1 Lagrangian

1.1 Lagrangian function

The Primal objective function is given by

$$\begin{aligned}\theta_*, \epsilon_* = \arg\min_{\theta,\epsilon} & C\sum_{n=1}^{N}(\epsilon_n^+ + \epsilon_n^-) + \|\mathbf{w}\|_2^2 \\ \text{s.t.} \quad \forall n \;\; & y_n \le f(\mathbf{x}_n, \theta) + \epsilon_n^+ \\ \forall n \;\; & y_n \ge f(\mathbf{x}_n, \theta) - \epsilon_n^- \\ \forall n \;\; & \epsilon_n^+ \ge 0 \\ \forall n \;\; & \epsilon_n^- \ge 0\end{aligned}$$

Let $\alpha_n, \beta_n, \gamma_n, \delta_n$ be Lagrangian multipliers. The Lagrangian function is given as

$$\begin{aligned}\mathcal{L}(\mathbf{w}, b, \epsilon_n, \alpha, \beta, \gamma, \delta) = & C\sum_{n=1}^{N}(\epsilon_n^+ + \epsilon_n^-) + \|\mathbf{w}\|_2^2 - \sum_{n=1}^{N}\alpha_n\left((\mathbf{w}\mathbf{x}_n^T + b) + \epsilon_n^+ - y_n\right) \\ & - \sum_{n=1}^{N}\beta_n\left(y_n + \epsilon_n^- - (\mathbf{w}\mathbf{x}_n^T + b)\right) - \sum_{n=1}^{N}\gamma_n\epsilon_n^+ - \sum_{n=1}^{N}\delta_n\epsilon_n^- \\ & \text{such that } \forall n,\ \alpha_n \ge 0,\ \beta_n \ge 0,\ \gamma_n \ge 0,\ \delta_n \ge 0\end{aligned} \tag{1}$$

1.2 Lagrangian dual formulation

$$\begin{aligned}\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}, b, \epsilon_n, \alpha, \beta, \gamma, \delta) &= 2\mathbf{w} - \sum_{n=1}^{N}\alpha_n\mathbf{x}_n - \sum_{n=1}^{N} -\beta_n\mathbf{x}_n \\ 2\mathbf{w} - \sum_{n=1}^{N}\alpha_n\mathbf{x}_n + \sum_{n=1}^{N}\beta_n\mathbf{x}_n &= 0 \\ \mathbf{w} &= \frac{1}{2}\sum_{n=1}^{N}(\alpha_n - \beta_n)\mathbf{x}_n\end{aligned} \tag{2}$$

$$\begin{aligned}\nabla_b\mathcal{L}(\mathbf{w}, b, \epsilon_n, \alpha, \beta, \gamma, \delta) &= -\sum_{n=1}^{N}\alpha_n - \sum_{n=1}^{N} -\beta_n \\ \sum_{n=1}^{N}\alpha_n - \sum_{n=1}^{N}\beta_n &= 0 \\ \sum_{n=1}^{N}\alpha_n &= \sum_{n=1}^{N}\beta_n\end{aligned} \tag{3}$$

$$\begin{aligned}\nabla_{\epsilon_n^+}\mathcal{L}(\mathbf{w}, b, \epsilon_n, \alpha, \beta, \gamma, \delta) &= C - \alpha_n - \gamma_n = 0 \\ C &= \alpha_n + \gamma_n, \quad \forall n\end{aligned} \tag{4}$$

$$\begin{aligned}\nabla_{\epsilon_n^-}\mathcal{L}(\mathbf{w}, b, \epsilon_n, \alpha, \beta, \gamma, \delta) &= C - \beta_n - \delta_n = 0 \\ C &= \beta_n + \delta_n, \quad \forall n\end{aligned} \tag{5}$$

Resubstituting back into the Lagrangian, we get

$$
\begin{aligned}
\mathcal{L}(\mathbf{w}, b, \epsilon_n, \alpha, \beta, \gamma, \delta) &= C\sum_{n=1}^{N}(\epsilon_n^+ + \epsilon_n^-) + \|\frac{1}{2}\sum_{m=1}^{N}(\alpha_m - \beta_m)\mathbf{x}_m\|_2^2 \\
&- \sum_{n=1}^{N}\alpha_n \mathbf{w}\mathbf{x}_n^T + \alpha_n b + \alpha_n\epsilon_n^+ - \alpha_n y_n + \beta_n y_n + \beta_n\epsilon_n^- - \beta_n \mathbf{w}\mathbf{x}_n^T - \beta_n b \\
&= C\sum_{n=1}^{N}(\epsilon_n^+ + \epsilon_n^-) + \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(\alpha_n - \beta_n)(\alpha_m - \alpha_m)\mathbf{x}_n\mathbf{x}_m^T \\
&- \sum_{n=1}^{N}(\alpha_n - \beta_n)\mathbf{w}\mathbf{x}_n^T - \sum_{n=1}^{N}(\alpha_n - \beta_n)b + \sum_{n=1}^{N}(\alpha_n - \beta_n)y_n - \sum_{n=1}^{N}\alpha_n\epsilon_n^+ + \beta_n\epsilon_n^- \\
&= C\sum_{n=1}^{N}(\epsilon_n^+ + \epsilon_n^-) + \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(\alpha_n - \beta_n)(\alpha_m - \alpha_m)\mathbf{x}_n\mathbf{x}_m^T \\
&- \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(\alpha_n - \beta_n)(\alpha_m - \beta_m)\mathbf{x_n}\mathbf{x}_m^T - \sum_{n=1}^{N}(\alpha_n - \beta_n)b + \sum_{n=1}^{N}(\alpha_n - \beta_n)y_n - \sum_{n=1}^{N}\alpha_n\epsilon_n^+ + \beta_n\epsilon_n^- \\
&= C\sum_{n=1}^{N}(\epsilon_n^+ + \epsilon_n^-) + \sum_{n=1}^{N}(\alpha_n - \beta_n)y_n - \sum_{n=1}^{N}\alpha_n\epsilon_n^+ + \beta_n\epsilon_n^-
\end{aligned} \tag{6}
$$

The Lagrangian dual is given by $q(\lambda) = \min_{x\in\mathcal{X}} \mathcal{L}(\mathbf{x}, \lambda)$

$$
\begin{aligned}
\min \mathcal{L}(\mathbf{w}, b, \epsilon_n, \alpha, \beta, \gamma, \delta) &= C\sum_{n=1}^{N}(\epsilon_n^+ + \epsilon_n^-) + \sum_{n=1}^{N}(\alpha_n - \beta_n)y_n - \sum_{n=1}^{N}\alpha_n\epsilon_n^+ + \beta_n\epsilon_n^- \\
\text{s.t.} \quad & \forall n \;\; 0 \le \alpha_n \le C \\
& \forall n \;\; 0 \le \beta_n \le C \\
& \forall n \;\; \epsilon_n^+ \ge 0 \\
& \forall n \;\; \epsilon_n^- \ge 0 \\
& \sum_{n=1}^{N}(\alpha_n - \beta_n) = 0
\end{aligned} \tag{7}
$$

1.3 Advanatages of Dual formulation

1. The dual formulation of the problem is a concave function, even if the primal function is not convex. Hence we can use off-the-shelf optimizers to find the optimal parameters.

2. Additionally, the dual constraints are fewer and easier to solve than the primal constraints. For eg, the dual problem can have box constraints ($a \le y \le b$) compared to inequality constraints of the primal problem ($y \ge 0$), which are easier to solve.

3. In this particular problem, the primal function is non-differentiable, so it cannot be solved with standard optimizers. When strong duality holds, we can maximize the dual objective and recover the optimal paramters for primal objective by reverse mapping.

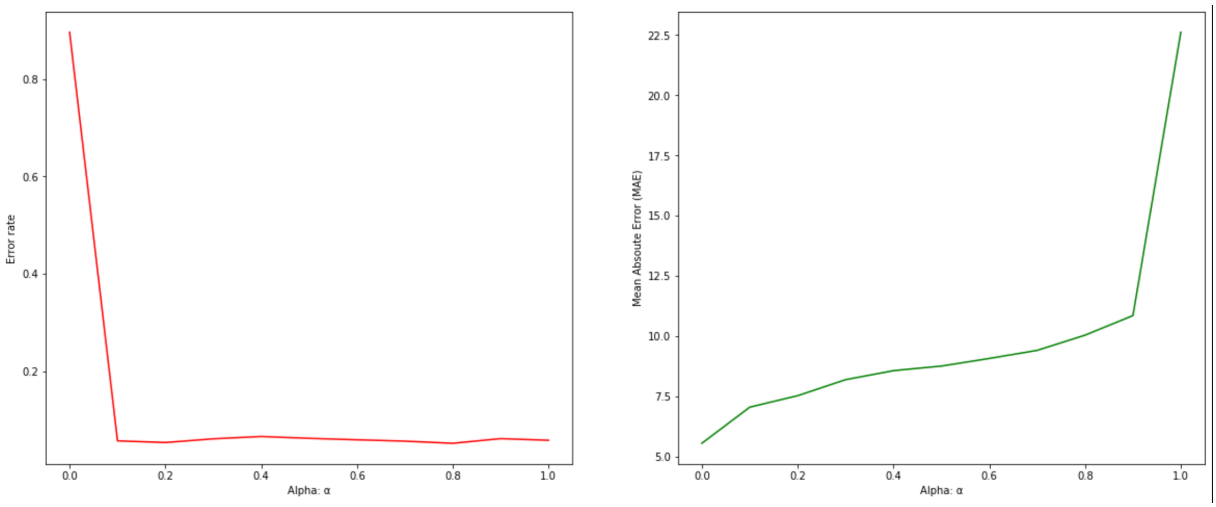
4. The dual constraints scale up with the number of data points (N). The primal constraints are linear in the number of data points as well the number of dimensions per data point.

2 Neural networks

2.1 Classification and angle error rate

1. **Training** Classification error rate = `0.001366`, Mean absolute error (MAE) = `8.03274`
2. **Validation** Classification error rate = `0.06199` Mean absolute error (MAE) for angle = `8.75306`

2.2 Effect of α



Learning the angle prediction task on its own gives the least MAE on the angles task. The error increases with the value of α. The joint learning of tasks does not give better performance.

However, using the angle prection task on its own ($\alpha = 0$), gives a poor classification accuracy. Setting $\alpha \geq 0.1$, approximately maintains accuracy. We can conclude that setting the value of $\alpha = \mathbf{0.1}$ gives good balance between classification and angle prediction accuracy.

3 More Neural networks

3.1 Architectures

1. **Optimizers**: I tried several optimizers, such as Stochastic gradient descent, AdaGrad and Adam. The best convergence was provided by **Adam**
2. **Batch size, learning rate**: I tried using different learning rates (sampled using a log scale). $\alpha \in \{1e-1, 1e-2, 1e-3, 1e-4\}$. A higher learning rate ($\geq 1e-2$), caused the model to overshoot the minimum. The batch sizes I tried were (64, 128, 256). Increasing the batch size, did not significanly improve accuracy, however lead to a significanly higher training time.
3. **Convoluational netowork**: Since the inputs are a single channel 28x28 pixel image, a convolutional network was a natural choice.
4. **BatchNorm, Dropout**: Adding batch normalization to each layer, improved accuracy. The dropout layer did not have much effect. The test was performed using an ablation study

3.2 Best Model

The Convolutional network provided the best classification accuracy. There were several ways to choose the architecture. By performing ablation study on different components, and considering the trade-off between train time and classification accuracy improvement. The procedure was as follows:

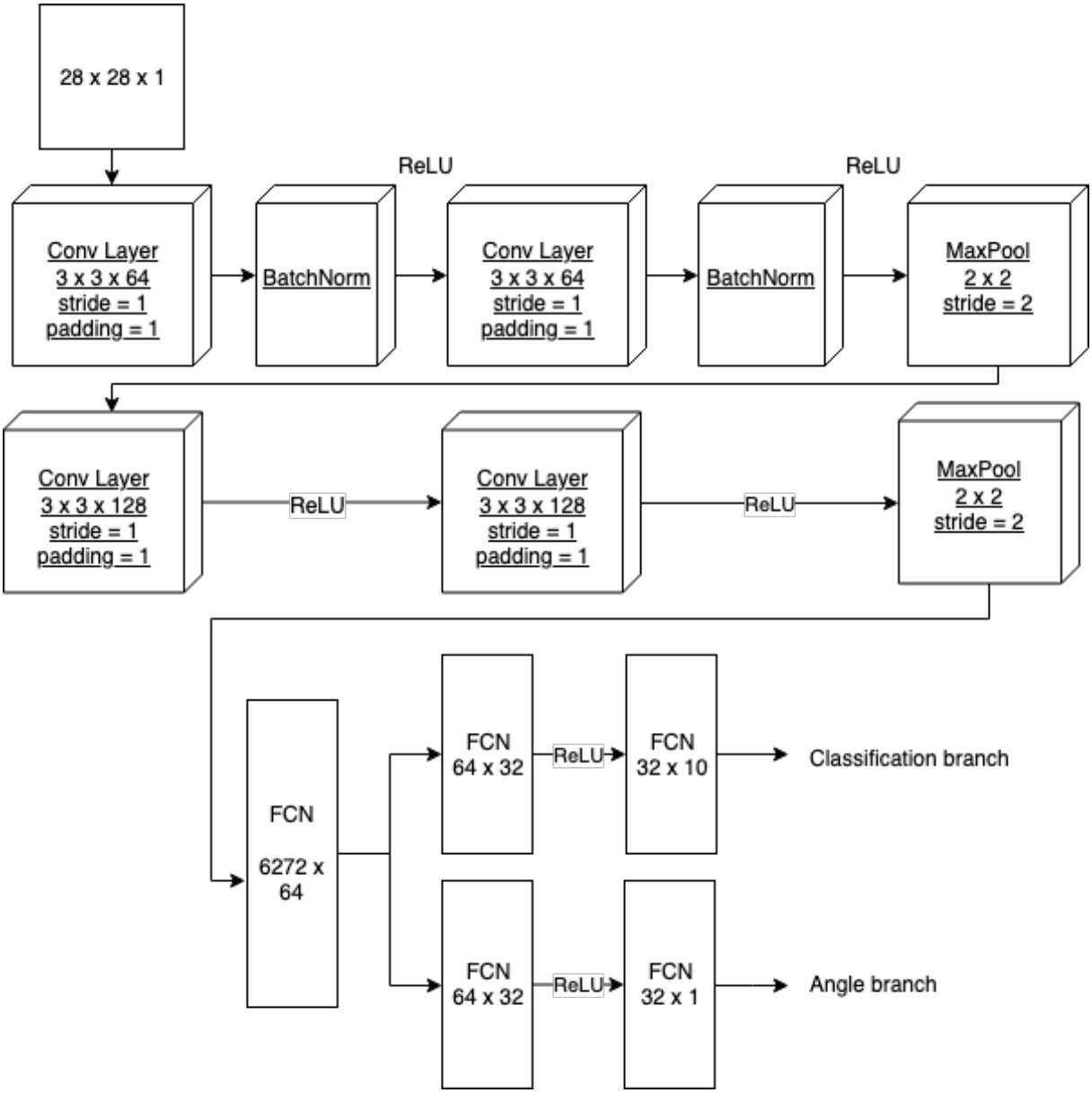
1. Fix a learning rate to $1e-4$, using **Adam** optimizer with a weight-decay of $1e-4$. Set the batch size to 64.
2. Add a (Conv - ReLU) **x 2** - MaxPool - Fully Connected layers. (+2% change)

3. Add Dropout (p = 0.8) layer before non-linearity and check accuracy. (+0%)
4. Add BatchNorm before each non-linearity and check accuracy. (+1% change)
5. Add more layers (Conv -ReLU) **x 4** and check accuracy. (+2% change, but more training time)

3.3 Architecture diagram

From the experiments performed above, the following is the final architecture of the neural network. Each layer is followed by non-linearity (ReLU).

1. Layer #1: Convoluational layer, 64 3x3 filters with stride=1 and padding=1 (To maintain input dimensions). Output: 28x28x64
2. Layer #2: BatchNorm (torch.nn.BatchNorm2d(64))
3. Layer #3: Convoluational layer, 64 3x53 filters with stride=1 and padding=1.
4. Layer #4: BatchNorm
5. Layer #5: MaxPool (torch.nn.MaxPool): stride=2, size=2x2.
6. Layer #5-7: Convoluational layers with 128 filters of size 3x3, padding=1.
7. Layer #8-10: Dense layers with one brach giving angle predictions and the other class predictions.


28 x 28 x 1
Conv Layer
3 x 3 x 64
stride = 1
padding = 1
BatchNorm
ReLU
Conv Layer
3 x 3 x 64
stride = 1
padding = 1
BatchNorm
ReLU
MaxPool
2 x 2
stride = 2
Conv Layer
3 x 3 x 128
stride = 1
padding = 1
ReLU
Conv Layer
3 x 3 x 128
stride = 1
padding = 1
ReLU
MaxPool
2 x 2
stride = 2
FCN
6272 x
64
FCN
64 x 32
ReLU
FCN
32 x 10
Classification branch
FCN
64 x 32
ReLU
FCN
32 x 1
Angle branch