Project-II by Group Sydney

Diego Antognini & Jason Racine
EPFL
diego.antognini@epfl.ch, jason.racine@epfl.ch

Abstract

This report provides a summary of the project two of the PCML class.

1 Data Description

The train-data for binary classification consists of $N = 6000$ images. As input we have 2 representations of the image : *the histogram of oriented gradients* (HOG) $\mathbf{X}_{hog}$ and the *overFEAT ImageNet CNN features* (CNN) $\mathbf{X}_{cnn}$. Our input $\mathbf{X}$ is the concatenation of $\mathbf{X}_{hog}$ with $\mathbf{X}_{cnn}$.

Each input sample is a vector $\mathbf{x}_n$ with dimension $D = 42273$ (5408 for the HOG and 36865 for the CNN) and is the concatenation of $\mathbf{x}_n^{hog}$ with $\mathbf{x}_n^{cnn}$ The output ($\mathbf{y}$) represents the classification of the images. For the binary classification, the label 1 represents a car, a horse or a plane and the label 2 anything else. For the multi-class classification, the label 1 represents a car, 22 a horse, 3 a plane and 4 anything else.

We also have test-data of size $N = XXX$ without their corresponding output. Our goal is to produce predictions for those data, as well as an approximation of the test-error using *Balanced Error Rate* (BER).

1.1 Histogram of Oriented Gradients

Histogram of oriented gradients is a feature used in computer vision in order to detect objects. To compute it, we first need to normalize the image, compute the gradients (of each pixel) for the different color channel and take those with the largest norm. Then we decompose the image in bins of size $w \times h$. For each of those bins, we compute an histogram of the orientation of the gradients using theirs angles and weighted by theirs magnitudes. The histogram has n bins from 0 to 180 degrees. We compute this histogram using 4 different normalizations.

In our case, the dimensions of this feature is $13 \times 13 \times 32 = 5408$, where the first 13 is the number of bins in the height, the second 13 the number of bins in the width. Finally 32 is 4×8 where the 4 is the different normalizations for the histogram and 8 the number of bins (each bin has a size of 22.5).

1.2 OverFEAT ImageNet CNN features

Those features are extracted from a convolutional network-based image features extractor OverFeat. They were trained on the ImageNet dataset (tens of millions images). The size is the output of the fifth layer of the neural network which is $1024 \times 6 \times 6 = 36864$. In our features, we also have an extra dimension which can be ruled out.

2 Data visualization and analysis

We first have plot the labels distribution for the binary and multi-class cases. For the binary case, we have 60.3% for the label 1 and 39.8% for the second one. We have observed that there is a

imbalance between the classes but this is quite small. For the multi-classes case, we obtain 16.07%, 19.37%, 24.87% and 39.70% for the labels 1, 2, 3 and 4. This time, the imbalance is more important, especially with the label 4 and we do have to take care about it during the training.

We also wanted to analyze the distribution of our features, and have observed that for the features HOG, each of them has a has similar distribution as showed in Figure XXX. The values are essentially in the range $[0, 0.2]$. We are also interested about the correlation between the input and output variables. We have observed that the correlation is in the range $[-0.24; 0.30]$, and so can conclude that features seem not to have a direct correlation between them. However, a combination of features can have a high correlation with the output but we couldn't find such combination. Finally, $\mathbf{X}_{hog}$ is rank-deficient with a rank of 3467 instead of 5408.

For the features CNN, first we can see that it is a sparse matrix and most of the values for each data is 0 (around 95.5%). This means that for each data, a small subset is able to describe the picture. Moreover, we can observe that we are faced to the problem of $D > N$ and the solution we have adopted is PCA which will describe during the next sections. The values are in mainly in the range $[0, 64.87]$. We also wanted to see the correlation of those features with the output and have found that they are in the range $[-0.24; 0.18]$, which is similar to the HOG features. Finally, $\mathbf{X}_{cnn}$ is rank-deficient with a rank of 5997 instead of 36865.

3 Techniques used

This section is a description of the technique we have used in the project. It describes mainly how it works, why we thought it could be useful in this project.

3.1 Principal component analysis

As we have seen in the section "Data visualization and analysis", we are faced to the $D > N$ problem with the CNN features. One way to solve this problem is to use PCA. Basically, PCA works as follow : we compute the mean and the covariance matrix $S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T$ of our data and then compute the M eigenvectors of S corresponding to the M largest eigenvalues. M is the number of features we have to take into account in order to minimize the reconstruction. Finally, the approximation of a data sample is $\tilde{x_n} = \sum_{i=1}^{M} x_{ni}u_i$. However, we couldn't compute PCA using this method, because computing the full eigenvector decomposition a matrix $D \times D$ runs in $O(D^3)$ and D is quite large. We have used an alternative proposed in [REF] which allows us to compute the same eigenvectors in $O(N^3)$, which is roughly at least 230 times faster. The tricks is to express $S = \frac{1}{N}XX^T$ and finally using $\frac{1}{N}XX^Tv_i = \lambda_i v_i$, where $v_i = Xu_i$, we obtain an eigenvector equation for the $N \times N$ matrix S.

The last parameter to set, is the parameter M which defines the M largest eigenvectors to take into consideration for our features selection.We introduce a distortion measure $J = \frac{1}{N}\sum_{n=1}^{N}||x_n - \tilde{x_n}||^2$ which represents the mean squared distance between the original data sample and our approximation. Without going into the steps, the solution to the minimization of J is $J = \sum_{i=M+1}^{D}\lambda_i$.

The Figure XX shows the plot of J for the HOG features and the CNN features. For the CNN, we have tried several values for M and have conclude that $M = 150$ was a good compromise between accuracy and speed. The corresponding distortion might be very high compared to the HOG features, but the range of the values are not the same, CNN has values going larger than 0.2. We tried some bigger values but we didn't get major improvements in terms of error. For the HOG features, the distortion measure seems more nice, and have decided to take $M = XXX$. BECAUSE BLA BLA BLA

3.2 Support Vector Machine

3.3 Neural Networks

This is the multi-layer perceptron (MLP) we have seen in class. The neural networks used in this project comes from the given DeepLearning matlab toolbox. It uses batch sizes in order to use less memory (take the first n data samples and works on them, then the next n etc). We have let the

default parameter (100). The number of epochs is simply the number of time where all the training data do a forward and a backward pass. We have put this value to 20, in order to have a little more accuracy without loosing a lot of time, it was a good compromise because larger epochs don't mean especially better accuracy. the number of input is M and output is either 2 or 4, depending of the type of classification (binary or multi-class). For the number of hidden layer, we have let 1 because we didn't see improvement by adding some more. However, the number of neurons in the hidden layer has been tested and have found 1000 for the binary classification and 700 for the multi-class case. We also have to set up the learning rate : the optimal value we have found is 3 for the binary case and 2 for the multi-class one. By default, the activation function used is the *tanh* one. We didn't see so much different using the *sigmoid*. Finally, the algorithm uses stochastic gradient descent using a momentum of 0.5, which might help to converge faster.

3.4 Decision Trees

3.5 Random Forests

4 Evaluation methods

5 Model comparison

5.1 Binary classification

5.2 Multi-class classification

6 Feature transformation

7 Conclusion

8 References

@miscIMM2012-06284, author = "R. B. Palm", title = "Prediction as a candidate for learning deep hierarchical models of data", year = "2012",

@miscPMT, author = Piotr Dollár, title = Piotr's Computer Vision Matlab Toolbox (PMT), howpublished = `http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html`