |f_*(x)| d\rho_{\mathcal{X}}(x) \leq \int_{\mathcal{X}_f} |f_*(x) - f(x)| d\rho_{\mathcal{X}}(x) \tag{103}$$

Given that the integral is over $\mathcal{X}_f$, $f_*(x)$, and $f(x)$ will always have different sign, we study the inequality for both cases:

- If $f(x) > 0$ and $f_*(x) < 0$. Then $|f_*(x) - f(x)| \geq |f_*(x)|$.
- If $f(x) < 0$ and $f_*(x) > 0$. Then $|f_*(x) - f(x)| \geq |f_*(x)|$.

Once again, equality can be assigned to any of the two cases and won't affect the result. We can proceed to acquire the desired result using the Cauchy Schwartz inequality, $\left(\int f(x)g(x)dx\right)^2 \leq \int f(x)^2 dx \int g(x)^2 dx$, where $g(x) = 1$:

$$\int_{\mathcal{X}_f} |f_*(x) - f(x)| d\rho_{\mathcal{X}}(x) = \sqrt{\left(\int_{\mathcal{X}_f} |f_*(x) - f(x)| d\rho_{\mathcal{X}}(x)\right)^2} \tag{104}$$

$$\leq \sqrt{\int_{\mathcal{X}_f} |f_*(x) - f(x)|^2 d\rho_{\mathcal{X}}(x) \int_{\mathcal{X}_f} d\rho_{\mathcal{X}}} \tag{105}$$

$$= \sqrt{\int_{\mathcal{X}_f} |f_*(x) - f(x)|^2 d\rho_{\mathcal{X}}} \tag{106}$$

$$= \sqrt{\mathbb{E}_{\rho_{\mathcal{X}}} \left(|f_*(x) - f(x)|^2\right)} \tag{107}$$

### 2.5.3

We start by re-writing the expression as:

$$\varepsilon(f) - \varepsilon(f_*) = \int_{\mathcal{X}} \left(\int_{\mathcal{Y}} l(f(x), y)\right) d\rho_{\mathcal{X}}(x) - \int_{\mathcal{X}} \left(\int_{\mathcal{Y}} l(f_*(x), y)\right) d\rho_{\mathcal{X}}(x) \tag{108}$$

$$= \int_{\mathcal{X}} \left(\int_{\mathcal{Y}} (f(x) - y)^2 - (f_*(x) - y)^2\right) d\rho_{\mathcal{X}}(x) \tag{109}$$

$$= \int_{\mathcal{X}} \left(\int_{\mathcal{Y}} f(x)^2 - f_*(x)^2 - 2y(f(x) - f_*(x))\right) d\rho_{\mathcal{X}}(x) \tag{110}$$

We recognise that $f_*(x) = \rho(y = 1|x) - \rho(y = -1|x) = \mathbb{E}_{\rho(y|x)}[y]$ and re-write the expression using expectations:

$$\varepsilon(f) - \varepsilon(f_*) = \int_{\mathcal{X}} \left( \int_{\mathcal{Y}} f(x)^2 - (\mathbb{E}_{\rho(y|x)}[y])^2 - 2y(f(x) - \mathbb{E}_{\rho(y|x)}[y]) \right) d\rho_{\mathcal{X}}(x) \tag{111}$$

$$= \mathbb{E}_{\rho_{\mathcal{X}}} \mathbb{E}_{\rho(y|x)} \left[ f(x)^2 - (\mathbb{E}_{\rho(y|x)}[y])^2 - 2y(f(x) - \mathbb{E}_{\rho(y|x)}[y] \right]. \tag{112}$$

We recognise that the only term dependent on $y$ is $2y$, and that $\mathbb{E}_{\rho(y|x)} \left[ \mathbb{E}_{\rho(y|x)} [y] \right] = \mathbb{E}_{\rho(y|x)} [y]$. Therefore, we can rewrite this as:

$$\varepsilon(f) - \varepsilon(f_*) = \mathbb{E}_{\rho_{\mathcal{X}}} \left[ f(x)^2 - (\mathbb{E}_{\rho(y|x)}[y])^2 - 2\mathbb{E}_{\rho(y|x)} [y] \left( f(x) - \mathbb{E}_{\rho(y|x)}[y] \right) \right] \tag{113}$$

$$= \mathbb{E}_{\rho_{\mathcal{X}}} \left[ f(x)^2 + (\mathbb{E}_{\rho(y|x)}[y])^2 - 2\mathbb{E}_{\rho(y|x)} [y] f(x) \right] \tag{114}$$

$$= \mathbb{E}_{\rho_{\mathcal{X}}} \left[ (f(x) - f_*(x))^2 \right] \tag{115}$$

$$= \mathbb{E}_{\rho_{\mathcal{X}}} \left[ |f(x) - f_*(x)|^2 \right] \tag{116}$$

We thus acquire the desired result:

$$\varepsilon(f) - \varepsilon(f_*) = \mathbb{E}_{\rho_{\mathcal{X}}} \left[ |f(x) - f_*(x)|^2 \right]. \tag{117}$$

# 3 Exercise 3: Kernel Perceptron

## 3.1 Theory

### 3.1.1 Classifier algorithms

We extend a basic 2-class kernel perceptron algorithm to $k$ classes. Specifically, we use $k = 10$ to classify hand-written digits (0 - 9) from the MNIST dataset. Our perceptron algorithms process data online for one training example at a time. We use two methods to extend the kernel perceptrion: One vs. Rest (OvR) and One vs. One (OvO).

**One vs. Rest classifier**

| | k Class Kernel Perceptron (OvR training) |
|---|---|
| **Input:** | $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \in (\mathbb{R}^n, \{1, \ldots, k\})^m$ |
| **Initialization:** | For each class $j$:<br>$\mathbf{w}_1^{(j)} = \mathbf{0}$ $(\alpha_0^{(j)} = 0)$ |
| **Prediction:** | Upon receiving the $t^{th}$ instance $\mathbf{x}_t$, predict<br>$\hat{y}_t = \arg\max_j(\mathbf{w}_t^{(j)}(\mathbf{x}_t)) = \arg\max_j \sum_{i=0}^{t-1} \alpha_i^{(j)} K(\mathbf{x}_i, \mathbf{x}_t)$ |
| **Update:** | Given $\mathbf{p} = \sum_{i=0}^{t-1} \alpha_i^{(j,l)} K(\mathbf{x}_i, \mathbf{x}_t)$<br>For each class $j$:<br>if $y_t = j$:<br>if sign$(j) = 1$ and sign$(\mathrm{p}_n) = -1$ then $\alpha_t^{(j)} = 1$<br>else if sign$(j) = -1$ and sign$(\mathrm{p}_n) = 1$ then $\alpha_t^{(j)} = -1$<br>else $\alpha_t^{(j)} = 0$<br>$\mathbf{w}_{t+1}^{(j)}(\cdot) = \mathbf{w}_t^{(j)}(\cdot) + \alpha_t^{(j)} K(\mathbf{x}_t, \cdot)$ |

Table 1: Initialization, prediction and update procedures for OvR classifier.

In our OvR classifier, we use one classifier for each class. A classifier represents its own class $j$ as 1 and all other classes as $-1$. We represent the model weights as a $N \times k$ matrix, where $N$ is the number of training samples and each column contains the $\alpha$ values for one class.

Our OvR method uses an adaptive learning mechanism where the update rule for the weight vector is conditional on the sign of both the class and prediction. Thus, for an incorrect prediction, the perceptron adjusts its parameters based on the direction of the misclassification.

**One vs. One classifier**

| | k Class Kernel Perceptron (OvO training) |
|---|---|
| **Input:** | $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \in (\mathbb{R}^n \times \{1, \ldots, k\})^m$ |
| **Initialization:** | For each pair of classes $(j, l)$, $j < l$:<br>$\mathbf{w}_1^{(j,l)} = \mathbf{0}$ $(\alpha_0^{(j,l)} = 0)$ |
| **Prediction:** | Upon receiving the $t^{th}$ instance $\mathbf{x}_t$, predict<br>$\hat{y}_t = \arg\max_{j<l} f_{j,l}(\mathbf{w}_t^{(j,l)}(\mathbf{x}_t)) = \arg\max_{j<l} f_{j,l}(\sum_{i=0}^{t-1} \alpha_i^{(j,l)} K(\mathbf{x}_i, \mathbf{x}_t))$<br>where $f_{j,l}$ is the decision function for the $(j,l)^{th}$ classifier |
| **Decision function:** | In majority voting function $f_{j,l}$ define votes $\mathbf{v} \in \mathbb{R}^k$<br>For each classifier $(j,l)$'s vote $m$:<br>$\mathrm{v}_m \mathrel{+}= 1$ |
| **Update:** | Given $\mathbf{p} = \sum_{i=0}^{t-1} \alpha_i^{(j,l)} K(\mathbf{x}_i, \mathbf{x}_t)$<br>For each $n$ classifier $(j,l)$:<br>if $y_t = j$ and $\mathrm{sign}(p_n) = 1$ then $\alpha_t^{(j,l)} = -1$<br>else if $y_t = l$ and $\mathrm{sign}(p_n) = -1$ then $\alpha_t^{(j,l)} = 1$<br>else $\alpha_t^{(j,l)} = 0$<br>$\mathbf{w}_{t+1}^{(j,l)}(\cdot) = \mathbf{w}_t^{(j,l)}(\cdot) + \alpha_t^{(j,l)} K(\mathbf{x}_t, \cdot)$ |

Table 2: Initialization, prediction, decision and update procedures for OvO classifier.

In our OvO classifier, one classifier is used for each pair of classes. For $k$ classes, we then have $\frac{k(k-1)}{2}$ classifiers. Each classifier represents its first class $j$ as $-1$ and its second class $l$ as 1. We represent the model weights as a $N \times \frac{k(k-1)}{2}$ matrix, where $N$ is the number of training samples and each column contains the $\alpha$ values for a unique pair of classes.

The OvO classifier utilizes a pairwise voting mechanism where each pair of classes gets a separate classifier, and the final prediction is made based on the majority vote across all classifiers.

### 3.1.2 Kernels

A feature map $\Phi(\mathbf{x})$ transforms data from a non-linearly separable lower dimension to a linearly separable higher dimension. This allows for multi-class classification by perceptrons, which require linearly separable data. To optimize our training process, we compute the kernel values for the entire training set before each run.

**Polynomial kernel**

Our polynomial kernel has the form $K_d(\mathbf{x}_i, \mathbf{x}_t) = (\mathbf{x}_i \cdot \mathbf{x}_t)^d$, where $\mathbf{x}_t$ is the $t^{th}$ data instance, and $d$ is a parameter controlling the dimension of the polynomial. We compute the polynomial kernel as:

$$K_d = (\mathbf{X}\mathbf{X}^T)^d = \begin{bmatrix} (\mathbf{x}_1^T \mathbf{x}_1)^d & \cdots & (\mathbf{x}_1^T \mathbf{x}_m)^d \\ \vdots & \ddots & \vdots \\ (\mathbf{x}_m^T \mathbf{x}_1)^d & \cdots & (\mathbf{x}_m^T \mathbf{x}_m)^d \end{bmatrix} \tag{118}$$

**Gaussian kernel**

Our Gaussian kernel has the form $K_c(\mathbf{x}_i, \mathbf{x}_t) = e^{-c\|\mathbf{x}_i - \mathbf{x}_t\|^2}$, where $\mathbf{x}_t$ is the $t^{th}$ data instance, and $c$ is a parameter controlling the factor of the norms. We expand the kernel function as:

$$K_c = e^{-c(\|\mathbf{x}_i\|^2 + \|\mathbf{x}_t\|^2 - 2\mathbf{x}_i \cdot \mathbf{x}_t)} \tag{119}$$

Using this expansion, we compute the Gaussian kernel as:

$$K_c = e^{-c[\mathbf{X}_{(\cdot)} + \mathbf{X}_{(\cdot)}^T - 2\mathbf{X}\mathbf{X}^T]} = \begin{bmatrix} e^{-c\|\mathbf{x}_1 - \mathbf{x}_1\|^2} & \dots & e^{-c\|\mathbf{x}_1 - \mathbf{x}_m\|^2} \\ \vdots & \ddots & \vdots \\ e^{-c\|\mathbf{x}_m - \mathbf{x}_1\|^2} & \dots & e^{-c\|\mathbf{x}_m - \mathbf{x}_m\|^2} \end{bmatrix} \tag{120}$$

## 3.2 Methods and Results

### 3.2.1 Summary of experiments

| Method | Kernel Function | Training Epochs | Cross-validation |
|---|---|---|---|
| OvR | Polynomial | 12 | None |
| OvR | Polynomial | 12 | 5-fold |
| OvR | Gaussian | 12 | None |
| OvR | Gaussian | 12 | 5-fold |
| OvO | Polynomial | 12 | None |
| OvO | Polynomial | 12 | 5-fold |

Table 3: Overview of experiments.

We use a subset of the MNIST dataset, with approximately $9,300$ labeled $16 \times 16$-pixel black-and-white images of hand-written digits $0 - 9$. We perform six experiments to assess different classification methods, kernels, and hyperparameters. For all experiments we perform 20 runs with a randomized 80/20 train/test split and average results across runs.

### 3.2.2 Determining optimal number of epochs

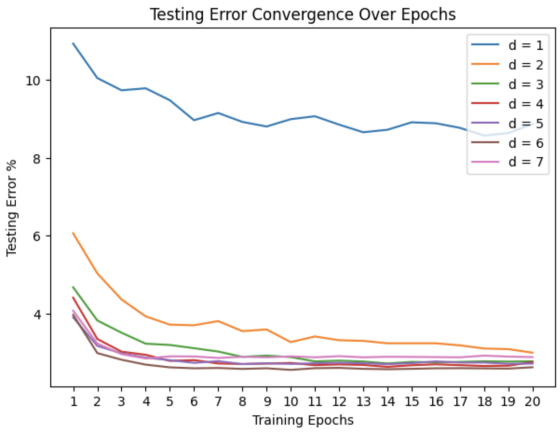


Figure 1: Mean testing error convergence for OvR method with polynomial kernel.

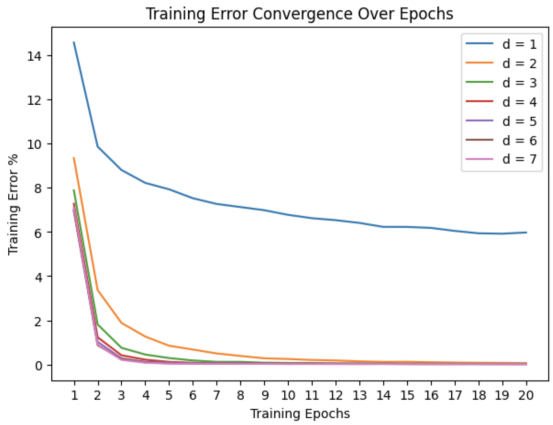


Figure 2: Mean training error convergence for OvR method with polynomial kernel.

We test each $d$ in the range $[1, 7]$ with each number of epochs in the range $[1, 20]$ and calculate the mean test and train error rates using the OvR method with a polynomial kernel. For $d$ values with overall lower error rates (e.g. $d = 3, .., 7$), we see error convergence around 12 epochs. Therefore, we choose to use 12 training epochs in all experiments for consistency.

### 3.2.3 Basic results

| d | Test Error % | Train Error % |
|---|---|---|
| 1 | 8.844 ± 1.252 | 6.543 ± 0.264 |
| 2 | 3.261 ± 0.373 | 0.163 ± 0.048 |
| 3 | 2.898 ± 0.313 | 0.043 ± 0.015 |
| 4 | 2.742 ± 0.351 | 0.037 ± 0.023 |
| 5 | 2.540 ± 0.421 | 0.023 ± 0.014 |
| 6 | 2.680 ± 0.322 | 0.026 ± 0.016 |
| 7 | 2.777 ± 0.429 | 0.018 ± 0.012 |

Table 4: Mean testing and training errors for OvR method with a polynomial kernel.

For the OvR method with polynomial kernel, we test each $d$ in the range $[1, 7]$ and calculate the mean test and train error rates. We observe the lowest test errors between $d$ values of 4 and 6.

### 3.2.4 Cross-validation

| d* | Test Error % |
|---|---|
| 4.700 ± 0.843 | 2.524 ± 0.389 |

Table 5: Mean $d*$ and testing error for OvR method with polynomial kernel.

We split the training set from within and perform 5-fold cross-validation to select the $d$ which yields the lowest average test error. We then retrain on the full training set using the selected $d$ and calculate the average $d$ and test error.

Our cross-validated $d*$ of 4.700 falls in the range we expected based on our basic results. A SD of 0.843 indicates that in some runs, $d$ parameters could have been selected across the range of $3 - 7$ (adding 3 SDs gives $2.171 - 7.229$). This is consistent with the error rates observed in the basic results which vary by at most 0.358% between $d = 3$ and $d = 7$.

### 3.2.5 Confusion matrix

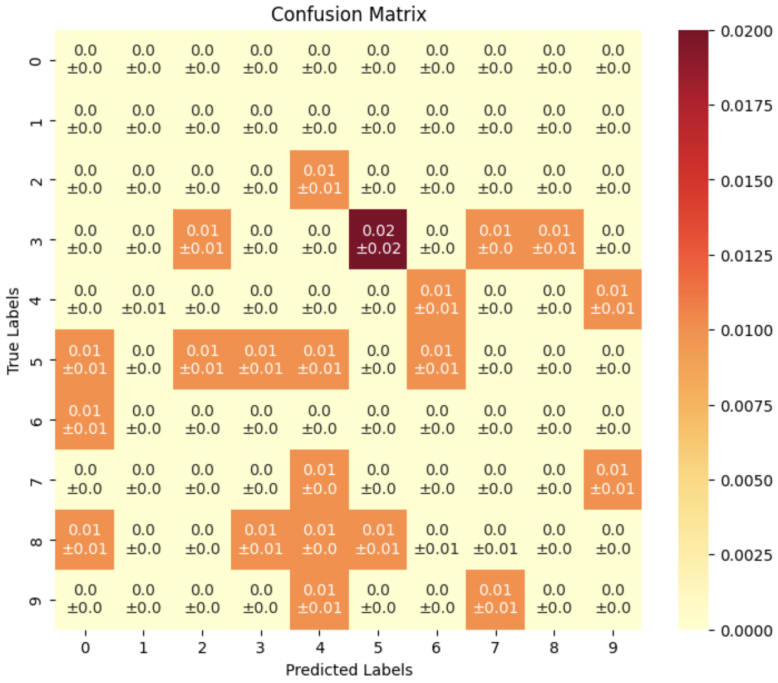


Figure 3: Predicted label-true label error rates across all test images in the cross-validation final runs of 3.2.4.

We save all predicted and ground-truth labels from the final testing runs in 3.2.4. We compute a $10 \times 10$ matrix where each cell contains a confusion error rate (CER) calculated as:

$$CER = \frac{\text{Number of times digit } a \text{ was mistaken for digit } b \text{ (test set)}}{\text{Number of digit } a \text{ points (test set)}} \tag{121}$$

We observe that the OvR method with polynomial kernel is most accurate in classifying 0s and 1s, with cumulative CERs of 0.00. It is least accurate in classifying 3s and 5s, with cumulative CERs of 0.05. The model confuses these two during prediction, with 3s misclassified as 5s at a rate of 0.02, and 5s misclassified as 3 at a rate of 0.01. The most frequent incorrect predicted label is 4, at a cummulative rate of 0.04, followed by a 5 or 0, at cummulative rates of 0.03. The model is unlikely to misclassify any digit as a 1, at a rate of 0.00.

### 3.2.6 Hardest to predict images

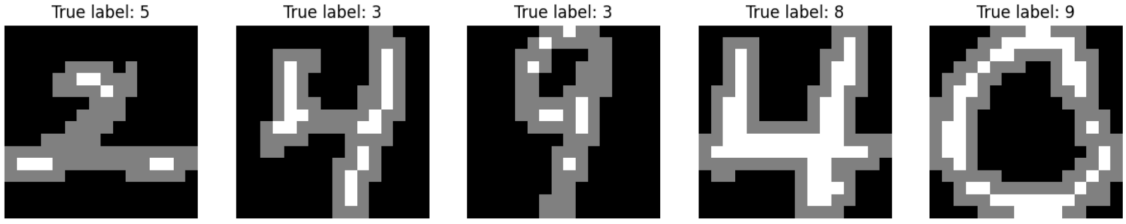


Figure 4: The five images which were most frequently misclassified in the cross-validation final runs of 3.2.4.

Using the same results from 3.2.4 with the corresponding image data, we identify which images had the highest misclassification rates. The results are not surprising, as each image more closely resemble a digit which does not correspond to its ground truth label. From left to right, the digits (subjectively) appear to be $[2, 4, 9, 4, 0]$, while their ground truth labels are $[5, 3, 3, 8, 9]$. This likely reveals an error in data labeling, and provides realistic expectations for the level of accuracy our models achieve.

### 3.2.7 Gaussian kernel

| c | Test Error % | Train Error % |
|---|---|---|
| 0.0000 | 13.567 ± 3.959 | 13.677 ± 0.492 |
| 0.0002 | 10.645 ± 3.778 | 8.972 ± 0.323 |
| 0.0013 | 5.395 ± 1.140 | 2.483 ± 0.201 |
| 0.0067 | 2.804 ± 0.346 | 0.034 ± 0.019 |
| 0.0357 | 3.161 ± 0.389 | 0.011 ± 0.012 |
| 0.1889 | 6.245 ± 0.676 | 0.000 ± 0.000 |
| 1.0000 | 6.589 ± 0.531 | 0.000 ± 0.000 |

Table 6: First experimental range for $S$. Mean testing and training errors for OvR method with Gaussian kernel.

| c | Test Error % | Train Error % |
|---|---|---|
| 0.0014 | 6.013 ± 1.291 | 2.199 ± 0.128 |
| 0.0024 | 4.110 ± 0.601 | 0.688 ± 0.116 |
| 0.0041 | 3.075 ± 0.253 | 0.136 ± 0.045 |
| 0.0071 | 2.686 ± 0.361 | 0.036 ± 0.016 |
| 0.0123 | 2.567 ± 0.523 | 0.022 ± 0.015 |
| 0.0213 | 2.516 ± 0.366 | 0.021 ± 0.012 |
| 0.0369 | 3.186 ± 0.450 | 0.014 ± 0.012 |

Table 7: Second experimental range for $S$. Mean testing and training errors for OvR method with Gaussian kernel.

| c | Test Error % | Train Error % |
|---|---|---|
| 0.0071 | 2.712 ± 0.319 | 0.034 ± 0.012 |
| 0.0093 | 2.616 ± 0.340 | 0.034 ± 0.023 |
| 0.0123 | 2.629 ± 0.391 | 0.027 ± 0.009 |
| 0.0162 | 2.583 ± 0.373 | 0.027 ± 0.009 |
| 0.0213 | 2.535 ± 0.344 | 0.013 ± 0.012 |
| 0.0280 | 2.876 ± 0.395 | 0.014 ± 0.012 |
| 0.0369 | 3.113 ± 0.332 | 0.015 ± 0.010 |

Table 8: Final selected range for $S$. Mean testing and training errors for OvR method with Gaussian kernel.

| c* | Test Error % |
|---|---|
| 0.0150 ± 0.0032 | 2.557 ± 0.264 |

Table 9: Mean $c*$ and testing error for OvR method with Gaussian kernel.

We first select a set of values $S$ from which to draw the parameter $c$. For our Gaussian kernel $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-c\|\mathbf{x}_i - \mathbf{x}_j\|^2}$, when $\mathbf{x}_i, \mathbf{x}_j$ are close together the kernel evaluates close to 1. When $\mathbf{x}_i, \mathbf{x}_j$ are far apart, the kernel evaluates close to 0. Therefore, we select an initial range for $S$ of $[0, 1]$. As the Gaussian kernel is exponential, we use $S = [e^{-10}, ..., e^{0}]$, with seven evenly spaced increments between $[-10, 0]$.

We use two initial experiments with the OvR method to refine our range for $S$. In the first experiment, $c = 0.0067$ produced the lowest testing error, so we center our search here in the second experiment, where we observe the lowest testing error for $c = 0.213$. We settle on an exponentially distributed range for S of $[0.0071, 0.0369]$.

We follow the procedures in 3.2.3 and 3.2.4 using the OvR method with Gaussian kernel to obtain our final results. We observe the lowest test errors between $c$ values of 0.0162 and 0.0213,

and our cross-validated $c*$ parameter is 0.0150, slightly below this range. Given, our exponential distribution of $S$, this discrepancy is not surprising. With a SD of 0.0032, in some runs $c$ parameters could have been selected across the range of $0.0071 - 0.213$ (adding 3 SDs yields $0.0054 - 0.0246$). This is consistent with the error rates observed in the basic results which only vary by at most 0.177% between $c = 0.0071$ and $c = 0.213$.

#### 3.2.8 One vs. One Classifier

| d | Test Error % | Train Error % |
|---|---|---|
| 1 | 6.462 ± 0.698 | 2.612 ± 0.217 |
| 2 | 3.484 ± 0.491 | 0.087 ± 0.049 |
| 3 | 3.231 ± 0.301 | 0.051 ± 0.022 |
| 4 | 3.086 ± 0.352 | 0.032 ± 0.018 |
| 5 | 3.331 ± 0.415 | 0.043 ± 0.021 |
| 6 | 3.417 ± 0.409 | 0.037 ± 0.022 |
| 7 | 3.460 ± 0.428 | 0.027 ± 0.015 |

Table 10: Mean testing and training errors for OvO method with polynomial kernel.

| d* | Test Error % |
|---|---|
| 3.500 ± 0.671 | 3.223 ± 0.450 |

Table 11: Mean $d*$ and testing error for OvO method with polynomial kernel.

We follow the procedures in 3.2.3 and 3.2.4 using the OvO method with polynomial kernel. We observe the lowest test errors between $d$ values of 3 and 5, and our cross-validated $d*$ is 3.223. A SD of 0.450 indicates that in some runs, $d$ parameters could have been selected across the range of $2 - 4$ (adding 3 SDs yields $1.873 - 4.57$). This is consistent with error rates observed in the basic results which only vary by at most 0.398% between $d = 2$ and $d = 4$.

### 3.3 Discussion

#### 3.3.1 Comparison of results between polynomial and Gaussian kernel

| Method | Kernel Function | d*/c* | Test Error % |
|---|---|---|---|
| OvR | Polynomial | $4.700 \pm 0.843$ | $2.524 \pm 0.389$ |
| OvO | Polynomial | $3.500 \pm 0.671$ | $3.223 \pm 0.450$ |
| OvR | Gaussian | $0.0150 \pm 0.0032$ | $2.557 \pm 0.264$ |

Table 12: Cross-validation results for each method and kernel combination.

We achieve the best results for this dataset by using an OvR method with a polynomial kernel, closely followed by an OvR method with a Gaussian kernel. The polynomial kernel likely performs better on the MNIST dataset as it captures the polynomial nature of pixel intensity relationships, which is useful for identifying digits and handling the sparse feature space of the image data. Given the exponential nature of the Gaussian kernel, our $c*$ parameter falls between [0,1], while our $d*$ parameters are much higher.

#### 3.3.2 Comparison of results between OvR and OvO methods

The OvO method likely requires a lower $d*$ as it deals with simpler binary classification tasks for each pair of classes, reducing the model complexity needed for class separation.

The OvO method's higher testing error may stem from the complexity of integrating numerous pairwise decisions, which can increase the overall variance. Additionally, the voting mechanism used to consolidate these binary decisions can skew results, especially if some digit pairs are inherently more challenging to differentiate.

### 3.3.3 Parameters which were not cross-validated over

**Learning rate**

We use a fixed learning rate in both the OvR and OvO kernel perceptrons. Selecting a learning rate through cross-validation could potentially yield higher accuracy by systematically testing and selecting the rate that best balances speed of convergence with risk of overshooting the loss function minimum for unseen data. However, this process can significantly increase the computational cost due to the iterative nature of cross-validation.

**Training epochs**

We also use a fixed number of training epochs in both the OvR and OvO kernel perceptrons, which we selected by observing where training and testing errors converged for the OvR method with a polynomial kernel. More training epochs could lead to better model accuracy as it allows more adjustments to the classifier's weights, but too many epochs may cause overfitting to the training data. Notably, we do not test for convergence in other combinations of classification methods and kernel functions. Using cross-validation to select the number of epochs for each classifier-kernel combination could help to optimize training processes, but would also increase computational time and complexity.