KERNEL METHODS SUPPORT VECTOR MACHINES (SVM), RIDGE REGRESSION & LOGISTIC REGRESSION

Ezukwoke K.I
Unversity Jean Monnet, Saint-Etienne.
{*ifeanyi.ezukwoke*}*@etu.univ-st-etienne.fr*

Abstract

Kernel methods for better classification result. The theory behind this intelligent method makes it the choice model for both linearly and non-linearly separable data.

1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is an unsupervised dimension reduction algorithm for scaling high dimensional data to low dimensional data. The idea was initially proposed by Karl Pearson [1] as a statistical method to find lines or planes of best fit in the context of regression. It hase been subsequently cited and reproduced in various context and expecially in machine learning for **denoising**.

This implcitly means it is used for feature extraction since some of the features in the original space may not be required when using the eigenvectors corresponding to the largest eigenvalue.

1.1 Classical PCA

The Classical PCA algorithm is to find a linear subspace of lower dimensionality than the original space. The PCA algorithm is described below in stepwise order. Note that $\sum$ is used to denote the **covariance**, which is different from summation $\sum_{i=i}^{N}$.

Algorithm 1: PCA Algorithm

```
   Input  : $x \in X$ where $x \in \mathbb{R}^D$
   Output: $\hat{x} \in \mathbb{R}^k$ where $k << D$
1  begin
2      $\hat{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$;
3      $dx_i = x_i - \hat{x}$;
4      $\sum = \frac{1}{N} dx_i dx_i^T$;
5      $\sum \mathbf{u}_k = \lambda_k \mathbf{u}_k$;
6      $\mathbf{u}_k = \arg\mathrm{sort}(\mathbf{u}_k)$;
7      $\hat{x} = \mathbf{u}_k \cdot dx_i$;
8  end
```

PCA is limited in use to finding a linear subspace, by taking advantage of the linear dependence between the feature vectors. This limitation makes the classical PCA unsuitable for non-linearly seperable data (*makemoondataset*).

1.2 Kernel PCA

The Kernel PCA is seeks to address the limitation of the classical PCA algorithm. By transforming the feature space x_i into a higher dimension space $\phi(x_i)$, the kernelized version is able to capture efficiently a subspace that reduces the dimension of the original feature space.

Algorithm 2: Kernel PCA Algorithm

```
   Input  : $\phi(x) \in \kappa(x_, x_j)$ where
            $\phi(x) \in \mathbb{R}^D$. Let
            $\kappa = \kappa(x_i, x_j)$
   Output: $\hat{\phi(x)} \in \mathbb{R}^k$ where $k << D$
1  begin
2      Select a kernel $\kappa$;
3      Construct Normalized kernel
         $\hat{\kappa} = \kappa - 2\kappa 1_{1/N} + 1_{1/N}\kappa 1_{1/N}$;
4      Solve eigen problem $\kappa\alpha_{\mathbf{i}} = \lambda\alpha_{\mathbf{i}}$;
5      Project data in new space
         $y_i = \sum_{i=1}^{N} \alpha\kappa$;
6  end
```

The kernel PCA algorithm is expressed only in terms of dot products, this trick allows us to construct different nonlinear versions of the classical PCA algorithm.

2 Kmeans

K-Means clustering is a fast, robust, and simple algorithm that gives reliable results when data sets are distinct or well separated from each other in a linear fashion. It is best used when the number of cluster centers, is specified due to a well-defined list of types shown in the data. However, it is important to keep in mind that K-Means clustering may not perform well if it contains heavily overlapping data, if the Euclidean distance does not measure the underlying factors well, or if the data is noisy or full of outliers

3 Support Vector Machine (SVM)

SVM is commonly referred to as the maximum margin classifier.

4 Kernel Ridge regression

We recall first the classical optimization formulation for ridge regression as

$$\min_{\beta} F(\beta, b) = \lambda||\beta||^2 + \sum_{i=1}^{m} (\beta^T x_i - y_i)^2 \quad (1)$$

where $\sum_{i=1}^{m} (\beta^T x_i - y_i)^2$ is the squared loss function between the actual and predicted values we try to minimize and $\lambda||\beta||^2$ is the regularization term.

The solution of this optimization problem is given as

$$\beta = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{Y} \quad (2)$$

Where β is the weight parameters that helps us predict we incoming data. We also recall that the solution of the ridge regression reverts to that of ordinary least square $\beta = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$ for very small value of λ (especially when λ is close to zero.)

Consequently, the solution of ridge regression only solves problems in a low dimensional data space. In high dimensional space when $N \to \infty$ we would need to map x into a higher feature space where learning the covariant with respect to the objective function y is much easier. Ridge regression solves this problem by projecting $x_i \to \phi(x_i)$.

The new solution of ridge in the kernel space is now given by $\beta = (\phi^T(x_i)\phi(x_i) + \lambda I)^{-1}\phi(x)y$ where $\phi^T(x_i)\phi(x_i) = \kappa(x_i, x_j)$ and $\alpha = (\phi^T(x_i)\phi(x_i) + \lambda I)^{-1}$.

To prove this ,we formalize ridge regression as an optimization problem [3] in the $\phi(x_i)$ space as

$$\min_{\beta,\xi} L_p = \sum_{i=1}^{N} \xi_i^2 \quad (3)$$

$$\text{subject to } y_i - \beta^T \phi(x_i) = \xi \ \forall i \quad (4)$$

$$||\beta||^2 \leq B^2 \quad (5)$$

The Langrangian formulation is therefore follows as

$$L_p = \sum_{i=1}^{N} \xi_i^2 + \sum_{i=1}^{N} \eta_i [y_i - \beta^T \phi(x_i) - \xi_i] + \lambda(||\beta||_2^2 - B^2)$$

Note that η and λ are teh lagrangian multipliers. By taking the first order derivative of L_p we have that,

$$\nabla_\xi L_p = 2\xi_i - \eta_i = 0$$
$$\eta_i = 2\xi_i$$
$$\xi_i = \frac{\eta_i}{2}$$

$$\nabla_\beta L_p = -\eta\phi(x_i) + 2\lambda\beta = 0$$
$$2\lambda\beta = \eta\phi(x_i),$$

From **representer theorem**, we know that $\beta = \alpha\phi(x_i)$. By substitution in the last equation we have that $\eta = 2\lambda\alpha$. If we substitute back the result of the derivatives into the lagrangian formulation we have that

$$\begin{aligned} L_p &= \sum_{i=1}^{N} \frac{\eta_i^2}{4} + \sum_{i=1}^{N} 2\lambda\alpha[y - \alpha\kappa(x_i, x_j) - \frac{\eta_i}{2}] \\ &\quad +\lambda\alpha^T\kappa(x_i, x_j)\alpha \\ &= \sum_{i=1}^{N} \lambda^2\alpha^2 + 2\lambda\alpha[y - \alpha\kappa(x_i, x_j) - \lambda\alpha] \\ &\quad +\lambda\alpha^T\kappa(x_i, x_j)\alpha \end{aligned}$$

By omitting two steps of expansion we can reduce the above equation to

$$\nabla_\alpha L_p = -2\lambda^2\alpha - 2\lambda\alpha\kappa(x_i, x_j) + 2\lambda y = 0 \tag{6}$$

$$\alpha = (\kappa(x_i, x_j) + \lambda\mathbf{I})^{-1}y \tag{7}$$

and the *beta* parameter can be estimated as

$$\beta = \alpha\phi(x) = (\kappa(x_i, x_j) + \lambda\mathbf{I})^{-1}\phi(x)y \tag{8}$$

We predict new sample using the function

$$\begin{aligned} y = \beta^T\phi(x) &= (\kappa(x_i, x_j) + \lambda\mathbf{I})^{-1} \\ &\quad \kappa(x_i, x_j)y \end{aligned}$$

Note that this is the closed form solution of kernel ridge regression, as it can also be solved using optimization algorithm like gradient descent or stochastic gradient descent.

5 Kernel Logistic Regression

Logistic regression is a binary classification algorithm. The algorithm is capable of classifying linearly separable dataset. The linear version of logistic regression is however not able to accurately classify non-linear data, hence its kernel version.

Using the Iterative Reweighted Least Square method (IRLS) proposed by Zhu et al.[2]. an extension of the Support vector machine was introduced to logistic regression (Import vector machines). This method uses much more smaller data points as support vectors than SVMs and hence, outperforms SVMs for classification purposes.

Given a set of training data $\{(x, y)\}^n$ where $x_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}^d$ where $i = 1, ..., N$. The output is a binary output controlled by the sigmoid function $\frac{1}{1+e^{-\mathbf{x}\cdot\beta}}$.

Classical logistic regression is define using the binomial distribution as

$$l(\beta_0, \beta) = \Pi_{i=1}^{n} p(x_i)^y(1 - p(x_i)^{(1-y_i)}) \tag{9}$$

and the log-likelihood follows as

$$\begin{aligned} L(\beta_0, \beta) = \sum_{i=1}^{n} y_i log p(x_i) + (1 - y_i) \\ log(1 - p(x_i)) \end{aligned}$$

where

$$p(x; \beta) = \frac{1}{1 + e^{-x\cdot\beta}} \tag{10}$$

and

$$1 - p(x; \beta) = \frac{1}{1 + e^{x\cdot\beta}} \tag{11}$$

However, classical logistic regression will fail to classify accurately non-linearly separable data, hence its kernel version.

The vector space can be expressed as a linear combination of the input vectors such that

$$\beta = \sum_{i=1}^{N} \alpha_i\phi(\mathbf{x_i}) \tag{12}$$

where $\alpha \in \mathbb{R}^{nx1}$ is the dual variable. The function $\phi(x_i)$ maps the data points from lower dimension to higher dimension.

$$\phi : \mathbf{x} \in \mathbb{R}^{\mathbb{D}} \to \phi(x) \in \mathbb{F} \subset \mathbb{R}^{\mathbb{D}'} \tag{13}$$

Let $\kappa(x_i, x)$ be a kernel function resulting from the inner product of $\phi(x_i)$ and $\phi(\mathbf{x_j})$, such that

$$\kappa(x_i, x) = \langle \phi(x_i)\phi(x_j) \rangle \tag{14}$$

From **representer theorem** we know that

$$\begin{aligned} F = \beta^T \phi(x) &= \alpha \left\langle \phi(\mathbf{x_i})\phi(\mathbf{x_j}) \right\rangle \\ &= \alpha\kappa(x_i, x_j) \end{aligned}$$

We can now express $p(x;\beta)$ is subspace of input vectors only such that

$$p(\phi;\alpha) = \frac{1}{1 + e^{-\alpha_i \kappa(x_i, x_j)}} \tag{15}$$

and

$$1 - p(\phi;\alpha) = \frac{1}{1 + e^{\alpha_i \kappa(x_i, x_j)}} \tag{16}$$

The logit function is mapped into the kernel space as

$$logit(\frac{p(\phi;\alpha)}{1 - p(\phi;\alpha)}) = \alpha\kappa(x_i, x) \tag{17}$$

Deriving the equation of kernel logistic regression requires the regularized logistic regression, precisely the regularized $l2-norm$ of the log-likelihood. This is in comparison to the SVM objective function.

$$\begin{aligned} L_\alpha = \sum_{i=1}^{n} y_i logp(x_i) + (1 - y_i)log(1 - p(x_i)) \\ -\frac{\lambda}{2}\alpha^{\mathbf{T}}\kappa(\mathbf{x_i}, \mathbf{x})\alpha \end{aligned}$$

5.1 Learning kernel logistic regression

As mentioned earlier, some of the methods for finding the maximum likelihood estimate include gradient descent (**GD**), iterative re-weighted least sqaures (**IRLS**) method. Here we employ the use of IRLS which is based on the Newton-Ralphson algorithm.

- Optimization function:

$$\begin{aligned} L_\alpha = \sum_{i=1}^{n} y_i logp(x_i) + (1 - y_i)log(1- \\ p(x_i)) - \frac{\lambda}{2}\alpha^{\mathbf{T}}\kappa(\mathbf{x_i}, \mathbf{x})\alpha \end{aligned}$$

We can expand the objective function as follows

$$\begin{aligned} L_\alpha = ylog\left(\frac{p}{1-p}\right) + log(1 - p(x_i)) \\ -\frac{\lambda}{2}\alpha^{\mathbf{T}}\kappa(\mathbf{x_i}, \mathbf{x})\alpha \\ = ylog\left(\frac{p}{1-p}\right) + log\left(\frac{1}{1 + e^{\alpha\kappa(x_i, x)}}\right) \\ -\frac{\lambda}{2}\alpha^{\mathbf{T}}\kappa(\mathbf{x_i}, \mathbf{x})\alpha \\ = y\alpha\kappa(x_i, x) - log(1 + e^{\alpha\kappa(x_i, x)}) \\ -\frac{\lambda}{2}\alpha^{\mathbf{T}}\kappa(\mathbf{x_i}, \mathbf{x})\alpha \end{aligned}$$

First order derivative of the log-likelihood

$$\begin{aligned} \nabla_\alpha L = y\kappa(x_i, x) - \frac{\kappa(x_i, x)e^{\alpha\kappa(x_i, x)}}{1 + e^{\alpha\kappa(x_i, x)}} \\ -\lambda\alpha\kappa(x_i, x) \\ = y\kappa(x_i, x) - p\kappa(x_i, x) - \lambda\alpha\kappa(x_i, x) \\ \nabla_\alpha L = \kappa(x_i, x)(y - p) - \lambda\alpha\kappa(x_i, x) \end{aligned}$$

Now we deduce the second derivative from the result of the first. We know from the first that

$$\begin{aligned} \nabla_\alpha L = \kappa(x_i, x)\left(y - \frac{1}{1 + e^{\alpha\kappa(x_i, x)}}\right) \\ -\lambda\alpha\kappa(x_i, x) \end{aligned}$$

$$\begin{aligned} \nabla_\alpha^2 L = -\kappa(x_i, x)^T \left(\frac{e^{\alpha\kappa(x_i, x)}}{(1 + e^{\alpha\kappa(x_i, x)})^2}\right) \\ \times\kappa(x_i, x) - \lambda\kappa(x_i, x) \\ = -\kappa(x_i, x)^T \left(\frac{e^{\alpha\kappa(x_i, x)}}{1 + e^{\alpha\kappa(x_i, x)}} \cdot \frac{1}{1 + e^{\alpha\kappa(x_i, x)}}\right) \\ \times\kappa(x_i, x) - \lambda\kappa(x_i, x) \end{aligned}$$

resulting

$$\nabla_\alpha^2 L = -\kappa(x_i, x)^T (p(1-p))\kappa(x_i, x) - \lambda\kappa(x_i, x)$$

The update for α is

$$\alpha_{j+1} = \alpha_j - (\nabla_\alpha^2 L)^{-1}\nabla_\alpha L \qquad (18)$$

$$\alpha_{j+1} = \alpha_j + (\kappa(x_i,x)^T W\kappa(x_i,x) + \lambda\kappa(x_i,x))^{-1}(\kappa(x_i,x)^T(y-p) - \lambda\kappa(x_i,x)\alpha)$$

Where W is the diagonal matrix corresponding to $p(1-p)$. For simplification let $\alpha_j = (\kappa(x_i,x)^T W\kappa(x_i,x) + \lambda\kappa(x_i,x))^{-1}(\kappa(x_i,x)^T W\kappa(x_i,x) + \lambda\kappa(x_i,x))\alpha_j$ and $\kappa(x_i,x) = K$.

so that

$$\begin{aligned}\alpha_{j+1} &= (K^T WK + \lambda K)^{-1}(K^T WK + \lambda K)\alpha_j + (K^T WK + \lambda K)^{-1}(K^T(y-p) - \lambda K\alpha)\\ &= (K^T WK + \lambda K)^{-1}((K^T WK + \lambda K)\alpha_j + K^T(y-p) - \lambda K\alpha)\end{aligned}$$

After expanding the above term we have that

$$\begin{aligned}\alpha_{j+1} &= (K^T WK + \lambda K)^{-1}(K^T WK\alpha_j + K^T(y-p))\\ &= (K^T WK + \lambda K)^{-1}K^T W(K\alpha_j + W^{-1}(y-p))\end{aligned}$$

if $z_j = (K\alpha_j + W^{-1}(y-p))$ we can summarize the solution of α_{j+1} with a shorthand equation thus.

$$\alpha_{j+1} = (K^T WK + \lambda K)^{-1}K^T Wz_j \qquad (19)$$

z_j is the adjusted response.

- Prediction
  Still using the representer theorem, we compute the posterior probability of a new data point such that

$$y = sign\left(\frac{1}{1+\exp^{-\alpha\kappa(x_i,x)}}\right) \qquad (20)$$

Here, the prediction is dependent only on α and the inner product of the training and test data.

Algorithm 3: KLR using **IRLS**

```
   Input  : $\kappa, y, \alpha_j$
   Output: $\alpha_{j+1}$
1  begin
2      $c = 0$;
3      while $\left|\frac{DEV^c - DEV^{c+1}}{DEV^{c+1}}\right| \geq \epsilon_1$ and
         $c \leq Max\ IRLS\ Iterations$ do
4          for $i \leftarrow 1$ to $N$ do
5              $\hat{p} = \frac{1}{1+\exp^{-\alpha\kappa(x_i,x)}}$;
6              $v_i = \hat{p}(1-\hat{p})$;
7              $z_i = K\alpha^c + \frac{y_i - \hat{p}}{\hat{p}(1-\hat{p})}$;
8          end
9          $\mathbf{V} = diag(v_1, ..., v_N)$;
10         $\hat{\alpha}^{c+1} =$
             $(K^T WK + \lambda K)^{-1}K^T Wz^c$;
11         $c = c + 1$
12     end
13 end
```

References

[1] Pearson K. On lineas and Planes of closest fit to systems of points in space. *Philos Mag A*, 6:559–572, 1901.

[2] Zhu J, Hastie T. Kernel logistic regression and import vector machine. *J Comput Graphic Stat*, 14:185–205, 2005.

[3] Saunders C., Gammerman A., Vovk V., Ridge Regression Learning Algorithm in Dual Variables. *ICML 15th Internationl Conforence on Machine Learning.*, 1998.