ECE ILLINOIS
Department of Electrical and Computer Engineering

Assignment 1: Binary Classification

Corresponding TA: Raymond Yeh

Overview

Release Date: 08/29/17

Due Date: 09/21/17

In this assignment you will:

- Derive properties of linear models.
- Learn basic image processing with Python.
- Implement linear models, including linear regression, logistic regression, and support vector machines.
- Implement optimizatoin algorithm to train the models with NumPy.
- Implement linear models and optimization algorithms with TensorFlow.

Written Section

Note: Answers without sufficient justification will not recieve any points.

Exercise:

Consider a linear classifer of the form $F = \mathbf{w}^T\mathbf{x} + b$, where $\mathbf{w}, \mathbf{x} \in \mathbb{R}^K$, $b \in \mathbb{R}$. Next, you have a training dataset of the form $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, where the subindex is the example index, and $y \in \{-1, 1\}$.

- Problem 1: Given $\hat{y}$ how do you make a prediction? ($i.e.$ whats the function mapping from $F \rightarrow y$?) for linear regression, logistic regression and support vector machine?
- Problem 2: What is the ideal loss function, which we hope to optimize but can't?
- Problem 3: What is the loss function for linear regression, logistic regression and support vector machine?
- Problem 4: What is the gradient w.r.t $\mathbf{w}$ and b for linear regression, logistic regression and support vector machine?
- Problem 5: If the dataset is linearly separable, will gradient descent find the optimal $\mathbf{w}$ for logitis regression (with / without ℓ_2 regularization)?

Programming Section

Part 1: Setup

- First remote connect to a ews machine.

```
ssh (netid)@remlnx.ews.illinois.edu
```

You may need vpn if you are not on a campus network.

- Load python module, this will also load pip and virtualenv

```
module load python/3.4.3
```

- Reuse the virtual environment from mp0.

```
source ~/ece544na_fa2017/bin/activate
```

- Copy mp1 into your svn directory, and change directory to mp1.

```
svn cp https://subversion.ews.illinois.edu/svn/fa17-ece544/_shared/mp1 .
cd mp1
```

- Create data directory and Download the data into the data directory.

```
mkdir data
wget --user (netid) --ask-password https://courses.engr.illinois.edu/ece544na/fa2017/secure/assignment1_data.zip  -O data/assignment1_data.zip
```

- Unzip assignment1_data.zip

```
unzip assignment1_data.zip -d data/
```

- Prevent svn from checking in the data directory.

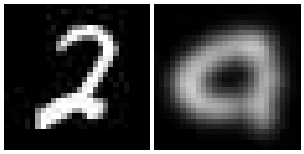
```
svn propset svn:ignore data .
```

- Follow the same steps as mp0 to checkout and setup for mp1 and install the requirements.

Part 2: Exercise

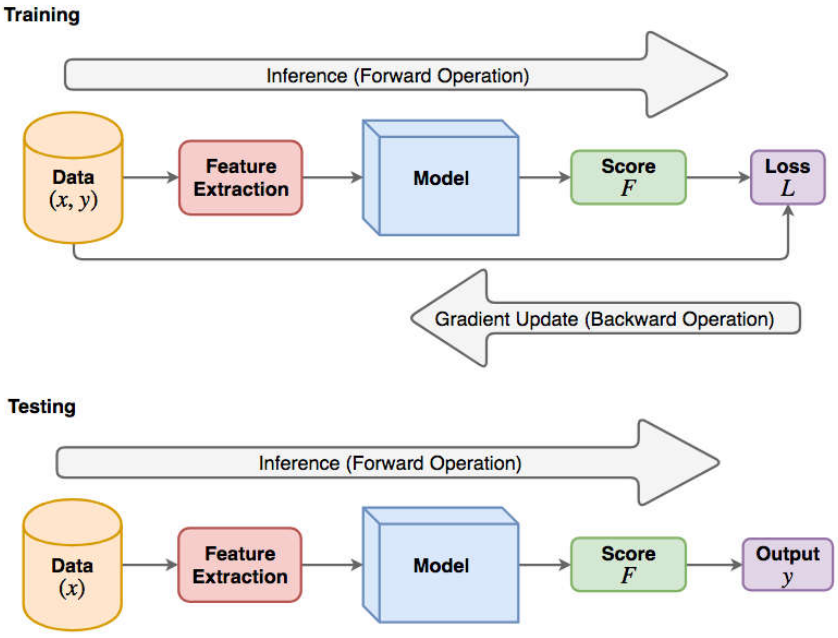
In this exercise you will build a system to classify between clear and blurry images using linear classifiers.

We are given a dataset of clear and blurry images, example shown below.



High level overview

The overall pipeline of the system is illustrated below, and you will implement each of the blocks.



In `main.py`, the overall program structure is provided for you.

Part 2.1: NumPy Implementation

Readining in data.

In `utils/io_tools.py` you will fill in one function for reading in the dataset. The dataset consists of the directory of images stored in png format, and txt files indicating the filename of the imaeg and label.

There are 3 txt files, `train_lab.txt`, `val_lab.txt`, `test_lab.txt`, each indicate the files and labels for the corresponding split the dataset.

The txt foarmat is tab separated in order of file_name, label.

For example:

`test_00000.png 1`

The image_file “train_00000.png” has the label “1”, meaning the imaeg is a clear image.

Revelant File: `utils/io_tools.py`

Hint: `skimage` provides helpful functions to read in images.

Data processing.

In `utils/data_tools.py` you will fill in 3 functions for extracting feature. The feature extraction process involes simple image manipulation, such as mean removal, filtering, element-wise scaling. More details are provided in the function docstring.

Revelant File: `utils/data_tools.py`

Hint: `skimage` provides helpful functions to extract image features.

Linear model implementation.

In `models/linear_model.py`, you will implement an abstract base class for linear modles (e.g linear regression, logistic regression, and support vector machine).

All of the models must support the follow operations:

Forward operation

Forward operation is the function which takes an input and outputs a score. In this case, for linear models, it is $F = \mathbf{w}^T\mathbf{x} + b$.

For simplicity, we will rewrite $x = [x, 1]$, and $w = [w, b]$, then equivalently, $F = \mathbf{w}^T\mathbf{x}$.

Loss Function

Loss function takes in a score, and ground-truth label and outputs a scalar. The loss function indicates how good the model’s predicted score fits to the ground-truth.

Backward operation

Backward opretaion is the operation for computing gradient of the loss function w.r.t to the model parameters. This is computed after the forward operation to update the model.

predict operation

The prediction operation is a function which takes a score as input and outputs a prediction $\in \mathcal{Y}$, in the case of binary classification, $\mathcal{Y} = \{-1, 1\}$.

Revelant Files: `models/linear_model.py`, `models/linear_regression.py`, `models/logistic_regression.py`, `models/support_vector_machine.py`

Gradient descent implementation.

Gradient descent is a optimization algorithm, where you adjust the model in the direction of the negative gradient of L.

Repeat until convergence:

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta\nabla L^{(t-1)}$$

The above equation is refered as an update step, which consists of one pass of the forward and backward operation.

Revelant Files: `models/train_eval_model.py`

Model Selection

Observe that dataset is divided into three parts, train, eval, and test, the splits were constructed to perform the following:

Training

The training set is used to train your model, meaning you use this set to find the optimal model parameters.

Validation

The validation set is used to evalute the quality of the trained models, and determine which set of *hyperparameters* to use. In our case, the hyperparameters are the feature extractions, loss function, the learning rate, and the number of training steps.

The idea is to pick the best hyperparamters based on the validation performance, this prevents overfitting to the training set.

Testing

Lastly, the testing set is never used to determine anything of the trained model. This set is purely used for evaluation. In the test split,we did not provie the label, you will be able to compute the testing performace through the Kaggle competition.

Revelant Files: `models/train_eval_model.py`,

Running Experiments

Experiment with different features, weights initialization and learning rate. We will not grade the `main.py` file, feel free to change it.

Revelant Files: `models/main.py`,

Part 2.2: TensorFlow Implementation.

You will reimplement the models in the previous section using TensorFlow.

Revelant Files: `models/linear_model_tf.py` `models/linear_regression_tf.py`, `models/logistic_regression_tf.py`, `models/support_vector_machine_tf.py`

Part 3: Kaggle competition (extra credit).

Top 3 ranked submissions across all mps will get a secret prize at the end of the semester.

We have hosted a Kaggle competition for this assignment [link], we encourage you to participate. The leaderboard should be helpful as a sanity check.

To submit to Kaggle, the submission format is in csv. The uploaded file should be of 8001 lines, format illustrated below:

```
Id,Prediction
test_00000.png,1
...
test_01234.png,-1
...
test_07999.png,-1
```

Revelant Files: `utils/io_tools.py`,

Part 5 Writing tests.

Similar to mp0, we have provided a basic test case. Feel free to write more.

Revelant Files: `test.py`

Part 6: Submit

Submitting your code is simply commiting you code. This can be done with the follow command:

```
svn commit -m "Some meaningful comment here."
```

Note: The assignment will be autograded. It is important that you do not use additional libraries, or change the provided functions input and output.