Data Augmentation for Support Vector Machines

Roland Nzala-backa
Master MVA
ENS Paris-Saclay
roland.nzala-backa@polytechnique.edu

Abstract

I present in this project a latent variable representation of regularized support vector machines (SVM) that uses EM (Expectation-Maximisation), ECME (Expectation Conditional Maximisation Either) or MCMC (Markov chain Monte carlo) algorithms. After explaining the core principles of Support Vector Machines (SVM), I will first review the mathematical theory of [1] for these algorithms and then I will test these algorithms on a breast cancer dataset and a soccer players dataset. Pros and cons of the methods will be presented.

1 Introduction

Support vector machines are supervised learning algorithms used to solve regression and classification problems. Given a dataset of training examples, where each example is assigned to one of two category, the algorithm builds a model that gives new examples a group. It's binary classification.
In this project I present data augmentation for SVM .The main idea is to use latent variable representation for different algorithms : Expectation Maximisation (EM), Expectation Conditional Maximisation Either (ECME) , and Markov Chain Monte Carlo Methods (MCMC).
At start after explaining their interests, I will explain Support Vector machines computed with data augmentation, then I will present different algorithms that can be computed with this method. Afterwards I will discuss the results of some experiments of two different datasets, one of breast cancer, the other on a soccer players. During all this report, I will provide an reasoned point of view exposing the advantages and disadvantages of the different methods proposed by the authors.

2 Support Vector Machines principles

Let n be the number of samples, and $k-1$ the number of parameters (a column for the constant term is added). Let $X = (x_{i,j}), i \in [[1:n]], j \in [[1:k]]$, a matrix representing the dataset of k-1 predictors and a constant term. Each line is $x_i = (1, x_{i,2}, .., x_{i,k})$.
For a binary outcome $y_i \in \{-1, 1\}$, with a linear SVM model, the goal is to find a β coefficient such that the quantity of equation (1) is minimised for norm L_α. α is the norm used, ν a parameter and σ_j is the standard deviation of the j^{th} element of X.

$$d_\alpha(\beta, \nu) = \sum_{i=1}^{n} max(1 - y_i x_i^T \beta, 0) + \nu^{-\alpha} \sum_{j=1}^{k} |\frac{\beta_j}{\sigma_j}|^\alpha \tag{1}$$

Contrary to the classic penalisation model of L2 norm of beta, this method is more adapted to dataset with different values with the scaling variable σ_j

Minimizing $d_\alpha(\beta, \nu)$ is equivalent to find the mode of the pseudo distribution (equation (2))

$$p(\beta|\nu, \alpha, y) \propto exp(-d_\alpha(\beta, \nu)) \propto C_\alpha(\nu) L(y|\beta) p(\beta|\nu, \alpha) \tag{2}$$

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

$C_\alpha(\nu)$ is a term normalisation term, $L(y|\beta)$ is expressed in equation(3) and $p(\beta|\nu,\alpha)$ is expressed in equation(5).

The pseudo likelihood can be expressed as follow (equation 3)

$$L(y|\beta) = \prod_{i=1}^{n} L_i(y_i|\beta) = exp\left(-2\sum_{i=1}^{n} max(1 - y_i x_i^T \beta, 0)\right) \tag{3}$$

It can be represented has a mixture and has the density function of $f_{y_i|\beta}(\lambda_i)$ expressed as equation (4)

$$f_{y_i|\beta}(\lambda_i) = \frac{1}{\sqrt{2\pi\lambda_i}} exp\left(-\frac{1}{2}\frac{(1+\lambda_i - y_i x_i^T\beta)^2}{\lambda_i}\right) 1_{0\leq\lambda_i} \tag{4}$$

This equation exponential prior distribution of β with the regularisation penalty.

$$p(\beta|\nu,\alpha) = \prod_{j=1}^{k} p(\beta_j|\nu,\alpha) = \left(\frac{\alpha}{\nu\Gamma(1+\alpha^{-1})}\right)^k exp\left(-\sum_{j=1}^{k} |\frac{\beta_j}{\nu\sigma_j}|^\alpha\right) \tag{5}$$

According to Pollard (1946) and West(1987), it has the following density function $f_{\beta_j|\nu,\alpha}$, where $\phi(.|\mu,\sigma^2)$ is the density function of a normal law of mean μ and variance σ^2 and $p(\omega_j|\alpha) \propto \omega_j^{-\frac{3}{2}} St^{+}_{\frac{\alpha}{2}}(\omega_j^{-1})$ and $St^{+}_{\frac{\alpha}{2}}$ is the density function f a positive stable random variable of index $\frac{\alpha}{2}$

$$f_{\beta_j|\nu,\alpha}(\omega_j) = \phi(\beta_j|0, \nu^2\omega_j\sigma_j^2)p(\omega_j|\alpha)1_{0\leq\omega_j} \tag{6}$$

Let $\Lambda = diag(\lambda)$, $\Omega = diag(\omega)$, $\Sigma = diag(\sigma_1^2, ..., \sigma_k^2)$, $\mathbf{X}$ matrix where row i is equal to $y_i x_i$. I introduce the following vector b and matrix B such that $B^{-1} = \nu^{-2}\sigma^{-1}\omega^{-1} + \mathbf{X}^T\Lambda^{-1}\mathbf{X}$ and $b = B\mathbf{X}^T(\mathbf{1} + \lambda^{-1})$ Then by calculation, it follows :

$$p(\beta|\nu,\lambda,\omega,y) \sim N(b, B) \tag{7}$$

Finally with some other calculation for $i \in [[1:n]]$, $j \in [[1:k]]$

$$p(\lambda_i^{-1}|\beta, y_i) \sim IG(|1 - y_i x_i^T\beta^{(g)}|^{-1}, 1) \quad p(\omega_j^{-1}|\beta_j,\nu) \sim IG(\nu\sigma_j/\beta_j, 1) \tag{8}$$

where IG represents the inverse Gaussian distribution.

3 Algorithms

3.1 Expectation- Maximisation algorithms

The first algorithm explained is EM algorithm. In this case it is an iterative method to find the maximum argument β for the pseudo likelihood. It alternates two step: the (Expectation-Step) E-Step updates the parameters used to compute the likelihood, and the M-Step computes β that maximise the likelihood.

The EM-SVM algorithm is computed when ν is fixed and β is the parameter to optimise for the likelihood (algorithm 1).

Algorithm 1 EM-SVM

```
Repeat until convergence
E-step Given a current estimate $\beta = \beta^{(g)}$, compute
$\lambda^{-1(g)} \leftarrow |1 - y_i x_i^T\beta^{(g)}|^{-1}$,   $\Lambda^{-1(g)} \leftarrow diag(\lambda^{-1(g)})$,   $\Omega^{-1(g)} \leftarrow diag(\omega_j^{-1(g)})$
M-step Compute $\beta^{(g+1)}$ as
$\beta^{(g+1)} \leftarrow (\nu^{-2}\sigma^{-1}\omega^{-1(g)} + \mathbf{X}^T\Lambda^{-1(g)}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{1} + \lambda^{-1(g)})$
```

This algorithm is easy to implement, stable because the pseudo log likelihood decreases at each step, but the convergence is quite slow and can converge to only a local optimum of β, not a global one.

The ECME-SVM algorithm follows also the same steps of the previous algorithm, but in that case ν is also a variable of optimisation (algorithm 2). It is assumed here that the prior distribution of ν is the inverse gamma distribution of parameters a_ν and b_ν, such that :

$$p(\nu^{-\alpha}) \propto \left(\frac{1}{\nu^\alpha}\right)^{a_\nu - 1} exp(-b_\nu \nu^{-\alpha}) \tag{9}$$

Algorithm 2 ECME-SVM

```
Repeat until convergence
**E-step** Identical to the E-step of EM-SVM with $\nu = \nu^{(g)}$
**CM-Step** Identical to the M-step of EM-SVM with $\nu = \nu^{(g)}$
**CME-Step** $(\nu^{-\alpha})^{(g+1)} = \frac{b_\nu + \sum_{j=1}^{k} |\beta_j^{(g+1)}|/\sigma_j|^\alpha}{k/\alpha + a_\nu - 1}$
```

3.2 Markov chain Monte Carlo algorithms

Monte Carlo methods are techniques for randomly sampling a probability distribution and approximating a desired quantity, while Markov chain methods generate a sequence of random variables where the current value is dependent in probability on the value of a prior variable. The idea is to combine both methods : random sampling of probability distributions that honors the probabilistic dependence between samples by constructing a Markov Chain that comprise the Monte Carlo sample. At first the authors develop a MCMC-SVM algorithm for $\alpha = 1$ (algorithm 3)

Algorithm 3 MCMC-SVM

```
Repeat until convergence
**Step 1** Draw $\beta^{(g+1)} \sim p(\beta|\nu, \Gamma^{(g)}, \Omega^{(g)}, y) \sim N(b^{(g)}, B^{(g)})$
**Step 2** Draw $\lambda^{(g+1)} \sim p(\lambda|\beta^{(g+1)}, y)$, for $1 \le i \le n$ where $\lambda_i^{-1}|\beta, \nu, y_i \sim IG(|1 - y_i x_i^T \beta|^{-1}|, 1)$
**Step 3** Draw $\omega^{(g+1)} \sim p(\omega|\beta^{(g+1)}, y)$, for $1 \le j \le k$ where $\omega_j^{-1}|\beta_j \sim IG(\nu\sigma_j|\beta_j|^{-1}, 1)$
**Step 4** Draw $\nu^{(g+1)}$ from the conditional $p(\nu^{-1}|\beta_j, \nu) \sim \Gamma(\nu\sigma_j|\beta_j|^{-1}, 1)$
```

To compute β, the Rao - Blackwellised estimate is used, where

$$\hat{\beta} = E(\beta|y) = \frac{1}{G}\sum_{g=1}^{G} b^{(g)} \tag{10}$$

The implementation is easy and and adapted to high dimensional probability distribution, but the convergence can be slow (slower than EM), and it can be not very consistent.

An other idea is to introduce a regularisation term with a "spike and slab prior" (algorithm 4). It is a statistical approach to help select important features for a model, where each feature is assigned a "spike" and a "slab". The "spike" is a probability distribution centered at zero, meaning the feature is not important. The "slab" is a probability distribution centered at some non-zero value, indicating that the feature is important. The algorithm adjusts the probabilities of each spike and slab based on the data at each step of the algorithm.
I introduce note $\gamma = (\gamma_j)_{j \in [[1:k]]} = (\mathbf{1}_{0 \neq \beta_j})_{j \in [[1:k]]}$, and its prior probability distribution is $p(\gamma) = \prod_{j=1}^{k} \pi^{\gamma_j}(1-\pi)^{1-\gamma_j}$ (π is a parameter). For a vector v and a matrix M, v_γ and M_γ are the subset of v with non zero element and rows of columns of M where $\gamma_j = 1$ respectively. The posterior distribution of γ is (equation 17)

$$p(\gamma|y, \lambda, \nu) \propto p(\gamma)\left(\frac{|\Sigma_\gamma^{-1}/\nu^2|}{|B_\gamma^{-1}|}\right)^{\frac{1}{2}} exp\left(-\frac{1}{2}\sum_{i=1}^{n}\frac{(1 + \lambda_i - y_i x_{i,\gamma}^T b_\gamma)^2}{\lambda_i} - \frac{1}{2\nu^2} b_\gamma^T \Sigma_\gamma^{-1} b_\gamma\right) \tag{11}$$

While this method is useful to reduce the numbers of predictors, unfortunately, because it is needed to compute γ and the corresponding subset of parameters (vectors, matrix) subset, the complexity is proportional to $O(2^k)$ (exponential).

Algorithm 4 MCMC-SVM spike-and-slab

```
Repeat until convergence
Step 1 Draw $\lambda^{(g+1)} \sim p(\lambda|\beta^{(g+1)}, y)$, for $1 \leq i \leq n$ where $\lambda_i^{-1}|\beta, \nu, y_i \sim IG(|1 - y_i x_i^T \beta|^{-1}|, 1)$
Step 2 Draw $\gamma$ proportional to equation (11)
Step 3 Draw $\beta^{(g+1)} \sim N(b_\gamma^{(g)}, B_\gamma^{(g)})$
```

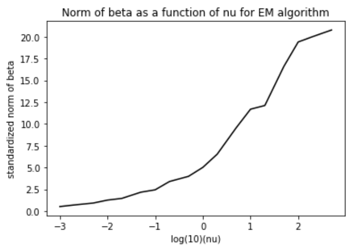

Figure 1: Norm L_2 of beta as a function of $log_{10}(\nu)$

4 Experimental results

The code can be seen at *'https://github.com/rolnz/MVA_Bayes_ML'* and is done with Python. Unfortunately, due to its time complexity, the spike and slab method can not be used for comparison. For each dataset, the training set is 75% of the data and the test set 25% of the data.

4.1 Cancer dataset

The data used is a dataset of breast cancer, present in the library of sklearn. It consist of 569 rows and contains 30 predictors of biological data from patient, in order to predict if the disease is benign (357 rows, 62,74 % of the data) or malignant (212 rows, 37,26 % of the data) .

At first for EM-algorithm, the norm of β is computed as a function of $log_{10}(\nu)$ (Figure 1) No coefficient of β is equal to zero in any case, but the norm of β increases with ν.
For the MCMC algorithm, I do 100 rehearsals of a sequence of 500 computations, and I keep the best β parameter. I compare the vales of β computed with EM, ECME, MCMC algorithm with the value of the default sklearn package of Python with penalisation of parameter $\nu = \nu_{ECME}$.

Globally that the values of β computed with EM and the ECME are really similar. The values computed by sklearn and MCMC are very different.

Finally, I plot the different values for every algorithm of the precision, the recall and the f1-score (Table 1). The baseline hypothesis is 'malignant'. The average computation MCMC does not product

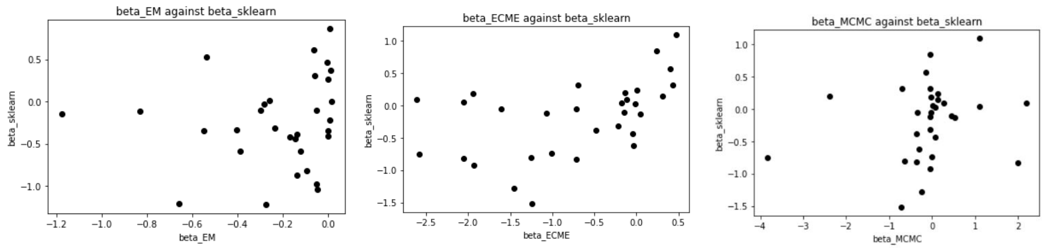

Figure 2: Coefficient from the EM algorithm plotted against coefficients from a standard sklearn SVM (left), Coefficient from the ECME algorithm plotted against coefficients from a standard sklearn SVM (center), Coefficient from the MCMC algorithm plotted against coefficients from a standard sklearn SVM (right)

Table 1: Precision, recall and f1-score of the different SVMalgorithms for the breast cancer dataset

Type of algorithm	Precision	Recall	F1-score
EM SVM	98.810	93.258	95.954
ECME SVM	98.837	95.506	97.143
sklearn SVM	98.864	97.753	98.305
MCMC SVM average	89.833	91.011	90.318
MCMC SVM best	96.703	98.876	97.778

Table 2: Precision, recall and f1-score of the different SVM algorithms for FM dataset (position: midfielder)

Type of algorithm	Precision	Recall	F1-score
EM SVM	70.896	72.519	71.698
ECME SVM	70.896	72.519	71.698
sklearn SVM	73.333	75.572	74.436
MCMC SVM average	70.331	70.702	70.437
MCMC SVM best	74.815	77.099	75.940

really interesting result, but the best one is nearly as efficient as the auther methods. The sklearn algorithm is a bit more efficient that the other methods.

4.2 Soccer Dataset

The objective is to study the dataset of players from the English and the Spanish league, to see if the characteristics of players are different. I use a dataset from Football Manager and I focus on attributes of players (height, weight and physical abilities, 52 parameters in total). I study 903 midfielders, because I found that it is the more discriminating position. The reference is the English Premier League.
I use the same algorithms as before and obtain the following results. (Table 2). The best computation of MCMC SVM method outperforms the other predictors, and based on a player characteristics, it is possible to predict relatively well if he plays in one league or other.

The confirmation of some clichés are visible on figure 3 : while the agility seems to be higher in the Spanish league (negative β_i coefficient), the work abilities seems to be higher in the English league (positive β_i coefficient).

5 Conclusion and Discussion

Data augmentation is a useful tool for SVM binary classification: the pseudo likelihood can be expressed as a mixture of normal distributions, and classical methods of optimisation like Expectation maximisation and Monte Carlo Chain methods are pertinent for SVM and can even outperform other regularised SVM algorithms. Nevertheless, the lack of consistency of MCMC, the exponential time complexity of spike and slab methods, and slow convergence of EM methods are limits to this approach.

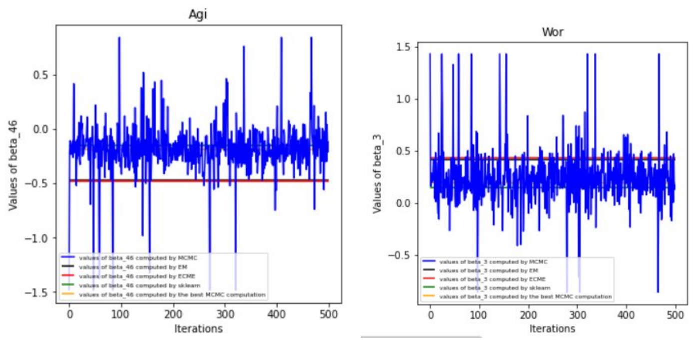

Figure 3: Sample paths and beta coefficients for the predictor 'agility' for MCMC(left), Sample paths and beta coefficients for the predictor 'work' for MCMC(rigth)