COMP9417 Machine Learning Project

Classification via Robust SVMs: A Comparison on the Wisconsin Breast Cancer Dataset

Daniel Woolnough, z5116128, Group h5116128, August 5, 2020

Introduction

Support Vector Machines (SVMs) are a useful machine learning tool for classification of labelled data into distinct sets, and are capable of modelling both linear and non-linear separable datasets via kernelization and projection into higher dimensional spaces. Recently, the Robust SVM (rSVM) has been proposed, which provides immunity to uncertainty in the dataset and has an easy reformulation as a second-order cone program (SOCP), a convex optimisation problem that is easily solvable by, e.g., interior-point methods. Other SVMs are the L_1-SVM, and the Doubly-regularized SVM (DrSVM).

The aim of this report is to assess some of these SVMs, and some other classification techniques (including a decision tree, a naive Bayes approach, and a neural network) on a well-used dataset, the Wisconsin Breast Cancer dataset (available here: `https://www.kaggle.com/uciml/breast-cancer-wisconsin-data`). In particular, we wish to evaluate the rSVM in the case where the given training set is noisy/uncertain, and classifications need to be made taking this into account.

This report is broken up into four sections:

1. We describe the dataset and the preprocessing carried out on it, to prepare it for use.
2. We compare a Decision Tree, a Gaussian Naive Bayes classifier, a Two-Layered Perceptron (Neural Network), and a standard SVM by their performance on the (preprocessed) dataset.
3. We define the mathematical formulations of the classic SVM, the DrSVM, and the rSVM, that are used to implement the methods in MATLAB. MATLAB has been chosen for its more powerful optimisation toolkits and its more user-friendly interface.
4. Finally, we evaluate these three SVMs on the dataset, now affected by noise.

The selected metric over which we evaluate models is simply the raw accuracy of the model on the dataset, as expected on the Kaggle webpage.

1 The Wisconsin Breast Cancer Dataset, and Preprocessing

The dataset contains 569 items and 32 columns:

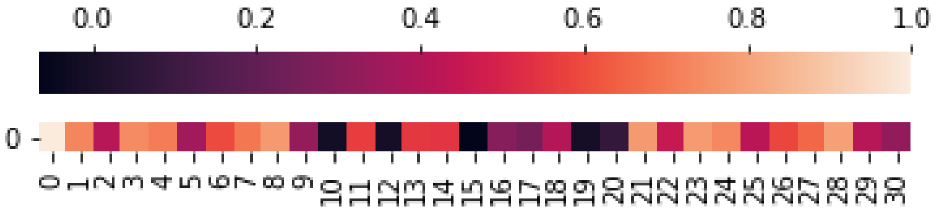
- The first column is the item ID, numbered from 0 through 568.

- The second column is the target, which is the diagnosis of the cancer as malignant (M) or benign (B). There are 357 benign diagnoses, and 212 malignant diagnoses, giving a class distribution of $[0.63, 0.37]$.
- The remaining thirty columns contain the mean, standard error, and extremal value (worst-case measurement) of 10 features: radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension.

For this dataset the following preprocessing was done:

- The ID column was dropped, as it does not affect the diagnosis.
- The diagnoses were mapped to the numeric values: 0 if benign, 1 if malignant.
- It was revealed there were no missing or NaN values in the dataset.
- Each column was normalised, by recentering at 0 (subtracting the mean), and dividing through byb the standard deviation.
- Finally, columns that had a correlation coefficient < 0.1 with the target value were dropped, as these were decided to be uncorrelated with the target, and therefore would not affect the diagnosis.

The correlation of each feature with the diagnosis (ordered as in the dataset, so diagnosis at position 0, etc.) is shown below.



This process is carried out by `preprocessing.py`, which then saves the resulting dataset in a new csv file. The corresponding notebook also demonstrates this process step by step.

2 Comparison of Non-Robust Methods

Taking our preprocessed dataset, we wish to make some preliminary comparisons of some common methods, to motivate the rest of the report. We chose the following (non-robust) methods:

- A decision tree, with the minimum number of samples at each leaf set to 2% of the training set, to avoid overfitting. It was determined that anything less than 1% would overfit, while anything over 4% would underfit.

- A Gaussian Naive Bayes classifier. A Gaussian model was chosen since our data was previously normalised. To accommodate the Bayesian assumption of independence of features (which clearly doesn't hold between, for example, radius mean, standard error, and extremal value), this classifier only considered the extremal values of each core feature.

- A Two-Layered Perceptron (neural network). The hidden layer was chosen to be twice the size of the input dimension (i.e. twice the number of features), with tanh activation at the hidden layer, and sigmoid activation at the output. The model was trained via stochastic gradient descent, and an adaptive learning rate initialised at 0.1.

- A Support Vector Machine, with a linear kernel function and regularisation parameter set to 0.05. The regularisation parameter was determined through a grid search which showed the optimal choice to mostly lie in the range $[0, 0.1]$.

For 100 simulations, a model was created and then trained on 80% of the dataset, and tested on the remaining 20%. For each simulation the accuracy (proportion of predictions which were correct) were recorded, as was the average of all simulations; see Table 1. We also recorded the F1 score for each model, for each simulation, to assess the reliability of the accuracy given the slight class imbalance; see Table 2.

Model	**Sim. 1**	**Sim. 2**	**Sim. 3**	**Sim. 4**	**Sim. 5**	**Average**
Decision Tree	0.9211	0.9386	0.9474	0.9298	0.9561	0.9326
Gaussian NB	0.9737	0.9474	0.9298	0.9474	0.9474	0.9445
Neural Network	0.9912	0.9825	1.0000	0.9386	0.9474	0.9729
Support Vector Machine	0.9912	0.9825	0.9912	0.9649	0.9737	0.9736

Table 1: Accuracy results for the four basic models. Sim i is the accuracy for the i^{th} simulation; Average is the average accuracy over 100 simulations.

Model	**Sim. 1**	**Sim. 2**	**Sim. 3**	**Sim. 4**	**Sim. 5**	**Average**
Decision Tree	0.9072	0.9213	0.9211	0.8974	0.9398	0.9082
Gaussian NB	0.9684	0.9348	0.9070	0.9231	0.9250	0.9252
Neural Network	0.9897	0.9738	1.0000	0.9091	0.9231	0.9633
Support Vector Machine	0.9897	0.9783	0.9877	0.9487	0.9620	0.9638

Table 2: F1 scores for the four basic models. Sim i is the F1 score for the i^{th} simulation; Average is the average F1 score over 100 simulations.

From the results we can see that, on average, the tuned SVM just outperforms the tuned Neural Network with one hidden layer, and is the best performing model of the four. We can also see the accuracies are reliable in that their deviation from the F1 score is small. Therefore, it is worth considering SVMs more generally in the rest of the report; in the following sections we will define four different SVM models and their mathematical formulations, and then compare their performance in a noisy setting.

3 Different SVM Models

We now present the four different SVM models that we will compare in the next section: the standard SVM, the DrSVM, the HHSVM, and the robust SVM. Each of these have been implemented from scratch in MATLAB, with the use of the optimisation package MOSEK, interfaced through YALMIP. The corresponding MATLAB live script gives a simple example demonstrating the implementation and solution retrieval for each.

We denote the number of datapoints as m, and the number of features as n.

3.1 Standard SVM

For this report, we take the standard SVM model as described in the lectures to be that with soft margins, as we do not want to assume linear separability. The primal formulation for this problem is

$$\begin{aligned} \min_{\boldsymbol{w}\in\mathbb{R}^n,\gamma,\boldsymbol{\xi}\in\mathbb{R}^m} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \lambda\boldsymbol{e}^T\boldsymbol{\xi} \\ \text{subject to} \quad & Y(X\boldsymbol{w} - \boldsymbol{e}\gamma) + \boldsymbol{\xi} \geq \boldsymbol{e} \\ & \boldsymbol{\xi} \geq 0 \end{aligned}$$

where $\boldsymbol{x}_i \in \mathbb{R}^n$ is the i^{th} data point, $X \in \mathbb{R}^{m\times n}$ is a matrix whose i^{th} row is $\boldsymbol{x}_i$, $\boldsymbol{y} \in \mathbb{R}^m$ is the classification vector (with each element y_i in $\{-1, 1\}$), $Y = \operatorname{diag}(\boldsymbol{y}) \in \mathbb{R}^{m\times m}$, and $\boldsymbol{e} = (1, 1, \ldots, 1)^T \in \mathbb{R}^m$. The optimisation variables are $\boldsymbol{w}$, the weight vector, γ the bias, and $\boldsymbol{\xi}$, the misclassification error for X: that is, $\xi_i = 0$ if $\operatorname{sign}(\boldsymbol{w}^T\boldsymbol{x}_i - \gamma) = y_i$, otherwise $\xi_i > 0$, $i = 1, \ldots, m$.

Typically we solve (SVM) by instead formulating and solving the dual problem, as this (a) allows for us to more efficiently the solve the problem if we wish to apply the kernel trick for high-dimensional embedding, and (b) still be able to retrieve the values for $\boldsymbol{w}$ and γ. The dual problem is given by

$$\begin{aligned} (SVM) \qquad \min_{\boldsymbol{u}\in\mathbb{R}^m} \quad & \frac{1}{2}\boldsymbol{u}^T YXX^TY^T\boldsymbol{u} - \boldsymbol{e}^T\boldsymbol{u} \\ \text{subject to} \quad & \boldsymbol{e}^TY^T\boldsymbol{u} = 0 \\ & 0 \leq \boldsymbol{u} \leq \lambda\boldsymbol{e} \end{aligned}$$

The original solution is then retrieved as follows: $\boldsymbol{x}_j$ is a support vector iff $0 < u_j < \lambda$, $j = 1, \ldots, m$; $\boldsymbol{w} = X^TY^T\boldsymbol{u}$; and γ satisfies $y_j(\boldsymbol{w}^T\boldsymbol{x}_j - \gamma) = 1$ for support vector $\boldsymbol{x}_j$.

In our experiments, we implement the dual formulation.

3.2 L_1-SVM [5]

The L_1-norm SVM is very similar to the standard SVM, except that it replaces the regularization term for $\boldsymbol{w}$ with that given by use of the L_1 norm instead.

$$\begin{aligned} \min_{\boldsymbol{w}\in\mathbb{R}^n,\gamma,\boldsymbol{\xi}\in\mathbb{R}^m} \quad & \|\boldsymbol{w}\|_1 + \lambda \boldsymbol{e}^T\boldsymbol{\xi} \\ \text{subject to} \quad & Y(X\boldsymbol{w} - \boldsymbol{e}\gamma) + \boldsymbol{\xi} \geq \boldsymbol{e} \\ & \boldsymbol{\xi} \geq 0 \end{aligned}$$

Due to the fact that $\|\boldsymbol{w}\|$ is not differentiable at $\boldsymbol{w} = \boldsymbol{0}$, an alternative reformulation for the above comes from Mangasarian [3], wherein we set $\boldsymbol{w} = \boldsymbol{p} - \boldsymbol{q}$, for $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^n_+$. This gives rise to the alternative formulation as a standard linear program:

$$\begin{aligned} (L_1SVM) \quad \min_{\boldsymbol{p},\boldsymbol{q}\in\mathbb{R}^n,\gamma,\boldsymbol{\xi}\in\mathbb{R}^m} \quad & \boldsymbol{e}_n^T(\boldsymbol{p}+\boldsymbol{q}) + \lambda \boldsymbol{e}_m^T\boldsymbol{\xi} \\ \text{subject to} \quad & Y(X(\boldsymbol{p}-\boldsymbol{q}) - \boldsymbol{e}_m\gamma) + \boldsymbol{\xi} \geq \boldsymbol{e}_m \\ & \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{\xi} \geq 0 \end{aligned}$$

One well documented advantage of the above method is that, for large values of λ, the problem is forced to restrict some weights to 0. In this sense, the problem is also ideal for feature selection: the task of choosing which features are necessary for classification. Feature selection will not be dealt with in this report however.

3.3 DrSVM [4]

The Doubly-Regularised SVM (DrSVM) aims to make the best of both of the previous methods, by using **two** regularization terms in the objective function: one is the previous L_2-norm regularization (or *ridge* regularization); the other is the L_1-norm (or *lasso*) regularization. The primal formulation is thus given by

$$\begin{aligned} (DrSVM) \quad \min_{\boldsymbol{w}\in\mathbb{R}^n,\gamma,\boldsymbol{\xi}\in\mathbb{R}^m} \quad & \frac{\lambda_1}{2}\|\boldsymbol{w}\|_2^2 + \lambda_1\|\boldsymbol{w}\|_1 + \boldsymbol{e}^T\boldsymbol{\xi} \\ \text{subject to} \quad & Y(X\boldsymbol{w} - \boldsymbol{e}\gamma) + \boldsymbol{\xi} \geq \boldsymbol{e} \\ & \boldsymbol{\xi} \geq 0 \end{aligned}$$

A recent method for efficiently solving the above problem was recently proposed in [1] which, inspired by the approach by Mangasarian as above, also uses the substitution $\boldsymbol{w} = \boldsymbol{p} - \boldsymbol{q}$ and then, under some simple assumptions, proposes a dual that is quadratic program:

$$\min_{\boldsymbol{u}\in\mathbb{R}^m} \quad \frac{1}{2}\boldsymbol{u}^T\left(\hat{Y}\left(\hat{X}\left(C+\nu I_{2n+m}\right)^{-1}\hat{X}^T + \boldsymbol{e}\boldsymbol{e}^T\right)\hat{Y} + M\right)\boldsymbol{u} - \boldsymbol{e}^T\boldsymbol{u}$$

where each matrix is in $\mathbb{R}^{(2n+m)\times(2n+m)}$ and is given by

$$\hat{Y} = \begin{pmatrix} Y & 0 \\ 0 & 0 \end{pmatrix}, \; \hat{X} = \begin{pmatrix} 0 & X & -X \\ 0 & 0 & 0 \end{pmatrix}, \; C = \lambda_1 \begin{pmatrix} 0 & 0 & 0 \\ 0 & I_n & 0 \\ 0 & 0 & I_n \end{pmatrix}, \text{ and } M = (C+\nu I_{2n+m})^{-1}(2\hat{Y}\hat{X} + I_{2n+m})$$

To retrieve our original solution, we do the following:

$$\begin{bmatrix} \boldsymbol{\xi} \\ \boldsymbol{p} \\ \boldsymbol{q} \end{bmatrix} = (C + \nu I_{2n+m})^{-1}((\hat{Y}\hat{X})^T + I_{2n+m})\boldsymbol{u}, \quad , \gamma = -\boldsymbol{e}^T\hat{Y}\boldsymbol{u}, \boldsymbol{w} = \boldsymbol{p} - \boldsymbol{q}$$

3.4 Robust SVM [2]

Finally we reach the robust SVM, the main model we wish to evaluate in this paper, in comparison to the non-robust approaches. In practice, the input data is sensitive to error, be it measurement error, data uncertainty, noise, etc. The robust approach makes no assumption on the nature of this uncertainty other than that, for some radius $r_i > 0$, $i = 1, \dots, m$, each data point is bounded in an n-dimensional ball:

$$\boldsymbol{x}_i \in \mathcal{U}_i(r_i) = \bar{\boldsymbol{x}}_i + r_i\mathbb{B}_n, \quad i = 1, \dots, m$$

where $\mathbb{B}_n = \{\boldsymbol{v} \in \mathbb{R}^n : \|\boldsymbol{v}\|_2 \leq 1\}$ is an n-dimensional ball. The robust methodology aims to take this uncertainty into account, and so find a solution to our standard SVM problem that is feasible (satisfies all constraints) no matter where the data points lie in their uncertainty sets $\mathcal{U}_i$:

$$\begin{aligned} \min_{\boldsymbol{w}\in\mathbb{R}^n, \gamma, \boldsymbol{\xi}\in\mathbb{R}^m} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \lambda \boldsymbol{e}^T\boldsymbol{\xi} \\ \text{subject to} \quad & y_i(\boldsymbol{x}_i^T\boldsymbol{w} - \gamma) + \xi_i \geq 1, \;\; \forall \boldsymbol{x}_i \in \mathcal{U}_i(r_i), \;\; i = 1, \dots, m \\ & \xi_i \geq 0, \;\; i = 1, \dots, m, \end{aligned}$$

Following the robust methodology and the approach in [2], we reformulate the above into a second-order cone program, that no longer has the semi-infinite constraints posed by the forall quantifier:

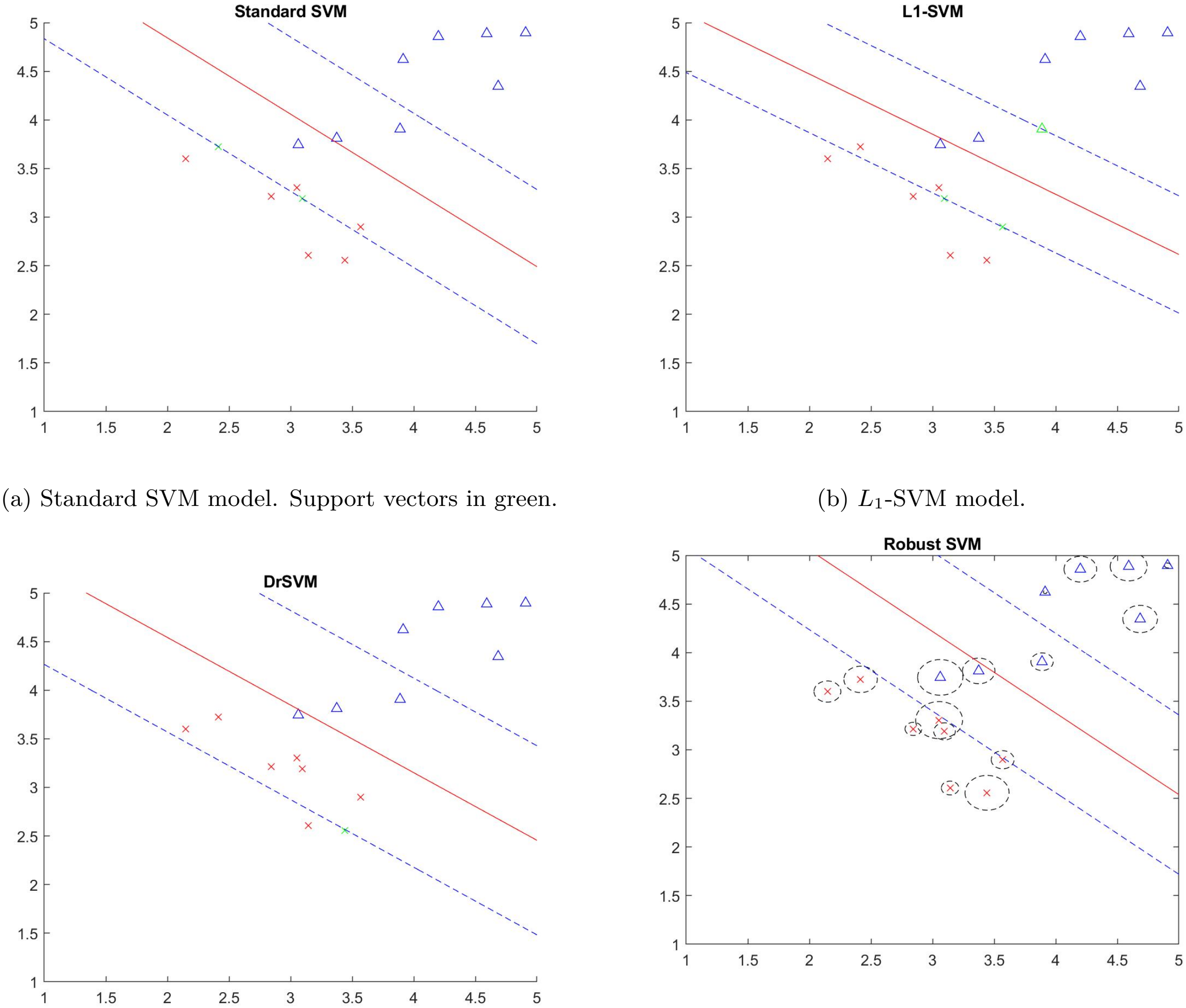
$$\begin{aligned} (rSVM) \quad \min_{\boldsymbol{w}\in\mathbb{R}^n, \gamma, \boldsymbol{\xi}\in\mathbb{R}^m, t_1, t_2} \quad & \frac{1}{2}t_1^2 + \lambda t_2 \\ \text{subject to} \quad & y_i\left(\bar{\boldsymbol{x}}_i^T\boldsymbol{w} - \gamma\right) - y_i r_i\|\boldsymbol{w}\|_2 + \xi_i \geq 1 \\ & \|\boldsymbol{w}\|_2 \leq t_1 \\ & \boldsymbol{e}^T\boldsymbol{\xi} \leq t_2 \\ & \boldsymbol{\xi} \geq 0 \end{aligned}$$

Notice that in the case where there is no uncertainty (i.e. $r_i = 0$, $i = 1, \dots, m$) then $(rSVM)$ reduces to (SVM), as would be expected.

Simple Comparison

A simple comparison between the four methods is demonstrated below. The plots were generated with `svms_example.m`, and can also be examined interactively with the accompanying MATLAB Live script.

As we can see, the robust method does not identify any of the points as support vectors. This is because, inherent in the methodology, the formulation actually considers the *worst-case* in the



(a) Standard SVM model. Support vectors in green.

(b) L_1-SVM model.

(c) DrSVM model.

(d) Robust SVM model. Uncertainty sets circled. In this case, the "support vector" is the cross located at approximate $(3.125, 3.25)$, as the margin borders the uncertainty set.

uncertainty set surrounding each point. To this effect, we can observe that the cross located at approximately $(3.125, 3.25)$ is the "support vector" chosen, as the margin borders the uncertainty set at its extremal value (in the 2-norm sense).

We can now progress to testing our four SVM models with regards to their resilience in the face of uncertainty.

4 Classification Under Noisy Data

In order to evaluate the worth of the rSVM method in comparison to the SVM, the L_1SVM, and the DrSVM, we will perform the following experiment:

- Firstly, after splitting our dataset into training and test sets, we will choose small values for r_i, $i = 1, \ldots, m$, and perturb our m training data points within the uncertainty set $\mathcal{U}_i(r_i)$.

- We will then train our four models on this perturbed training set.
- The four models will then be tested on the *original* test set, and evaluated on (a) their accuracy, and (b) their F1 score.

The motivation behind this is as follows: given a noisy dataset, we will be testing how well our four models can classify true data.

References

[1] M. Dunbar, J. Murray, L. A. Cysique, B. J. Brew, V. Jeyakumar: "Simultaneous classification and feature selection via convex quadratic programming with application to HIV-associated neurocognitive disorder assessment", *European Journal of Operational Research*, (206), 470-478, 2010.

[2] M. A. Goberna, V. Jeyakumar, G. Li: "Calculating Radius of Robust Feasibility of Uncertain Linear Conic Programs via Semidefinite Programs", *UNSW Preprint*, `https://arxiv.org/abs/2007.07599`, 2020.

[3] O. L. Mangasarian: "Exact 1-Norm Support Vector Machines via Unconstrained Convex Differentiable Minimization", *Journal of Machine Learning Research*, (7), 1517-1530, 2006.

[4] L. Wang, J. Zhu, and H. Zou: "The Doubly Regularized Support Vector Machine", *Statistica Sinica*, (16), 589–615, 2006.

[5] J. Zhu, S. Rosset, T. Hastie, R. Tibshirani: "1-Norm support vector machines", `http://wwwstat.stanford.edu/~hastie/Papers/`, 2003.