CSC 2515: Introduction to Machine Learning

Homework 3

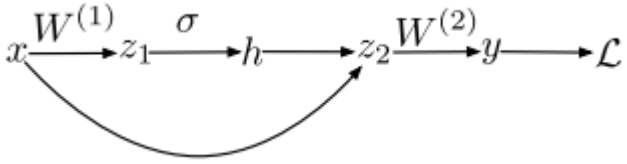
By: Alexis Bruneau:

Collaborated with: Ramy ElMallah, Anton Korikov

Presented to Dr.Amir-massoud Farahmand

Nov 25th, 2022

Question 1 Backpropagation - 20 pts



$$z_1 = W^1 x, with\ x \epsilon R^d \tag{1}$$

$$h = \sigma(z_1), with\ h \epsilon R^d \tag{2}$$

$$z_2 = h + x \tag{3}$$

$$y = W^2 z_2 \tag{4}$$

$$L = \frac{1}{2}(y - t)^2, with\ t \epsilon R \tag{5}$$

a) Determine the dimensions of $W^1, W^2, z_1, and\ z_2$

We assume that x is a point dimension (d,1). Looking at equation 3, we can assume that h and z_2 are the same dimension as x (d,1). We can also say that z_1 is of dimension (d,1) since the activation function does not change its dimension. Afterwards we can get the dimension of W^1 with the following equation:

$$z_1 = W^1 x$$

$$(d,1) = (a,b) \cdot (d,1)$$

Hence, we can say that a and b needs to be equal to d, w^1 is of shape (d,d).

Since we know that t is a real number, we know that the dimension of y,t and L is (1,1). We can find the shape of W^2 using the following equation:

$$y = W^2 z_2$$

$$(1,1) = (e,f) \cdot (d,1)$$

Hence, we can say that the dimension of W^2 is (1,d). To conclude we found the following:

- $z_1 = (d,1)$
- $z_2 = (d,1)$
- $W^1 = (d,d)$
- $W^2 = (1,d)$

b) Calculate the number of parameters in this network, as a function of d. You need to show how you get to the solution.

We know that W^1 is of size (d,d). Hence it has d^2 parameters. The same logic is done for W^1, it has d parameters. Hence, the total network parameter can be found by combining the parameters of W^1 and W^2.

$$Total\ network\ parameters = d^2 + d$$

c) Compute the gradient of loss L with respect to all variables

i)

$$\bar{y} = \frac{dL}{dy}$$

$$\frac{dL}{dy} = \frac{d}{dy}\left(\frac{1}{2}(y-t)^2\right) = y - t$$

ii)

$$\bar{W}^2 = \frac{dL}{dW^2}$$

$$= \frac{dL}{dy} * \frac{dy}{dW^2}$$

$$= (y-t)\left(\frac{d}{dW^2}(W^2 z_2)\right) = (y-t)(z_2)$$

iii)

$$\bar{z_2} = \frac{dL}{dz_2} = \frac{dL}{dy} * \frac{dy}{dz_2}$$

$$= (y-t)\left(\frac{d}{dz_2}(W^2 z_2)\right) = (y-t)(W^2)$$

iv)

$$\bar{h} = \frac{dL}{dy} * \frac{dy}{dz_2} * \frac{dz_2}{dh}$$

$$= (y-t)(W^2)\left(\frac{d}{dh}(h+x)\right) = (y-t)(W^2)$$

v)

$$\bar{z} = \frac{dL}{dy} * \frac{dy}{dz_2} * \frac{dz_2}{dh} * \frac{dh}{dz_1}$$

$$= (y - t)(W^2)\left(\frac{d}{dz_1}\big(\sigma(z_1)\big)\right)$$

$$= (y - t)(W^2)(\sigma'(z_1))$$

vi)

$$\overline{W^1} = \frac{dL}{dy} * \frac{dy}{dz_2} * \frac{dz_2}{dh} * \frac{dh}{dz_1} * \frac{dz_1}{dW^1}$$

$$= (y - t)(W^2)\big(\sigma'(z_1)\big) * \left(\frac{d}{dW^1}(W^1 x)\right)$$

$$= (y - t)(W^2)\big(\sigma'(z_1)\big)(x)$$

vii)

$$\bar{x} = \frac{dL}{dy} * \frac{dy}{dz_2} * \frac{dz_2}{dh} * \frac{dh}{dz_1} * \frac{dz_1}{dx} + \frac{dL}{dy} * \frac{dy}{dz_2} * \frac{dz_2}{dx}$$

$$\frac{dx}{dx} = 1$$

$$\frac{dh}{dx} = \frac{dh}{dz_1} * \frac{dz_1}{dx} = \sigma'(Z_1)W^1$$

$$\frac{dz_1}{dx} = \frac{d}{dx}(W^1 x) = W^1$$

$$\bar{x} = \frac{dL}{dx} = \frac{dL}{dz_2} * \frac{dz_2}{dx} = (y - t)(W^2)\big[\big(\sigma'(z_1)\big)(W^1) + 1\big]$$

2. Multi-Class Logistic Regression -10pts

$$z = Wx$$

$$y = softmax(z)$$

$$L_{CE}(t, y) = -t^T \log(y) = -\sum_{k=1}^{K} t_k \log(y_k)$$

$$y_k = softmax(z_1, \ldots., z_K)_k = \frac{e^{z_k}}{\sum_{k'=1}^{K} e^{z_{k'}}}$$

a) Compute:

$$\frac{dy_k}{dz_{k'}}$$

$$\frac{dy_k}{dz_{k'}} = \frac{d}{dz_{k'}}\left(\frac{e^{z_k}}{\sum_{k'=1}^{K} e^{z_{k'}}}\right)$$

Using quotient rule:

$$f(x) = \frac{g(x)}{h(x)} \rightarrow f'(x) = \frac{g'(x)h(x) - g(x)h'(x)}{\big(h(x)\big)^2}$$

$$g(x) = e^{z_k}, h(x) = \sum_{k'=1}^{K} e^{z_{k'}}$$

$$g'(x) = \begin{cases} e^{z_k} \; for \; k = k' \\ 0 \; oherwise \end{cases}$$

$$h'(x) = e^{z_{k'}}$$

When $k = k'$:

$$\frac{dy_k}{dz_{k'}} = \frac{(e^{z_k})\big(\sum_{k'=1}^{K} e^{z_{k'}}\big) - (e^{z_k})(e^{z_{k'}})}{\big(\sum_{k'=1}^{K} e^{z}k'\big)^2}$$

$$= \frac{e^{z_k}\big(\sum_{k'=1}^{K} e^{z}k' - e^{z_{k'}}\big)}{\big(\sum_{k'=1}^{K} e^{z}k'\big)^2} = \frac{e^{z_k}}{\sum_{k'=1}^{K} e^{z}k'} * \frac{\big(\sum_{k'=1}^{K} e^{z}k' - e^{z_{k'}}\big)}{\sum_{k'=1}^{K} e^{z}k'}$$

$$= y_k - y_k y_k'$$

$$= y_k(1 - y_k')$$

When $k \neq k'$:

$$\frac{dg_k}{dz_k'} = 0$$

$$\frac{dy_k}{dz_{k'}} = \frac{0 - e^{z_k}(e^{z_{k'}})}{\left(\sum_{k'=1}^{K} e^{z}k'\right)^2}$$

$$= -\frac{e^{z'_k}}{\sum_{k'=1}^{K} e^{z'_k}} * \frac{e^{z_k}}{\sum_{k'=1}^{K} e^{z'_k}}$$

Hence:

$$\frac{dy_k}{dz_{k'}} = \begin{cases} y_k(1 - y'_k) \; for \; k = k' \\ -y_k y'_k \; for \; k \neq k' \end{cases}$$

b)

$$L_{CE} = -\sum_{k'=1}^{K} t_k \log y_k$$

$$\frac{dL}{dw_k} = -\sum_{k'}^{K} t'_k \frac{d}{dy'_k}(\log y'_k) \frac{dy'_k}{dz_k} \frac{dz_k}{dw_k}$$

$$\frac{dz_k}{dw_k} = \frac{d}{dw_k}(w_k x) = x$$

$$\frac{d}{dy'_k}(\log(y'_k)) = \frac{1}{y'_k}$$

$$\frac{dL}{dw_k} = -\frac{t_k y_k}{y_k}(1 - y_k)x - \sum_{k' \neq k} \frac{t'_k y'_k y_k}{y'_k}(-1)x$$

$$= -t_k x + t_k y_k x + \sum_{k' \neq k} t'_k y_k x$$

$$= -t_k x + y_k x(t_k + \sum_{k' \neq k} t'_k)$$

Because the target is a one-hot encoded vector and so $\sum_k t_k = 1$

$$= -t_k x + y_k x$$

$$= (y_k - t_k)x$$

Question 3 Some Interpretations of Regularization

a) Compute the partial derivative of $\phi(x)^T w$ w.r.t its input $x \epsilon R^d$.

We know that $\phi(x) \epsilon R^p$, $w \epsilon R^p$

$$\frac{df(x;w)}{dx} = \frac{d}{dx}(\phi(x)^T w)$$
$$= \left[\frac{d\phi x}{dx_1} \dots \frac{d\phi x}{dx_d}\right]^T w = J_\phi(x)^T w$$

Where $J_\Phi(x)$ has dimension p x d

$$J_\phi = \begin{bmatrix} \frac{d\phi_1 x}{dx_1} & \cdots & \frac{d\phi_1 x}{dx_d} \\ \vdots & \ddots & \vdots \\ \frac{d\phi_p x}{dx_1} & \cdots & \frac{d\phi_p x}{dx_d} \end{bmatrix}$$

Hence:

$$\frac{df(x;w)}{dx} = J_\phi(x)^T w$$
$$dimension = (d,p)\ x\ (p,1) = d$$

b) Compute:

$$\left|\left|\frac{df(x;w)}{dx}\right|\right|_2^2 = \left[J_\phi(x)^T w\right]^T \left[J_\phi(x)^T w\right] = w^T J_\phi(x) J_\phi(x)^T w$$

$$dimension = (1,p)\ x\ (p,d)\ x\ (d,p)\ x\ (d,p) x\ (p,1)$$

$$dimension = (1,1) = scalar$$

As a sanity check, we checked that the result is a scalar.

c) Compute:

$$R(w) = \int_x \left|\left|\frac{df(x;w)}{dx}\right|\right|_2^2 dx$$
$$= \int_x w^T J_\phi(x) J_\phi(x)^T w\, dx$$
$$R(w) = w^T w \int_x J_\phi(x) J_\phi(x)^T\, dx$$

The expended form of $R(w)$ is the following:

$$= [w_1 \dots w_p]_{(1,p)} \int_x \begin{bmatrix} \frac{d\phi_1 x}{dx_1} & \cdots & \frac{d\phi_1 x}{dx_d} \\ \vdots & \ddots & \vdots \\ \frac{d\phi_p x}{dx_1} & \cdots & \frac{d\phi_p x}{dx_d} \end{bmatrix}_{(p,d)} \begin{bmatrix} \frac{d\phi_1 x}{dx_1} & \cdots & \frac{d\phi_p x}{dx_1} \\ \vdots & \ddots & \vdots \\ \frac{d\phi_1 x}{dx_d} & \cdots & \frac{d\phi_p x}{dx_d} \end{bmatrix}_{(d,p)} dx \begin{vmatrix} w_1 \\ \dots \\ w_p \end{vmatrix}_{(p,1)}$$

Breaking down equation (too long too fit on one line

$$\begin{bmatrix} \frac{d\phi_1 x}{dx_1} & \cdots & \frac{d\phi_1 x}{dx_d} \\ \vdots & \ddots & \vdots \\ \frac{d\phi_p x}{dx_1} & \cdots & \frac{d\phi_p x}{dx_d} \end{bmatrix}_{(p,d)} \begin{bmatrix} \frac{d\phi_1 x}{dx_1} & \cdots & \frac{d\phi_p x}{dx_1} \\ \vdots & \ddots & \vdots \\ \frac{d\phi_1 x}{dx_d} & \cdots & \frac{d\phi_p x}{dx_d} \end{bmatrix}_{(d,p)} =$$

$$= \begin{bmatrix} \left(\left(\frac{d\phi_1 x}{dx_1}\right)^2 + \cdots + \left(\frac{d\phi_1 x}{dx_d}\right)^2\right) & \cdots & \left(\left(\frac{d\phi_1(x) d\phi_p(x)}{dx_1 dx_1}\right) + \cdots + \left(\frac{d\phi_1(x) d\phi_p(x)}{dx_d dx_d}\right)\right) \\ \vdots & \ddots & \vdots \\ \left(\left(\frac{d\phi_p x}{dx_1} * \frac{d\phi_1 x}{dx_1}\right) + \cdots + \left(\frac{d\phi_p x}{dx_d} * \frac{d\phi_1 x}{dx_d}\right)\right) & \cdots & \left(\left(\frac{d\phi_p x}{dx_1}\right)^2 + \cdots + \left(\frac{d\phi_p x}{dx_d}\right)^2\right) \end{bmatrix}_{(p,p)}$$

Hence, the expended form of R(w):

$$R(w) = w_1^2 \int_x \left(\left(\frac{d\phi_1 x}{dx_1}\right)^2 + \cdots + \left(\frac{d\phi_1 x}{dx_d}\right)^2\right) dx + \cdots + w_p^2 \int_x \left(\frac{d\phi_1(x) d\phi_p(x)}{dx_1 dx_1}\right) + \cdots + \left(\frac{d\phi_1(x) d\phi_p(x)}{dx_d dx_d}\right) dx$$

d) If $\Phi(x) = x$, write $R(w)$ in terms of w alone. How is it related to the regularizers that we have seen? Assume $X = [0,1]^d$, the d-dimensional cube with side length of 1.

We see that for those conditions, $\frac{d\phi_i(x)}{dx_i'} = 0, unless\ i = i'$

$$R(w) = \int_x \left|\left|\frac{df(x;w)}{dx}\right|\right|_2^2 dx$$

$$= [w_1 \dots w_P]_{(1,p)} \int_x \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}_{(d,d)} \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}_{(d,d)} dx \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}_{p,1}$$

$$= [w_1 \dots w_P]_{(1,d)} \int_x \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}_{(d,d)} dx \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}_{d,1}$$

$$= [w_1 \dots w_P]_{(1xd)} \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}_{(d,d)} dx \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}_{d,1}$$

It is to note that the integral above in the matrix resulted in 1 for non-zero elements because $X = [0,1]^d$.

$$= w^T = \sum_{i=1}^{d} w_i^2$$

$$= \|w\|_2^2$$

Hence, we can see that it gives the regularizer L2-Norm Regularizer (Ridge Regularizer).

e) We know consider X = [0,1] , so the input x = x is a scalar.

$$\int_0^1 |g_1'(x)|^2 \, dx = \int_0^1 (2\pi \cos(2\pi x))^2 dx$$

$$= 4\pi^2 \int_0^1 \cos^2(2\pi x) dx$$

$$\int_0^1 |g_2'(x)|^2 = \int_0^1 (20\pi \cos(20\pi x))^2 dx$$

$$= 20^2\pi^2 \int_0^1 \cos^2(20\pi x) dx$$

$$\frac{20^2\pi^2}{2} = 200\pi^2$$

f) Consider the following feature map based on the Fourier bases:

$$\phi(x) = [\sin(1\ x\ 2\pi x), \dots, \sin(k\ x\ 2\pi x), \dots, \sin(p\ x\ 2\pi x)]^T$$

Note $\phi(x) \epsilon R^p$. Compute $R(w)$

$$R(w) = \int_x \left\| \frac{df(x;w)}{dx} \right\|_2^2 dx$$

$$= \begin{bmatrix} w_1 \dots w_p \end{bmatrix}_{1,p} \int_x \begin{bmatrix} \left(\frac{d\phi_1}{dx}\right)^2 & \dots & \frac{d\phi_1\phi_p}{dxdx} \\ \vdots & \ddots & \vdots \\ \frac{d\phi_1\phi_p}{dxdx} & \dots & \left(\frac{d\phi_p}{dx}\right)^2 \end{bmatrix}_{p,p} dx \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}_{d,1}$$

$$\int_0^1 \left(\frac{d\phi_k}{dx}\right)^2 dx = \int_0^1 \left(2\pi k cos(2\pi k x)\right)^2 dx = \frac{(2\pi k)^2}{2}$$

It is to note that all other integrations that combine partial derivatives of different ϕ is 0.

$$R(w) = \int_x \left\| \frac{df(x;w)}{dx} \right\|_2^2 dx$$

$$\begin{bmatrix} w_1 \dots w_p \end{bmatrix}_{1,p} \begin{bmatrix} \frac{(2\pi k)^2}{2} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \frac{(2\pi k)^2}{2} \end{bmatrix}_{p,p} \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}_{p,1}$$

Hence:

$$R(w) = 2\pi^2\left(w_1^2 + \cdots + p^2 w_p^2\right) = 2\pi^2 \sum_{k=1}^{p} k^2 w_k^2$$

g) We can say that the regularizers penalizes more function that are less smooth.

3.2) Bayesian Interpretation of Regularizer

a) Suppose that our prior distribution for the parameters w is a Gaussian distribution with mean zero and covariance $\sum_W$ that is:

$$P\{w\} = \frac{\exp\left(-\frac{1}{2} w^T \sum_w^{-1} w\right)}{(2\pi)^{\frac{p}{2}} \sqrt{\det\left(\sum_W\right)}}$$

Suppose the same gaussian noise model is the same as:

$$P\{y^i | x^i, w\} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}\left(y^i - w^T x^i\right)^2}$$

$$P(D|w)P(w) = \frac{1}{\sqrt{2\pi\sigma^2}}\left[\prod_{i=1}^{N} \exp\left(-\frac{1}{2\sigma^2}\left(y^i - w^T x^i\right)^2\right] * \frac{1}{(2\pi)^{\frac{p}{2}}\sqrt{\det\left(\sum_w\right)}} \exp\left(\frac{1}{2} w^T \sum\nolimits_w^{-1} w\right)$$

We now remove terms that are not dependent on w

$$\widehat{w} \leftarrow argmax(w) \frac{1}{\sqrt{\det\left(\sum_w\right)}} \exp\left(-\frac{1}{2\sigma^2}\left(y^i - w^T x^i\right)^2\right) - \frac{1}{2} w^T \sum_w^{-1} w$$

$$\widehat{w} \leftarrow argmax(w) - \frac{1}{2}\ln\left(det\left(\sum_w\right)\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(y^i - w^T x^i\right)^2 - \frac{1}{2} w^T \sum_w^{-1} w$$

b) Consider $\sum_w = \sigma_w^2 I$

$$\widehat{w} \leftarrow argmax(w) - \frac{1}{2}\ln\left(\sigma_w^{2p}\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(y^i - w^T x^i\right)^2 - \frac{1}{2\sigma_w^2} w^T w$$

Multiply everything by -2 and convert argamx to argmin

$$\widehat{w} \leftarrow argmin(w)\ ln\left(\sigma_w^{2p}\right) + \frac{1}{\sigma^2}\sum_{i=1}^{N}\left(y^i - w^T x^i\right)^2 + \frac{1}{\sigma_w^2} w^T w$$

Only keep terms that are dependent on w

$$\widehat{w} \leftarrow argmin(w)\frac{1}{2}\sum_{i=1}^{N}\left(y^i - w^T x^i\right)^2 + \frac{\sigma^2}{2\sigma_w^2}w^T w$$

c) Consider the Laplace Distribution:

$$P\{w\} = \frac{1}{2b}\exp\left(-\frac{\|w\|_1}{b}\right)$$

$$\widehat{w} \leftarrow argmax(w)\frac{1}{\sqrt{2\pi\sigma^2}}\left[\prod_{i=1}^{N}\exp\left(\frac{1}{2\sigma^2}\left(y^i - w^T x^i\right)^2\right)\right] * \frac{1}{2b}\exp\left(\frac{\|w\|_1}{b}\right)$$

Removing therms not depend on w

$$\widehat{w} \leftarrow argmax(w)\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(y^i - w^T x^i\right)^2 - \frac{\|w\|_1}{b}\right)$$

Taking ln of the terms and multiply by -1 converting argmax to argmin

$$\widehat{w} \leftarrow argmin(w)\frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(y^i - w^T x^i\right)^2 + \frac{\|w\|_1}{b}$$

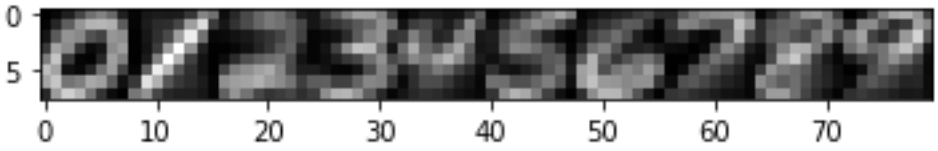
$$\widehat{w} \leftarrow argmin(w)\frac{1}{2}\sum_{i=1}^{N}\left(y^i - w^T x^i\right)^2 + \frac{\sigma^2}{b}\|w\|_1$$

d) The previous MAP estimates are related to the **L1 regularizer**.

Question 4 Handwritten Digit Classification:

The code for section 4 can be found in the submitted code.

4.0) The plot and code for the 10 means is shown underneath:



4.1.1a-b) The result when K = 1 for the training accuracy was 1, and the test accuracy 0.968. When K =15, the training accuracy was 0.959, and the testing accuracy was 0.958. The accuracy for both training and testing lowered a bit when we increased the value of K.

```
#print("K = 1: Train Accuracy = %f" %(train_acc_1))
print(f" K = 1: Train Accuracy: {train_acc_1:f}, Test Accuracy: {test_acc_1:f}\n
K = 15: Train Accuracy: {train_acc_15:f}, Test Accuracy: {test_acc_15:f}")
```

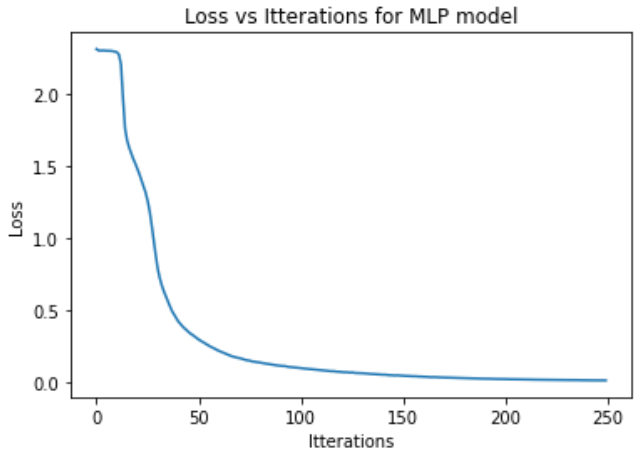
```
 K = 1: Train Accuracy: 1.000000, Test Accuracy: 0.968750
K = 15: Train Accuracy: 0.959429, Test Accuracy: 0.958500
```

4.1.2 When K > 1, the tie-breaking is done by scipy.stats.mode() by selecting the smallest of the tied values.

4.1.3 Using 10-fold cross validation, the highest validation accuracy averaged across folds occurs at k = 1 with an accuracy of 0.964. Underneath is the results of the average accuracy for each K.

```
At k = 1: Validation Accuracy = 0.9657142857142856
At k = 2: Validation Accuracy = 0.9558571428571427
At k = 3: Validation Accuracy = 0.9648571428571427
At k = 4: Validation Accuracy = 0.9629999999999999
At k = 5: Validation Accuracy = 0.9624285714285715
At k = 6: Validation Accuracy = 0.9597142857142857
At k = 7: Validation Accuracy = 0.9592857142857143
At k = 8: Validation Accuracy = 0.9571428571428571
At k = 9: Validation Accuracy = 0.9555714285714284
At k = 10: Validation Accuracy = 0.9545714285714284
At k = 11: Validation Accuracy = 0.9545714285714286
At k = 12: Validation Accuracy = 0.9528571428571428
At k = 13: Validation Accuracy = 0.9524285714285714
At k = 14: Validation Accuracy = 0.9500000000000002
At k = 15: Validation Accuracy = 0.9484285714285715
```

4.1.1 A MLP Neural Network Classifier was designed. In the first layer, there was one unit for each pixel and 10 units for the output layer. Originally, 500 iterations were used, but the code stopped before 500 iterations because the loss converged before reaching the max iteration. For this reason, we lowered the max iteration to 250. The accuracy for the test label was 0.979 (Code in notebook).




```
The accuracy score is: 0.963
```

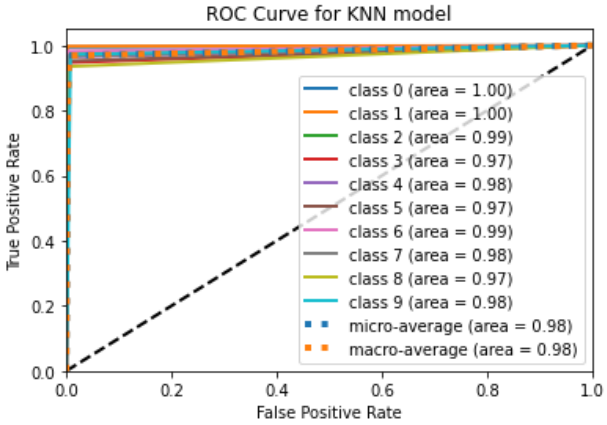
4.2.2 The gridsearch method was used to find the best parameters for our SVM model. Different values of C, gamma, and kernel was tried and 5 cross fold validation was used. The accuracy of the SVM model on the test set was 0.979 (Code in notebook)

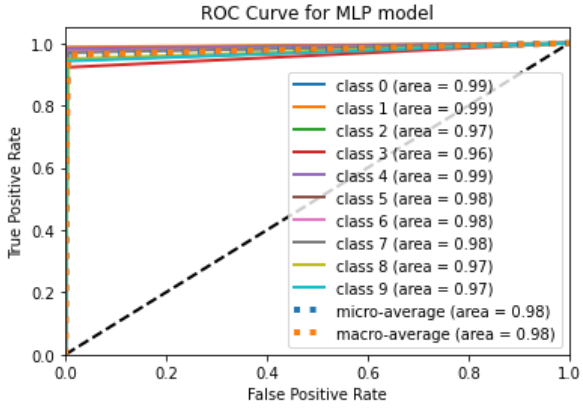
4.2.3 The grid search method was used for the AdaBoost Classifier method. We decided to use the 'SAMM.R' algorithm. The accuracy on the test set using AdaBoost was 0.9005 (Code in notebook).

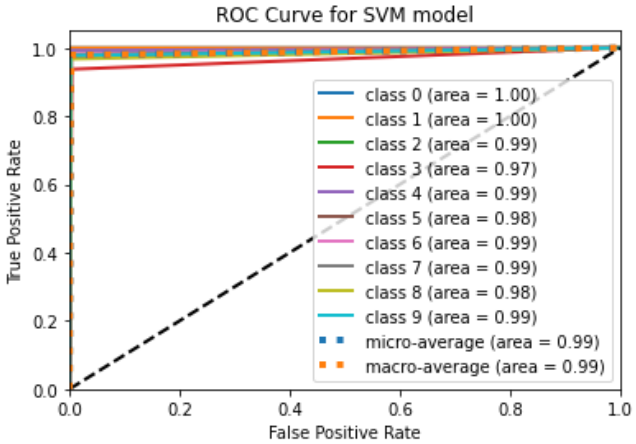
4.3 In this section, we used different metrics to compare our models. We used the ROC curve, the confusion matrix, and the accuracy, precision, and recall for each model.

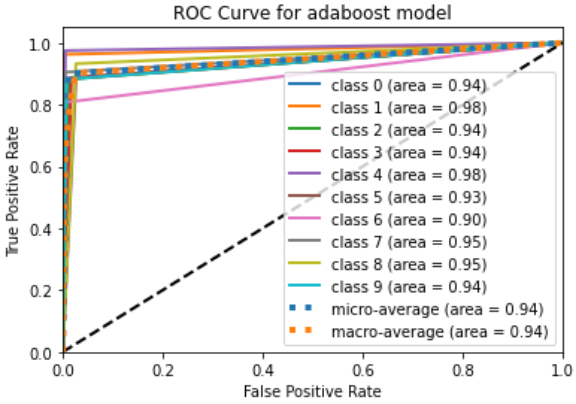
ROC: The average from least to highest for each model was the following:

- Adaboost – average = 0.94
- KNN – average = 0.98
- ROC curve – average 0.98
- SVM – average = 0.99

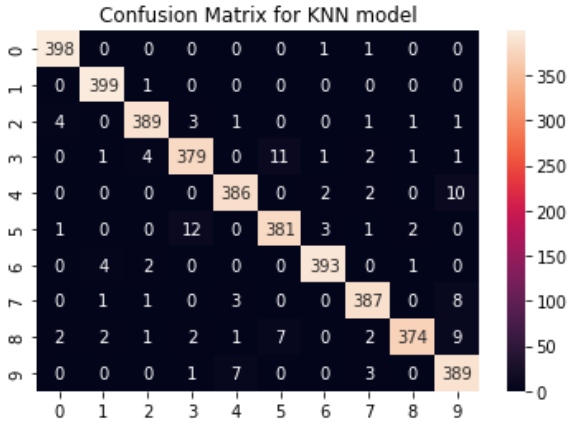


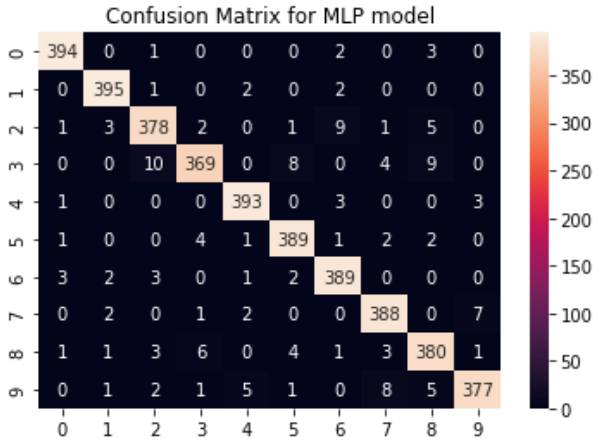


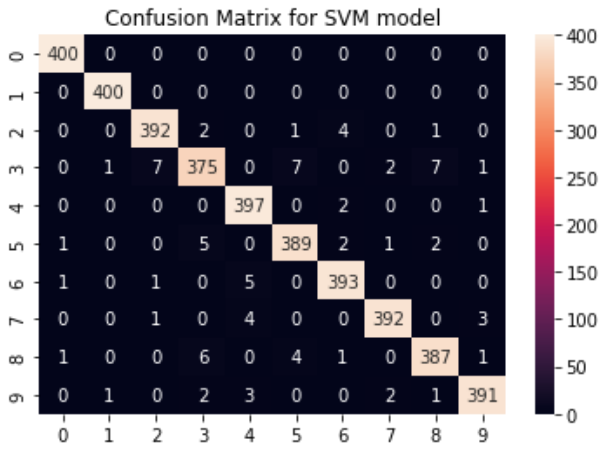


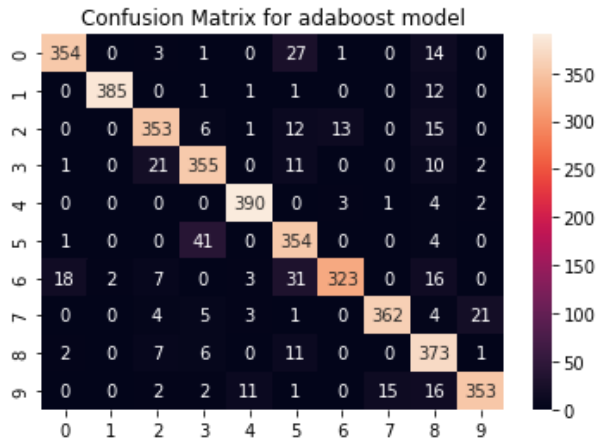


Confusion Matrix:









From the confusion matrix, we can see that the SVM has the highest accuracy and adaboost has the lower accuracy.

Accuracy, Precision (Micro, Macro), Recall (Micro, Macro):

We computed the results for the accuracy, precision, and recall of each model. The results are shown underneath. SVM had the highest score for each of those metrics, followed by KNN, then MLP, and finally adaboost.

Model	Accuracy	Precision (Micro)	Precision (Macro)	Recall (Micro)	Recall (Macro)
KNN	0.96875	0.96875	0.96897	0.96875	0.96875
MLP	0.963	0.963	0.963041	0.963	0.963
SVM	0.979	0.979	0.978968	0.979	0.979
adaboost2	0.9005	0.9005	0.905485	0.9005	0.9005

The results are expected. SVM are often used for image recognition. I thought that MLP would get a higher accuracy compared to SVM. However, this article goes through image classification using MLP and SVM and had higher accuracy using SVM [1]. SVM works well when there's a clear margin of separation between classes and works well in high dimensional spaces [2] which is the case for our problem.

The lowest result was AdaBoost. This was not expected as ad boost normally is used to increase the accuracy of classifiers. The data might be overfitted. Another reason could be the presence of outliers. Adaboost is sensitive to outliers. The loss function for Adaboost is defined das the following:

$$L\big(y, f(x)\big) = e^{-yf(x)}$$

This loss function penalizes outliers strongly.

For the computational time, it was hard for us to determine which method was more computationally expensive since some methods used grid search, and not the same number of parameters were used. However, based on research we found the big Oh (O) for each method:

- SVM $O(nd^2)\ if\ d < n, where\ d\ is\ the\ dimension$

When dealing with a large dataset, the O notation of SVM is $O(kd)$ where k is the number of support vectors and d the number of data points ($0 < k \leq d$) [3].

The time complexity of KNN is O(nd) where n is the total number of data points and d the total number of features. [4]

Between SVM and KNN, we can see that the time complexity of SVM is higher. From our observations, it seemed that the MLP took longer. Also, based on this paper, the time complexity of MLP vs SVM was higher for each test [5]

Finally, adaboost will take long as its time complexity is trivially O(t f), where f is the runtime of the weak learner in use.

Reference

[1] Thomas, M. (2020, April 13). *Comparing SVM and MLP machine learning models*. Medium. Retrieved November 23, 2022, from https://becominghuman.ai/comparing-svm-and-mlp-machine-learning-models-348d08efea6b

[2] K, D. (2020, December 26). *Top 4 advantages and disadvantages of support vector machine or SVM*. Medium. Retrieved November 23, 2022, from https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107

[3] Banerjee, A. (2020, August 31). *Computational complexity of SVM*. Medium. Retrieved November 23, 2022, from https://alekhyo.medium.com/computational-complexity-of-svm-4d3cacf2f952

[4] Shah, R. (2021, March 5). *Introduction to K-nearest neighbors (knn) algorithm*. Medium. Retrieved November 23, 2022, from https://ai.plainenglish.io/introduction-to-k-nearest-neighbors-knn-algorithm-e8617a448fa8

[5] *Training time comparison between MLP and SVM. - researchgate*. (n.d.). Retrieved November 24, 2022, from https://www.researchgate.net/figure/Training-time-comparison-between-MLP-and-SVM_fig3_216890094