KERNEL METHODS

Chih-Wei Chang   Chung-Yen Hung

January 29, 2015

Department of Mathematics, National Taiwan University

INTRODUCTION

Motivation

Linear Model

1. Simple and Well Studied
2. Computationally Feasible

Real-world Problems

1. Non Linear
2. Sophisticated Interdependencies

Question Can we apply **Linear Model** on **Real-world Problems**?

Motivation Cont.

How about **Feature Transform**?

Mapping features to higher dimensional space. For example, the quadratic transformation from $\mathbb{R}^2$ to $\mathbb{R}^6$

$$\Phi : (x_1, x_2) \mapsto (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1).$$

Line in $\mathbb{R}^6$ corresponds to quadratic curve in $\mathbb{R}^2$. Common techniques in the machine learning world.

Learning Stage Fit linear model $h : \mathbb{R}^6 \to \{0, 1\}$ on $\Phi(\mathrm{x})$.

How about **Feature Transform**?

Mapping features to higher dimensional space. For example, the quadratic transformation from $\mathbb{R}^2$ to $\mathbb{R}^6$

$$\Phi : (x_1, x_2) \mapsto (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1).$$

Line in $\mathbb{R}^6$ corresponds to quadratic curve in $\mathbb{R}^2$. Common techniques in the machine learning world.

Learning Stage Fit linear model $h : \mathbb{R}^6 \to \{0, 1\}$ on $\Phi(\mathrm{x})$.

Testing Stage $h(\Phi(\mathrm{x}_{test}))$ gives the prediction.

Motivation Cont.

However …

Pros Fitting complicated underlying rules – the higher the dimension, the more powerful the model.

Motivation Cont.

However ...

Pros Fitting complicated underlying rules – the higher the dimension, the more powerful the model.

Cons Extra computational costs, e.g., inner products – the higher the dimension, the harder the computation.

Kernel Trick

Luckily, **Kernel Trick** comes to rescue ...

1. Mathematical shortcut for preventing computing inner product directly.

Kernel Trick

Luckily, **Kernel Trick** comes to rescue ...

1. Mathematical shortcut for preventing computing inner product directly.
2. Combine the favors of Linearity and Non-Linearity.

Kernel Trick

Consider the inner product in the previous example

$$\begin{aligned} \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle &= \Phi(\mathbf{x})^T \Phi(\mathbf{y}) \\ &= x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 x_2 y_1 y_2 + 2x_1 y_1 + 2x_2 y_2 + 1 \\ &= (\mathbf{x}^T \mathbf{y} + 1)^2 = (\langle \mathbf{x}, \mathbf{y} \rangle + 1)^2 \end{aligned}$$

It implies we can define a special function

$$K(\mathbf{x}_1, \mathbf{x}_2) = \langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \rangle,$$

which is the so-called kernel function.

COMMON KERNEL

Polynomial Kernel

Suppose we use the polynomial feature transformation $\Phi_d(x)$, which collects all the possible coefficients in the d-degree polynomial.

We then have the polynomial kernel

$$K_{\Phi_d}(x, y) = (\langle x, y \rangle + \alpha)^d$$

Quadratic Transformation as An Special Case

Let $d = 2$, the transformation

$$\begin{aligned}\Phi_2(x) =&(x_n^2, \ldots, x_1^2, \sqrt{2}x_n x_{n-1}, \ldots, \sqrt{2}x_n x_1, \sqrt{2}x_{n-1}x_{n-2}, \ldots, \\ &\sqrt{2c}x_n, \ldots, \sqrt{2c}x_1, c)\end{aligned}$$

And the kernel function

$$\begin{aligned}K(x, y) &= (\sum_{i=1}^{n} x_i y_i + c)^2 \\ &= \sum_{i=1}^{n} x_i^2 y_i^2 + \sum_{i=2}^{n}\sum_{j=1}^{i-1}(\sqrt{2}x_i x_j)(\sqrt{2}y_i y_j) + \sum_{i=1}^{n}(\sqrt{2c}x_i)(\sqrt{2c}y_i) + c^2\end{aligned}$$

Gaussian Kernel

The Gaussian Kernel is defined as

$$K(x, y) = \exp(-\gamma \|x - y\|^2), \text{with } \gamma > 0$$

Gaussian Kernel Cont.

As An Extreme Case of Polynomial Kernel

For $\gamma = 1$

$$
\begin{aligned}
K(x, y) &= \exp(\|x - y\|^2) \\
&= \exp(-x^2)\exp(-y^2)\exp(2xy) \\
&= \exp(-x^2)\exp(-y^2)\left(\sum_{i=0}^{\infty}\frac{(2xy)^i}{i!}\right) \\
&= \sum_{i=0}^{\infty}\left(\exp(-x^2\sqrt{\frac{2^i}{i!}})x^i\right)\left(\exp(-y^2\sqrt{\frac{2^i}{i!}})y^i\right) \\
&= \Phi(x)^T\Phi(y)
\end{aligned}
\tag{1}
$$

It implies $\Phi(x) = \exp(-(x)^2)(1, \sqrt{\frac{2}{1!}}x, \sqrt{\frac{2^2}{2!}}x^2, \ldots)$.

KERNEL MACHINES

Kernel SVM

Soft Margin SVM dual

$$\max_{\forall\alpha\geq 0,\forall\beta\geq 0}\left(\min_{b,w}\frac{1}{2}w^Tw+C\sum_{n=1}^{N}\xi_n+\sum_{n=1}^{N}\alpha_n(1-\xi_n-y_n(w^Tz_n+b))+\sum_{n=1}^{N}\beta_n(-\xi)\right)$$

by KKT condition

$$\max_{\forall 0\leq\alpha\leq C,\beta_n=C-\alpha_n}-\frac{1}{2}\sum_{n=1}^{N}||\alpha_n y_n x_n||^2+\sum_{n=1}^{N}\alpha_n$$

Kernel SVM Cont.

Soft Margin SVM dual

Unfold formula and multiply -1.

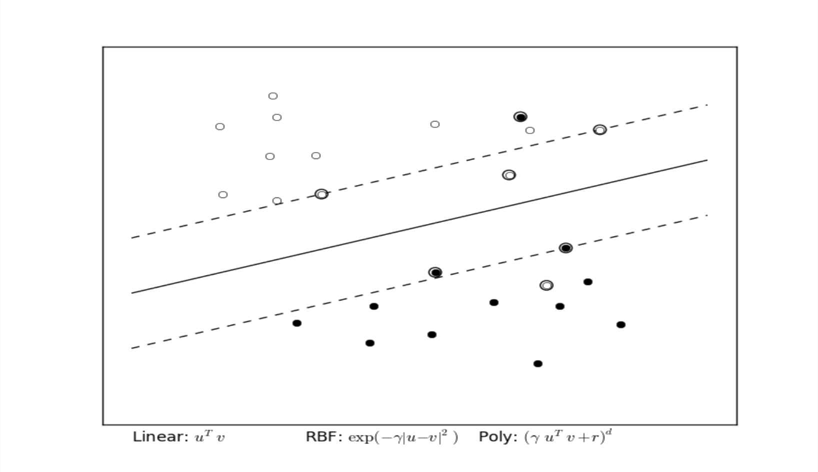
$$\min_{\alpha} \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m {x_n}^T x_m - \sum_{n=1}^{N} \alpha_n$$

$$\text{subject to } \sum_{n=1}^{N} y_n \alpha_n = 0; \alpha_n \geq 0, \forall n \in [0, N], n \in \mathbb{N}$$

$$\text{implicitly } w = \sum_{n=1}^{N} \alpha_n y_n x_n; \beta_n = C - \alpha_n, \forall n \in [0, N], n \in \mathbb{N}$$

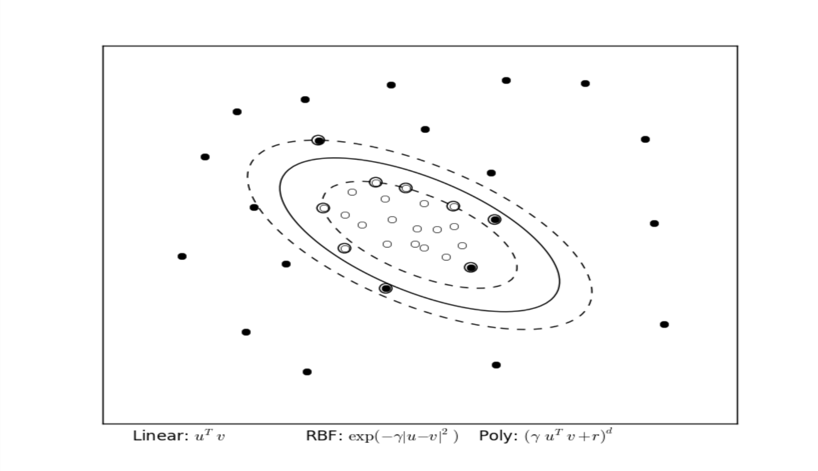
We can do kernel method on red side ${x_n}^T x_m$

Example: Linear SVM

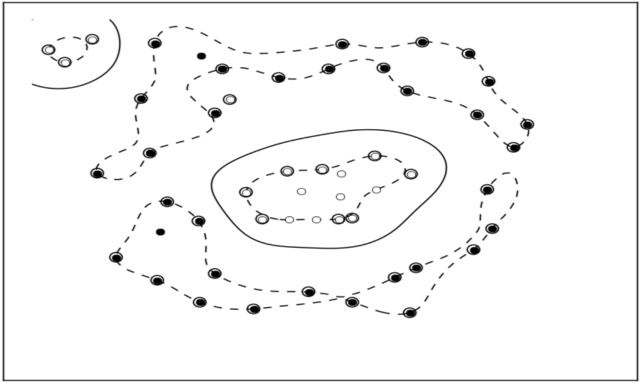


Kernel SVM Cont.

Example: Degree 2 Poly Kernel SVM



Example: Gaussian Kernel SVM



Linear: $u^T v$ RBF: $\exp(-\gamma|u-v|^2)$ Poly: $(\gamma\, u^T v + r)^d$

Kernel Ridge Regression

Ridge Regression

Minimize the quadratic cost as well as the regularization term

$$C(\mathbf{w}) = \frac{1}{2}\sum_i (y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \frac{1}{2}\lambda\|\mathbf{w}\|^2$$

We have analytic solution

$$\mathbf{w} = (\lambda\mathbf{I} + \sum_i \mathbf{x}_i\mathbf{x}_i^T)^{-1}(\sum_j y_j\mathbf{x}_j)$$

Plug the Kernel Function In

Note that

$$(P^{-1} + B^T R^{-1} B)^{-1} B^T R^{-1} = PB^T (BPB^T + R)^{-1}$$

Therefore,

$$\begin{aligned}
\mathbf{w} &= (\lambda \mathbf{I} + \sum_i \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^T)^{-1} (\sum_j y_j \phi(\mathbf{x}_j)) \\
&= (\lambda \mathbf{I} + \Phi\Phi^T)^{-1} \Phi \mathbf{y} \\
&= \Phi(\Phi^T \Phi + \lambda I)^{-1} \mathbf{y}
\end{aligned}$$

Kernel Ridge Regression Cont.

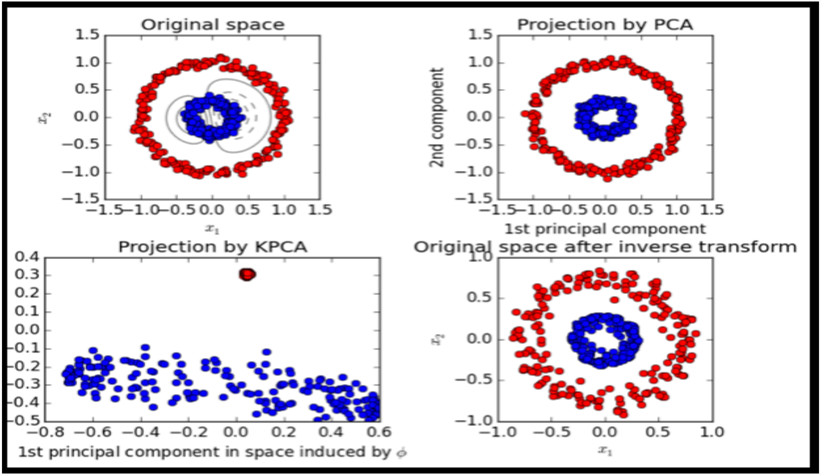
Prediction

In the prediction stage, we first transform $\phi(\mathbf{x}_{test})$

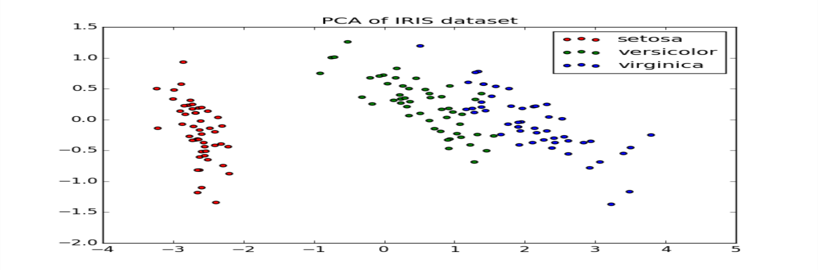
$$y = \mathbf{w}^T\phi(\mathbf{x}_{test}) = \mathbf{y}^T(\Phi^T\Phi + \lambda I)^{-T}\Phi^T\phi(\mathbf{x}_{test})$$

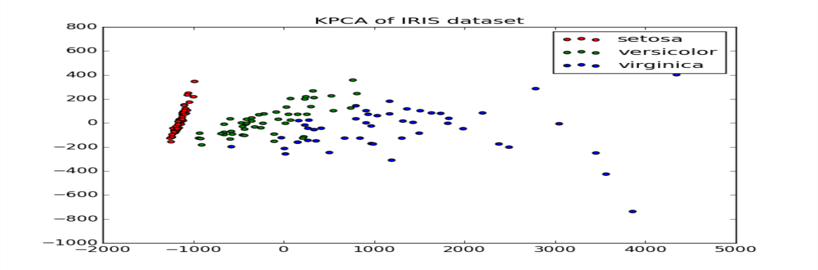
All the calculation can be done in terms of inner product, and hence we can apply the kernel trick.

Example: Kernel PCA v.s. PCA

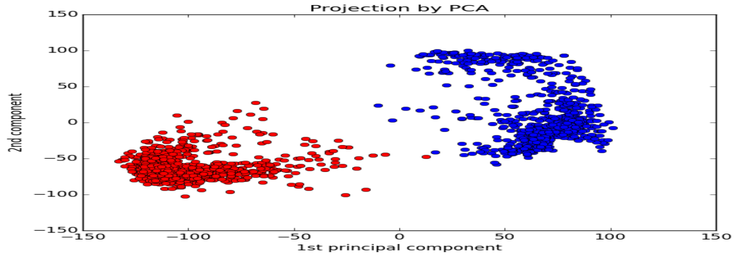


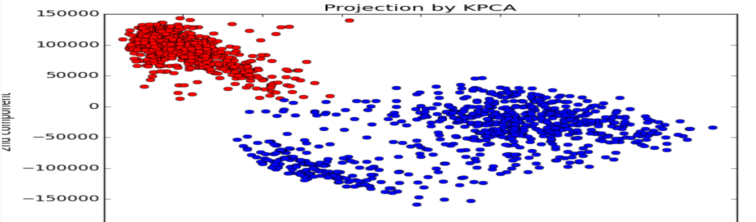
Example: Kernel PCA v.s. PCA of IRIS dataset





The first two class





CONCLUSION

Summary

- Linear to Non-linear
- Computationally Feasible
- Ability to Fit Arbitrarily Complicated Model

Questions?