Regularization and transformation: CSE802 Project

Jonny Dowdall, James Peterkin II, Eric Alan Wayman

Submitted May 4, 2018

1 Introduction

1.1 Motivation

The fact that the performance of classifiers can be improved through methods that 1) impose some form of regularization on the estimated parameters, or 2) exploit useful transformations of the feature space, suggests that understanding of the intracies and interrelationships between these techniques is an important part of any practitioner's toolbox in the field of pattern recognition and machine learning. Our project seeks to build such an understanding and to test that understanding against the application of the techniques to real-world datasets.

The basic idea of regularization is that when fitted to a particular training data set models may learn the idiosyncracies of that data set (overfitting) and that will be reflected in poorer performance on an unseen test data set compared to the performance of the average of a series of models fit to various training sets. Regularization simulates this procedure by restricting the magnitude of components of the parameter vector, forcing the fitted model to be more "smooth" than if the parameter vector were unrestricted. Overfitting can also be addressed through sparsity-encouraging regularization (ℓ_1 regularization, which increases the "signal to noise ratio" by encouraging the parameter coefficients of features that do not contribute to reducing model error to go to zero).

Underfitting occurs when the model is too simple for the data. Since the models in this project produce linear decision boundaries, this will occur when features are not linearly separable in the original feature space. Transforming the data to a higher-dimensional feature space, in this project demonstrated by the use of the RBF kernel, may allow the data to become linearly separable in that feature space. Through the use of the kernel trick, the model can be fit in the higher-dimensional space through performing calculations in the original feature space, and the resulting decision boundary visualized in the original feature space.

Overfitting can also be addressed through fitting the data using a subset of the training points, and by choosing a linear boundary that separates the data points with as large a "margin", or distance from the closest points to the boundary, as possible. Both of these qualities are characteristics of the Support Vector Machine (SVM).

Another classifier, logistic regression, can be used in conjunction with the ℓ_1 and ℓ_2 norms for regularization, as well as with transformed data (kernelized logistic regression). The details of all these techniques will be explained in sections to follow.

1.2 Literature review

Regularization was first proposed by Tychonoff (Theodoridis 2015, 72) in 1977 (Tychonoff and Arsenin 1977). ℓ_1 and ℓ_2 regularization are common techniques described in many machine learning textbooks, for example Murphy 2015.

Logistic regression, the first of the two classifiers that were used in this project, was invented by David Cox in 1958 (Cox 1958).

According to Wikipedia, kernel classifiers were likely first mentioned in the 1960s (Aizerman et al. 1964) and gained widespread attention due to the introduction of the support vector machine (SVM) in the 1990s due to the SVM's competitive performance on tasks such as handwriting recognition. This SVM, which uses the "kernel trick" to be described below was invented by Boser et al. in 1992 (Boser et. al 1992).

The usage of kernels with logistic regression is demonstrated in Zhu and Hastie (2002).

The techniques explained and used in this project are widely used in pattern classification (for example see).

1.3 Problem statement

To explain how techniques of regularization and data transformation can be used in conjunction with the binary classifers logistic regression and SVM, use this understanding to hypothesize regarding the performance of the various techniques on datasets with differing characteristics, and test these hypotheses on real-world datsets. Our hypotheses are as follows:

On datasets which are linearly separable, if most of the features are useful towards classification we expect logistic regression with ℓ_2 regularization and SVM with a linear kernel to perform well. If only a few of the features are useful towards classification on such a dataset, we expect logistic regression with ℓ_1 regularization to perform better.

On datasets which are not linearly separable, if most of the features are useful towards classification we expect the SVM with an RBF kernel to perform well. If only a few of the features are useful towards classification on such a dataset, we expect logistic regression with an RBf kernel to perform better.

2 Approach

2.1 Logistic regression with ℓ_1 and ℓ_2 regularization

Logistic regression is a discriminative classifier. It corresponds to a binary classification model:

$$p(y|\boldsymbol{x}, \boldsymbol{w}) = \mathrm{Ber}(y|\mathrm{sigm}(\boldsymbol{w}^T\boldsymbol{x}))$$

(Murphy 2012, 245) where sigm is the sigmoid function. If the possible values of y are either -1 or $+1$, then $p(y = 1) = 1/(1 + \exp(-\boldsymbol{w}^T\boldsymbol{x}))$ and $p(y = -1) = 1/(1 + \exp(\boldsymbol{w}^T\boldsymbol{x}))$. We minimize the error by maximizing the negative log-likelihood:

$$NLL(\boldsymbol{w}) = \log(1 + \exp(-y_i\boldsymbol{w}^T\boldsymbol{x}))$$

(Murphy 2012, 245). There is not a closed-form solution for the MLE of $\boldsymbol{w}$, so it must be estimated by an optimization algorithm. However, we must often design constraints to prevent the parameters from overfitting the training data and losing generality.

2.1.1 The effects of ℓ_1 and ℓ_2 regularization

When the ℓ_2 regularization term is included, maximizing the NLL function with respect to $\boldsymbol{w}$ and λ tries to reduce the norm of $\boldsymbol{w}$ (the vector of parameters) while at the same time minimizing the error given by the log-likelihood cost function (maximizing the negative of this function). This helps prevent overfitting: by restricting the ℓ_2 norm of $\boldsymbol{w}$, the "complexity" of the model is restricted, so it is prevented from "learning too much about the idiosyncrasies of the specific training data set" (Theodoridis 2015, 74).

If only a few features contain significant information and there are a large number of features, the "true" model generating the data will have the coefficients of most components of $\boldsymbol{w}$ equal to zero. Therefore it

The following figure (Figure 1, taken from Theodoridis 2015, 406, Figure 9.2) shows the relationship between a given component θ of the parameter vector $\boldsymbol{\theta}$ (what we call $\boldsymbol{w}$) and its contribution to $\|\boldsymbol{\theta}\|_p$, $|\theta|^p$, for given levels of p. For ℓ_p norms with $p \geq 1$, components θ with larger $|\theta|^p$ give a larger contribution to the norm, so assuming for example's sake that two components θ_1 and θ_2 have the same effect on the fit of the model and $|\theta_1|^p > |\theta_2|^p > 1$, the minimization will try to reduce the size of θ_1 more than θ_2. Conversely, for $p > 1$, any θ_j with $|\theta_j|^p < 1$ will not have its size reduced very much at all, irrespective of the amount to which it contributes to minimizing the error of the model.

2.1.2 Applying regularization to logistic regression

ℓ_1 regularization is achieved by adding the term $\lambda\|\boldsymbol{w}\|_1{}^2$ where $\|\boldsymbol{w}\|_1 = \sum_{i=1}^{c} |\boldsymbol{w}_i|$ (Theodoridis 2015, 404), so

$$NLL(\boldsymbol{w}, \lambda) = \log(1 + \exp(-y_i \boldsymbol{w}^T \boldsymbol{x})) + \lambda\|\boldsymbol{w}\|_1$$

ℓ_2 regularization is achieved by adding the term $\frac{\lambda}{2}\|w\|_2{}^2$ to $NLL(\boldsymbol{w})$ above, giving

$$NLL(\boldsymbol{w}, \lambda) = \log(1 + \exp(-y_i \boldsymbol{w}^T \boldsymbol{x})) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2{}^2$$

However, for $p = 1$, even components θ_j with $|\theta_j|^1 < 1$ will have the regularization applied to them. Therefore irrespective of the size of a true θ_j, the regularization will force θ_j to 0 if it does not contribute to minimizing model error.

(The above discussion was based Theodoridis 2015, 406-407)

2.2 Kernels: linear vs RBF

Kernels are commonly used to model similarities over pairs of data points. A Mercer kernel is a kernel whose Gram matrix

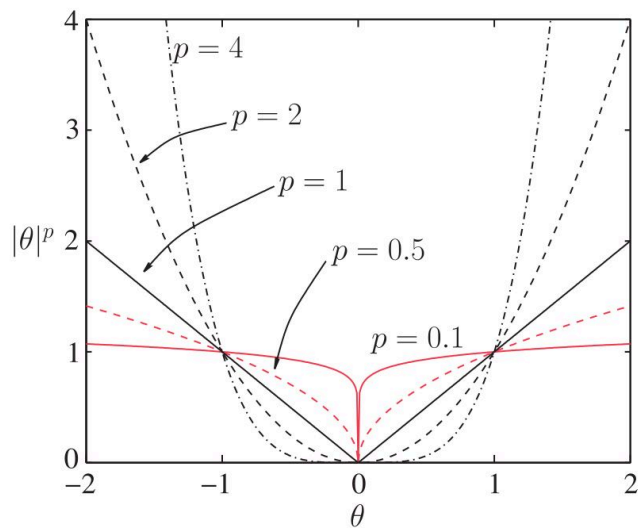

Figure 1: CHANGE THISSS!!!!!!!!!

$$\boldsymbol{K} = \begin{pmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ & \vdots & \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

is positive semi-definite for any set of inputs $\{\boldsymbol{x}_i\}_{i=1}^N$ (Murphy 2012, 481). For any Mercer kernel there exists a function $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^D$ for which then $K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{x})$. Note that D can be infinite, as explained in the section "SVM and RBF kernel relationship explanation."

In this project we use two kernels, linear kernels and the RBF kernel, both of which are Mercer kernels. The kernels will be used in this project as transformations of data to be input to classifiers which produce a linear decision boundary (if transformed data is input to a classifier, the resulting decision boudary will be linear in that transformed space).

Note that usually it is hard to derive the feature vector $\boldsymbol{\phi}(\boldsymbol{x})$ from a Kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}')$, but the reverse is not difficult for a Mercer kernel since $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})$.

The linear kernel is $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$, which corresponds to the case where $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{x}$, so $\boldsymbol{\phi}(\boldsymbol{x})$ takes points in $\mathcal{X}$ to $\mathcal{X}$. This kernel is useful in the case where the decision boundary is linear in the original feature space, so transforming them to a higher-dimensional feature space is not necessary (Murphy 2012, 482).

The RBF kernel is defined as follows:

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

As noted above, the D in $\boldsymbol{\phi}(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}^D$ is infinite in the case of the RBF kernel. To understand the transformation, following Abu-Mostafa et al. (8-37), let $\gamma = 1$ and $\boldsymbol{x}$ be a scalar. Then

$$
\begin{aligned}
K(x, x') &= \exp\left(-\left\|x - x'\right\|^2\right) \\
&= \exp\left(-(x)^2\right) \cdot \exp\left(2xx'\right) \cdot \exp(-\left(x'\right)^2) \\
&= \exp\left(-(x)^2\right) \cdot \left(\sum_{k=0}^{\infty} \frac{2^k (x)^k \left(x'\right)^k}{k!}\right) \cdot \exp\left(-\left(x'\right)^2\right)
\end{aligned}
$$

Defining

$$
\boldsymbol{\phi}(x) = \exp(-x^2) \cdot \left(1, \sqrt{\frac{2^1}{1!}} x, \sqrt{\frac{2^1}{2!}} x^2, \sqrt{\frac{2^1}{3!}} x^3, \ldots\right)
$$

we see that $K(x, x') = \boldsymbol{\phi}(x)^T \boldsymbol{\phi}(x)$. The right hand side is an inner product in an infinite-dimensional feature space, which shows that the D in the range of K can be infinite.

2.2.1 The "kernel trick"

If it is difficult to compute $\boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{x})$, instead we can compute $K(\boldsymbol{x}, \boldsymbol{x}')$ in the original $\mathcal{X}$ space since the results are equal. For the kernels used in this project, this is useful for the RBF kernel, as exact calculation of $\boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{x})$ in the range space of $\boldsymbol{\phi}$ is impossible.

2.3 SVMs

The SVM is a classifier that incorporates sparsity of data points (as opposed to features) into its loss function (Murphy 2012, 497). SVMs for classification use a loss function called hinge loss, which is of the form $L_{\text{hinge}}(y, \eta) = \max(0, 1 - y\eta) = (1 - y\eta)_+$ where $\eta = f(\boldsymbol{x})$ is the "confidence" (not necessarily a probability) in choosing label $y = 1$ (Murphy 2012, 499). The objective function is

$$
\min_{\boldsymbol{w}, w_0} \frac{1}{2} \|\boldsymbol{w}\|_2^2 + C \sum_{i=1}^{N} (1 - y_i f(\boldsymbol{x}_i))_+
$$

This is non-differentiable, but by introducing slack variables, the minimization problem can be transformed to one solvable by quadratic programming (Murphy 2012, 499).

2.3.1 Generalization and the large-margin principle

The minimization problem mentioned in the previous paragram can be obtained through a different approach, namely maximizing the size of the margin $f(\boldsymbol{x}) / \|\boldsymbol{w}\|_2$. This approach also depends on the introduction of slack variables which allows the problem to handle certain cases. The resulting objective function is the same as the approach from minimizing the hinge loss function.

The importance of the large-margin is that it helps the model's generalization performance (Theodoridis 2015, 550). An intuitive way to see this is by Figure 2 (this is Figure 14.11 from Murphy 2012, 500).

2.3.2 Generalization and support vectors

The solution for the weights for the SVM has the form $\widehat{\boldsymbol{w}} = \sum_i \alpha_i \boldsymbol{x}_i$ where $\boldsymbol{\alpha}$ has many entries equal to 0; the $\boldsymbol{x}_i$ corresponding to non-zero α_i are called support vectors. Since the parameter vector for the fitted SVM depends only on a subset of data points, this helps model generalizability (Theodoridis and Koutroumbas 2009, 206).

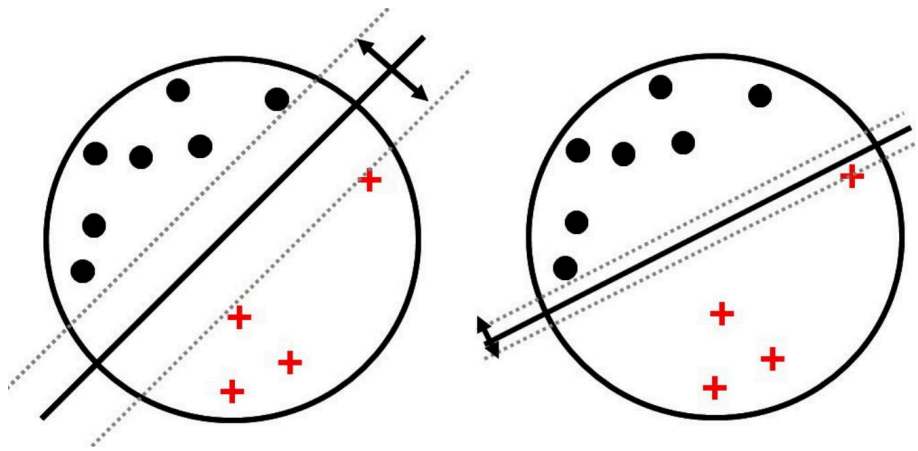

Figure 2: A simple caption

The SVM is used in this project with both the linear and RBF kernels.

2.4 SVM and logistic regression with the RBF kernel: a close relationship

In this section, we explain the effects of using logistic regression on data that has been transformed with the RBF kernel, and how this relates to the case where an SVM is used with such transformed data.

The optimal $f(\boldsymbol{x})$ in fitting an SVM is of the form $f(\boldsymbol{x}) = \sum_{i=1}^n \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i')$ (Zhu and Hastie 2002, 186). Also since the negative log-likelihood (NLL) for logistic regression has a similar shape to the NLL of the SVM, replacing the NLL of the SVM with the NLL of the logistic regression gives roughly the same solution (Zhu and Hastie 2002, 186). Then for a Mercer kernel, the interpretation of the probability $p(\boldsymbol{x})$ (which equals $P(y = 1|\boldsymbol{X} = \boldsymbol{x})$, Lin 2002) is

$$
\begin{aligned}
p(\boldsymbol{x}) &= \frac{e^{f(\boldsymbol{x})}}{1 + e^{f(\boldsymbol{x})}} = \frac{1}{1 + \exp(-f(\boldsymbol{x}))} \\
&= \frac{1}{1 + \exp(-\sum_{i=1}^n \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i'))} && \text{plugging in the optimal solution} \\
&= \frac{1}{1 + \exp(-\sum_{i=1}^n \alpha_i \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{\phi}(\boldsymbol{x}))} && \text{using the kernel trick} \\
&= \frac{1}{1 + \exp(-\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}))}
\end{aligned}
$$

where the last step is by defining $\boldsymbol{w} = \sum_i \alpha_i \boldsymbol{\phi}(\boldsymbol{x}_i)$ is the weighted sum of transformed support vectors. The last two steps here were taken from Guestrin (2007). This implies that the kernel

trick can be used to run logistic regression on data that has been transformed to an infinite-dimensional feature space using the ϕ corresponding to the RBF kernel.

2.5 Summary of model fitting strategies and data transformations

The following table summarizes the combinations of model fitting strategies and data transformations used in this project. Each column indicates a different model fitting strategy (used in conjunction, of course, with minimizing model error as represented by a loss function), while each row indicates a type of data transformation. Each cell indicates the classifier that was used in conjuction with the fitting strategy and data transformation. Note that the model used for any particular combination is deterministic: in other words, the desired model fitting strategy and data transformation indicate a model choice.

	vanilla	ℓ_1	ℓ_2	few data pts & large margin
linear		log reg	log reg	SVM
RBF	log reg			SVM

vanilla: loss function with no regularization term

loss + ℓ_1: loss function with ℓ_1 regularization term

loss + ℓ_2: loss function with ℓ_2 regularization term

few data pts & large margin model attempts to utilize small subset of of training data in parameter estimation, and attempts to maximize size of margin while minimizing error

3 Datasets

4 Experimental analysis

5 Findings

6 Summary and future work

7 Works cited

Abu-Mostafa, Malik Magdon-Ismail and Hsuan-Tien Lin. 2012. *Learning with data*, e-Chapter 8 (“Support vector machines”) AMLBook.

Gretton, Arthur et al. 2012. ”A Kernel Two-Sample Test.” Journal of Machine Learning Research. Vol 13, p. 723-773.

Guestrin, Carlos. 2007. “Support vector machines.” Lecture slides for “Machine Learning – 10701/15781“ at Carnegie Mellon University.

Hsu, Chih-Wei et al. 2016. ”A Practical Guide to Support Vector Classification.” Department of Computer Science, National Taiwan University.

Lin, Yi. 2002. “Support vector machines and the Bayes rule in classification.” *Data Mining and Knowledge Discovery* (6): 259–275.

Murphy, Kevin. 2012. *Machine learning: a probabilistic perspective.* MIT Press: Cambridge, MA.

Theodoridis, Sergios and Konstantinos Koutroumbas. 2009. *Pattern recognition.* Academic Press: Burlington, MA.

Theodoridis, Sergios. 2015. *Machine learning: a Bayesian and optimization perspective.* Academic Press: London, United Kingdom.

Zhu, Ji and Trevor Hastie. 2004. “Kernel logistic regression and the import vector machine.” *Journal of Computational and Graphical Statistics.* Volume 14, 2005 - Issue 1.

8 References cited in literature review

Aizerman, M. A., Emmanuel M. Braverman and Rozoner, L. I. 1964. ”Theoretical foundations of the potential function method in pattern recognition learning”. *Automation and Remote Control.* 25: 821–837.

Boser, Bernhard E., Isabelle M. Guyon and Vladimir N. Vapnik. 1992. ”A training algorithm for optimal margin classifiers”. *Proceedings of the fifth annual workshop on Computational learning theory – COLT ’92.* p. 144.

Cox, DR. 1958. ”The regression analysis of binary sequences (with discussion)”. *J Roy Stat Soc B.* 20: 215–242.

Tychonoff, A.N. and V.Y. Arsenin. *Solution of ill-posed problems.* Winston & Sons: Washington, 1977.