Project report
Machine Learning with Kernel Methods

Maha ELBAYAD
Ecole Normale Superieure de Cachan - Master MVA

maha.elbayad@student.ecp.fr

1 Introduction

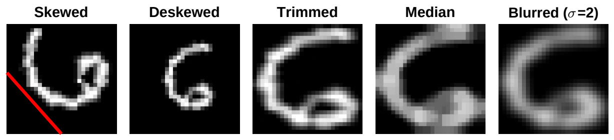
This project aims at implementing a kernel based method from scratch to train a classifier on the MNIST dataset. We focused solely on variants of support vector machines.

2 Outline

We distinguish three different phases, first the pre-processing of the training samples (noise filtering, scaling, skew correction...) to enhance the quality of the images. The second main task is the features extraction where we design other features than the raw pixel intensities and choose appropriate kernels.Finally, the third phase tackles the optimization problem and prepare the classifier outputs for prediction.

3 Pre-processing

Our training set $\mathcal{T} = \{(x^{(i)}, y^{(i)})\}_{1\leq i\leq m}$ consists of 5000 gray-scale 28×28 images distributed quite evenly among the 10 classes $\{0, 1, ..., 9\}$. We experimented with skew correction via Hough transform so that we vertically align the dominant line.



Another pre-processing that improves the performance of our classifier is trimming the image black border as shown above. Afterwards, we either apply the median filter to denoise the image or the Gaussian blur to lessen the variance within each class. The final pixel intensities are adjusted in $[0, 1]$ and with this pre-processing, normalizing the features globally doesn't improve the learning performance.

4 Features extraction

4.1 HOG features

In order to include information about the image gradients, we append HOG features (histograms of oriented gradients) to the raw pixels . They can be seen as a nonlinear function of the edge orientations : we first evaluate the vertical and horizontal gradients of the image and pool them into local neighborhoods to remove sensitivity to exact localization of the edges. For each block of size $b \times b$ we compute an n-bins histogram of the gradient orientations and the final feature would be the concatenation of all histograms

4.2 The kernels

For the raw and HOG features we tested the following kernels :

Linear	$K(x, y) = x^t y$	Raw, HOG
RBF	$K_\sigma(x, y) = \exp(-x^t y/(2\sigma^2))$	Raw, HOG
Poly	$K_d(x, y) = (1 + x^t y)^d$	Raw
χ_2	$K(x, y) = \sum_i (x_i - y_i)^2/(x_i + y_i)$	HOG
Intersection	$K(x, y) = \sum_i \min(x_i, y_i)$	HOG

Our implementation of those kernels is straightforward, thus the intersection and χ_2 kernels are computationally expensive which hinder their efficiency although they're proven suitable for the HOG features.

5 Support vector machines

5.1 Sequential Minimal Optimization (SMO)

We wish to solve the QP dual problem of the regularized SVM :

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & W(\alpha) = \mathbf{1}^t\alpha - \frac{1}{2}\alpha^t H\alpha \\ \text{subject to} \quad & \mathbf{0} \preceq \alpha \preceq C\mathbf{1} \\ & \alpha^t Y = 0 \end{aligned} \tag{QP1}$$

Where $Y = [y^{(1)}, ..., y^{(m)}]$ and $H = \operatorname{diag}(Y)K\operatorname{diag}(Y)$
For a vector x, we will predict :

$$\hat{y}(x) = \operatorname{sign} f(x) = \sum_{i=1}^{m} \alpha_i y^{(i)} K(x^{(i)}, x) + b$$

A point is optimal if and only if the KKT conditions are fulfilled :

$$\begin{aligned} \alpha_i = 0 \Rightarrow \quad & y^{(i)} f(x^{(i)}) \geq 1 \\ \alpha_i \in (0, C) \Rightarrow \quad & y^{(i)} f(x^{(i)}) = 1 \\ \alpha_i = C \Rightarrow \quad & y^{(i)} f(x^{(i)}) \leq 1 \end{aligned} \tag{KKT}$$

We implemented a simplified version of the SMO algorithm to solve (QP1) :

Repeat until convergence :

1. Select a violating term α_i i.e doesn't satisfy (KKT)
2. optimize $W(\alpha)$ with respect to α_i and a randomly selected term α_j while the other components are fixed.

We consider that the algorithm has converged if α remains unchanged after a few iterations (cf. full algorithm in appendix).

Other intricate heuristics to select the most violating pair (α_i, α_j) or a larger working selection were tested, but this version of SMO is sufficiently efficient.

5.2 Multiple kernel learning (MKL)

Given a set pf base kernels $\{K_k\}_{1ß\leq n}$, the objective of linear MKL is to jointly learn the SVM parameters and the linear combination of the base kernels :

$$K = \sum_{i=1}^{n} d_i K_i$$

The new QP problem is :

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \mathbf{1}^t\alpha - \frac{1}{2}\sum_{i=1}^{n} d_i \alpha^t H_i \alpha + \frac{\lambda}{2}\|d\|_2^2 \\ \text{subject to} \quad & \mathbf{0} \preceq \alpha \preceq C\mathbf{1} \\ & \alpha^t Y = 0 \end{aligned} \tag{QP2}$$

where for each kernel $H_i = \text{diag}(Y)K_i\,\text{diag}(Y)$. SMO is suitable to solve this new problem as well (cf. full algorithm in appendix)

6 Experiments and results

To compare our models we considered k-fold cross-validation ($k = 5$) so that all observations are used for both training and validation, in the following we report both the k-fold estimated accuracy and the Kaggle leader-board score.

6.1 One vs One

The default SVM algorithm solves for binary classifiers, and since our problem is a multi-class classification (10 classes) we consider all possible classifiers between two classes (a total of $\binom{10}{2} = 45$)

6.2 One vs Rest

For each class we learn a classifier where all the remaining classes are considered as one class. In this setting the positive class is a minority ; to weight out the imbalance effect we consider two different regularization parameters per class ($C_+ = \beta C_-$, $\beta > 1$)

6.3 Final prediction

With either one vs. one or one vs. rest strategy we end up with multiple scores for each observation. To combine those scores and output a final prediction we use the following :

- One vs. one : each classifier cast a vote in the predicted class and we outputs the class with the most votes.
- One vs. rest : in order to compare the 10 scores we learn the posterior class probability for each class using Platt's scaling technique :

$$\mathbb{P}(y = 1|x) \approx p_{a,b}(s) \equiv \text{sigmoid}(-(as + b))$$

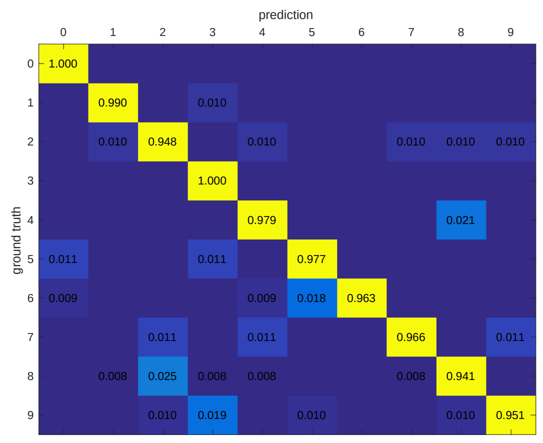
where s is the SVM score and a, b are optimized via logistic regression.

6.4 Results

Kernel	parameters	Acc(1)	Acc(2)
Raw(linear)	$C_- = 10, \beta = 1$	87.30%	
Raw(poly, $d = 3$)	$C_- = 10, \beta = 1$	95.80%	
Raw(poly, $d = 5$)	$C_- = 10, \beta = 1$	95.10%	
Raw(rbf, $\sigma = 4$)	$C_- = 10, \beta = 2$	95.70%	
Raw(rbf, $\sigma = 7$)	$C_- = 10, \beta = 2$	96.10%	95.42%
Raw(rbf, $\sigma = 10$)	$C_- = 10, \beta = 2$	95.90%	
HOG(linear)	$C_- = 10, \beta = 1$	92.30%	
HOG(intersection)	$C_- = 10, \beta = 1$	93.70%	
HOG(rbf, $\sigma = 2$)	$C_- = 10, \beta = 1$	96.10%	
HOG(rbf, $\sigma = 4$)	$C_- = 10, \beta = 1$	96.00%	
K_1= Raw(rbf, $\sigma = 7$)			
K_2= HOG(rbf, $\sigma = 4$)	$C_- = 10, \beta = 2$	96.90%	
K $= K_1 \odot K_2$			
K_1= Raw(rbf, $\sigma = 7$)			
K_2= HOG(rbf, $\sigma = 4$)			
K_3= HOG(linear)			
$K = K_1 \odot (K_2 + K_3 + K_2 \odot K_3)$	$C_- = 10, \beta = 3$		
with blurring and deskewing		97.10%	97.26%
without blurring nor deskewing		97.00%	97,08%
without trimming		94.90%	
MKL $\{K_1, K_2, K_3\}$		97.20%	96.90%

Table 1: *Acc(1) : k-fold accuracy - Acc(2) : Kaggle accuracy*

6.4.1 Best performance statistics



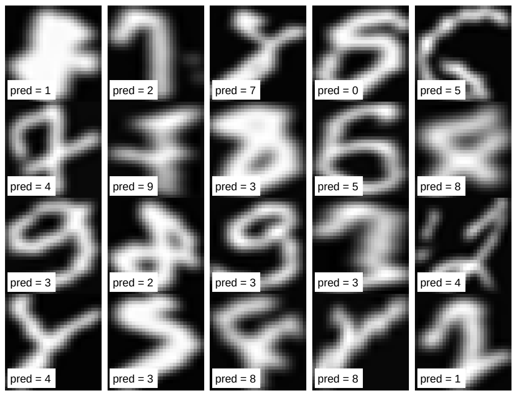


Figure 1: *Some misclassified samples are unrecognisable even by a human classifier, although the median filter might help us get rid of the outlier elements in the figures like the ! character above but the overall performance drops*

7 Conclusion

On the MNIST problem, some classes are more distinctive than others $\{0, 1, 3\}$ and although in one vs. one the classes are accurately separated the overall accuracy is generally lower than the one vs rest model with Platt's scaling. As for the chosen kernels the linear, rbf and exponential χ_2 yield better results on the hog space while the rbf is performant on the raw pixels space. Our implementation of the MKL smo is too mush slower than the standard svm smo and generally the elementwise product of kernels represnets the samples similarity better than a linear combination which is the only version we tested with MKL.

Références

Bottou, L., and Lin, C.-J. 2007. Support vector machine solvers. *Large scale kernel machines*, 301–320.

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

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., and Murthy, K. R. K. 2001. Improvements to platt's smo algorithm for svm classifier design. *Neural Computation 13*, 3, 637–649.

Lin, H.-T., Lin, C.-J., and Weng, R. C. 2007. A note on plattś probabilistic outputs for support vector machines. *Machine learning 68*, 3, 267–276.

Platt, J. C. 1999. 12 fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods*, 185–208.

Sun, Z., Ampornpunt, N., Varma, M., and Vishwanathan, S. 2010. Multiple kernel learning and the smo algorithm. In *Advances in neural information processing systems*, 2361–2369.

Algorithm 1: SVM SMO

```
**Inputs :** samples : $(x^{(i)}, y^{(i)})_{1 \le i \le m}$, $tol$ : tolerance, regularization $C \in \mathbb{R}^m$
**Outputs :** Lagrange multipliers $\alpha$ and the bias $b$
**Initialization :**
$\alpha = \mathbf{0}$, $b = 0$
$f = \mathbf{0}$ to store $f(x_i) = \sum_{j=1}^{m} \alpha_j y^{(j)} K(x^{(j)}, x^{(i)}) + b$
$epoch = 0$
while $epoch < 10$ do
    diff $= 0$ // number of updated coefficients $\alpha_i$

    for $i = 1, ..., m$ do
        compute $E_i = f(i) + b - y^{(i)}$
        if $[\, y^{(i)} E_i < -tol \,\wedge\, \alpha_i < C(i) \,] \vee [\, y^{(i)} E_i > tol \,\wedge\, \alpha_i > C(i) \,]$ then
            select $j \in \{1, ..., m\} \backslash \{i\}$
            compute $E_j = f(j) + b - y^{(j)}$
            /* compute bounds of $\alpha_j$ to satisfy the problem's conditions */

            if $y^{(i)} = y^{(j)}$ then
                $L = \max(0, \alpha_i + \alpha_j - C(i))$
                $H = \min(C(j), \alpha_i + \alpha_j)$
            else
                $L = \max(0, \alpha_j - \alpha_i)$
                $H = \min(C(j), C(i) + \alpha_j - \alpha_i)$
            if $|L - H| < tol$ then
                continue to the next $i$
            $\eta = 2K(i,j) - K(i,i) - K(j,j)$
            if $\eta \ge 0$ then
                continue to the next $i$
            /* update $\alpha_i$ and $\alpha_j$ */

        $a_j = \alpha_j - y^{(j)} \frac{E_i - E_j}{\eta}$
        crop $a_j$ to $[L, H]$
        if $|a_j - \alpha_j| < tol$ then
            continue to the next $i$
        $a_i = \alpha_i + y^{(i)} y^{(j)} (\alpha_j - a_j)$
        /* choose a suitable bias b */

        $b_1 = b - E_i - y^{(i)}(a_i - \alpha_i)K(i,i) - y^{(j)}(a_j - \alpha_j)K(i,j)$

        $b_2 = b - E_j - y^{(i)}(a_i - \alpha_i)K(i,j) - y^{(j)}(a_j - \alpha_j)K(j,j)$

        if $a_i \in [0, C(i)]$ then
            $b = b1$
        if $a_j \in [0, C(j)]$ then
            $b = b2$
        if $a_i \notin [0, C(i)] \,\wedge\, a_j \notin [0, C(j)]$ then
            $b = \frac{b_1 + b_2}{2}$
        update the terms of $f$
        $\alpha_i = a_i$
        $\alpha_j = a_j$
        diff ++
    if $diff = 0$ then
        epoch ++
    else
        epoch = 0
```

Algorithm 2: MKL SMO

```
**Inputs :** samples : $(x^{(i)}, y^{(i)})_{1 \le i \le m}$, $tol$ : tolerance, regularization $C \in \mathbb{R}^m$
**Outputs :** Lagrange multipliers $\alpha$ and the bias $b$
**Initialization :**
$\alpha = \mathbf{0}$, $b = 0$
$f = \mathbf{0}$ to store $f(x_i) = \sum_{j=1}^{m} \alpha_j y^{(j)} K(x^{(j)}, x^{(i)}) + b$
$epoch = 0$
while $epoch < 10$ do
    diff $= 0$ // number of updated coefficients $\alpha_i$

    for $i = 1, ..., m$ do
        compute $E_i = f(i) + b - y^{(i)}$
        if $[\, y^{(i)} E_i < -tol \,\wedge\, \alpha_i < C(i) \,] \vee [\, y^{(i)} E_i > tol \,\wedge\, \alpha_i > C(i) \,]$ then
            select $j \in \{1, ..., m\} \backslash \{i\}$
            $s = -y^{(i)} y^{(j)}$
            /* compute bounds of $\alpha_j$ to satisfy the problem's conditions */

            if $s = 1$ then
                $L = \max(-\alpha_i, -\alpha_j)$
                $H = \min(C(j) - \alpha_j, C(i) - \alpha_i)$
            else
                $L = \max(-\alpha_i, \alpha_j - C(j))$
                $H = \min(C(i) - \alpha_i, \alpha_j)$
            if $|L - H| < tol$ then
                continue to the next $i$
            /* compute the optimal step */

            for $1 \le l \le n$ do
                $A_l = K_l(i,i) + K_l(j,j) - 2K_l(i,j)$
                $B_l = y^{(i)} (\alpha \odot Y)^t (K_l(i,:) - K_l(j,:))$
            /* solve the cubic equation */

            $\Delta^* = \arg\max_{L \le \Delta \le U} (1+s)\Delta - ...$
                    $\frac{1}{8\lambda} (\sum_{l=1}^{n} (a_l \Delta^2 + 2 b_l \Delta + c_l)^2)$
            if $|\Delta| < tol$ then
                continue
            /* update $\alpha_i$ and $\alpha_j$ */

            $\alpha_i += \Delta$
            $\alpha_j += s\Delta$
            /* update the kernel coefficients $d$ */

            $d = d + \frac{1}{2\lambda} \Delta B$
            update the projections $f$
            /* choose a suitable bias b */

            $b_1 = y^{(i)} - f(i)$
            $b_2 = y^{(j)} - f(j)$
            if $a_i \in [0, C(i)]$ then
                $b = b1$
            if $a_j \in [0, C(j)]$ then
                $b = b2$
            if $a_i \notin [0, C(i)] \,\wedge\, a_j \notin [0, C(j)]$ then
                $b = \frac{b_1 + b_2}{2}$
            diff ++
            if $diff = 0$ then
                epoch ++
            else
                epoch = 0
```