Soumya Saha

HW02

1.

$$\theta_* = \arg\min_{\theta} C \sum_{n=1}^{N} |y_n - f(x_n, \theta)| + \|W\|_2^2$$

$$\theta = [w, b]$$

$$f(x, \theta) = wx^T + b$$

The expanded objective function is,

$$\theta_*, \epsilon_* = \arg\min_{\theta, \epsilon} C \sum_{n=1}^{N} (\epsilon_n^+ + \epsilon_n^-) + \|W\|_2^2$$

S.t. $\forall n \quad \epsilon_n^+ + f(x_n, \theta) - y_n \geq 0 \qquad \forall n \quad \epsilon_n^+ \geq 0$

$\forall n \quad \epsilon_n^- - f(x_n, \theta) + y_n \geq 0 \qquad \forall n \quad \epsilon_n^- \geq 0$

(a) The Lagrange function for the constrained formulation is,

$$\mathcal{L}(w, b, \epsilon^+, \epsilon^-, \lambda_1, \lambda_2, \lambda_3, \lambda_4) = C \sum_{n=1}^{N} (\epsilon_n^+ + \epsilon_n^-) + \|W\|_2^2 - \sum_{n=1}^{N} \lambda_{1n} (\epsilon_n^+ + wx_n^T + b - y_n)$$

$$- \sum_{n=1}^{N} \lambda_{2n} (\epsilon_n^- - wx_n^T - b + y_n) - \sum_{n=1}^{N} \lambda_{3n} \epsilon_n^+ - \sum_{n=1}^{N} \lambda_{4n} \epsilon_n^-$$

$\lambda_{1n}, \lambda_{2n}, \lambda_{3n}, \lambda_{4n}$ are the lagrange multipliers.

(b) $$\nabla_w L = 2w - \sum_{n=1}^{N} \lambda_{1n} x_n + \sum_{n=1}^{N} \lambda_{2n} x_n = 0$$

$$w = \frac{1}{2} \sum_{n=1}^{N} (\lambda_{1n} - \lambda_{2n}) x_n$$

$$\nabla_b L = -\sum_{n=1}^{N} \lambda_{1n} + \sum_{n=1}^{N} \lambda_{2n} = 0 \Rightarrow \sum_{n=1}^{N} (\lambda_{1n} - \lambda_{2n}) = 0$$

$$\nabla_{\epsilon_n^+} L = C - \lambda_{1n} - \lambda_{3n} = 0 \Rightarrow \lambda_{1n} + \lambda_{3n} = C$$

$$\nabla_{\epsilon_n^-} L = C - \lambda_{2n} - \lambda_{4n} = 0 \Rightarrow \lambda_{2n} + \lambda_{4n} = C$$

Now, we will substitute the value of w in the primal,

$$L = C \sum_{n=1}^{N} (\epsilon_n^+ + \epsilon_n^-) + \left\| \sum_{m=1}^{N} \frac{(\lambda_{1m} - \lambda_{2m}) x_m}{2} \right\|_2^2$$

$$- \sum_{n=1}^{N} \lambda_{1n} \left(\epsilon_n^+ + \sum_{m=1}^{N} \frac{(\lambda_{1m} - \lambda_{2m}) x_m}{2} \cdot x_n^T + b - y_n \right) \quad \ldots -$$

Scanned by CamScanner

$$- \sum_{n=1}^{N} \lambda_{2n}\left(\epsilon_n^- - \sum_{m=1}^{N} \frac{(\lambda_{1m}-\lambda_{2m})x_m}{2} x_n^T - b + y_n\right) - \sum_{n=1}^{N} \lambda_{3n}\epsilon_n^+ - \sum_{n=1}^{N} \lambda_{4n}\epsilon_n^-$$

$$L = C\sum_{n=1}^{N}(\epsilon_n^+ + \epsilon_n^-) + \frac{1}{4}\sum_{n=1}^{N}\sum_{m=1}^{N}(\lambda_{1n}-\lambda_{2n})(\lambda_{1m}-\lambda_{2m})x_m^T x_n$$
$$- \sum_{n=1}^{N}(\lambda_{1n}+\lambda_{3n})\epsilon_n^+ - \sum_{n=1}^{N}\sum_{m=1}^{N}\frac{1}{2}\lambda_{1m}(\lambda_{1n}-\lambda_{2n})x_n^T x_m$$
$$+ \sum_{n=1}^{N}\sum_{m=1}^{N}\frac{1}{2}\lambda_{2m}(\lambda_{1n}-\lambda_{2n})x_n^T x_m - \sum_{n=1}^{N}(\lambda_{2n}+\lambda_{4n})\epsilon_n^-$$
$$- \sum_{n=1}^{N}\lambda_{1n}b + \sum_{n=1}^{N}\lambda_{2n}b + \sum_{n=1}^{N}\lambda_{1n}y_n - \sum_{n=1}^{N}\lambda_{2n}y_n$$

$$L = C\sum_{n=1}^{N}(\epsilon_n^+ + \epsilon_n^-) + \frac{1}{4}\sum_{n=1}^{N}\sum_{m=1}^{N}(\lambda_{1n}-\lambda_{2n})(\lambda_{1m}-\lambda_{2m})x_m^T x_n$$
$$- C\sum_{n=1}^{N}(\epsilon_n^+ + \epsilon_n^-) - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(\lambda_{1n}-\lambda_{2n})(\lambda_{1m}-\lambda_{2m})x_m^T x_n$$
$$- b\sum_{n=1}^{N}(\lambda_{1n}-\lambda_{2n}) + \sum_{n=1}^{N}(\lambda_{1n}-\lambda_{2n})y_n \qquad \left[\text{Putting } \lambda_{1n}+\lambda_{3n}=C,\ \lambda_{2n}+\lambda_{4n}=C\right]$$

Putting $\sum_{n=1}^{N}(\lambda_{1n}-\lambda_{2n}) = 0$, we get,

$$L = \sum_{n=1}^{N}(\lambda_{1n}-\lambda_{2n})y_n - \frac{1}{4}\sum_{n=1}^{N}\sum_{m=1}^{N}(\lambda_{1n}-\lambda_{2n})(\lambda_{1m}-\lambda_{2m})x_m^T x_n$$

The dual is,

$$q(\lambda_1, \lambda_2) = \min_{x}(\lambda_1-\lambda_2)^T y - \frac{1}{4}(\lambda_1-\lambda_2)^T x_m^T x_n (\lambda_1-\lambda_2)$$

(Ans.)

The constraints are, $\forall n \quad \lambda_1 \geq 0,\ \lambda_2 \geq 0,\ \lambda_3 \geq 0,\ \lambda_4 \geq 0$

As, $\lambda_1 + \lambda_3 = C$ and $\lambda_2 + \lambda_4 = C$

So, $0 \leq \lambda_1 \leq C,\quad 0 \leq \lambda_2 \leq C$

Scanned by CamScanner

(c) In the dual problem, we have to solve a much simpler optimization problem which doesnot have any complex inequality constraints. The dual problem is also a convex and quadratic in expanded set of variables, whereas the primal problem was non-differentiable at some points. Also, compared to the primal, dual problem has fewer parameters.

2. $$w_* = \underset{w}{\operatorname{argmin}}\ C \sum_{n=1}^{N} \max(0, 1 - y_n(wx_n^T + b)) + \|w\|_2^2$$

(a) $$\frac{\partial L}{\partial w} = C \sum_{n=1}^{N} z_n + 2w \quad \text{where, } z_n = \begin{cases} 0 & \text{if } 1 < y_n(wx_n^T + b) \\ [0, -y_n x_n] & \text{if } 1 = y_n(wx_n^T + b) \\ -y_n x_n & \text{otherwise} \end{cases}$$

$$\frac{\partial L}{\partial b} = C \sum_{n=1}^{N} v_n \quad \text{where } v_n = \begin{cases} 0 & \text{if } 1 < y_n(wx_n^T + b) \\ [0, -y_n] & \text{if } 1 = y_n(wx_n^T + b) \\ -y_n & \text{otherwise.} \end{cases}$$

Scanned by CamScanner

2. b. To learn the model, I first calculated the objective function and the subgradient function. I initialized the value of parameters w and b as zeros of proper dimensions. Then, I have used minibatch subgradient descent to minimize the objective function. Subgradient over a batch helps in accelerating the learning process and preventing overfitting of the model. I have used a batch size of 200 for each iteration of the training process and the batch was taken from the data randomly with replacement. After lots of experimentation, I have found the learning rate 0.0009 to give better accuracies. Also, I have set the regularization parameter, C to be 1 and total iterations to perform for training purpose to be 600. After learning the model, the learned value of w and b are then used to predict on test data.

2. c.

Value of the SVC objective function at the optimal model parameters: 15.65
Classification error rate on the training data: 0.2 %

2. d.

Value of logistic regression objective function at w_{LR} and b_{LR} : 27.04
Value of SVC objective function at w_{LR} and b_{LR} : 27.75
Classification error rate of logistic regression at w_{LR} and b_{LR} on the training data: 0%
Classification error rate of SVC at w_{LR} and b_{LR} on the training data: 0%

2. e. For same weight and bias, the value of objective function is almost similar for both SVC and logistic regression. SVC will have lower generaliation error than logistic regression when the data is linearly separable and the classes are far apart i.e if there exists high margin between the data. Also, if the number of outliers is fewer in the data, SVC will perform better.

3. a. I have used PyTorch (version 0.4.1) to develop my implementation of the multi-output neural network. The neural network contains two different branches for different outputs. I have derived nn.Module base class of PyTorch to implement my neural network. I used nn.Linear to create the layers of the network. And then used proper Relu activation function at the output of the hidden layers. The weights and the biases of the Linear layers are initialized automatically with uniform distributed values. Also, one of the main benefits of using PyTorch over NumPy is the automatic backward propagation of derivatives on the basis of the computation graph created during the forward pass. To calculate the objective of the Classification output, I have used the PyTorch function BCEWithLogitsLoss to calculate the cross-entropy loss for classification and for the Localisation output, I have calculated the square of the norm of the error. To optimize the loss minimization process, I have used the Adam optimizer, which is widely used optimizer in deep learning. The learning rate 0.001 has seemed to generate good accuracies after lots of trials. To accelerate the learning process, I have used a batch size of 200 for the calculation of automatic differentiation of weights and biases. I have trained the data over 400 epochs. After the learning process, the learned weights and biases are used for predict output for new test data. For prediction, the probability values of the classification output layer were binarized to match with the test output data.

3. b.

Classification error rate on training set for $\alpha = 0.5$: 3.83%
Classification error rate on testing set for $\alpha = 0.5$: 3.17%
Mean Squared Error for localization output on training set for $\alpha = 0.5$: 2.10
Mean Squared Error for localization output on testing set for $\alpha = 0.5$: 6.92

3. c.

I calculated the Test accuracy of the Classification output and the MSE of the localisation output on the test data on the trained neural network with values of α in the range 0 to 1 with steps of 0.1. The values can be seen in the table.

α	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Accuracy	50.97	92.34	95.46	95.74	96.17	96.83	97.02	96.88	98.11	98.77	99.29
MSE	7.97	6.98	7.87	7.23	7.88	6.92	7.80	7.72	7.76	7.92	464.52

I have also plotted the Test Accuracy and Test MSE with values of Alpha with the values of alpha in the range of 0 to 0.95. And it can be seen from the graph that at α = 0.5 the MSE is minimum and the test accuracy is considerably high. Thus at this setting of α learning the two tasks provides benefit.