Report

Eric Alan Wayman

Due May 4, 5 PM

1 Introduction

1.1 Motivation

1.2 Literature review

1.3 Problem statement

2 Approach

ℓ_1 regularization (sparsity) vs. ℓ_2 regularization: ℓ_1 regularization will work better when a small subset of the features are significant. l2 regularization will work better when many of the features are significant.

linear vs rbf kernel: the linear kernel will work fine when the data is already linearly separable. the rbf kernel will help when the data is not linearly separable.

svm rbf and svm linear will perform similarly if there is no need to transform data to higher dimensional space. svm rbf will outperform svm linear if data is not linearly separable in original space.

SVM rbf vs RBF logistic: SVM will outperform logistic when generalization is important (large margin/sparsity of data points that comes from support vectors is not there for RBF logistic so it gives poor results on leukimia (poor generalization)).

2.1 Logistic regression with ℓ_1 and ℓ_2 regularization

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

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

(Murphy p. 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 $p(y = -1) = 1/(1 + \exp(\boldsymbol{w}^T\boldsymbol{x}))$, so the

negative log-likelihood, which equals negative one times the error of the model and which is therefore to be maximized, is

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

(Murphy p. 245). There is not a closed-form solution for the MLE of $\boldsymbol{w}$, so it must estimated by an optimization algorithm.

L2 regularization is achieved by adding the term $\frac{\lambda}{2}\|w\|_2^{\ 2}$ to $NLL(\boldsymbol{w})$ above, giving

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

L1 regularization is achieved by adding the term $\lambda\|\boldsymbol{w}\|_1^{\ 2}$ where $\|\boldsymbol{w}\|_1 = \sum_{i=1}^{c} |\boldsymbol{w}_i|$ (Theodoridis p. 404), so

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

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, p.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 p. 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.

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, p. 406-407)

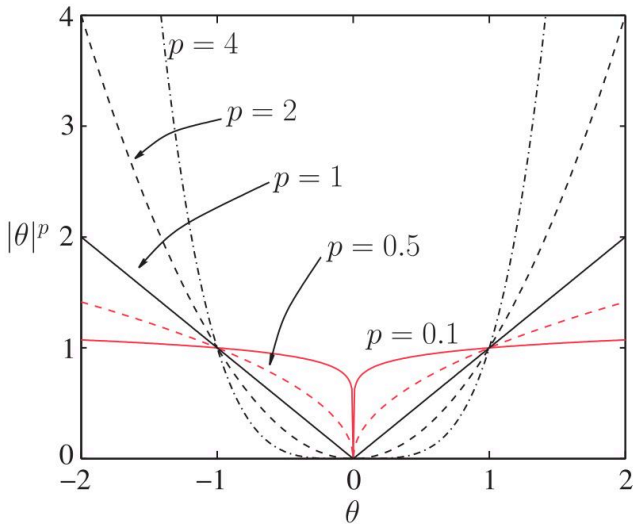

Figure 1: A simple caption

2.2 Kernels: linear vs RBF

Kernels were discussed in lecture. A Mercer kernel is a kernel whose Gram matrix

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

is positive semi-definite for any set of inputs $\{\boldsymbol{x}_i\}_{i=1}^N$ (Murphy p. 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, p. 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. (p. 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. This will be further explained where it appears later.

2.3 SVMs

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 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.

	loss	loss + ℓ_1	loss + ℓ_2	few data pts
linear		log reg	log reg	SVM
RBF	log reg			SVM

loss: 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 model attempts to utilize small subset of of training data in parameter estimation

2.4 Kernelized logistic regression using RBF kernel

2.5 SVM with linear and RBF kernels

2.6 SVM and RBF kernel relationship explanation

kernel trick

sparsity

large margin

2.6.1 Logistic regression

MLE:

Therefore

$$p(y=1) = \frac{1}{1+\exp\left(-\boldsymbol{w}^T\boldsymbol{x}\right)}$$

and likewise $p(y=-1) = \frac{1}{1+\exp(\boldsymbol{w}^T\boldsymbol{x})}$ (Murphy p. 246). If $\boldsymbol{x}$ is replaced with $\boldsymbol{\phi}(\boldsymbol{x})$, this becomes

$$p(y=1) = \frac{1}{1+\exp\left(-\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\right)}$$

Switching for a moment to SVMs, SVMs have 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 p. 499).

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, p.2).

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, p. 2). 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

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

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).

3 Datasets

4 Experimental analysis

5 Findings

6 Summary and future work

7 References

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

Guestrin, Carlos. 2007. "Support vector machines." Lecture slides for "Machine Learning – 10701/15781" at Carnegie Mellon 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. 2015. *Machine learning: a Bayesian and optimization perspective*. Elsevier: London.

Zhu, Ji and Trevor Hastie. 2004. "Kernel logistic regression and the import vector machine." Preprint.