CSE 455-INTRO TO PATTERN RECOGNITION

Neural Networks

Project Assignment 4

Ankit Sigroha

50245902

ankitsig@buffalo.edu

Problem 1:

In this problem set, we will train a neural network to identify the digit on a image in the MNIST data set using Tensorflow/Keras. This neural network has 10 softmax output nodes generating $\log p(t=m|x;w)$ where m=0,1,...,9. Let $x_n \in R^{28\times 28}$ be the 28×28 images, t_n be the label of the image x_n, w be the synaptic weights of the neural network, and n be the index of a pattern in the training data set.

1. Demonstrate that a neural network to maximize the log likelihood of label is one that has softmax output nodes and minimizes the criterion function of the negative log probability of training data set: $J_0(w) = -\log p(\{(x_n,t_n): n = 1,2,\cdots,\};w) = -\log \prod_n \prod_{m=0}^{9} p(t_n = m|x_n;w)$.

Demonstrate that a neural network to maximize the a posterior likelihood of observing the training data given a Gaussian prior of the weight distribution $p(w;\alpha) = N(0,\alpha I)$ is one that minimizes the criterion function with L2 regularization $J(w) = J_0(w) - \log p(w;\alpha^{-1})$.

Ans:

A classification model for a problem with classes typically generates:

y∈RCy∈RC, a vector of scores where we might use multivariate linear regression with a vector output as seen in the last chapter. Usually these scores will be arbitrary real numbers. To go from arbitrary scores y∈RCy∈RC to normalized probability estimates p∈RCp∈RC for a single instance, we use exponentiation and normalization:

$$p_i = \frac{\exp y_i}{\sum_{c=1}^{C} \exp y_c}$$

here i,c∈{1,…,C}i,c∈{1,…,C} range over classes, and pi,yi,yc pi,yi,yc refer to class probabilities and scores for a single instance. This is called the softmax function. A model that converts the unnormalized values at the end of a linear regression to normalized probabilities for classification is called the softmax classifier. We need to figure out the backward pass for the softmax function.

Let's find the partial derivative of one component of p with respect to one component of y:

$$\frac{\partial p_i}{\partial y_j} = \frac{[i=j]\exp y_i(\sum_c \exp y_c) - \exp y_i \exp y_j}{(\sum_c \exp y_c)^2} = [i=j]\, p_i - p_i p_j$$

To use the softmax function in neural network, we need to compute its derivative. If we say

$$\Sigma C = \sum_{d=1}^{e} c e z_d \text{ for } c=1$$

so that

$$y_c = \frac{e z_c}{\Sigma c y_c} = \frac{e z_c}{\Sigma C}$$

then this derivative

$\left(\frac{\partial y_i}{\partial z_j}\right)$ · ~~∂y~~ of the output y of the softmax function w.r.t its input z can be calculated as:

If i=j: $$\frac{\partial y_i}{\partial z_1} = \frac{\partial \frac{e^{z_1}}{\Sigma C}}{\partial z_i} =$$

$$= \frac{e^{z_1}\Sigma C - e^{z_i} e^{z_1}}{\Sigma^2 C}$$

$$= \frac{e^{z_1}}{\Sigma_c}\,\frac{\Sigma_c - e^{z_1}}{\Sigma_c} = \frac{e^{z_1}}{\Sigma_c}\left(1 - \frac{e^{z_1}}{\Sigma C}\right) = y_i(1-y_i)$$

If i≠j: ~~$\frac{\partial e^{z_1}}{\partial}$~~ $$\frac{\partial y_1}{\partial z_j} = \frac{\partial \frac{e^{z_1}}{\Sigma C}}{\partial z_j} = \frac{0 - e^{z_1} e^{z_j}}{\Sigma^2 C} = -\frac{e^{z_1}}{\Sigma_c}\,\frac{e^{z_j}}{\Sigma_c} = -y_i y_j$$

To derive cost function for softmax fnxn we start with likelihood function:-

the maximization of likelihood fnxn can be written as:

$$\underset{(\theta)}{\arg\max}\ \mathcal{L}(\theta \mid t, z)$$

The likelihood $\mathcal{L}(\theta \mid t, z)$ can be rewritten as joint probability of generating t & z.

$$P(t, z \mid \theta) = P(t \mid z, \theta) P(z \mid \theta)$$

$$P(t \mid z) = \prod^{C} P(t_c \mid z)^{t_c} = \prod^{C} \xi(z)_c^{t_c}$$

$$= \prod^{C} y_c^{t_c}$$

We can either maximize this likelihood or minimize negative log-likelihood which is:

$$-\log \mathcal{L}(\theta \mid t, z) = \xi(t, z) = -\log \prod_{i=c}^{C} y_c^{t_c}$$

$$= -\sum_{i=c}^{C} t_c \cdot \log(y_c)$$

Hence proved by minimizing log-likelihood.

ii) We say that

Posterior $\propto$ likelihood $\propto$ prior

or $$p(Y|X) = \frac{P(X|Y)\, p(Y)}{p(X)}$$

We maximize likelihood by minimizing the objective error function & finding its gradient.

i.e $$\frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}\{y(x_n, W_{ML}) - t_n\}^2$$

Finding posterior for given data with m data points & n features:-

$$p(W|\alpha) = \mathcal{N}(W|0, \alpha^{-1}I) = \left(\frac{\alpha}{2\pi}\right)^{\frac{(M+1)}{2}} \cdot \exp\{\frac{\alpha}{2}W$$

where α is hyper parameter

$$= \quad p(W|x, t, \alpha, \beta) \propto P(t|x, W, \beta)\, p(W|\alpha)$$

$$= \beta\tilde{E}(W) = \frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, W) - t_n)^2 + \frac{\alpha}{2}W^T W$$

$$p(W|x, t, \alpha, \beta) \propto P(t|x, W, \beta)\, P(W|\alpha)$$

$$\boxed{\beta\tilde{E}(W) = \frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, W) - t_n\}^2 + \frac{\alpha}{2}W^T W}$$

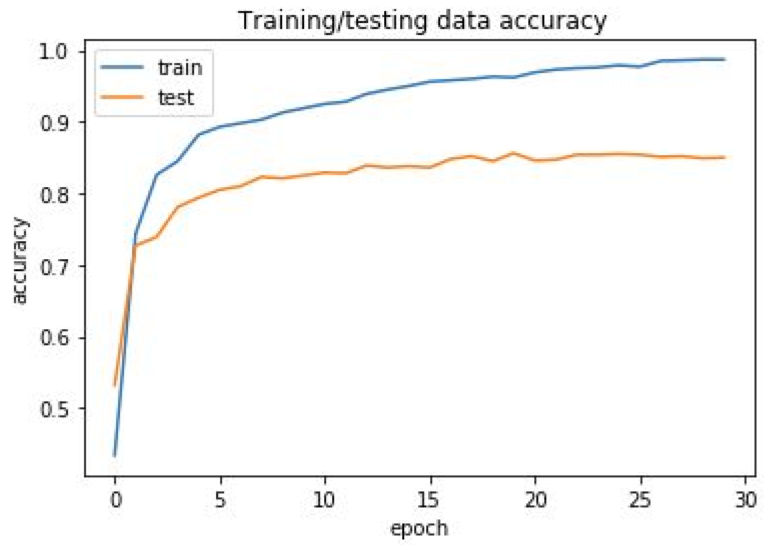
so, we can say that maximizing posterior is equivalent to minimizing L2 regularized objective error function.

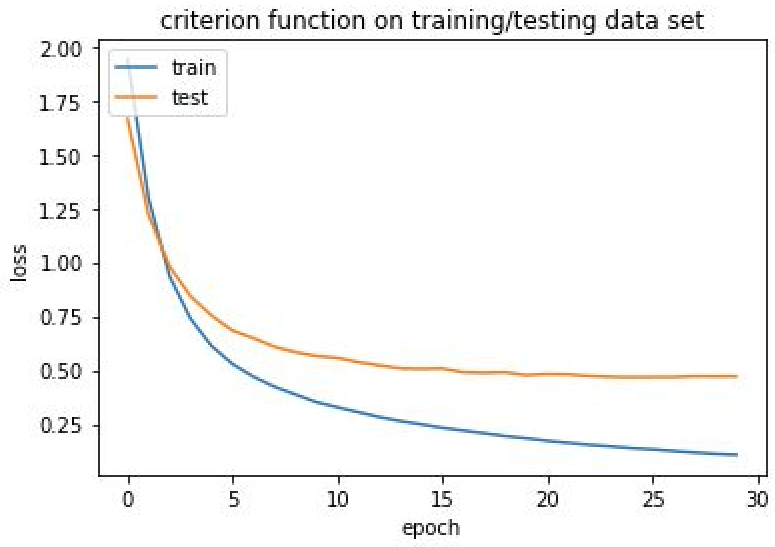
<u>Problem2:</u>

a) **Without regularization:**

Hidden layer: 1

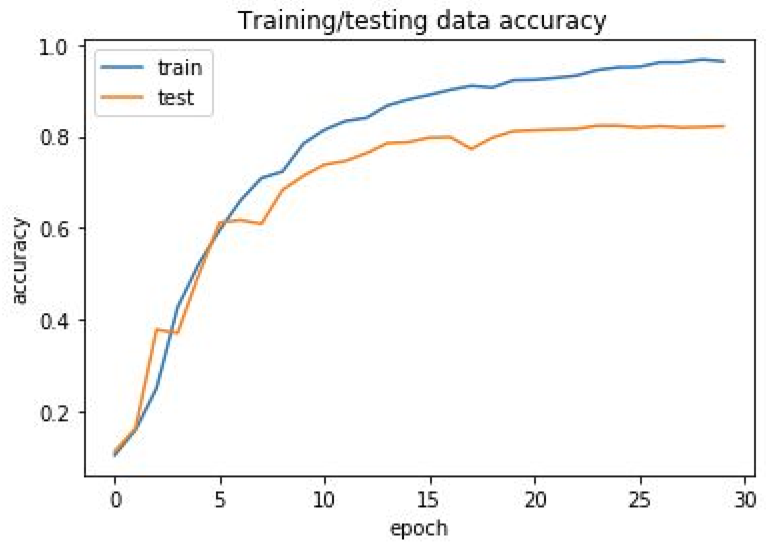
```
Test accuracy: 0.85
Train accuracy: 0.993
```

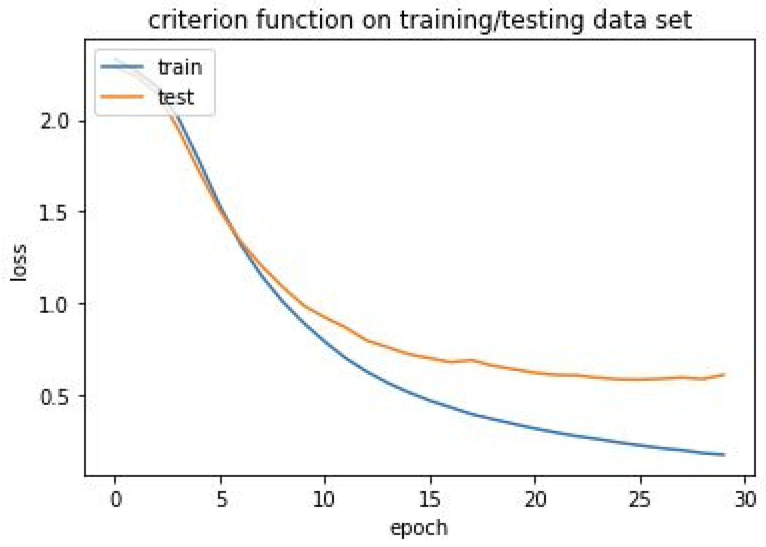




Hidden layer: 2

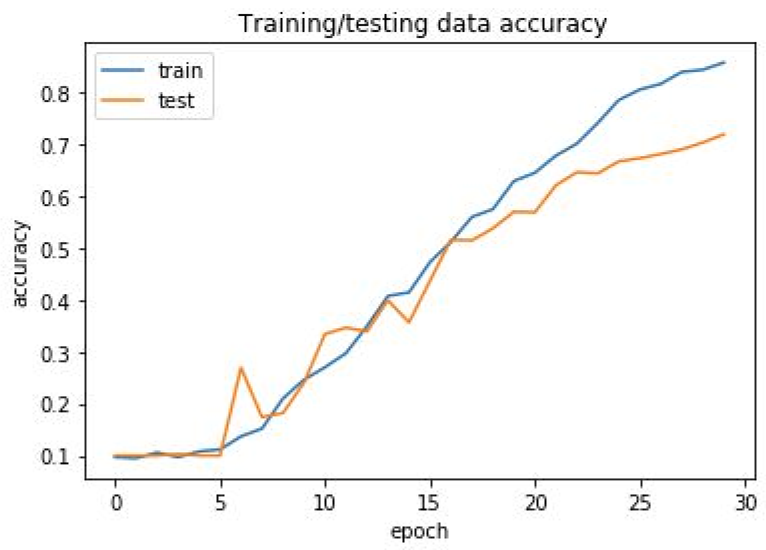
```
Test accuracy: 0.822
Train accuracy: 0.973
```

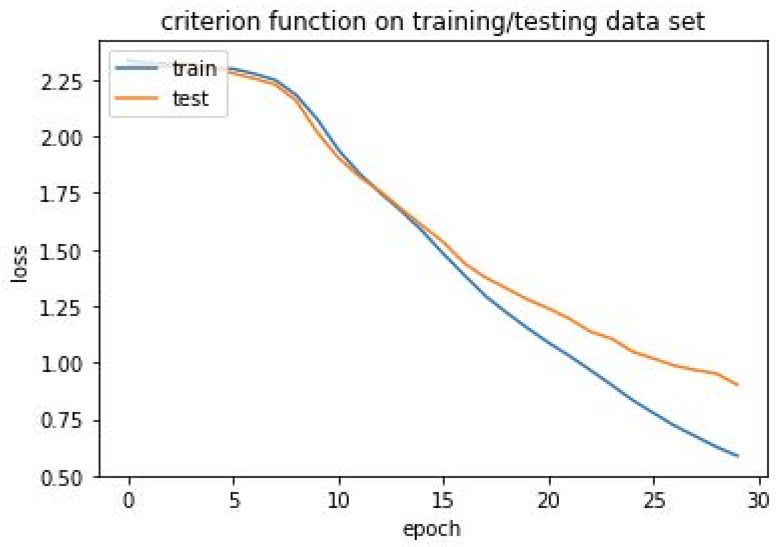




Hidden layer: 3

```
Test accuracy: 0.718
Train accuracy: 0.879
```





b) **With regularization:**

Hidden layer: 1

```
Test accuracy: 0.1
Train accuracy: 0.101
```

Hidden layer: 2

```
Test accuracy: 0.1
Train accuracy: 0.1
```

Hidden layer: 3

```
Test accuracy: 0.1
Train accuracy: 0.1
```

References:

a) https://math.stackoverflow.com/
b) Our textbook & slides
c) Lecture slides and Machine Learning slides by Dr. Chandola
d) CEDAR deptt. Of UB.