Kernel Trick contd…

Aarti Singh

Machine Learning 10-315
Nov 2, 2020

Dual formulation only depends on dot-products, not on w!

α_i - Lagrange multiplier

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i . \mathbf{x}_j$$ ← original feature

n-dim problem

$$\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$ ← high-dim features

$$\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

Φ(**x**) – High-dimensional feature space, but never need it explicitly as long as we can compute the dot product fast using some Kernel K

Dot Product of Polynomials

$\Phi(\mathbf{x}) =$ polynomials of degree exactly d

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

d=1 $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = x_1 z_1 + x_2 z_2 = \mathbf{x} \cdot \mathbf{z}$

d=2 $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) = \begin{bmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{bmatrix} \cdot \begin{bmatrix} z_1^2 \\ \sqrt{2} z_1 z_2 \\ z_2^2 \end{bmatrix}$

$$\begin{aligned} &= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2 \\ &= (x_1 z_1 + x_2 z_2)^2 \\ &= (\mathbf{x} \cdot \mathbf{z})^2 \end{aligned}$$

d $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) = K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^d$

Common Kernels

Using kernels, cost of computing dot products depends on dimension of original features x, and NOT transformed features $\phi(x)$

- Polynomials of degree d

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v})^d$$

- Polynomials of degree up to d

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v}+1)^d$$

- Gaussian/Radial kernels (polynomials of all orders – recall series expansion of exp)

$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||^2}{2\sigma^2}\right) = \phi(u)\cdot\phi(v)$$

- Sigmoid

$$K(\mathbf{u},\mathbf{v}) = \tanh(\eta\mathbf{u}\cdot\mathbf{v}+\nu)$$

Mercer Kernels

What functions are valid kernels that correspond to feature vectors $\varphi(\mathbf{x})$?

$$K(x, x') = \phi(x) \cdot \phi(x') = \phi(x') \cdot \phi(x) = K(x', x)$$

Answer: **Mercer kernels** K

- K is continuous
- K is symmetric
- K is positive semi-definite, i.e. $\mathbf{x}^T K \mathbf{x} \geq 0$ for all $\mathbf{x}$

$$\underset{D\times n}{\Phi} = \left[\left[\phi(x_1)\right] \left[\phi(x_2)\right] \cdots \right]$$

Ensures optimization is concave maximization

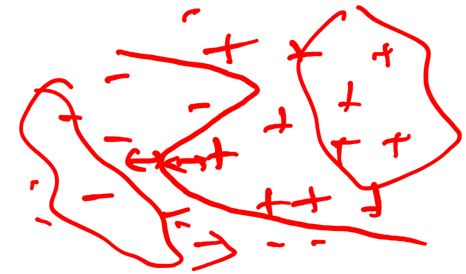
$$x^T K x = x^T \, \Phi \cdot \Phi \, x \geq 0$$

$$K_{ij} = \phi(x_i) \cdot \phi(x_j)$$

$$K = \Phi \cdot \Phi = \Phi^T \Phi$$

$$\underset{n\times n}{K} = \Phi^T \Phi = (n\times D)(D\times n)$$

Overfitting



- Huge feature space with kernels, what about overfitting???
  - Maximizing margin leads to sparse set of support vectors $\alpha_i = 0$
  - Some interesting theory says that SVMs search for simple hypothesis with large margin
  - Often robust to overfitting

What about classification time?

- For a new input **x**, if we need to represent $\Phi(\mathbf{x})$, we are in trouble!
- Recall classifier: sign(**w**.Φ(**x**)+b)

$$\mathbf{w} = \sum_i \alpha_i y_i \Phi(\mathbf{x}_i)$$

$$b = y_k - \mathbf{w}.\Phi(\mathbf{x}_k)$$

for any k where $C > \alpha_k > 0$

$w.\Phi(x)$

$= \sum_i \alpha_i y_i K(x_i, x)$

$w.\Phi(x_k)$

- Using kernels we are cool!

$$K(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v})$$

SVMs with Kernels

- Choose a set of features and kernel function
- Solve dual problem to obtain support vectors α_i
- At classification time, compute:

$$\mathbf{w} \cdot \Phi(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i)$$

$$b = y_k - \sum_i \alpha_i y_i K(\mathbf{x}_k, \mathbf{x}_i)$$

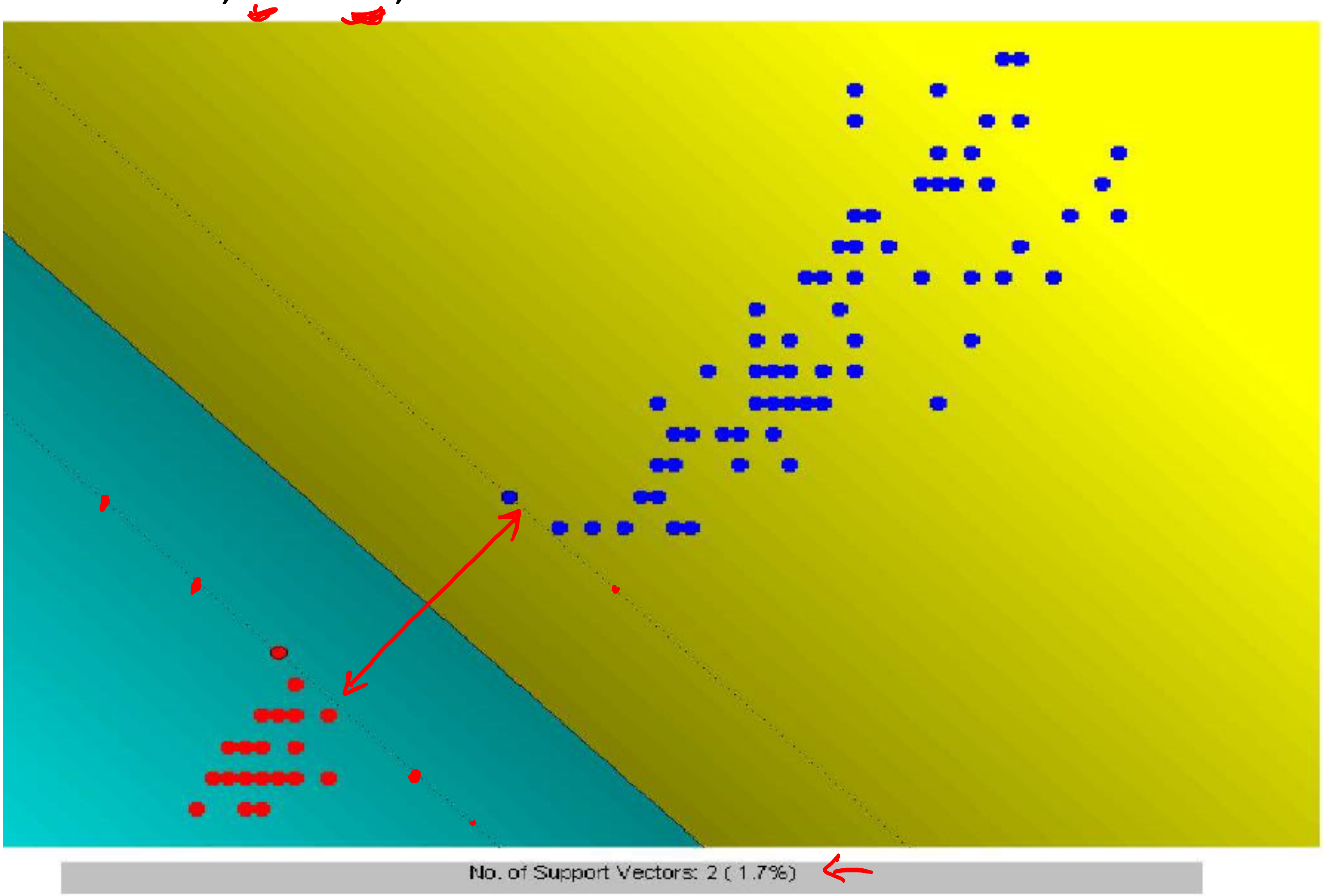
for any k where $C > \alpha_k > 0$

Classify as

$$sign\left(\mathbf{w} \cdot \Phi(\mathbf{x}) + b\right)$$

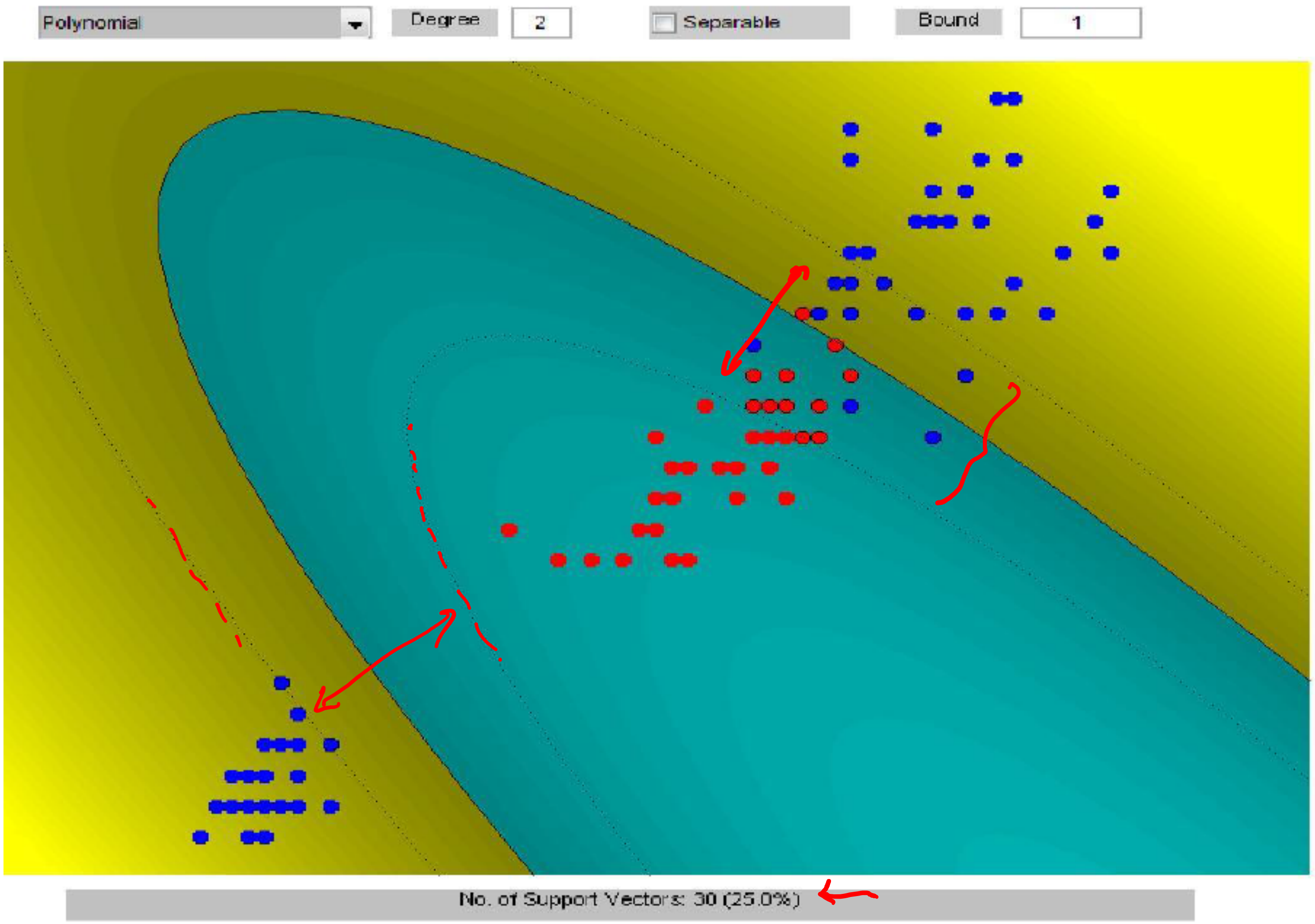
SVMs with Kernels

- Iris dataset, 2 vs 13, Linear Kernel



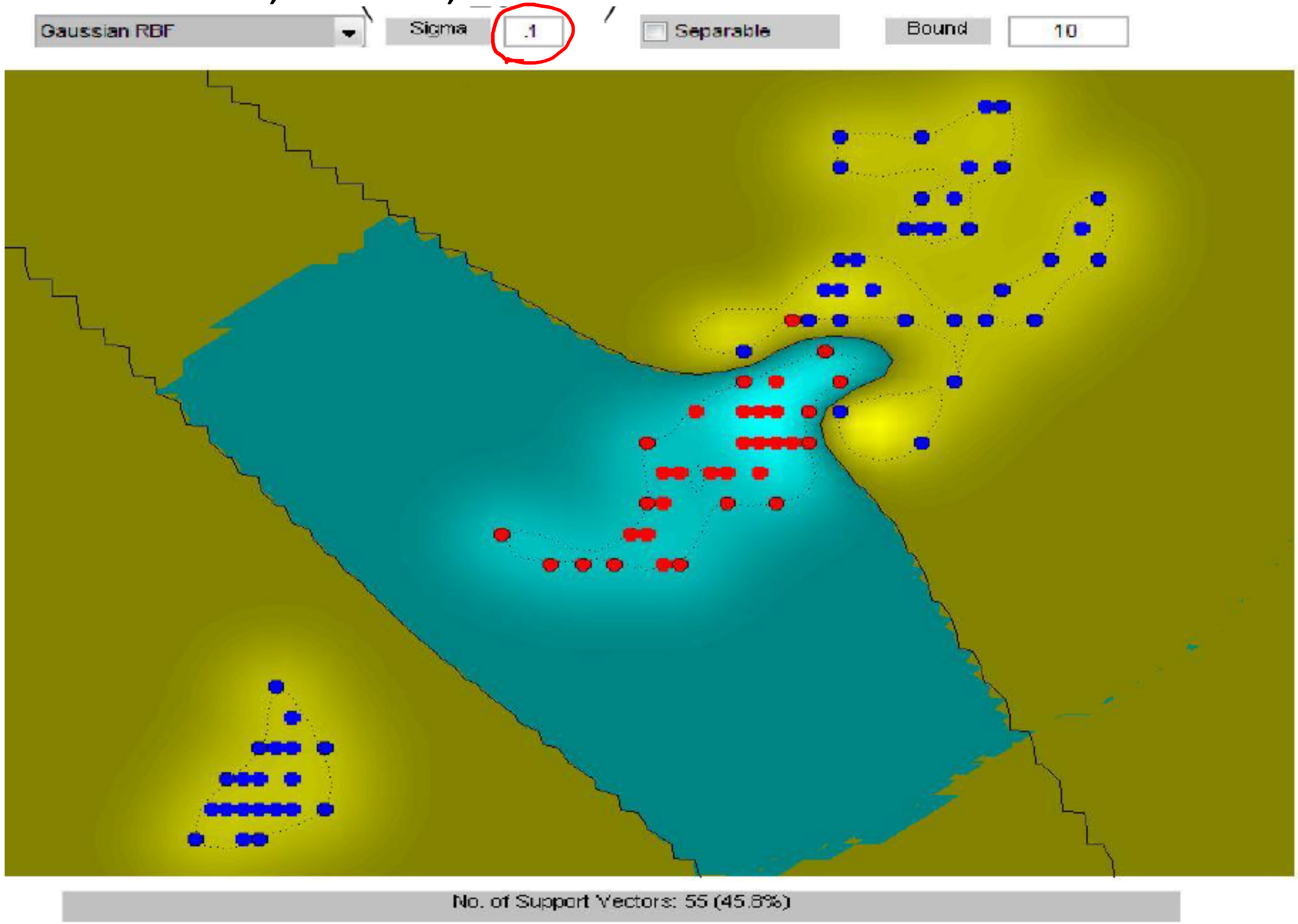
SVMs with Kernels

- Iris dataset, 1 vs 23, Polynomial Kernel degree 2



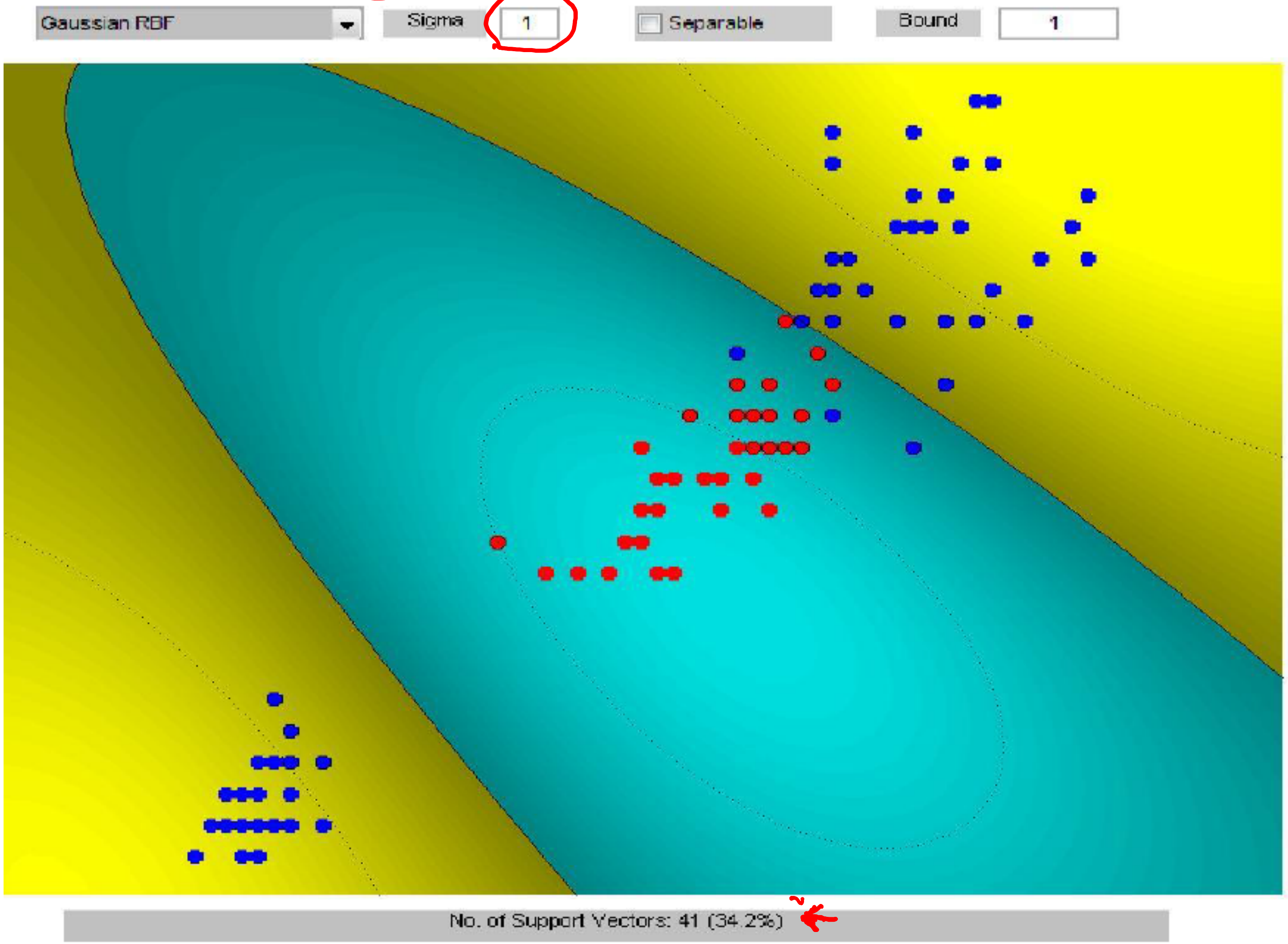
SVMs with Kernels

- Iris dataset, 1 vs 23, Gaussian RBF kernel



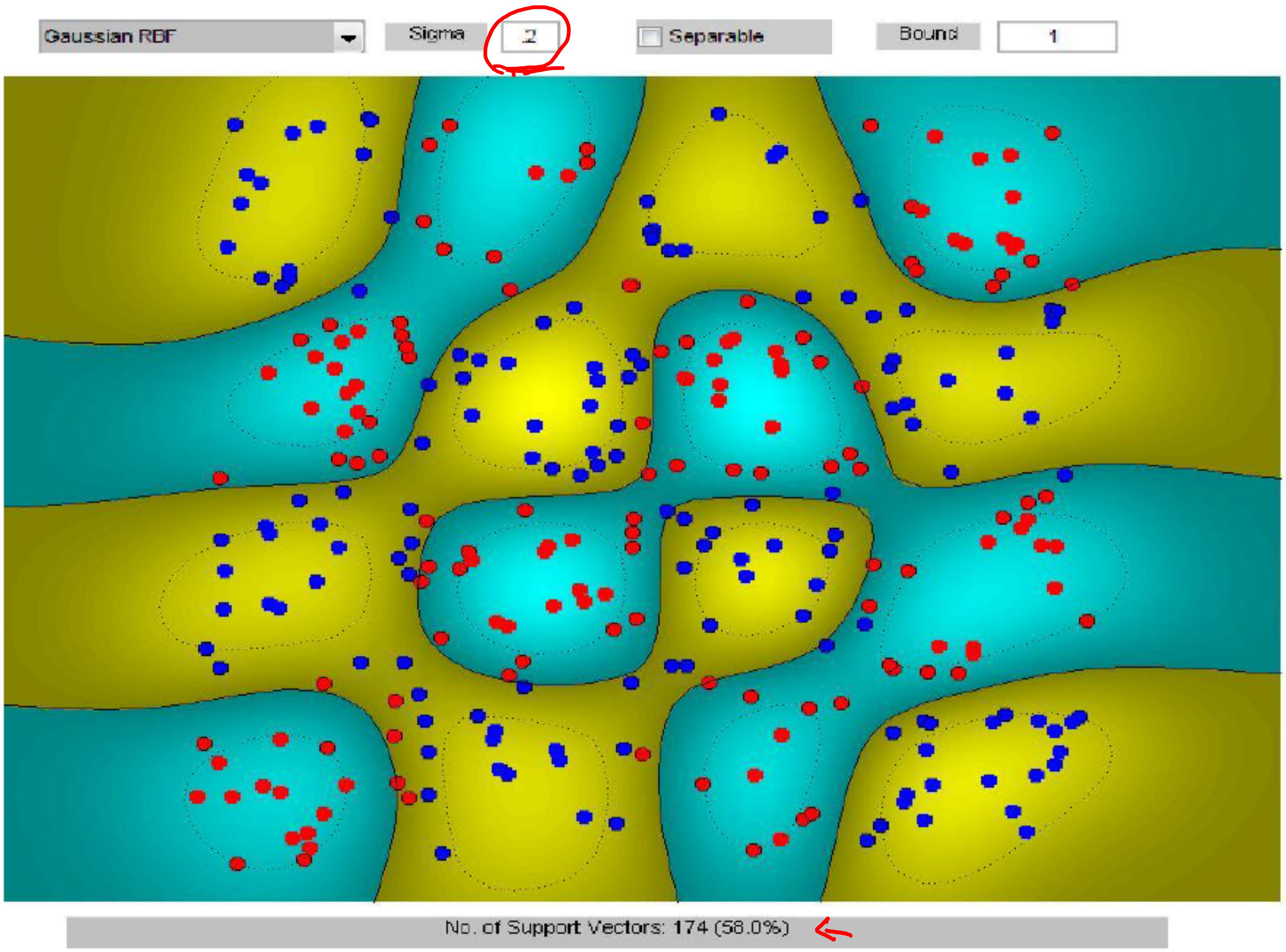
SVMs with Kernels

- Iris dataset, 1 vs 23, Gaussian RBF kernel



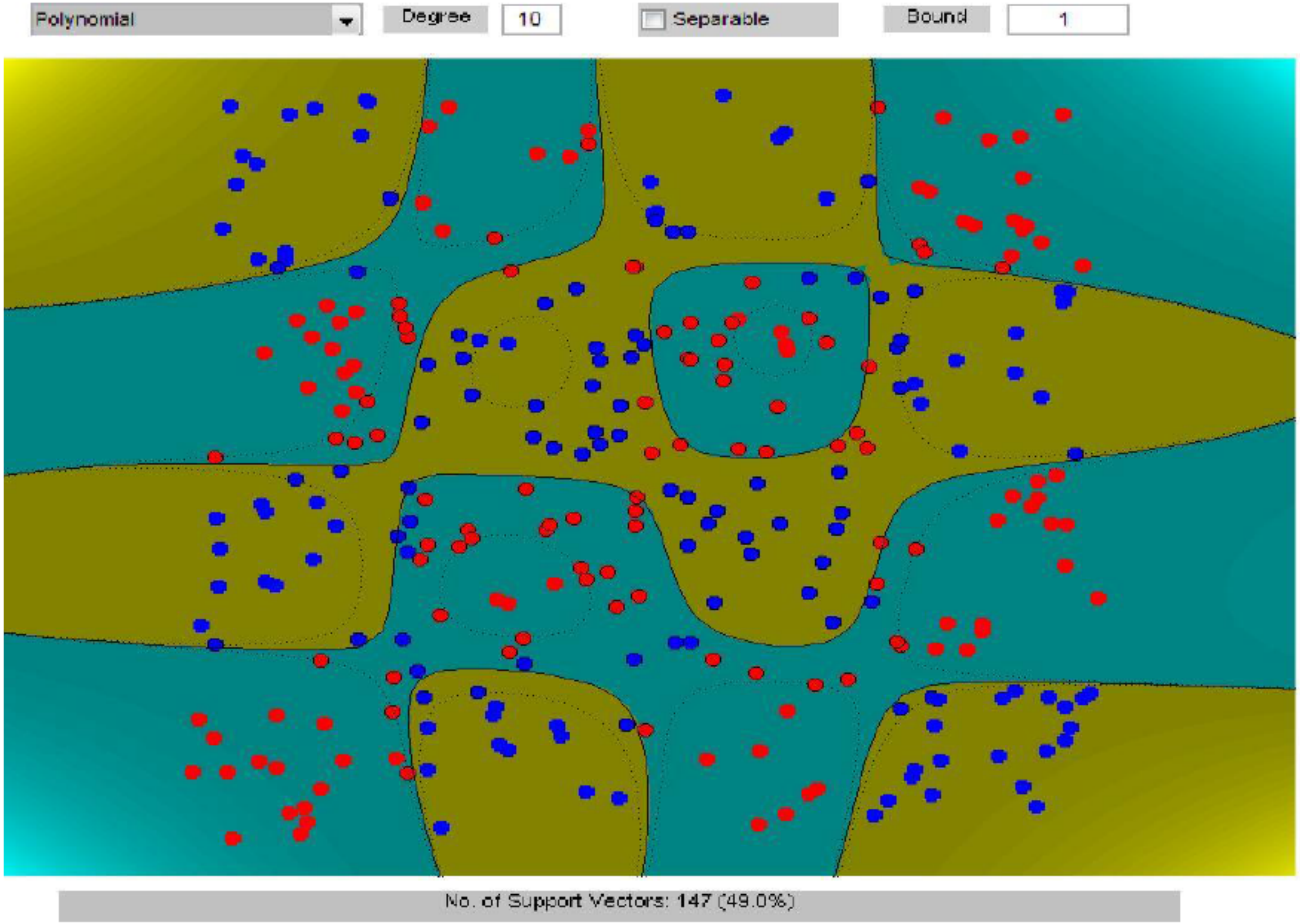
SVMs with Kernels

- Chessboard dataset, Gaussian RBF kernel

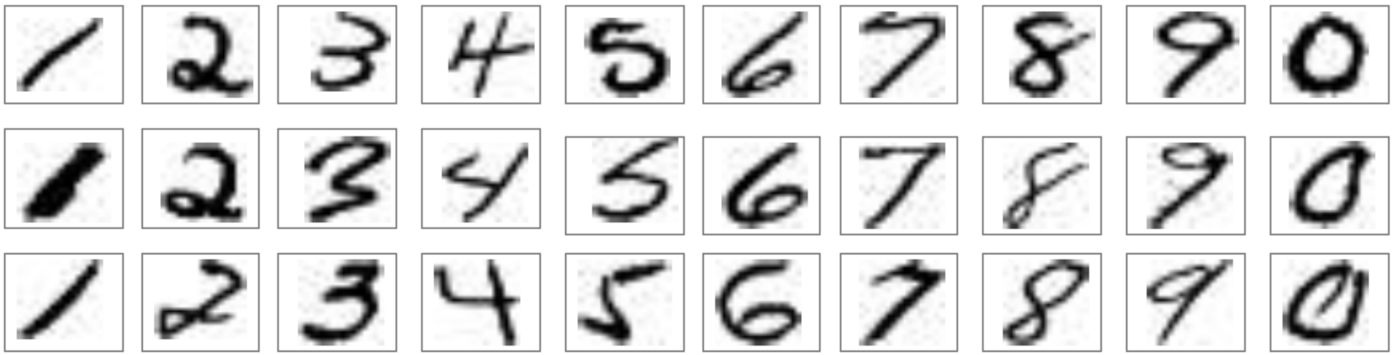


SVMs with Kernels

- Chessboard dataset, Polynomial kernel



USPS Handwritten digits



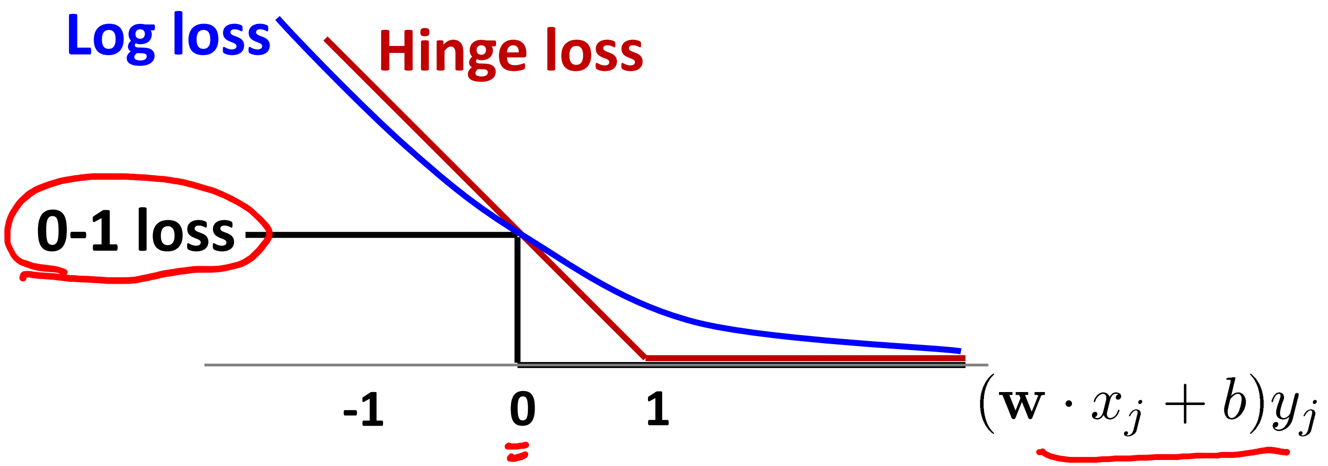
- ❑ 1000 training and 1000 test instances

Results:
SVM on raw images **~97%** accuracy

SVMs vs. Logistic Regression

	SVMs	**Logistic Regression**
Loss function	Hinge loss	Log-loss

M(C) LE



SVMs vs. Logistic Regression

	SVMs	**Logistic Regression**
Loss function	Hinge loss	Log-loss
High dimensional features with kernels	Yes!	Yes!

Kernels in Logistic Regression

$$P(Y = 1 \mid x, \mathbf{w}) = \frac{1}{1 + e^{-(\mathbf{w} \cdot \Phi(\mathbf{x}) + b)}}$$

- Define weights in terms of features:

$w = \sum_i \alpha_i y_i \phi(x_i)$

$$\mathbf{w} = \sum_i \alpha_i \Phi(\mathbf{x}_i)$$

i ← data points

$$P(Y = 1 \mid x, \mathbf{w}) = \frac{1}{1 + e^{-(\sum_i \alpha_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) + b)}}$$

$$= \frac{1}{1 + e^{-(\sum_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b)}}$$

- Derive simple gradient descent rule on α_i

SVMs vs. Logistic Regression

	SVMs	Logistic Regression
Loss function	Hinge loss	Log-loss
High dimensional features with kernels	Yes!	Yes!

SVMs vs. Logistic Regression

	SVMs	Logistic Regression
Loss function	Hinge loss	Log-loss
High dimensional features with kernels	Yes!	Yes!
Solution sparse	Often yes!	Almost always no!

SVMs vs. Logistic Regression

	SVMs	Logistic Regression
Loss function	Hinge loss	Log-loss
High dimensional features with kernels	Yes!	Yes!
Solution sparse	Often yes!	Almost always no!
Semantics of output	“Margin”	Real probabilities

Can we kernelize linear regression?

Linear (Ridge) regression

$$\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2$$

$$\widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y} \qquad \hat{f}_n(X) = X\hat{\beta}$$

Recall

$$\mathbf{A} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} X_1^{(1)} & \dots & X_1^{(p)} \\ \vdots & \ddots & \vdots \\ X_n^{(1)} & \dots & X_n^{(p)} \end{bmatrix}$$

$\mathbf{A}^T\mathbf{A}$ is a p x p matrix whose entries denote the (sample) correlation between the features

NOT inner products between the data points – the inner product matrix would be $\mathbf{A}\mathbf{A}^T$ which is n x n (also known as Gram matrix)

Ridge regression (dual)

$\Phi(X)\hat{\beta}$

$\phi(X_i)$

$$\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2$$

$$\hat{f}_n(X) = \sum_i \hat{\alpha}_i \Phi(X) \cdot \Phi(X_i)$$

$= K(X, X_i)$

- Define weights in terms of features: $\beta = \sum_i \alpha_i \Phi(X_i)$

$$\min_{\alpha} \sum_{i=1}^{n} (Y_i - \sum_j \alpha_j \Phi(X_i) \cdot \Phi(X_j))^2 + \lambda \sum_{ij} \alpha_i \alpha_j \Phi(X_i) \cdot \Phi(X_j)$$

$K(X_i, X_j)$ $K(X_i, X_j)$

$(Y_{n\times 1} - K_{n\times n} \alpha_{n\times 1})^T (Y - K\alpha)$ $\alpha^T_{1\times n} K_{n\times n} \alpha_{n\times 1}$

$$\rightarrow \min_{\alpha} (\mathbf{Y} - \mathbf{K}\boldsymbol{\alpha})^\top (\mathbf{Y} - \mathbf{K}\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha}$$

$\hat{\alpha} = (K + \lambda I)^{-1} Y$

$Y^TY + \alpha^T K^T K \alpha - 2\alpha^T K^T Y + \lambda \alpha^T K \alpha$

$2K^2\alpha - 2K^TY + 2\lambda K\alpha = 0$

$K(K + \lambda I)\alpha = KY$

Kernel ridge regression

$$\hat{f}_n(X) = \sum_i \hat{\alpha}_i K(X, X_i) = \mathbf{K}_X \hat{\boldsymbol{\alpha}}$$

1xn nx1

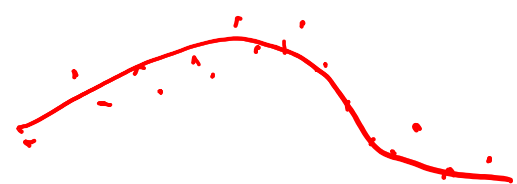
where $$\hat{\boldsymbol{\alpha}} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{Y}$$

Kx vector 1xn

$$\mathbf{K}_X(i) = \Phi(X) \cdot \Phi(X_i)$$

$$\mathbf{K}(i, j) = \Phi(X_i) \cdot \Phi(X_j)$$

Work with kernels, never need to write out the high-dim vectors



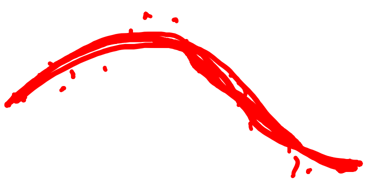
Ridge Regression with (implicit) nonlinear features $\Phi(X)$!

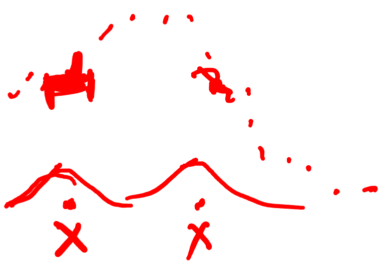
$$f(X) = \Phi(X)\beta$$

Kernel ridge regression vs. (local) Kernel Regression

$$\hat{f}_n(X) = \sum_i \hat{\alpha}_i K(X, X_i)$$

Kernel Ridge Regression	(Local) Kernel Regression
$\hat{\boldsymbol{\alpha}} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{Y}$	$\hat{\alpha}_i = \frac{Y_i}{\sum_i K(X, X_i)} = (\mathbf{1}^\top \mathbf{K}_X)^{-1}\mathbf{Y}$
	Weights depend on test point *X*
Global fit	Local fit
Interpret as weighted Nonlinear features	Interpret as weighted Least Squares





What you need to know

- Maximizing margin
- Derivation of SVM formulation
- Slack variables and hinge loss
- Tackling multiple class
  - One against All
  - Multiclass SVMs
- Dual SVM formulation
  - Easier to solve when dimension high d > n
  - Kernel Trick
- Relationship between SVMs and logistic regression
- Kernelizing linear regression e.g. Kernel Ridge Regression