McGill University

COMP-551

Applied Machine Learning

Kaggle - Identify Hand Drawn Images

Author:
Yu Zheng
Wei Zheng
David Valensi
Team: **Skynet**

Student Number & Email:
260731427(yu.zheng3@mail.mcgill.ca)
260829004(wei.zheng2@mail.mcgill.ca)
260873585(david.valensi@mail.mcgill.ca)

November 26, 2018

Yu Zheng
Wei Zheng
David Valensi

Contents

Yu Zheng
Wei Zheng
David Valensi
Team: **Skynet**

1 Introduction

This project will study specific learning algorithms for image Classification and investigate a systematic approach including image prepossessing, hyper-parameters turning in order to find model optimization. The goal is to produce algorithm for image classification, starting with linear learner and and comparing with the performance of neural network methods. The best approach among these methods is **Convolutional neural network** with best performance of 69% accuracy.

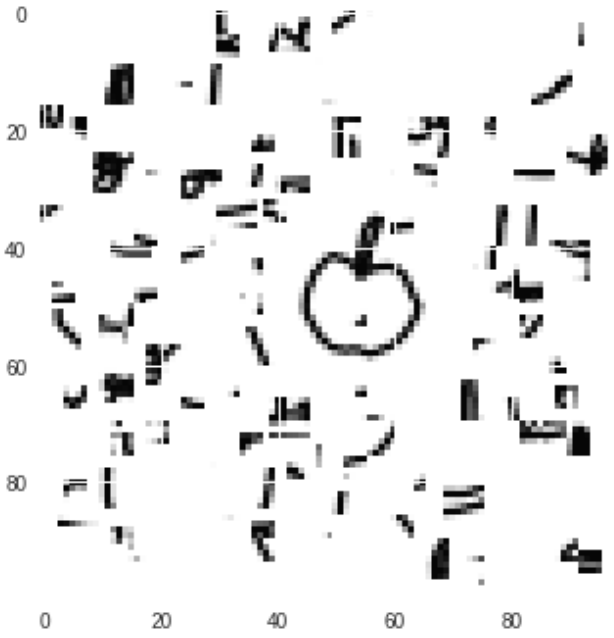
2 Feature Design & Image Prepossessing

Raw images contain both noise and blank canvas background, which would introduce lot of interference features to slow and corrupt the prediction model.
Also the image contains 10000 pixel, which means it has 10000 features, because the number of data samples is same as the number of features, we have to reduce the number of features to improve the training model

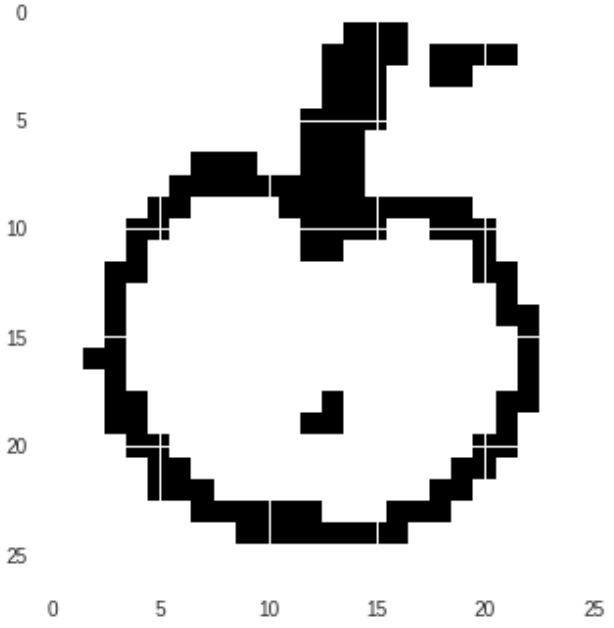
2.1 Image Cropper

- Locate the center of image in $100 * 100$ pixel blank canvas.
- To isolate the image from background. We crop the image into size of $27 * 27$ pixel, large enough to cover entire image, based on the center location of image. The tool we used here is **openCV** library, it can find out the most important contour for the given raw images.
- Map the color code from integer values $(0-255)$ to $(0, 1)$ — either 0 or 1

The first two steps are used to reduce the number of annoyed features; the mapping step is to aim to avoid numerical overflows when feeding large value, e.g. 255 , into the network. The weights is initial randomly with mean=0 and variance=1, multiplying the small weights with large pixel value will result in saturation of neurons because of the property of sigmoid activation functions, and gradients of sigmoid function will be too small to have significant update. After cropping the image, only **729** features are left. As shown in Fig.1a and Fig.1b, the background is almost removed, only the target image left.



(a) Raw image(apple)



(b) Image after cropping(apple)

Figure 1: Image crpper

2.2 Univariate Feature Selection

After truncation of images, there are still many non-informative features exist in images, these features includes randomly generated noise around the drawing and empty space as shown in Fig.2.

To get rid of these features, we apply univariate feature selection of **sklearn** to select removes all but the **600** most significant features. This number is chosen based on observation and testing.

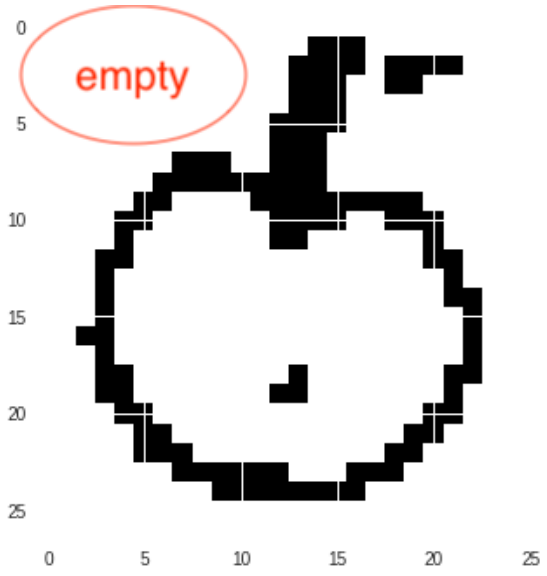


Figure 2: Image with empty space and noise around

3 Algorithms

3.1 Kernalized SVM

Support vector machines attempts to find a separating hyperplane between sets X and Y. Mathematically, the condition for a separating hyperplane is: $w*xi-b<0$ and $w*xi-b>0$. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick. 4 types of kernel functions has been implemented in SVM.

- Linear: $< x, x' >$
- Polynomial:$(gamma||x-x'||+r)^d$—degree
- Gaussian RBF: $exp(-gamma||x-x'||2)$ –gamma must be greater than 0
- Sigmoid: $\tanh(gamma < x, x' > +r)$

3.2 Feed Forward Back-propagation Neural Network

Feed-forward neural network contains threes parts: input layer, hidden layers(more than one layer) and output layer. These layers are connected with wires, each wire has its own weight. All the neurons in same layer are fully connected to the neurons of next layer.
Each neuron has activation function to allow network to learn the non-linear model. For feed-forward network, we implemented whole network into a python class, with functions to update the parameters, compute the feed-forward output result and evaluate the performance of prediction.

3.3 CNN

For this part, we used the Keras interface to implement the CNN using their Sequential model. This allowed us to define several models, using different hyper-parameters values. This is a powerful tool which can achieve state-of-the-art performances when the hyper-parameters tuning is done correctly.

4 Methodology

4.1 Linear Classifier

4.1.1 Training/validation split:

5-fold cross validation is used to determine the best model parameters. Here's the process:

- Split the 10000 data into 60% of training set ,20% of validation set and 20% as test set.
- validation set is used to tune the hyperparameter by GridSerch, record the accuracy to determine the best model hyper parameters
- Fit the trained model by the combination of training and validation set.
- The remaining 20% test set is used to compute the accuracy.

4.1.2 Hyper-parameters

- **Linear SVM:** Parameter–C; Loss–hinge, squared hinge
- **Kernelized SVM** Kernel; Parameter C; gammas; decision function shape

The parameter C, common to all SVM kernels, trades off misclassification of training examples against simplicity of the decision surface.

4.1.3 Optimization tricks

Computing the (soft-margin) SVM classifier amounts to minimizing an expression of the form

$$E = [\frac{1}{n}\sum_{i=1}^{n} max(0, 1-y_i(w*x_i-b))] + \lambda||w||^2 \qquad (1)$$

- In linear SVM model, using the dual problem which is a quadratic function. it is efficiently solvable by quadratic programming algorithms.
- In kernelized SVM, we are given a kernel function K. We want to learn a nonlinear classification rule which corresponds to a linear classification rule.

4.2 Fully Connected Feed Forward Neural Network

4.2.1 Training/validation split:

Because **5-fold cross validation**[1] is used to determine the best model parameters. Here's the process to split the dataset:

- Shuffle the training data before each epoch
- Split the 10000 data into 80% as 5-folds(training= 6400 & test set= 1600) and 20% as development set.
- For each fold, within an epoch loop, training set is used to train the neural network and development set is used to monitor and decide whether to continues the training.
- At the end of each fold, test set is used to computer the accuracy. The overall accuracy is the mean of 5-fold accuracy.
- Record the accuracy to determine the best model parameters, and retrain the neural network with whole dataset.

4.2.2 Hyper-parameters

- **Number of layers:** Due to vanish of gradient problem, only **1** hidden layer is used.
- **Learning rate:** The range of learning rate to be considered is $[0.3, 1, 3, 10]$ because I have already mapped the pixel value from 255 to 1 and the initial parameters are generated from Gaussian with zero mean and 1 variance, then learning rate smaller than 0.3 would take long time to converge, while learning rate greater than 10 would skip the minimum points. The optimal learning rate based on the 5-fold cross validation are: **10.0**
- **Number of nodes in i/p and o/p layer:** Number of input node is determined by the number of features of input, in this case, I truncated the image from $100 * 100$ pixel to **600** features 1-D array. For output layer, one-hot representation is used, therefore a **31** 1-D array represents 31 classes.
- **Number of nodes in hidden layers:** The range of learning rate to be considered is $[60, 120, 240, 480]$, to promote generalizability, I only select the number of nodes between input and output to avoid overfitting. The optimal number of hidden node based on the 5-fold cross validation are: **240**

4.2.3 Optimization tricks

Using **Mini-Batch Gradient Descent** to update the parameters relatively fast while reduce the variance to maintain the stability. The size of mini-batch is 40 to try to cover all 31 different classes during each update.

4.3 CNN

The CNN input is a 2D image (array) and the output are weights for the several output classes. The higher output value index is the predicted class for each sample.

4.3.1 Hyper-parameters & Optimization tricks

- The number of hidden layers and units
- The activation functions used in each layer
- The size of the convolution kernel and dimensionality of output space
- Batch size
- Dropout regularization values
- Number of epochs
- Optimization method

As we describe it here, there are some important hyper-parameters tuning and optimization tricks.

- The model offers convenient methods to avoid CNN problems, like Vanishing Gradient.
  In order to solve this problem, we used a Batch Normalization layer before each one of the convolutional blocks. That step is essential to us because it avoids the activation functions to saturate, thus, avoid vanishing points. Indeed, activation functions gives 0 derivatives when input is not in the dynamic range.
- Another important point in our architecture is the Relu activation function which essentially is: f(x) = max (0, x) (and may have many variants).
  It has many benefits like:
  - Gives sparsity for the hidden layers, and the resulting representation is more efficient.
  - It helps to avoid vanishing gradients
  - Allow faster training

  Moreover, it is the most widely used in the today networks since it has shown better empirical results.
- We also tried several activation functions we learned in class like Sigmoid and Tanh.
- During the tuning time, we tried optimization methods like Adam, SGD and RMSprop.
- Another trick was to use image generation to avoid overfitting. Indeed this method generates similar images but with slight differences to prevent the model to overfit with the same training images.

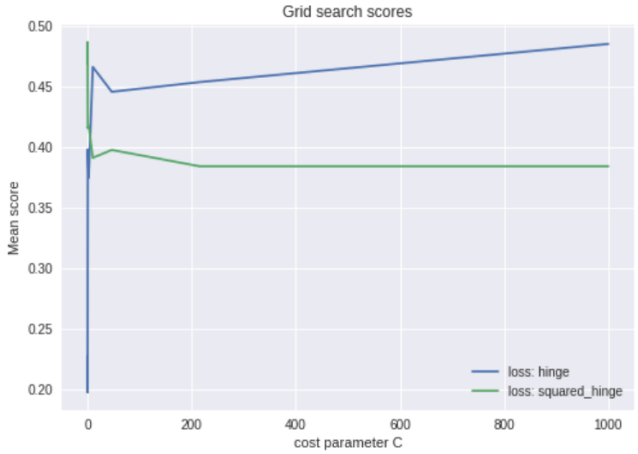


Figure 3: Mean Score vs. Cost parameter

5 Results

5.1 SVM

5.1.1 Linear SVM

- C: np.logspace(-3, 3, 10), best para = 0.46
- Loss: hinge, squared hinge, best para = squared hinge
- When tuning the hyper parameter, best score= 0.469, as shown in Fig.3

After building model by these hyper paras, we use the 20% test set to predict our accuracy, The accuracy is 0.525.

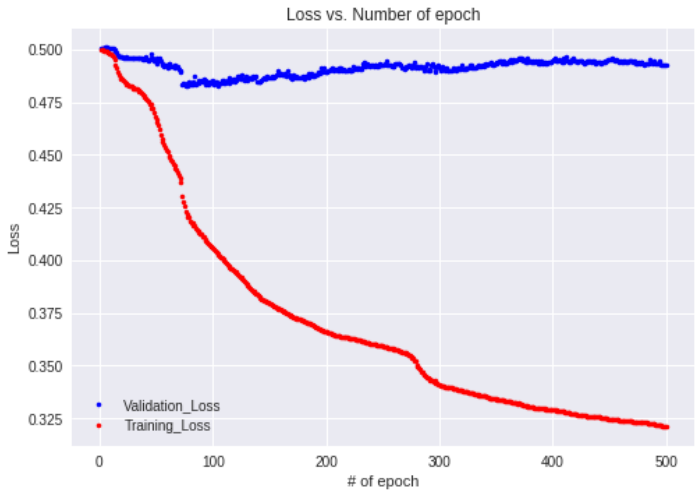
5.1.2 Kernelized SVM

- Kernel:(rbf, sigmoid, poly), best para =poly
- C: np.logspace(-3, 3, 10), best para = 0.001
- gammas = [0.001, 0.01, 0.1, 1], best para =0.001
- decision function shape=one against one scheme is always used as multi-class strategy, best score= 0.2755

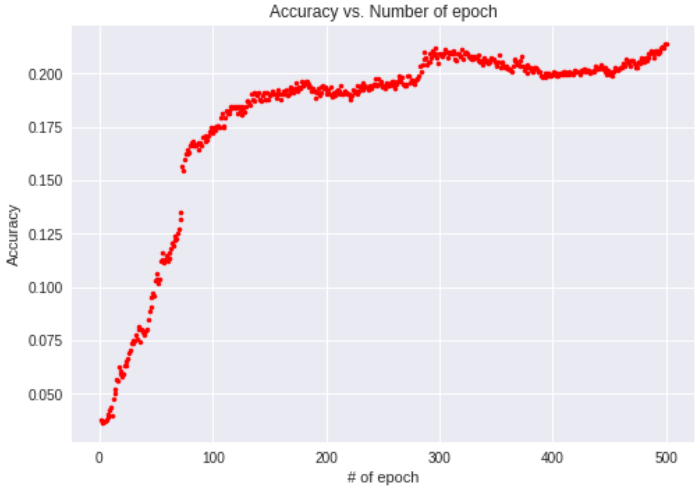
By setting kernel as "poly", C and gamma equals to 0.001, the final accuracy achieved is 0.436.

5.2 Fully Connected Neural Network

To find the optimal hyper-parameters of neural network, two 5-fold validation tests are performed, number of nodes in hidden layer and learning rate. The grid search is not used because too much combinations which require very high computational cost. Therefore, each parameter is tested individually.
For learning rate, with hidden node number equals to 200, $\alpha = 10.0$ gives the highest accuracy after 30 epoch(6%) and 500 epoch(19%). For number of hidden layer node, the best



(a) Training/validation loss vs.Number of epoch



(b) Accuracy changes with number of epoch

Figure 4: Image crpper

model is 240 nodes, around 10% after 30 epoch and 24% after 500 epoch.

From Fig.4a, the fully connected neural network is easy to over-fit with the training data set, the loss of training set would keep decreasing during the training but none significance change of validation loss is observed.

From Fig.4b, the accuracy is saturated after 200 epoch, which means the network converged to local minimum. By increasing the mini-batch size of SGD, based on the historical training, the maximum accuracy can reach around **30%**.

5.3 CNN

As expected, the CNN gives the better accuracies among all the classification models we trained. The highest accuracy we achieved was 69% on validation set. This was achieved after

Use of Batch Normalization	Number of conv2D layers (numb of dropout blocks)	Kernel size	Accuracy
Yes	16 (4 blocks)	3 and 5	67%
Yes	12 (3 blocks)	3 and 5	69%
Yes	12 (3 blocks)	10	60%
Yes	12 (no dropout)	3 and 5	62%
No	16(4 blocks)	3 and 5	5.5%

Tableau 1Several hyperparameters tuning. Optimizer: Adam, Activation func.: RELU

Figure 5: Performance of CNN with different hyper-parameters

several parameters tuning as described below, Fig.5

First of all, the activation functions we tried, Tanh and Sigmoid didn't give better results than Relu. The same kind of observations was made concerning the optimization method. SGD gave poorer results and RMSprop gave no better results than with Adam. Essentially, they didn't show specific advantages for our classification task.
Then, the use of Batch Normalization was critical and boosted the performances, it showed that this is an essential part of the CNN model and its wide use is justified.
We set the number of epochs to 25 or 30 depending on the saturation observed.

6 Discussion

6.1 SVM

- **Pros:** (1) SVM is one of the Non-parametric methods which have a potentially infinite number of parameters, where the number of support vectors depends on the dataset complexity. SVMs is flexible and powerful which require less assumptions.
  (2) It is effective in high dimensional spaces and also when the features greater the examples(dual problems), in this case, after preprocessing (hog features) we have 8192 features greater than the examples.
  (3) Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
  (4) Different Kernel functions can be specified for the decision function.(linear, RBF, polynomial, sigmoid etc)
- **Cons:** (1) Requires full labeling of input data, it needs a lot of data, which means the computational expensive, SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.
  (2) If the number of features is much greater than the number of samples, , it is more prone to overfitting.
- **Future work:** Through Kaggle competition, It is very essential to well pre-processing the data set before training the models, noise removal and the features selection plays an important role. Easy preprocessing results in a poor performance.

6.2 Fully Connected Neural Network

- **Pros:** Very good autodidact, it can discover the complex relation between input and output. The hidden layers of neural network provides non-linearity mapping, which is absent in linear SVM, it helps to create non-linear boundary between classes.
- **Cons:** The data fed into the network is 1-D array instead of 2-D, which means the neural network does not consider the spatial structure of the images. Sensitive to the small change of initial parameters, very easy to converge local minimum instead of global minimum. Easy to suffer the over-fitting and over-training. Therefore, it is not well-adapted to classify images.
- **Future work:** More hyper-parameters turning and features selection are required. Also, turning the hyper-parameters using grid search would be the better option but requires more computational time. Try to use other activation functions to allow more hidden layers. Optimize the training process to skip the local minimum.

6.3 CNN

- **Pros:** CNN is by far the best method among the tried. It allows to find complex patterns and to cluster efficiently the data. The best performances in image processing is probably linked to a CNN.
  The number of hyper-parameters is important and offers a lot of options of definition of models, there are theoretically an infinity of different models and our goal as Machine Learning people is to find the one that fits better to a given dataset.
- **Cons:** High computational cost, long training process and complex model. selection
  Requires a huge amount of data to create big complexity.
- **Future work:** It's definitely possible to improve the network parameters and achieve higher results.
  The hyper-parameters are wide and would permit to get much higher accuracy in object classification, as shown in recent work through the world.
  The amount of data available is also a track to improve our model, since the CNN needs huge datasets to operates as wanted.
  A method one can use to improve results apart from tuning parameters is to use Boosting or Bagging with other

7 Statement of Contributions

- **Yu Zheng:** Data analysis, Images prepossessing with coding implementation of **Image Cropper** and **Univariate Feature Selection**; developing the methodology and coding the solution of entire **fully connected neural network**; writing the report of relevant parts.
- **Wei Zheng:** Coding the **linear and kernelized SVM**, write the relevant part of report.
- **David Valensi:** Image preprocessing, Image feature detection using **HoG algorithm**. **CNN** study and implementation. Write the relevant part of the report.

We hereby state that all the work presented in this report is that of the authors.

References

[1] KURT JUNSHEAN ESPINOSA. K-fold cross validation in neural networks, aug 2016.