Multiclass Classification with Pegasos

Gabriele Cerizza

Università degli Studi di Milano
gabriele.cerizza@studenti.unimi.it
https://github.com/gabrielecerizza/smml

Introduction

In this report we detail the results of our experiments on the Pegasos algorithm [2] over a multiclass classification task, pursuant to the project specifications set out for the Statistical Methods for Machine Learning course of the Università degli Studi di Milano[1].

In Section 1 we illustrate the dataset used in the experiments. In Section 2 we briefly describe the algorithm and its implementation. In Section 3 we show the results of our experiments and provide comments thereon. Finally, Section 4 contains our concluding remarks.

1 Dataset

Our analysis was carried out on the USPS dataset[2], comprising 9298 16×16 grayscale images. These images depict handwritten digits ranging from 0 to 9. A sample of these digits can be seen in Figure 1. The classes do not present severe imbalance issues, as shown in Figure 2.

To gauge the complexity of multiclass classification on this dataset, we performed dimensionality reduction by way of the t-distributed Stochastic Neighbor Embedding (t-SNE) [1] algorithm and projected the instances onto a two-dimensional space. The result is given in Figure 3. Within this figure the instances are clustered into ten clearly distinguishable groups, thus suggesting that a satisfying classification can be attained.

The USPS dataset is made available with predetermined training and test sets. In our experiments, however, we merged these sets and proceeded to form our own training and test sets, as illustrated in Section 3.

2 Pegasos Algorithm

In the present section we describe the Pegasos algorithm

[1] https://cesa-bianchi.di.unimi.it/MSA/index_21-22.html
[2] https://www.kaggle.com/datasets/bistaumanga/usps-dataset

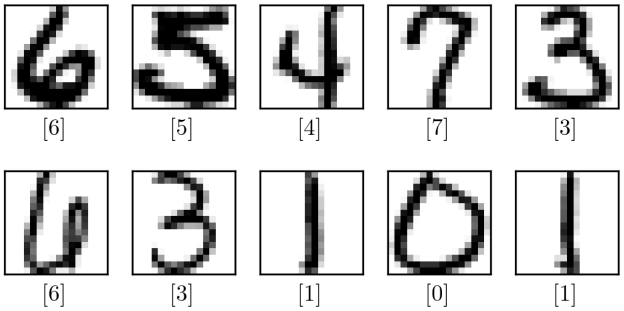

Fig. 1. Sample of images from the USPS dataset. The label is indicated below each image.

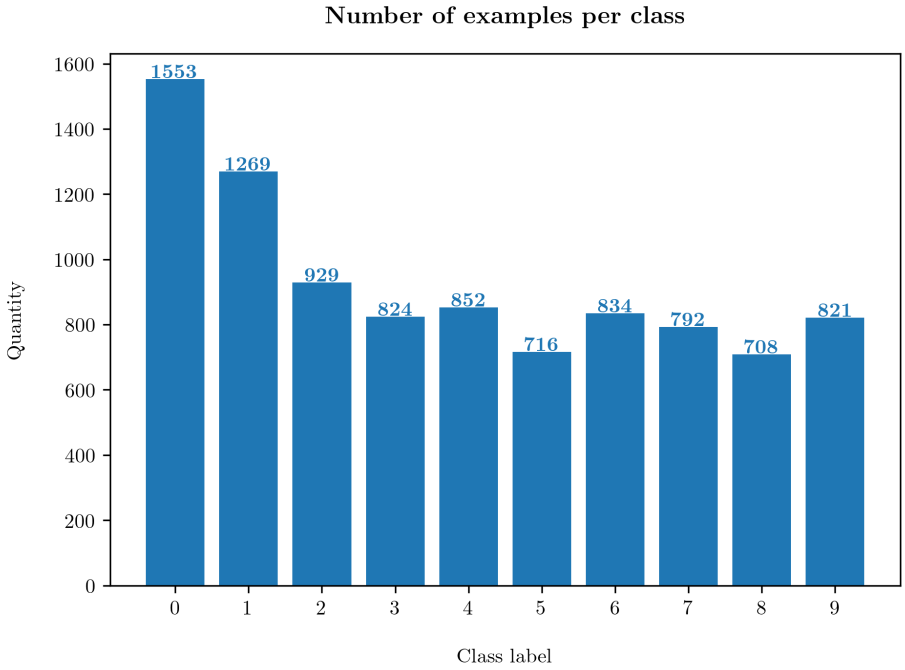

Fig. 2. Frequency of the classes in the USPS dataset.

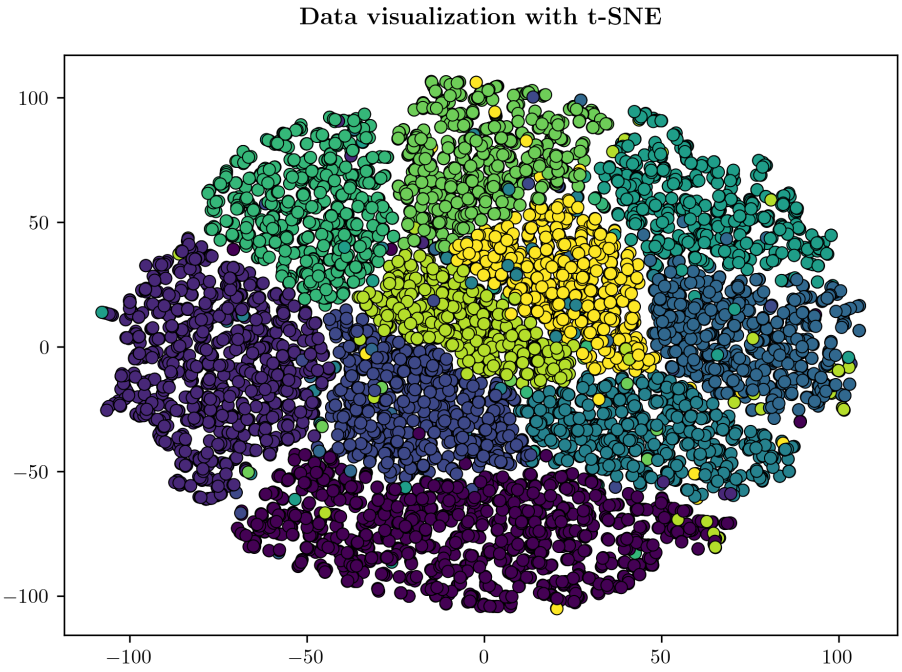

Fig. 3. Projection of the USPS dataset onto a two-dimensional space using t-SNE.

2.1 Description

In order to perform the multiclass classification task, we employed the Pegasos algorithm [2]. This algorithm was introduced to solve the Support Vector Machine (SVM) [3] convex optimization problem for binary classification, which can be formalized as follows.

Let $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$ be a training set with $\boldsymbol{x}_t \in \mathbb{R}^d \; \forall t = 1, \ldots, m$ and $y \in \{-1, 1\}$; and let $\phi_K : \mathbb{R}^d \to \mathcal{H}_K$ be a function such that $\dim(\mathcal{H}_K) \gg d$ and $\langle \cdot, \cdot \rangle_K$ is the inner product in $\mathcal{H}_K$. The optimization problem aims to find the maximum margin separating hyperplane in $\mathcal{H}_K$, which separates the examples according to their labels and maximizes the distance between the hyperplane and the closest example. This results in the following quadratic programming problem:

$$\begin{aligned} \min_{g \in \mathcal{H}_K, \boldsymbol{\xi} \in \mathbb{R}^m} \quad & \frac{\lambda}{2}\|g\|_K^2 + \frac{1}{m}\sum_{t=1}^{m} \xi_t \,, & (1) \\ \text{subject to} \quad & y_t \langle g, \phi_K(\boldsymbol{x}_t) \rangle_K \geq 1 - \xi_t \qquad t = 1, \ldots, m \,, \\ & \xi_t \geq 0 \qquad\qquad\qquad\quad t = 1, \ldots, m \,, \end{aligned}$$

where λ is a regularization coefficient dictating the relative importance of the two terms in the objective function; and ξ_t are slack variables that allow, but penalize, examples lying on the wrong side of the hyperplane.

This formulation can be turned into the following unconstrained optimization problem:

$$\min_{g \in \mathcal{H}_K} \quad \frac{\lambda}{2}\|g\|_K^2 + \frac{1}{m}\sum_{t=1}^{m} h_t(g) \,, \tag{2}$$

where $h_t(g) = [1 - y_t \langle g, \phi_K(\boldsymbol{x}_t) \rangle_K]_+$ is the hinge loss.

Pegasos solves this optimization problem using a stochastic gradient descent approach. The algorithm performs T iterations, each of which consists in drawing a training example uniformly at random and updating g through a gradient step. In particular, at iteration t Pegasos computes

$$g_{t+1} = \frac{1}{\lambda t}\sum_{r=1}^{t} \mathbb{I}\{h_{s_r}(g_r) > 0\} y_{s_r} K(\boldsymbol{x}_{s_r}, \cdot) \,, \tag{3}$$

where $\mathbb{I}$ is the indicator function, s_r is the index of the example randomly drawn at iteration r, and $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a kernel function such that $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi_K(\boldsymbol{x}), \phi_K(\boldsymbol{x}') \rangle_K$.

Note that g_{t+1} is a function, not a vector. Indeed, the dot in $K(\boldsymbol{x}_{s_r}, \cdot)$ stands for a missing argument, corresponding to the example on which the function is evaluated. Therefore, unlike vectors, g_{t+1} cannot be stored in memory as is and used to perform inference on new examples, since we first have to evaluate g_{t+1} on them.

In order to simplify the process, it was observed in [2] that (3) can be rewritten as

$$g_{t+1} = \frac{1}{\lambda t} \sum_{j=1}^{m} \boldsymbol{\alpha}_{t+1}[j] y_j K(\boldsymbol{x}_j, \cdot)\,, \tag{4}$$

where $\boldsymbol{\alpha}_{t+1} \in \mathbb{R}^m$ is a vector representing how many times each example $\boldsymbol{x}_j$ in the training set has been drawn and resulted in a non-zero loss, that is

$$\boldsymbol{\alpha}_{t+1}[j] = \left|\{r \leq t : s_r = j \wedge y_j \langle g_r, \phi_K(\boldsymbol{x}_j)\rangle_K < 1\}\right|.$$

The vector $\boldsymbol{\alpha}_{t+1}$ can be kept in memory and used to quickly evaluate g_{t+1} on new examples during both training and test. Algorithm 1 shows the pseudocode for the kernelized Pegasos algorithm using the abovementioned approach.

Algorithm 1: Pegasos algorithm in a kernel space

Hyperparameters: T, $\lambda > 0$
Data: Training set $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$
$\boldsymbol{\alpha}_1 \leftarrow (0, \ldots, 0)$
for $t = 1$ **to** T **do**
  Choose $s_r \in \{0, \ldots, |S|\}$ uniformly at random
  if $\frac{1}{\lambda t} \sum_{j=1}^{m} \boldsymbol{\alpha}_{t+1}[j] y_j K(\boldsymbol{x}_j, \boldsymbol{x}_{s_r}) < 1$ **then**
    $\boldsymbol{\alpha}_{t+1}[s_r] = \boldsymbol{\alpha}_t[s_r] + 1$
  end
end
return $\boldsymbol{\alpha}_{T+1}$

Once the vector $\boldsymbol{\alpha}_{T+1}$ has been obtained, the label of a new example $\boldsymbol{x}$ can be predicted by computing

$$\hat{y} = \operatorname{sgn}\left(\sum_{j=1}^{m} \boldsymbol{\alpha}_{T+1}[j] y_j K(\boldsymbol{x}_j, \boldsymbol{x})\right). \tag{5}$$

It is not necessary to multiply the sum by $\frac{1}{\lambda T}$, since scaling does not affect the sign.

So far we illustrated the Pegasos algorithm for binary classification. In order to handle multiclass classification, we prepare a different binary classifier for each class. These binary classifiers are trained using labels encoded as 1 for the examples belonging to the class of the classifier, and encoded as -1 for the examples belonging to all the other classes.

Finally, to determine the class of a new example $\boldsymbol{x}$, we first compute

$$h_i = \sum_{j=1}^{m} \boldsymbol{\alpha}_{T+1}[j] y_j K(\boldsymbol{x}_j, \boldsymbol{x}) \tag{6}$$

for each classifier c_i, and then predict with

$$\hat{y} = \arg\max_i h_i \,.$$

Also in this case scaling h_i is unnecessary, given that $\frac{1}{\lambda T}$ is always positive and therefore does not change the order of the h_i values.

What sets the Pegasos algorithm apart from other algorithms developed to solve the SVM problem is its ability to converge to a solution that differs from the optimal solution by less than ϵ in $O(\frac{1}{\lambda\epsilon})$ iterations with high probability with respect to the random draws of the training examples. In other terms, the convergence rate does not depend on the size of the training set, but only on the number of iterations, making Pegasos a suitable algorithm to perform classification over large datasets.

2.2 Kernel functions

The Pegasos algorithm is agnostic of the kernel function. In our experiments, we employed two kernel functions: the Gaussian kernel and the polynomial kernel. The Gaussian kernel is defined as

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2\gamma}\|\boldsymbol{x} - \boldsymbol{x}'\|^2\right),$$

with $\gamma > 0$. The polynomial kernel is defined as

$$K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \boldsymbol{x}^\top \boldsymbol{x}')^n \,,$$

where $n \in \mathbb{N}$ is the degree of the polynomial.

2.3 Implementation

In this section we expound on two implementation details employed to improve the runtime of the algorithm.

Kernel matrix. In order to perform multiclass classification, we trained multiple classifiers, each slated to run the Pegasos algorithm over the same training set. Within this framework, the kernel function between two given examples may be computed multiple times, resulting in redundant computations.

Moreover, at each iteration the Pegasos algorithm computes the kernel function between a randomly drawn training example and all the other training examples. Drawing the same example multiple times results in further redundant computations.

Redundant computations adversely affect the runtime of the algorithm and should therefore be avoided. To this effect, we opted to compute ahead of time the kernel matrix K, such that

$$K_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad \forall i, j = 1, \ldots, m \,.$$

This matrix was then dispatched to each classifier and used in place of the kernel function evaluation.

The drawback of the kernel matrix lies in its memory footprint. However, given the size of the USPS dataset, such a matrix would require in the worst case $\frac{9298 \times 9298 \times 32}{2^3 \times 10^9} \approx 0.34$ gigabytes of memory, which in modern machines constitutes a negligible amount.

Vectorization. The summations within the Pegasos algorithm can be vectorized, thus allowing to exploit the highly optimized array operations made available by math libraries.

In particular, the condition of the `if` statement within Algorithm 1 can be rewritten as

$$\frac{1}{\lambda t}(\boldsymbol{\alpha}_t \odot y)^\top K_{s_r} < 1\,,$$

where $\odot$ is the Hadamard product and K is the kernel matrix between the training examples. Whereas the formula for h_i in (6) can be rewritten as

$$\boldsymbol{h}_i = (\boldsymbol{\alpha}_{T+1} \odot y)^\top K'\,,$$

where K' is the kernel matrix between the training examples and the test examples and $\boldsymbol{h}_i$ is the vector containing the predictions for all test examples returned by classifier c_i.

Also the computation of the kernel matrices can be vectorized. In the case of the polynomial kernel, the vectorization is trivial.

alpha is a vector, bold

This is exacerbated

Remove tags for equations unless needed

Under scrutiny

3 Experiments

The experiments were run on a machine with 16 GB of RAM and a CPU Intel(R) Core(TM) i7-9700K 3.60GHz with 8 cores.

4 Conclusions

Statement

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

References

1. van der Maaten, L., Hinton, G.: Viualizing data using t-sne. Journal of Machine Learning Research **9**, 2579–2605 (11 2008)
2. Shalev-Shwartz, S., Srebro, N., Cotter, A.: Pegasos: Primal estimated sub-gradient solver for svm. Math. Program. **127**, 3–30 (03 2011)
3. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, second edn. (November 1999)