Kernel methods for image classification task

Victoria BRAMI, Margot COSSON

October 4, 2022

Support vector machines are powerful tools to achieve image classification tasks. This report describes a multi-classes' SVC model built to classify a dataset of 5000 training samples in $p = 10$ classes and gives the main results obtained.

1 Classification model

1.1 Kernels

Among all existing kernels, we chose to use the polynomial kernel parametrized by its power $d \in \mathbb{N}$:

$$K(x, y) = (x^T y)^d \tag{1}$$

Indeed, the task requires a non linear kernel. We also benchmarked the linear and the RBF kernels which gave respectively 20% and 4% smaller accuracies on the validation set (20% of the un-seen training set) compared to the polynomial kernel's performances.

1.2 Binary Support Vector Classifier

The base unit of the model is the binary support vector classifier which aims to separate the two classes of samples with a separating hyper-surface of equation $f(x) + b = 0$ with f, function in the RKHS of the chosen kernel K. The associated optimization problem is :

$$\begin{aligned} \min_{f,b,\xi_i} \quad & \frac{1}{2}\|f\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & y_i(f(x_i) + b) \geq 1 - \xi_i \quad \forall i \in [1, n] \\ & \xi_i \geq 0 \quad \forall i \in [1, n] \end{aligned} \tag{2}$$

whose dual is :

$$\begin{aligned} \min_{\alpha} l(\alpha) = \quad & \frac{1}{2}(y \odot \alpha)^T K (y \odot \alpha) - \mathbb{1}^T \alpha \\ \text{s.t.} \quad & y^T \alpha = 0 \\ & 0 \leq \alpha_i \leq C \quad \forall i \in [1, n] \end{aligned} \tag{3}$$

There exists no closed-form solution to this minimization problem. However, one can find the optimal solution α^* thanks to a gradient descent or a quadratic programming solver. Then, the support vectors S can be identified as follows :

$$\begin{cases} C > \alpha_i^* > 0 & \text{iif } x_i \text{ is a } \textbf{support vector}, \\ \alpha_i^* = 0 & \text{iif } x_i \text{ is } \textbf{outside} \text{ the margins}, \\ \alpha_i^* = C & \text{iif } x_i \text{ is } \textbf{inside} \text{ the margins}. \end{cases} \tag{4}$$

from which, one can build the optimal separating hyper-surface :

$$f^* = \sum_{i \in S} \alpha_i^* y_i k_{x_i} \quad \text{and} \quad b^* = \frac{1}{|S|}\sum_{i \in S}(y_i - f^*(x_i)) \tag{5}$$

and infer the resulting binary classifier:

$$h^*(x) = f^*(x) + b^* \tag{6}$$

$$\text{class}(x) = c^*(x) = 2 \times (h^*(x) > 0) - 1 \tag{7}$$

1.3 Multi-class Support Vector Classifier

As the model aims to classify data distributed in $p = 10$ classes, it has to combine several binary SVC [MA99]. We tried two different policies :

- The *One Vs All* (ova) strategy builds p binary SVC: $\{h_i^*\}_{i \in [1,p]}$. Each classifier is trained to separate one class from the rest of the datapoints. Then, the predicted class of a sample is the one who gets the highest classification score $h_i^*(x)$.

$$\text{class}(x) = \arg\max(\{h_1^*(x), ..., h_p^*(x)\}) \tag{8}$$

- The *One Vs One* (ovo) strategy builds p^2 binary SVC: $\{h_{ij}^*\}_{i,j \in [1,p]^2}$. Each classifier separates only the samples of two classes. Then, the predicted class of a sample is the one who gets the higher sum of prediction scores $c_{ij}^*(x) = 2 \times (h_{ij}^*(x) > 0) - 1$. Note also that only $\frac{p(p-1)}{2}$ classifiers are required since one can state $\forall i \in [1, p], \forall j \in [i + 1, p], h_{ji}^* = -h_{ij}^*$.

$$\text{class}(x) = \arg\max(\{\sum_{j=2}^{p} c_{1j}^*(x), ..., \sum_{j=1}^{p-1} c_{pj}^*(x)\}) \tag{9}$$

On the validation set, the *One Vs One* policy gave accuracy smaller by 1 point of percent than the one obtained with the *One Vs All* strategy. Therefore, we opted for the *ova* technique. Note however that the *ovo* algorithm is way faster in terms of computational time.

1.4 Hog features extractor

Raw images contain a lot of information but they are also too *pixel-specific* to allow the SVC to classify efficiently. Therefore, a pre-processing step for feature extraction is required, like the Histogram of Oriented Gradients descriptor (HOG). It outputs a simplified representation $H^N(I) \in \mathbb{R}^{|\text{patches}|\times|\text{bins}|}$ focused on the structure and the shape of the picture's elements of an image I [DT05]. It iteratively builds the following objects :

$$\nabla_h(x,y) = I(x+1) - I(x-1), \quad \forall (x,y) \in I \tag{10}$$

$$\nabla_v(x,y) = I(y+1) - I(y-1), \quad \forall (x,y) \in I \tag{11}$$

$$M(x,y) = \sqrt{\nabla_h(x,y)^2 + \nabla_v(x,y)^2}, \quad \forall (x,y) \in I \tag{12}$$

$$\phi(x,y) = \arctan \frac{\nabla_v(x,y)}{\nabla_h(x,y)}, \quad \forall (x,y) \in I \tag{13}$$

$$H_q(b) = \sum_{(x,y)\in q} \mathbb{1}_{\substack{\phi(x,y)\in \\ [v(b),v(b+1)[}} \times \frac{M(x,y)\phi(x,y)}{v(b+1)-v(b)}, \quad \forall q \in \text{patches}, \forall b \in \text{bins} \tag{14}$$

$$H_q^N(b) = \frac{H_q(b)}{\sum_{t\in Q}\sum_b H_t(b)}, \quad \forall Q \in \text{normalization patches}, \quad \forall q \in Q, \forall b \in \text{bins} \tag{15}$$

Features are extracted for each sample and the model is trained on the features' dataset: $\{H^N(I)\}_{I\in\text{dataset}}$.

2 Results

To solve the optimization problem described by the equation 3, we used the quadratic solver of CVXOPT [MSA13] with $P = (yy^T) \odot K$, $q = -\mathbb{1}_n$, $A = y$, $b = 0$, $G = [-I_n|I_n]$, $h = [0 * \mathbb{1}_n | C * \mathbb{1}_n]$.We determined the best hyper-parameters by cross-validation :

- Regularization constant $C = 1$
- Polynomial kernel with power $q = 5$
- Hog : patch radius $r_{\text{patch}} = 4$ pixels, normalization patch radius $r_{\text{normalization}} = 7$ blocks, bins number $b = 9$.

With these hyper-parameters, the model described above yields the following results:

	Train	Test
Accuracy	100.00 %	58.60 %
Time	568.82 s	—

The corresponding test predictions' file `Yte.csv` can be obtained by running the command `python start.py`. To further explore the implementation, one can access the folder `src/` where `hog.py` contains the code relative to the HOG features' extractor tool, `kernels.py` and `svm.py` are the core part of the model and `predict.py` compiles the whole pipeline and can be called via `python predict.py` with some particular values for the hyper-parameters through the `--hyperparameter value` style. The code is available through the following link :

https://github.com/Victoria-brami/kernel_data_challenge

3 Discussion

To conclude, we built a multi-classes' Support Vector Classifier (SVC) with $p = 10$ binary SVC based on the polynomial kernel. Each classifier discriminates one class from the rest. Then, the predicted class of a sample is the one corresponding to the highest SVC score. The trained model gave an accuracy of 58.60% on a 2000 samples test set.

We thought about few improvements for this image classification model. First of all, one could opt for a more elaborate features' extractor in the pre-processing step. For instance, it would be interested to try the Scale Invariant Feature Transformation (SIFT) algorithm. Furthermore, it could be relevant to use the classification scores $h^*(x)$ to post-process the solution. For instance, in *ova* strategy, if, for a certain sample, the two best classes' scores are very close, then it could be interesting to fit a binary classifier to distinguish only samples from these two classes in order to refine the final prediction.

References

[DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

[MA99] E. Mayoraz and E. Alpaydin. Support vector machines for multi-class classification. In J. Mira and J.V. Sánchez-Andrés, editors, *Engineering Applications of Bio-Inspired Artificial Neural Networks*, volume 1607 of *Lecture Notes in Computer Science*, pages 833–842. Springer, 1999.

[MSA13] Lieven Vandenberghe Martin S Andersen, Joachim Dahl. *CVXOPT: A Python package for convex optimization*, 2013.