Machine Learning Engineer Nanodegree

Capstone Proposal

Ashwani Kumar
April 8, 2018

Proposal

Domain Background

One motivation for representation learning is that learning algorithms can design features better and faster than humans can. To this end, this Kaggle problem tries to find the expression depicted by a human face. Here, the goal is to find the expression from an image not train the machine to understand the emotions. The problem can be found here.

Following are some related problems on Kaggle where the learning and solutions developed for this problem can be helpful:

1. Facial Expression Prediction
2. VISUM - Facial Expression Analysis
3. Facial Expression Prediction

Problem Statement

In this Kaggle problem, the challenge is to predict the likelihood that a given photo depicts one of the seven facial expressions which in turn leads to a multi-class classification machine learning problem.

The seven classes which are to identified are *Angry*, *Disgust*, *Fear*, *Happy*, *Sad*, *Surprise*, *Neutral*.

Here we need to train a CNN which will predict (after the model is trained) if a new image when given to the model belongs to one of the above-mentioned classes.

Datasets and Inputs

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression into one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

train.csv contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image. The "pixels" column contains a string surrounded by quotes for each image. The contents of this string a space-separated pixel values in row-major order. test.csv contains only the "pixels" column and our task is to predict the emotion column.

- **The training set consists of 28,709 images.**
- **There are two test sets each containing 3589 images.**

Some sample images from the data set are:

As the data given is pixel intensity, we will normalize the data between 0-1 by dividing by 255.

There is also a class imbalance problem in the data. Class 0 has 4953 samples while Class 1 has 547 samples. In order to counter this, we will use data augmentation to generate more data.

Solution Statement

This is a classical image classification problem. Over the years, many successful image classification models have been developed. Here we will be using CNN to classify the images into 7 classes. We will also use data augmentation to generate data from the existing images so that we have more data to train our model upon.

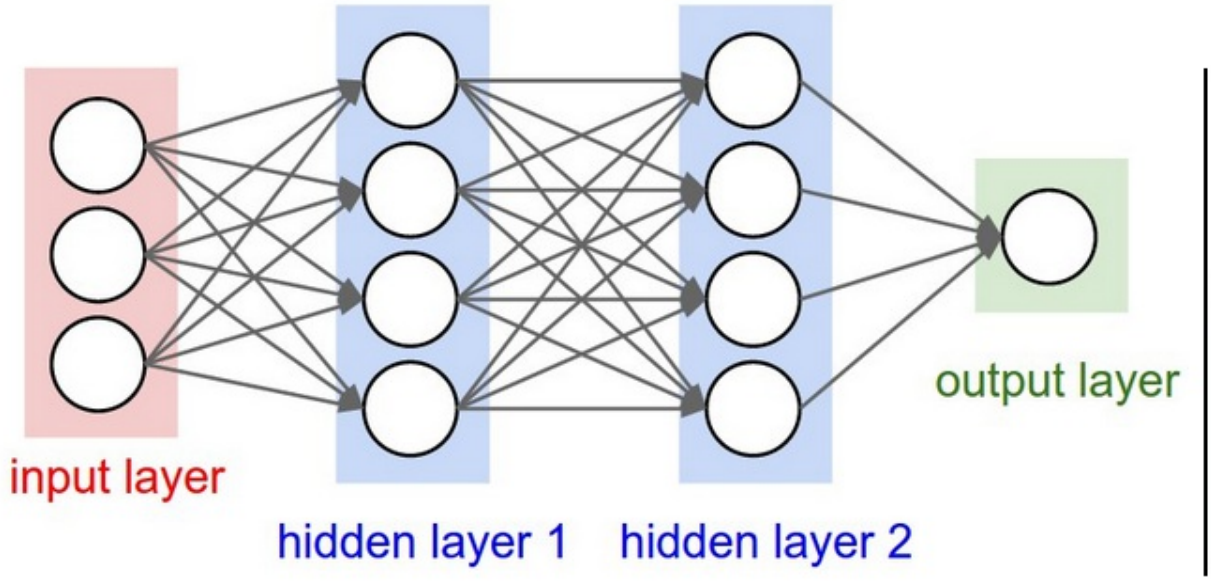
Benchmark Model

We will be solving this problem first with a linear classifier where we will use

```
Y = Wx + B
```

and softmax cross-entropy to classify the images.

After this, we will be using a 2 hidden layer CNN to classify the images.

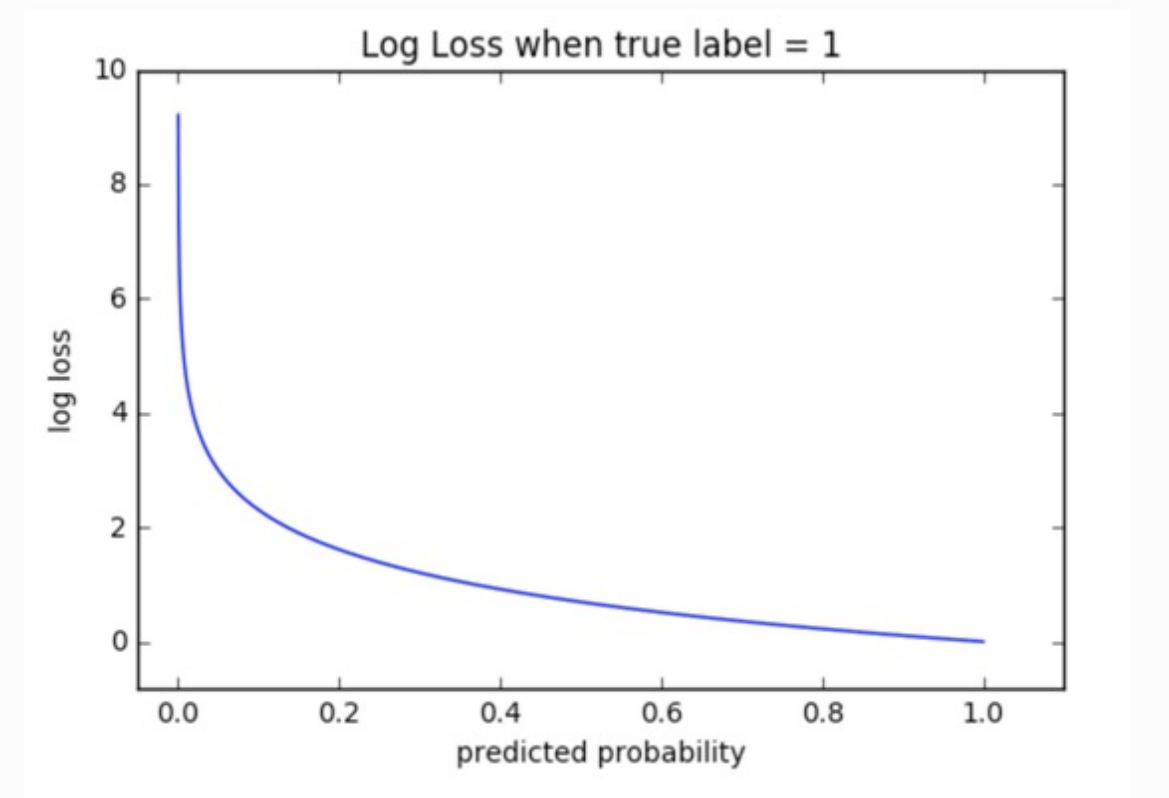


If the computing power supports, then more hidden layers will be added to check the performance.

And as this is a Kaggle problem, we can also compare our solution against the top Kaggle solution. If the accuracy is within ~**10%** of the top solution, we can assume that the model is performing as per the expectations.

Evaluation Metrics

We will be using Cross Entropy as the metrics to evaluate this model. Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverge from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.



The graph above shows the range of possible loss values given a true observation (isDog = 1). As the predicted probability approaches 1, log loss slowly decreases. As the predicted probability decreases, however, the log loss increases rapidly. Log loss penalizes both types of errors, but especially those predictions that are confident and wrong!

Cross-entropy and log loss are slightly different depending on context, but in machine learning when calculating error rates between 0 and 1 they resolve to the same thing.

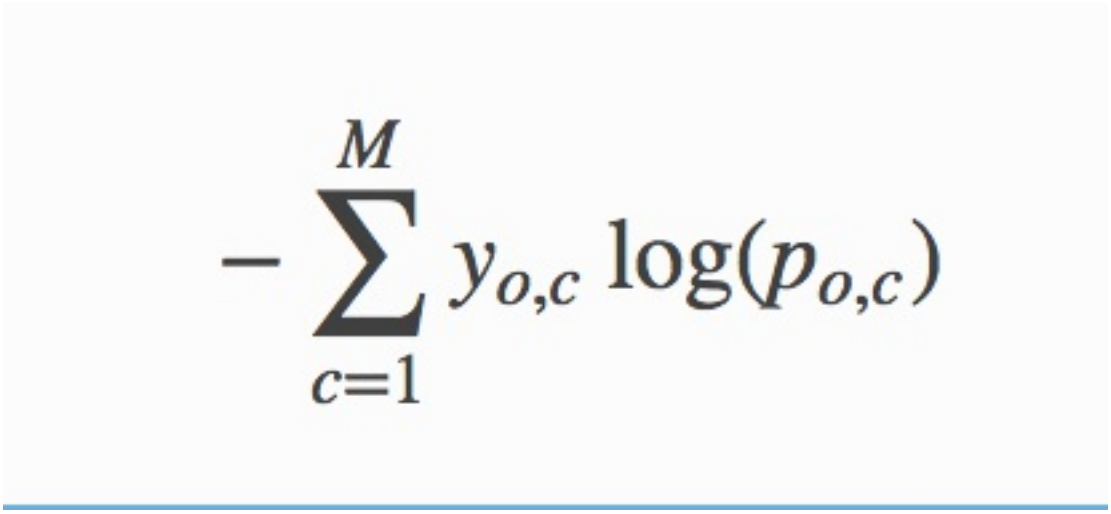
Math

In binary classification, where the number of classes M

equals 2, cross-entropy can be calculated as: $-(y\log(p)+(1-y)\log(1-p))$

If $M>2$

(i.e. multiclass classification), we calculate a separate loss for each class label per observation and sum the result.



$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

Here:

- M – the number of classes (anger, fear, sad etc)
- log – the natural log
- y – binary indicator (0 or 1) if class label c
- is the correct classification for observation o
- p – predicted probability observation o is of class c

Project Design

Environment

Python 3.+

Libraries

- Tensorflow
- Scikit-learn
- Numpy
- Matplotlib

Implementation

1. Data normalization: We will normalize the data so that the values are within 0-1.

2. Data Augmentation: This will be done to control class imbalance as mentioned in data-input.
3. The baseline will be established using a linear classifier model.
4. Then a CNN with two hidden layers will be trained to classify the data.
5. Once we have a working model in place, then we will tune the hyper-parameter to see if we can optimize the model.

References

1. Convolutional Neural Networks
2. Cross Entropy
3. Cross Entropy vs RMSE
4. Data Augmentation