统计机器学习

（小班研讨）

主讲教师：刘峤

第4章 支持向量机与核方法

Support Vector Machines and Kernel Methods

Kernelize Logistic Regression

- Kernelize

$$\mathbf{x} \to \phi(\mathbf{x}), \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i)$$

$$K(\mathbf{w}, \mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}_i, \mathbf{x})$$

$$p(y|\mathbf{x}) = \frac{1}{1+\exp(-yK(\mathbf{x},\mathbf{w}))} = \frac{1}{1+\exp\left(-y\sum_{i=1}^{N} \alpha_i K(\mathbf{x}_i,\mathbf{x})\right)}$$

- Dual Problem

$$\mathcal{L}(\mathbf{w}, \alpha) = \sum_{i=1}^{N} \log \frac{1}{1+\exp\left(-y_i \sum_{j=1}^{N} \alpha_j K(\mathbf{x}_j,\mathbf{x}_i)\right)} - \lambda \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$$

Representer Theorem

- For any L2-regularized linear model : $\mathbf{w}^* = \sum_{i=1}^{N} \beta_i \mathbf{z}_i$

- let optimal $\mathbf{w}^* = \mathbf{w_z} + \mathbf{w}_p$ where

$$\mathbf{w_z} \in \text{span}(\mathbf{z}_i) \text{ and } \mathbf{w}_p \perp \text{span}(\mathbf{z}_i) \quad \mathbf{w}_p = 0?$$

- if $\mathbf{w}_p \neq 0$, consider : $\mathbf{w_z}$
  - of same err as $\mathbf{w}^*$: $err(y_i, \mathbf{w}^{*T}\mathbf{z}_i) = err(y_i, (\mathbf{w_z} + \mathbf{w}_p)^T\mathbf{z}_i)$
  - of smaller regularizer as $\mathbf{w}^*$:

$$\mathbf{w}^{*T}\mathbf{w}^* = \mathbf{w}_\mathbf{z}^T\mathbf{w_z} + 2\mathbf{w}_\mathbf{z}^T\mathbf{w}_p + \mathbf{w}_p^T\mathbf{w}_p > \mathbf{w}_\mathbf{z}^T\mathbf{w_z}$$

- if $\mathbf{w}_p \neq 0$, consider : $\mathbf{w_z}$
- if $\mathbf{w}_p \neq 0$, consider : $\mathbf{w_z}$

Simplify ξ_i and β_i

$$\mathcal{L}(b, \mathbf{w}, \xi, \alpha, \beta) = \tfrac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\alpha_i(1 - \xi_i - y_i(\mathbf{w}^T\mathbf{x}_i + b)) + \sum_{i=1}^{N}\beta_i(-\xi_i)$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \Rightarrow \beta_i = C - \alpha_i$$ **implicit constraint**

$$\alpha_i \geq 0, \beta_i \geq 0 \Rightarrow 0 \leq \alpha_i \leq C$$ **explicit constraint**

- **no loss of optimality if solving with implicit & explicit constraint**

$$\max_{0 \leq \alpha_i \leq C, \beta_i = C - \alpha_i} \left(\min_{b, \mathbf{w}} \tfrac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N}\alpha_i(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \right)$$

Note: ξ_i can also be removed in this way :-)

Estimate the Margin

- 对偶问题

$$\max_{\alpha} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j)+\sum_{i=1}^{N}\alpha_i$$

$$\text{s.t.}\quad \sum_{i=1}^{N}\alpha_i y_i=0;\quad 0\le\alpha_i\le C,\ i=1,2,\cdots,N$$

- 求解对偶问题得到：

$$0<\alpha_j<C$$

$$\mathbf{w}^*=\sum_{i=1}^{N}\alpha_i^* y_i\mathbf{x}_i \qquad b^*=y_j-\sum_{i=1}^{N}\alpha_i^* y_i(\mathbf{x}_i\cdot\mathbf{x}_j)$$

- 分类超平面可以表示为：

$$\sum_{i=1}^{N}\alpha_i^* y_i(\mathbf{x}\cdot\mathbf{x}_i)+b^*=0$$

Learning the Maximum Margin Classifier

- Idea 2.0: Minimize

  w.w + C (distance of error points to their correct place)

- What should our quadratic optimization criterion be?

$$\frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{k=1}^{N}\epsilon_k$$

m = # input dimensions

- Our original (noiseless data) QP had ***m+1*** variables:

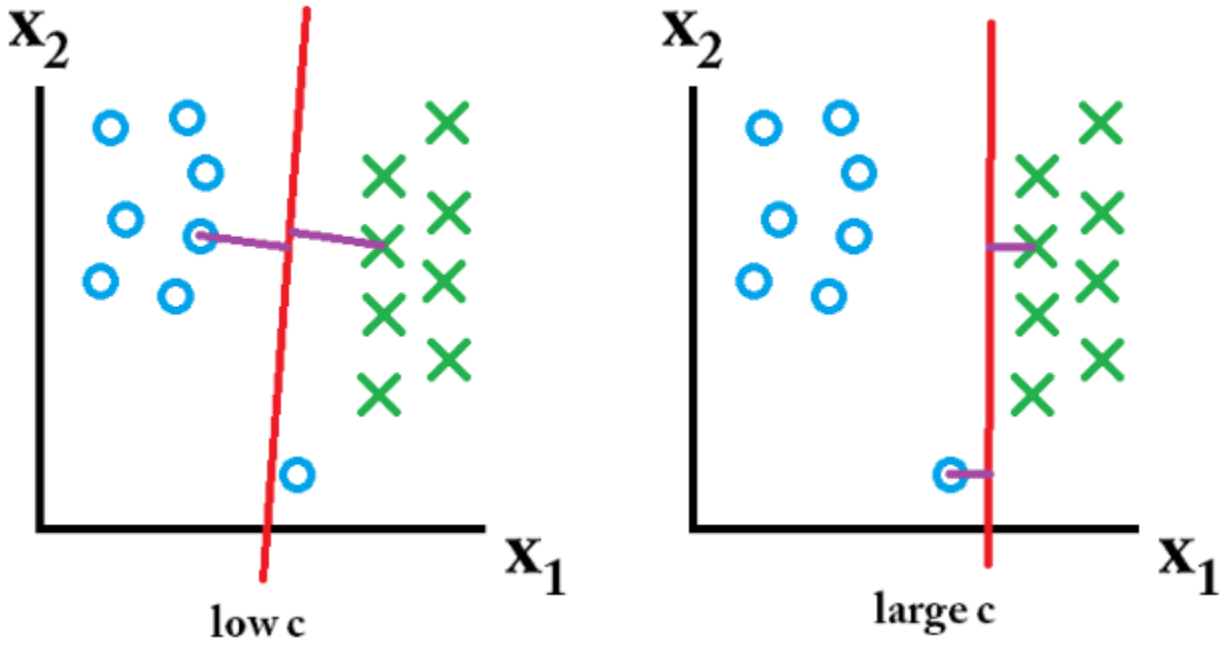
  $w_1, w_2, \ldots w_m$, and b.

- Our new (noisy data) QP has ***m+1+N*** variables:

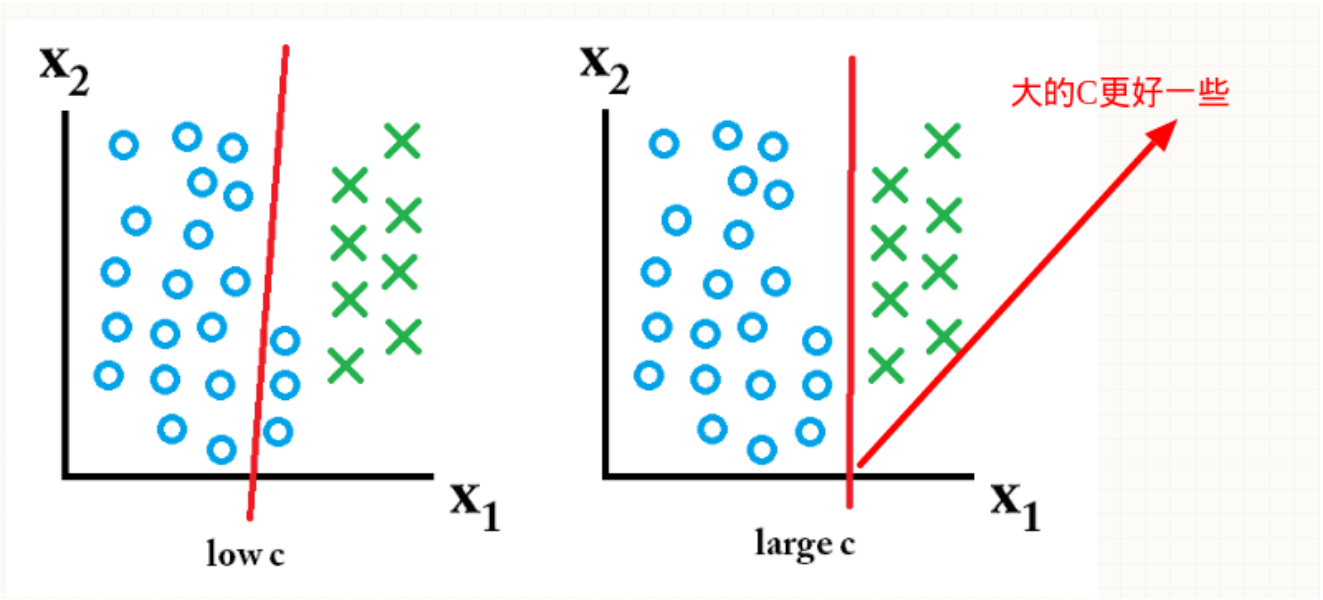
  $w_1, w_2, \ldots w_m, b, \varepsilon_k, \varepsilon_1, \ldots \varepsilon_N$

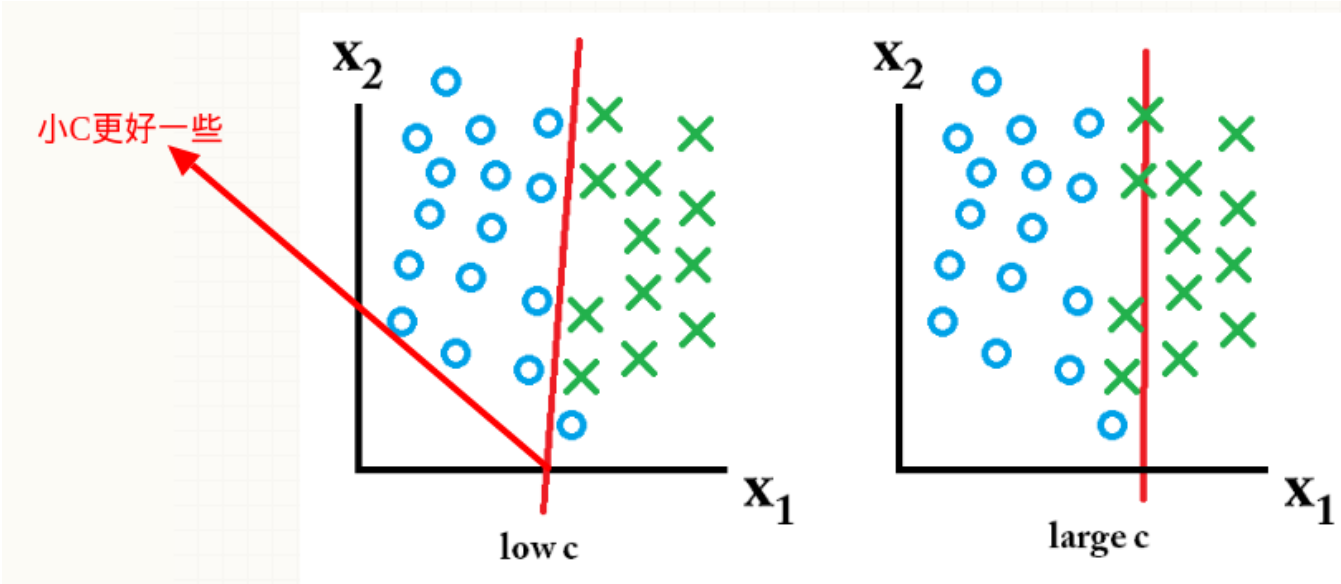
How do we determine the appropriate value for c ?

Impact of the penalize parameter C



Impact of the penalize parameter C



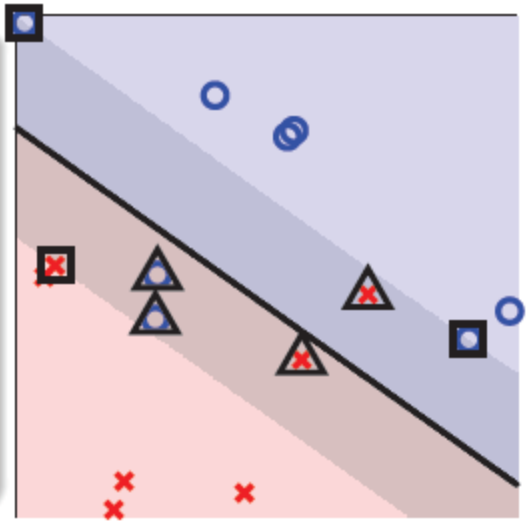


Physical Meaning of α_n

complementary slackness:

$$\alpha_n(1-\xi_n - y_n(\mathbf{w}^T\mathbf{z}_n + b)) = 0$$
$$(C - \alpha_n)\xi_n = 0$$

- non SV ($0 = \alpha_n$): $\xi_n = 0$, 'away from'/on fat boundary
- □ free SV ($0 < \alpha_n < C$): $\xi_n = 0$, on fat boundary, locates b
- △ bounded SV ($\alpha_n = C$): ξ_n = violation amount, 'violate'/on fat boundary



α_n can be used for **data analysis**

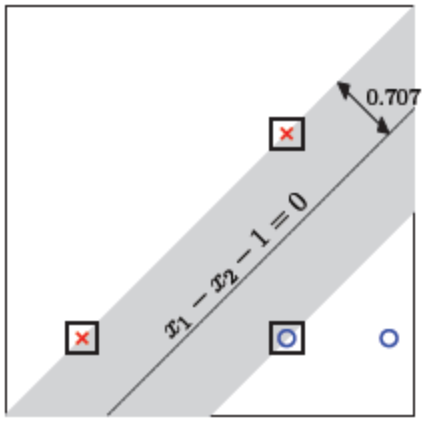
Leave-One-Out CV Error for SVM

recall: $E_{\text{loocv}} = E_{\text{cv}}$ with N folds

claim: $E_{\text{loocv}} \leq \frac{\#\text{SV}}{N}$

- for $(\mathbf{x}_N, y_N)$: if optimal $\alpha_N = 0$ (non-SV)
  $\Longrightarrow$ $(\alpha_1, \alpha_2, \ldots, \alpha_{N-1})$ still optimal when leaving out $(\mathbf{x}_N, y_N)$
  key: **what if there's better** α_n?
- SVM: $g^- = g$ when leaving out non-SV

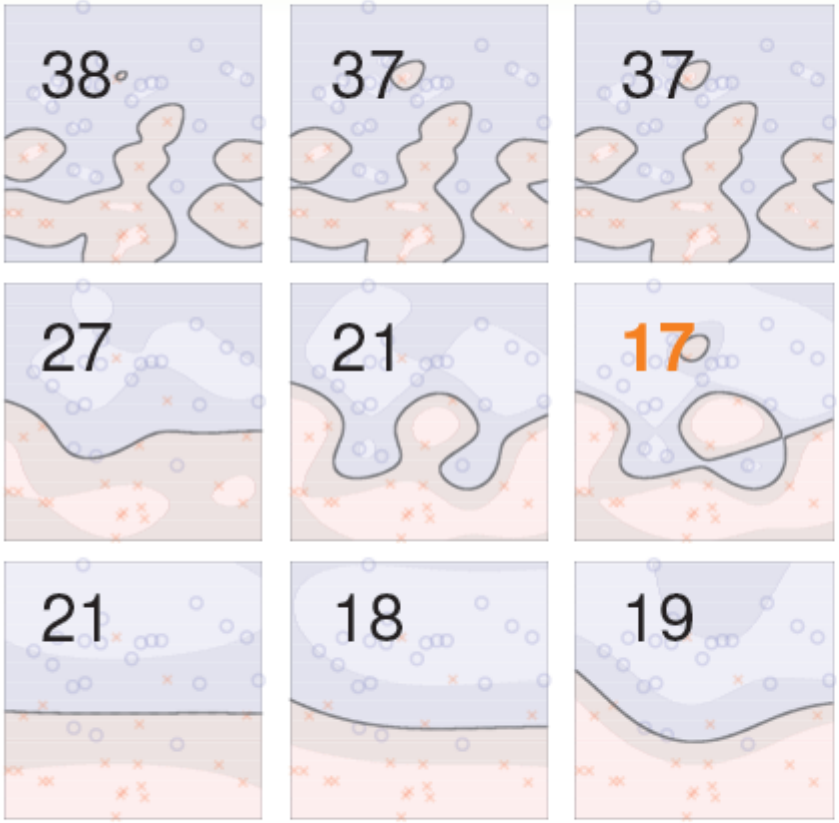
$$
\begin{aligned}
e_{\text{non-SV}} &= \text{err}(g^-, \text{non-SV}) \\
&= \text{err}(g, \text{non-SV}) = 0 \\
e_{\text{SV}} &\leq 1
\end{aligned}
$$



motivation from hard-margin SVM: only **SVs needed**

scaled #SV bounds leave-one-out CV error

Selection by # SV



- nSV(C, γ): 'non-smooth' function of (C, γ) —**difficult to optimize**
- **just an upper bound!**
- dangerous models can be ruled out by **nSV** on **a few grid values of** (C, γ)

nSV: often used as a **safety check** if computing E_{cv} is too time-consuming

- Scaling before applying SVM is very important. Part 2 of Sarle's Neural Networks FAQ Sarle (1997) explains the importance of this and most of considerations also apply to SVM.
    - The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges.
    - Another advantage is to avoid numerical difficulties during the calculation. Because kernel values usually depend on the inner products of feature vectors, e.g. the linear kernel and the polynomial kernel, large attribute values might cause numerical problems.
- We recommend linearly scaling each attribute to the range [-1; +1] or [0; 1].
- Of course we have to use the same method to scale both training and testing data. For example, suppose that we scaled the rst attribute of training data from [-10; +10] to [-1; +1]. If the rst attribute of testing data lies in the range [-11; +8], we must scale the testing data to [-1.1; +0:8].

QP with basis functions

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\boldsymbol{\Phi}(\mathbf{x}_k).\boldsymbol{\Phi}(\mathbf{x}_l))$

Subject to these constraints: $0 \leq \alpha_k \leq$

We must do $R^2/2$ dot products to get this matrix ready.
Each dot product requires $m^2/2$ additions and multiplications
The whole thing costs $R^2 m^2 /4$.
Yeeks!

...or does it?

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \boldsymbol{\Phi}(\mathbf{x}_k)$$

$$b = y_K (1 - \varepsilon_K) - \mathbf{x}_K . \mathbf{w}_K$$

$$\text{where } K = \arg\max_k \alpha_k$$

Most important changes:

$$\boldsymbol{x} \rightarrow \phi(\boldsymbol{x})$$

Higher Order Polynomials

Poly-nomial	$\phi(\boldsymbol{x})$	Cost to build Q_{kl} matrix tradition ally	Cost if 100 inputs	$\phi(\boldsymbol{a}).\phi(\boldsymbol{b})$	Cost to build Q_{kl} matrix sneakily	Cost if 100 inputs
Quadratic	All $m^2/2$ terms up to degree 2	$m^2 R^2/4$	2,500 R^2	$(\boldsymbol{a}.\boldsymbol{b}+1)^2$	$m R^2/2$	50 R^2
Cubic	All $m^3/6$ terms up to degree 3	$m^3 R^2/12$	83,000 R^2	$(\boldsymbol{a}.\boldsymbol{b}+1)^3$	$m R^2/2$	50 R^2
Quartic	All $m^4/24$ terms up to degree 4	$m^4 R^2/48$	1,960,000 R^2	$(\boldsymbol{a}.\boldsymbol{b}+1)^4$	$m R^2/2$	50 R^2

Common SVM Kernel functions

$$\mathbf{Q}_{ij} = y_i y_j (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j))$$

- We must do **$R^2/2$** dot products to get this matrix ready.
- Each dot product now only requires ***m*** additions and multiplications
- But there are still worrying things lurking away. What are they?
  - The fear of overfitting with this enormous number of terms
    - The use of **Maximum Margin** magically makes this not a problem
  - The evaluation phase will be very expensive *(why?)*
    - evaluation phase : doing a set of predictions on a test set
    - Because each $\mathbf{w} \cdot \Phi(\mathbf{x})$ needs $R^2 m^2/4$ operations. *What can be done?*

$$\mathbf{w} \cdot \Phi(\mathbf{x}) = \sum_{i \ s.t. \ \alpha_i > 0} \alpha_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) = \sum_{i \ s.t. \ \alpha_i > 0} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x} + 1)^p$$

Only *Sm* operations (*S*=#support vectors)

Common SVM Kernel functions

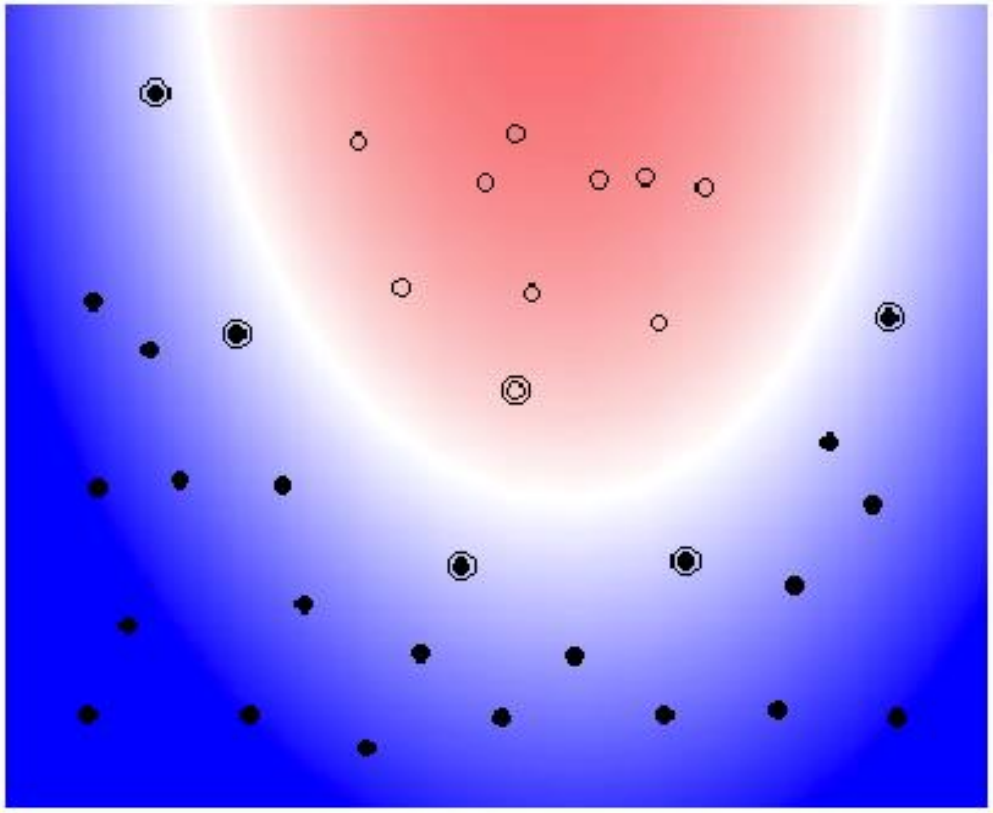
Andrew's opinion of why SVMs don't overfit as much as you'd think

- No matter what the basis function, there are really only up to R parameters: α_1, α_2 .. α_R, and usually most are set to zero by the Maximum Margin.
- Asking for small **w.w** is like "weight decay" in Neural Nets and like Ridge Regression parameters in Linear regression and like the use of Priors in Bayesian Regression---all designed to **smooth the function** and **reduce overfitting**.

Nonlinear Kernel

- Example: SVM with Polynomial of Degree 2

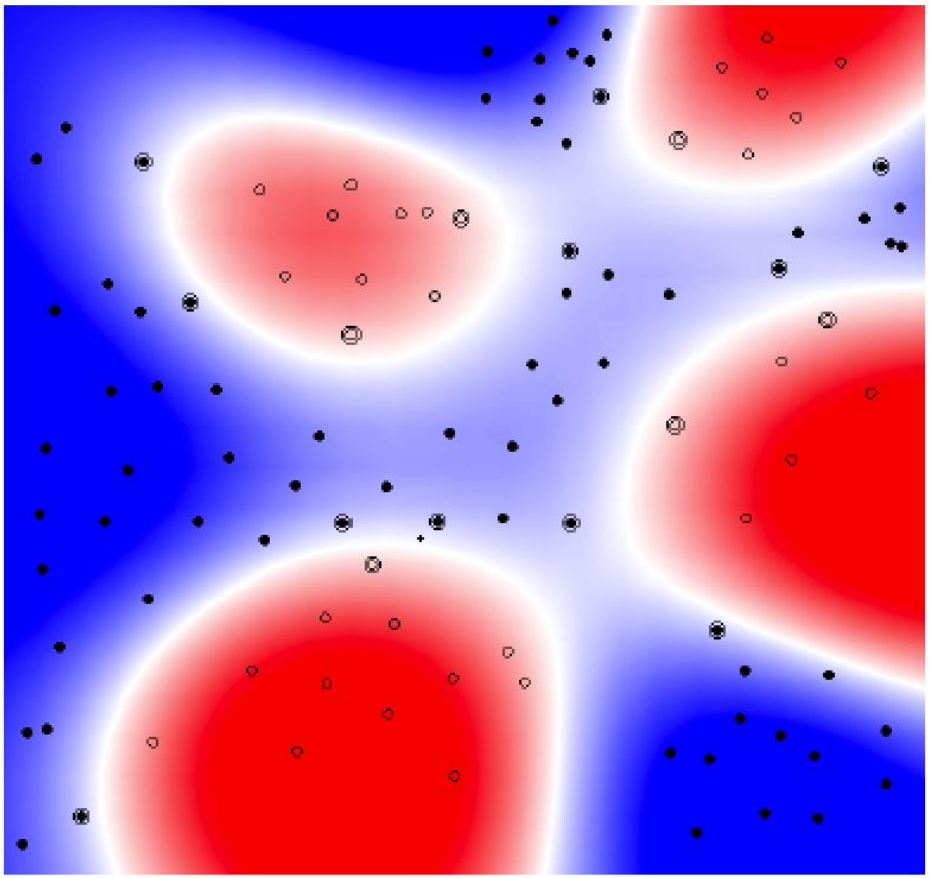
$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2$$



Nonlinear Kernel

- Example: SVM with RBF-Kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \ \gamma > 0$$




UESTC
1956