# Theoretical Assignment

## (1) SVM with multiple classes.

Since the data is linearly separable, there exists $w^* = w_1^*, \dots, w_K^*$ such that $y_i = \arg\max_y w_y^* \cdot x_i$ for every $i$. Notice that for every $c > 0$ the classifier $cw^* = cw_1^*, \dots, cw_K^*$ is the same classifier as $w^*$ because $\arg\max_{j\in[K]} cw_j^* \cdot x = \arg\max_{j\in[K]} w_j^* \cdot x$ for every $x \in \mathcal{X}$.

By the correctness of the true classifier $w^*$ it holds that $w_j^* \cdot x_i - w_{y_i}^* \cdot x_i < 0$ for every $j \neq y_i$. For every $i$ define $\gamma_i = -\max_{j\neq y_i}\left(w_j^* \cdot x_i - w_{y_i}^* \cdot x_i\right)$ and by the former statement $\gamma_i > 0$ for all $i$. Therefore,

$$f(cw^*) = \frac{1}{n}\sum_{i=1}^{n} \ell(cw^*, x_i, y_i) = \frac{1}{n}\sum_{i=1}^{n} \max_{j\in[K]}\left(cw_j^* \cdot x_i - cw_{y_i}^* \cdot x_i + \mathbb{1}(j \neq y_i)\right)$$
$$= \frac{1}{n}\sum_{i=1}^{n} \max\left\{0, \max_{j\neq y_i}\left(cw_j^* \cdot x_i - cw_{y_i}^* \cdot x_i + 1\right)\right\}$$
$$= \frac{1}{n}\sum_{i=1}^{n} \max\{0, 1 - c\gamma_i\} \leq \frac{1}{n}\sum_{i=1}^{n} |1 - c\gamma_i| \leq \left|1 - c\min_i \gamma_i\right|$$

Denote $\gamma = \min_i \gamma_i > 0$. So, by taking $c \to \frac{1}{\gamma}$ we get that $f(cw^*) \to 0$ ■

## (2) Expressivity of ReLU networks.

Denote: $ReLU(x) = h(x) = \max\{0, x\}$ and recall the following identity: $\max\{x_1, x_2\} = \frac{x_1+x_2}{2} + \frac{|x_1-x_2|}{2}$. Let $z_0 = (x_1, x_2)$ be the input of the neuron natural network. Consider the following learned parameter matrix $W$ and learned parameter vector $w$:

$$W = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix}; \; w = \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Therefore, by using one hidden layer with 4 neurons and ReLU activations we have:

$$z_1 = h(W \cdot z_0) = \begin{pmatrix} h(w_{11}x_1 + w_{12}x_2) \\ h(w_{21}x_1 + w_{22}x_2) \\ h(w_{31}x_1 + w_{32}x_2) \\ h(w_{41}x_1 + w_{42}x_2) \end{pmatrix} = \begin{pmatrix} \max\{0, w_{11}x_1 + w_{12}x_2\} \\ \max\{0, w_{21}x_1 + w_{22}x_2\} \\ \max\{0, w_{31}x_1 + w_{32}x_2\} \\ \max\{0, w_{41}x_1 + w_{42}x_2\} \end{pmatrix}$$

$$f(x_1, x_2) = w \cdot z_1$$
$$= w_1 \max\{0, w_{11}x_1 + w_{12}x_2\} + w_2 \max\{0, w_{21}x_1 + w_{22}x_2\}$$
$$+ w_3 \max\{0, w_{31}x_1 + w_{32}x_2\} + w_4 \max\{0, w_{41}x_1 + w_{42}x_2\}$$
$$= \frac{1}{2}\max\{0, x_1 + x_2\} + \frac{1}{2}\max\{0, -x_1 - x_2\} + \frac{1}{2}\max\{0, x_1 - x_2\}$$
$$+ \frac{1}{2}\max\{0, -x_1 + x_2\} = \frac{x_1 + x_2}{2} + \frac{|x_1 - x_2|}{2} = \max\{x_1, x_2\} \; ■$$

## (3) Soft SVM with $\ell^2$ penalty.

(a) Notice that the objective that we want to minimize contains each $\xi_i$ as squared so the non-negativity of $\xi_i$ does not change the objective function value and so the optimal value. Moreover, let's say that we don't have the non-negativity constraints. Those, if there exists $\xi_i < 0$ which satisfies the constraint $y_i(w^T x_i + b) \geq 1 - \xi_i$ in the optimal solution, then $-\xi_i > 0$ also satisfies the constraint and leads to same contribute in the objective function. Therefore, each optimal solution of the problem without the non-negativity constraints can be achieved with the non-negativity constraints. Of course, all possible solutions of the problem with the non-negativity constraint contains in the possible solutions without these constraints.

(b) The Lagrangian of this problem:

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) &= \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 + \sum_{i=1}^{n}\alpha_i\left(1 - \xi_i - y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b)\right) \\ &= \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{2}\|\boldsymbol{\xi}\|^2 + \sum_{i=1}^{n}\alpha_i - \boldsymbol{\alpha}^T\boldsymbol{\xi} - \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{w}^T\boldsymbol{x_i} \\ &- b\sum_{i=1}^{n}\alpha_i y_i\end{aligned}$$

(c) The derivative of the Lagrangian:

$$\nabla_{\boldsymbol{w},b,\boldsymbol{\xi}}\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \begin{pmatrix}\nabla_{\boldsymbol{w}}\mathcal{L} \\ \nabla_b\mathcal{L} \\ \nabla_{\boldsymbol{\xi}}\mathcal{L}\end{pmatrix}$$

So,

$$\frac{\partial\mathcal{L}}{\partial w_j} = w_j - \sum_{i=1}^{n}\alpha_i y_i x_i^{(j)}$$

Where $x_i^{(j)}$ is the j'th coordinate in the $x_i$ data vector. And we can present it as vectors:

$$\nabla_{\boldsymbol{w}}\mathcal{L} = \boldsymbol{w} - \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x_i}$$

Where $\boldsymbol{X}$ is the matrix of all data set vectors and $\boldsymbol{y}$ is the vector of all correspondent classifications.

$$\nabla_b\mathcal{L} = -\sum_{i=1}^{n}\alpha_i y_i = -\boldsymbol{\alpha}^T\boldsymbol{y}$$

$$\nabla_{\boldsymbol{\xi}}\mathcal{L} = C\boldsymbol{\xi} - \boldsymbol{\alpha}$$

We want to minimize the Lagrangian, so set $\nabla_{\boldsymbol{w},b,\boldsymbol{\xi}}\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = 0$:

$$\nabla_{\boldsymbol{\xi}}\mathcal{L} = C\boldsymbol{\xi} - \boldsymbol{\alpha} = 0 \Leftrightarrow \boldsymbol{\alpha} = C\boldsymbol{\xi}$$

$$\nabla_b\mathcal{L} = -\sum_{i=1}^{n}\alpha_i y_i = -\boldsymbol{\alpha}^T\boldsymbol{y} = 0$$

$$\nabla_{\boldsymbol{w}}\mathcal{L} = \boldsymbol{w} - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x_i} = 0 \Leftrightarrow \boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x_i}$$

Now, by substituting the constraints we have:

$$g(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}\cdot\boldsymbol{w} + \frac{C}{2}\frac{\|\boldsymbol{\alpha}\|^2}{C^2} + \sum_{i=1}^{n}\alpha_i - \frac{\boldsymbol{\alpha}^T\boldsymbol{\alpha}}{C} - \boldsymbol{w}\cdot\boldsymbol{w}$$

$$= -\frac{1}{2}\boldsymbol{w}\cdot\boldsymbol{w} - \frac{1}{2C}\boldsymbol{\alpha}\cdot\boldsymbol{\alpha} + \sum_{i=1}^{n}\alpha_i$$

$$= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{x_i}\cdot\boldsymbol{x_j} - \frac{1}{2C}\sum_{i=1}^{n}\alpha_i^2 + \sum_{i=1}^{n}\alpha_i$$

(d) The dual function is defined as:

$$g(\boldsymbol{\alpha}) = \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha})$$

The dual problem under the constraint $\sum_{i=1}^{n}\alpha_i y_i = 0$ is:

$$\max_{\boldsymbol{\alpha}\geq\boldsymbol{0}} g(\boldsymbol{\alpha}) = \max_{\boldsymbol{\alpha}\geq\boldsymbol{0}} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{x_i}\cdot\boldsymbol{x_j} - \frac{1}{2C}\sum_{i=1}^{n}\alpha_i^2 + \sum_{i=1}^{n}\alpha_i$$

$$= \min_{\boldsymbol{\alpha}\geq\boldsymbol{0}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{x_i}\cdot\boldsymbol{x_j} + \frac{1}{2C}\sum_{i=1}^{n}\alpha_i^2 - \sum_{i=1}^{n}\alpha_i$$

## (4) Gradient of cross-entropy loss over softmax.

First, we simplify the expression of $\ell_y(w)$:

$$\ell_{\boldsymbol{y}}(\boldsymbol{w}) = -\boldsymbol{y}\cdot\log\big(softmax(\boldsymbol{w})\big) = -\sum_{i=1}^{d} y_i \log\left(\frac{e^{w_i}}{\sum_{j=1}^{d} e^{w_j}}\right)$$

$$= -\sum_{i=1}^{d} y_i w_i - y_i \log\left(\sum_{j=1}^{d} e^{w_j}\right) =$$

(*) Notice that for one-hot vector $y$ it holds that $\sum_i y = 1$.

Second, consider the partial derivative:

$$\frac{\partial \ell_{\boldsymbol{y}}}{\partial w_i}(\boldsymbol{w}) = -\left(y_i - y_i\frac{e^{w_i}}{\sum_{j=1}^{d} e^{w_j}} - \sum_{k\neq i} y_j \frac{e^{w_i}}{\sum_{j=1}^{d} e^{w_j}}\right) = -y_i + \sum_{j=1}^{d} y_j \frac{e^{w_i}}{\sum_{j=1}^{d} e^{w_j}}$$

$$= -y_i + softmax(\boldsymbol{w})_i \sum_{j=1}^{d} y_j =_{(*)} -y_i + softmax(\boldsymbol{w})_i$$

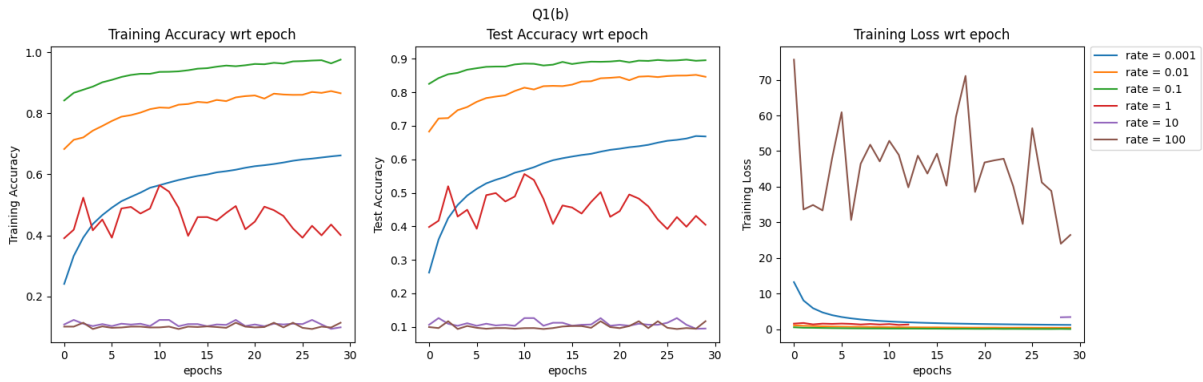
Therefore,

$$\nabla\ell_{\boldsymbol{y}}(\boldsymbol{w}) = softmax(\boldsymbol{w}) - \boldsymbol{y} \quad \blacksquare$$

# Programming Assignment

## 1. Neural Networks

(b) Plots:




Discussions:

As we can see, we achieve the best accuracy (training and test) after 30 epochs where the learning rate is 0.1 (then 0.01 and 0.001). Learning rate of 10 and 100 gives very bad accuracy. The training loss plot gives us the idea that learning rate of 100 is very unstable and the loss is very high for every epoch of this rate. On the other hand, learning rate of 0.001 gives us initially bad loss (until epoch 15), and then it sticks together to the rates 0.1 and 0.01 whose get close to 0 loss.

The learning rate determines the size of the SGD steps in each iteration. For big rates (i.e., 10 or 100), the steps might be too large, and we can easily "escape" the desired minimum. That's why the loss of rate=100 is so unstable. On the other hand, if the rates are too small (i.e., 0.001), the process might end before getting to the desired minimum. Other explanation may be that we get close enough to a local minimum of the loss which is not the optimum, and the tiny steps bring us to stuck there and not to find the optimum point.

(c) The test accuracy over all data set (training set and test set) of the final epoch is: 94.1%

(d) Improvement suggestion: A network of 3 levels: [768,400,10] with 30 epochs, mini baches on size 4 and learning rate of 0.1 gives (on the entire data) a test accuracy of 96.49% (0.9649).

## 2. Training a deep Convolutional Neural Network

(a) The validation accuracy of the Adam optimization is larger than the SGD approach as the loss of Adam optimization is smaller.

<u>Running results</u>

SGD with the given parameters

```
Epoch [10/10], Step [71/71], Loss: 1.1394
Accuracy of the network on the 5000 validation images: 49.58 %
```

Adam optimization, lr=0.0005

```
Epoch [10/10], Step [71/71], Loss: 0.6372
Accuracy of the network on the 5000 validation images: 61.72 %
```

(b) Using Adam with lr=0.05. The training process is much worse than the former learning rate of 0.00005.

```
Epoch [10/10], Step [71/71], Loss: 2.2048
Accuracy of the network on the 5000 validation images: 15.42 %
```

(c) Adam without Batch Normalization or Dropout layers: The learning loss starts from 2.04 and decreases by approximately 0.1 to 0.93 at the 10th epoch. The former Adam optimization (with these layers) gives us a batter loss, that starts from 1.88 and end with 0.637. It is not seemed that this change does not accelerate the learning process.

```
Epoch [10/10], Step [71/71], Loss: 0.9331
Accuracy of the network on the 5000 validation images: 52.86 %
```