Data Challenge : Kernel Methods for Machine Learning

Youssef DAOUD
M2 SIAM Data Science
Grenoble-INP Ensimag
Youssef.Daoud@grenoble-inp.org

Anas MEJGARI
M2 SIAM Data Science
Grenoble-INP Ensimag
Anas.Mejgari@grenoble-inp.org

Abstract

Most machine learning applications often involve classification tasks with many classes. Such tasks have become increasingly important within the research field in recent years. This report will present different kernel methods and machine learning techniques used in order to perform a multi class classification task.

1. Introduction

The data challenge we tackled is a multi class classification problem that aims to predict the right class among the 10 existing classes. The data available in this work is divided between 5000 labeled images for training, and 2000 unlabeled images for testing. The different images in the data set are 32x32x3 (RGB). The data is balanced, with an equal distribution between all classes. In this work, we used and implemented different kernel methods and Machine Learning techniques from scratch in order to achieve the best results and solve the limitations of non-kernel methods. *Our work helped us reaching the 7th position (equally with another team) in the private leaderboard with 56.59%, and a score of 56.2% on the public leaderboard.*

2. Data Visualization

The images were not easy to visualize because of the transformation made on the pixels. But, we have managed to show the images in order to discover our 10 classes. To do so, we first reshaped the images into the correct 32x32x3 RGB format. Then, we used an inverse normalization by performing a combination of the hyperbolic tangent function and its inverse on the pixels :

$$h(x) = \frac{1}{2}\tanh(\operatorname{arctanh}(x + c) + d)$$

with x is the pixel and c and d are the parameters of the function which are chosen respectively as 0.3 and 1.2. Then, after plotting the images we identified the ten classes: (from 0 to 9) aircraft, boat, bird, dog, cat, car, truck, deer, frog, horse.

A horse A car A boat A truck A dog

3. Feature Extraction

3.1. Histogram of Oriented Gradients

After implementing our algorithms on the given features, it appeared that using just the raw features will not lead to better performances. Then, we thought about using feature engineering techniques in order to extract those that contain useful information. To do so, we used Histogram of Oriented Gradients (HOG), which is a feature descriptor that is usually used to extract important variations from an image data. Without distorting the original relationships or significant information, it compresses the amount of data into manageable quantities for algorithms to process. Based on the paper of Dalal and Tiggers [1], the HOG implemented in our work used the following parameters:

- **orientations** Number of orientation bins : 9
- **pixels_per_cell Size** (in pixels) of a cell : 8x8
- **cells_per_block** Number of cells in each block : 2x2
- **block_norm** Normalization using L2-norm

Finally, the used parameters lead to 324 features per image.

3.2. Kernel PCA

In order to construct more robust features and feed it to our models, we used Kernel PCA over the image pixels to extract new characteristics from them and combine those features to the ones generated by the HOG. Since every pixel distribution is centered over 0, we can perform the kernel PCA directly. We chose to work with the polynomial kernel and to extract the top 64 features that explain the images.

3.3. K-means

Another approach we adopted in our problem is running the K-means clustering algorithm for K = 10 in order to explore the dataset in a different manner compared with the HOG and kernel PCA. We used as features *the distance of each sample to the 10 centroids* generated by the algorithm.

4. Classification Algorithms

4.1. Kernels used

- **Polynomial Kernel** $K(x, y) = (x^T y)^d$

- **Gaussian Kernel** $K(x,y) = \exp(\frac{-||x-y||_2^2}{2\sigma^2})$
- **Laplacian Kernel** $K(x,y) = \exp(\frac{-||x-y||_1^2}{\sigma})$
- **Chi-Square Kernel** $K(x,y) = \sum_{i=1}^{n} \frac{2x_i y_i}{x_i + y_i}$
- **Generalized Histogram Intersection** $K(x,y) = \sum_{i=1}^{n} \min(|x_i|^\alpha, |y_i|^\alpha)$
- **Log kernel** $K(x,y) = -\log(||x-y||^d + 1)$

4.2. Classifiers

We implemented the different classifiers seen in class:

- *Kernel Ridge Regression*;
- *Kernel Logistic Regression*;
- *SVM*.

For each classifier, we implemented the corresponding loss function, the optimization method (for the Ridge regression we do not use any optimization method since the prediction can be calculated directly using the kernel matrix, for the Kernel Logistic regression we choose to implement the gradient descent algorithm and for the SVM we used the library CVXOPT [2] dedicated to solve QP problems), a function *fit* for fitting the data, and finally a function *predict* to do the final predictions.

For each one we implemented 2 versions: **One Vs All** and **One Vs One**. In fact, we started by implementing the One Vs All classifiers, which ended up with 2 major inconveniences:

1. For each of our 10 classifiers, the corresponding dataset was imbalanced (only 10% are labeled 1);
2. The computations for the optimization algorithm were very expensive since it takes the whole dataset into consideration.

In general, for the SVM and Kernel Logistic Regression we got the best results using the One Vs One classifier.

5. Experiments and Results

We have implemented our work in python using the following libraries: numpy, matplotlib, cvxopt, skimage and seaborn.

5.1. Features selected and dimensionality reduction

As presented in section 3, we proposed 3 methods for selecting the important features that can help us to classify our images correctly: HOG, Kernel PCA, and distances to the centroids generated by K-means algorithm. Our different approaches were tested on 4 combinations of those features: 1 - HOG features only; 2 - HOG and Kernel PCA features; 3 - HOG and K-means features only and finally 4 - HOG, kernel PCA and K-means features.

5.2. Results and Hyperparameters finetuning

In order to choose the best parameters, we applied an exhaustive search over the proposed combination of features, the kernels introduced in 4.1, the parameters of the kernels and the hyperparameters of each model. We obtained our top 2 results by the following combinations and by evaluating them on a validation set of 1000 images:

1. **Kernel Ridge Regression** with the One Vs All approach using HOG features only, $\lambda = 0.001$ and the polynomial kernel of degree 12, resulting 58.7% on the validation set.
2. **SVM** with One Vs One approach using HOG features only with $C = 5$, the Laplacian kernel and $\sigma = 1$, resulting 52.3% on the validation set.

We can mention also that our best results using the Logistic Regression were obtained using the HOG and Kernel PCA (with polynomial kernel of degree 4) features.

6. Conclusion

This challenge gave us the opportunity to put our theoretical knowledge into practice. In this work, we have succeeded in visualizing the data using inverse normalization. Then, we used different approaches to select the best features such as **HOG**, **Kernel PCA** and **K-means**. We also experimented with different kernel methods with different hyperparameters like **Kernel Logistic Regression**, **Kernel Ridge Regression** and **Support Vector Machines**. Additionally, seeing that our problem is a multiclass classification task, we have used the two main methods to classify the images which are **One vs All** and **One vs One**. Finally, we thought about different possible improvements. First of all, based on the work [3], we started implementing Convolutional Kernel Networks from scratch but we did not have the time to finish and to debug our implementation. Secondly, we can use different methods for feature extraction such as SIFT and Fisher Vectors.

References

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 886–893. IEEE, 2005.

[2] CVXOPT : A free software package for convex optimization. `https://cvxopt.org/`

[3] Julien Mairal, Piotr Koniusz, Zaid Harchaoui, Cordelia Schmid. Convolutional Kernel Networks. Advances in Neural Information Processing Systems (NIPS), Dec 2014, Montreal, Canada.