Kernel Machines

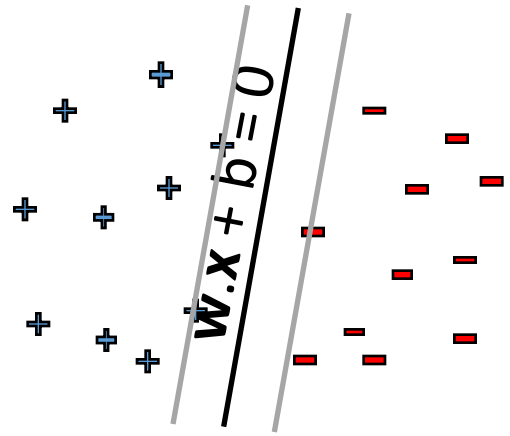
Pradeep Ravikumar
Co-instructor: Manuela Veloso

Machine Learning 10-701

SVM – linearly separable case

n training points — $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$

d features — $\mathbf{x}_j$ is a d-dimensional vector



- Primal problem: $\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}.\mathbf{w}$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \forall j$$

w – weights on features (d-dim problem)

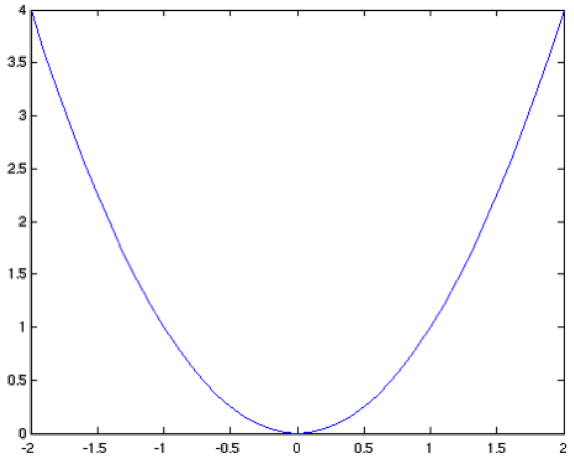
- Convex quadratic program – quadratic objective, linear constraints
- But expensive to solve if d is very large
- Often solved in dual form (n-dim problem)

Constrained Optimization

$\min_x \ x^2$
s.t. $x \geq b$

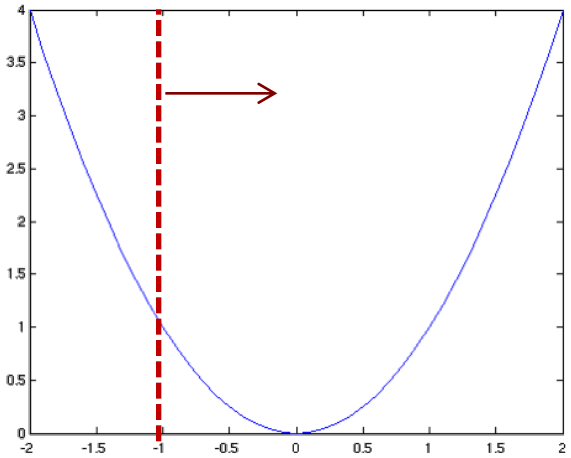
$$x^* = \max(b, 0)$$

$\min_x \ x^2$



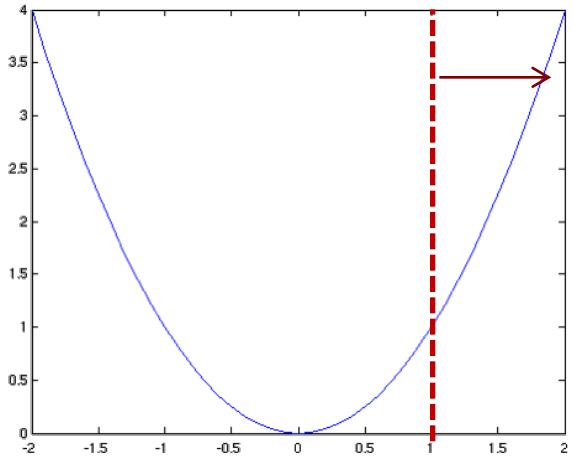
$x^* = 0$

$\min_x \ x^2$
s.t. $x \geq -1$



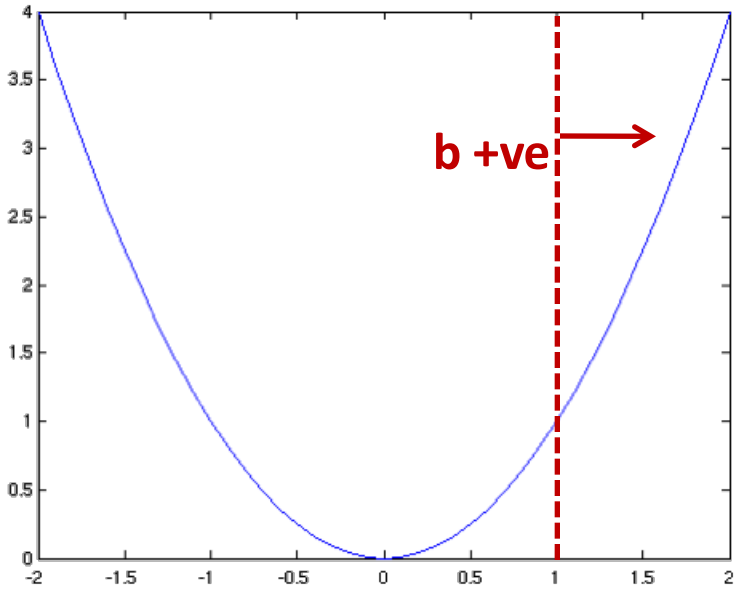
$x^* = 0$
Constraint inactive

$\min_x \ x^2$
s.t. $x \geq 1$



$x^* = 1$
Constraint active and tight

Constrained Optimization – Dual Problem



Primal problem:

$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq b$$

Moving the constraint to objective function

Lagrangian:

$$L(x, \alpha) = x^2 - \alpha(x - b)$$
$$\text{s.t.} \quad \alpha \geq 0$$

Dual problem:

$$\max_\alpha \; d(\alpha) \rightarrow \min_x L(x, \alpha)$$
$$\text{s.t.} \quad \alpha \geq 0$$

α = 0 constraint is inactive

α > 0 constraint is active

Connection between Primal and Dual

Primal problem: p* = $\min_x \; x^2$ s.t. $x \geq b$

Dual problem: d* = $\max_\alpha \; d(\alpha)$ s.t. $\alpha \geq 0$

- **Weak duality:** The dual solution d* lower bounds the primal solution p* i.e. d* ≤ p*

  To see this, recall $L(x, \alpha) = x^2 - \alpha(x - b)$

  For every feasible x (i.e. x ≥ b) and feasible α (i.e. α ≥ 0) , notice that

  $d(\alpha) = \min_x L(x, \alpha) \leq p^*$

- **Dual problem (maximization) is always concave even if primal is not convex**

Connection between Primal and Dual

Primal problem: p* = $\min_x \; x^2$ s.t. $x \geq b$

Dual problem: d* = $\max_\alpha \; d(\alpha)$ s.t. $\alpha \geq 0$

- **Weak duality:** The dual solution d* lower bounds the primal solution p* i.e. d* ≤ p*
- **Strong duality:** d* = p* holds often for many problems of interest e.g. if the primal is a feasible convex objective with linear constraints

Connection between Primal and Dual

What does strong duality say about α^* (the α that achieved optimal value of dual) and x^* (the x that achieves optimal value of primal problem)?

Whenever strong duality holds, the following conditions (known as KKT conditions) are true for α^* and x^*:

- 1. $\bigtriangledown L(x^*, \alpha^*) = 0$ i.e. Gradient of Lagrangian at x^* and α^* is zero.
- 2. $x^* \geq b$ i.e. x^* is primal feasible
- 3. $\alpha^* \geq 0$ i.e. α^* is dual feasible
- 4. $\alpha^*(x^* - b) = 0$ (called as complementary slackness)

We use the first one to relate x^* and α^*. We use the last one (complimentary slackness) to argue that $\alpha^* = 0$ if constraint is inactive and $\alpha^* > 0$ if constraint is active and tight.

Solving the dual

Solving:

$$\max_{\alpha} \min_{x} \underbrace{x^2 - \alpha(x-b)}_{L(x,\alpha)}$$
$$\text{s.t.} \quad \alpha \geq 0$$

Dual: d(alpha)

Optimization over x is unconstrained.

$$\frac{\partial L}{\partial x} = 2x - \alpha = 0 \Rightarrow x^* = \frac{\alpha}{2}$$

$$L(x^*, \alpha) = \frac{\alpha^2}{4} - \alpha\left(\frac{\alpha}{2} - b\right)$$
$$= -\frac{\alpha^2}{4} + b\alpha$$

Now need to maximize $L(x^*,\alpha)$ over $\alpha \geq 0$

Solve unconstrained problem to get α' and then take $\max(\alpha',0)$

$$\frac{\partial}{\partial \alpha} L(x^*, \alpha) = -\frac{\alpha}{2} + b \Rightarrow \alpha' = 2b$$

$$\Rightarrow \alpha^* = \max(2b, 0) \qquad \Rightarrow x^* = \frac{\alpha^*}{2} = \max(b, 0)$$

α = 0 constraint is inactive, $\alpha > 0$ constraint is active and tight

Dual SVM – linearly separable case

n training points, d features

($\mathbf{x}_1$, …, $\mathbf{x}_n$) where x_i is a d-dimensional vector

- Primal problem:

$$\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}.\mathbf{w}$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \ \forall j$$

w – weights on features (d-dim problem)

- Dual problem (derivation):

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}.\mathbf{w} - \sum_j \alpha_j \left[\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j - 1\right]$$
$$\alpha_j \geq 0, \ \ \forall j$$

alpha – weights on training pts (n-dim problem)

Dual SVM – linearly separable case

- Dual problem:

$$\max_{\alpha} \min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}.\mathbf{w} - \sum_j \alpha_j \left[\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j - 1\right]$$

$$\alpha_j \geq 0, \ \forall j$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \qquad \Rightarrow \mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

$$\frac{\partial L}{\partial b} = 0 \qquad \Rightarrow \sum_j \alpha_j y_j = 0$$

If we can solve for αs (dual problem), then we have a solution for **w**,b (primal problem)

Dual SVM – linearly separable case

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

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

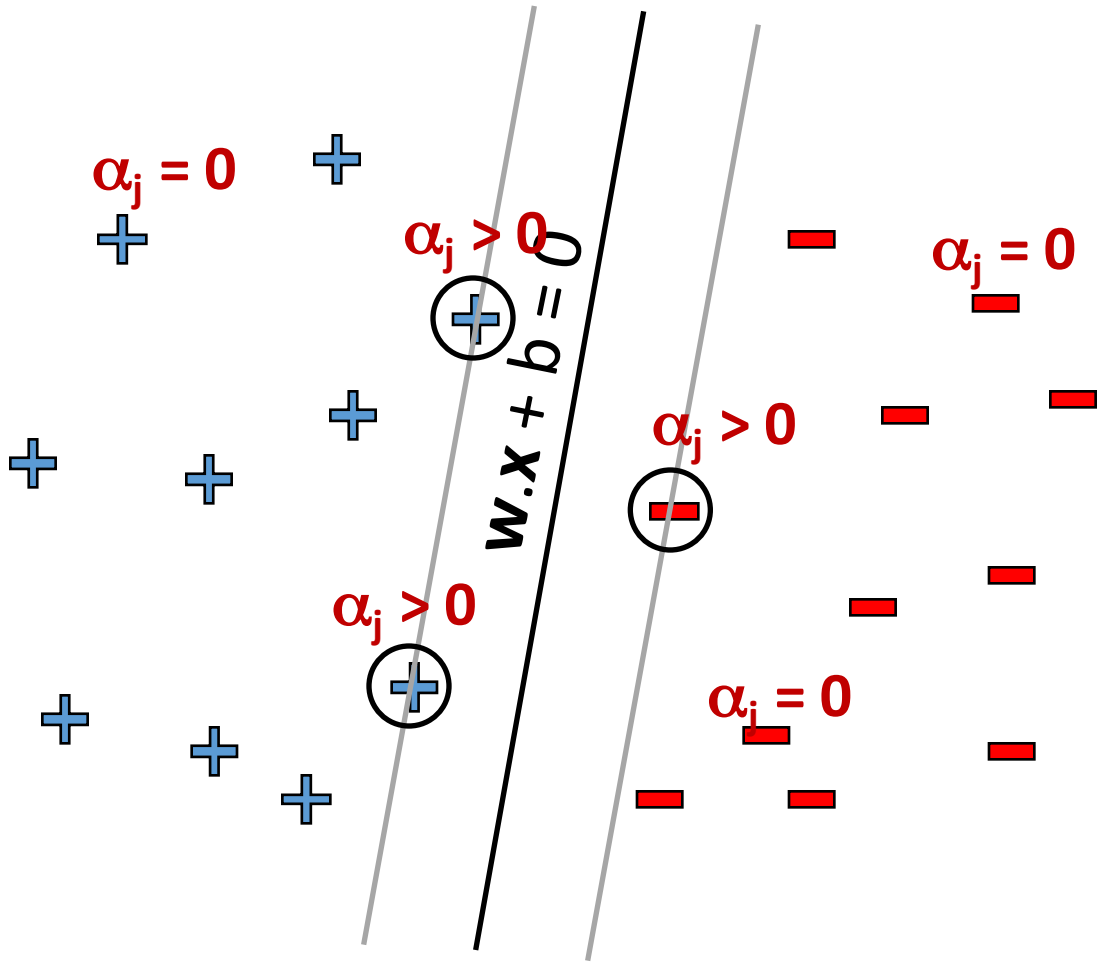
Dual problem is also QP

Solution gives α_js

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

What about b?

Dual SVM: Sparsity of dual solution



$$\mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

Only few α_js can be non-zero : where constraint is active and tight

$$(\mathbf{w}.\mathbf{x}_j + b)y_j = 1$$

Support vectors – training points j whose α_js are non-zero

Dual SVM – linearly separable case

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

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

Dual problem is also QP

Solution gives α_js

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

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

for any k where $\alpha_k > 0$

Use support vectors with α_k>0 to compute b since constraint is tight $(w.x_k + b)y_k = 1$

Dual SVM – non-separable case

- Primal problem:

$$\text{minimize}_{\mathbf{w},b,\{\xi_j\}} \frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_j \xi_j$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j, \ \forall j \qquad \alpha_j$$
$$\xi_j \geq 0, \ \forall j \qquad \mu_j$$

- Dual problem:

Lagrange Multipliers

$$\max_{\alpha,\mu} \min_{\mathbf{w},b,\{\xi_j\}} L(\mathbf{w}, b, \xi, \alpha, \mu)$$
$$s.t. \alpha_j \geq 0 \quad \forall j$$
$$\mu_j \geq 0 \quad \forall j$$

Dual SVM – non-separable case

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

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

comes from $\frac{\partial L}{\partial \xi} = 0$

Intuition:

If C→∞, recover hard-margin SVM

Dual problem is also QP

Solution gives α_js

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

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

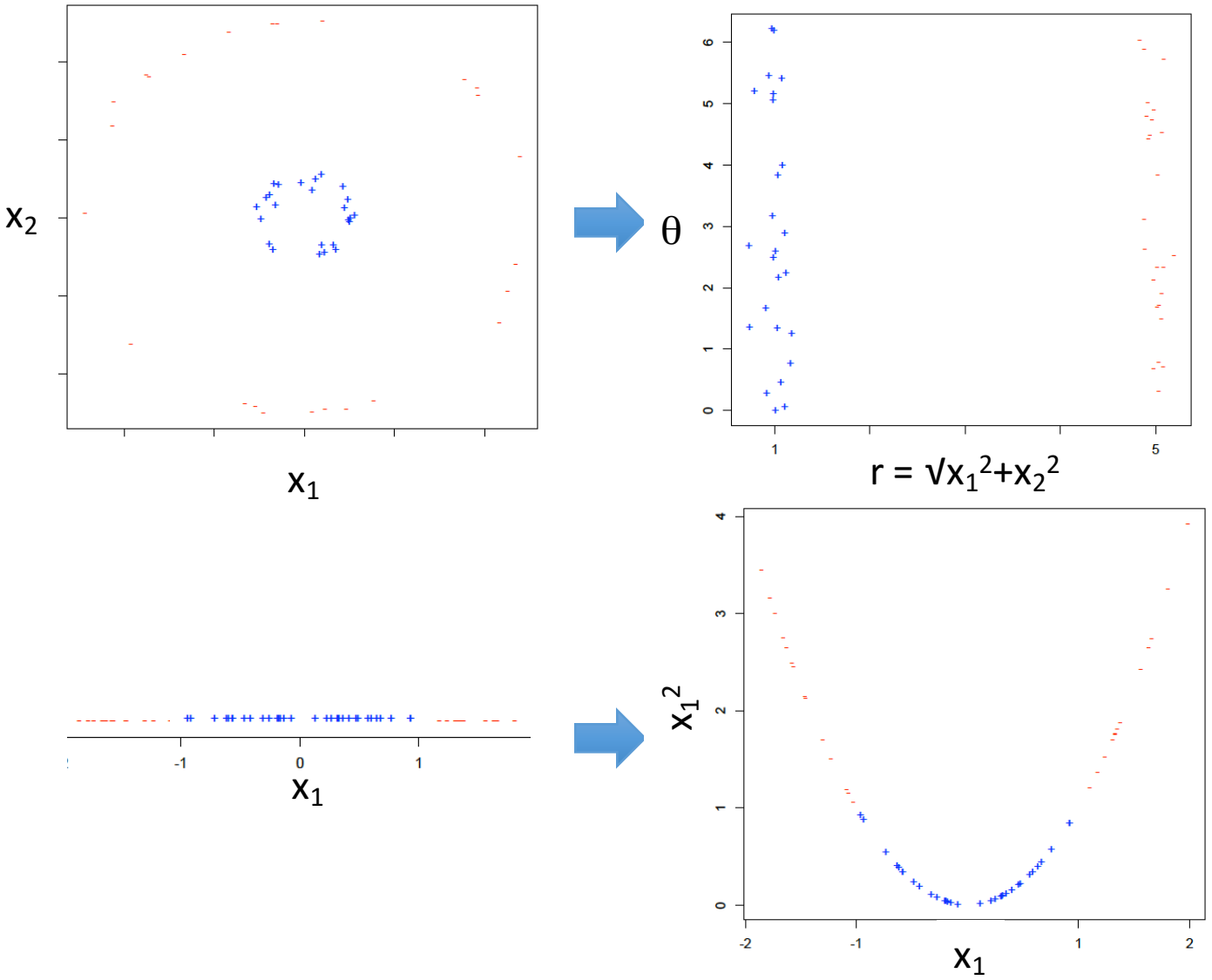
for any k where $C > \alpha_k > 0$

So why solve the dual SVM?

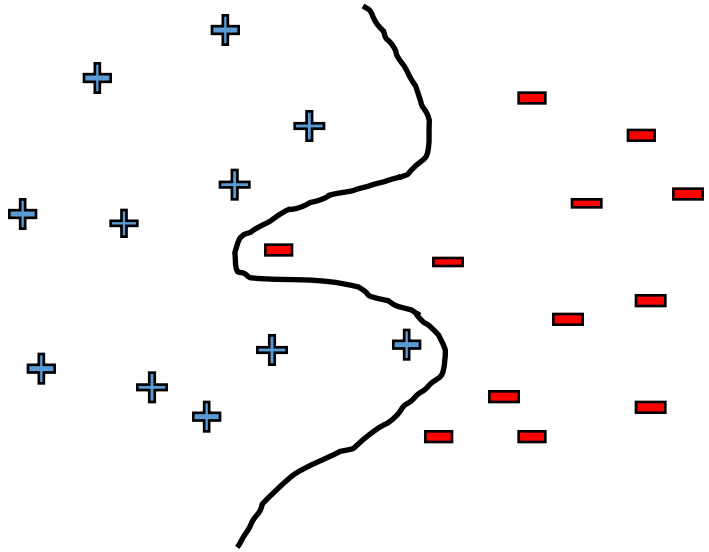
- There are some quadratic programming algorithms that can solve the dual faster than the primal, (specially in high dimensions d>>n)

- But, more importantly, the "**kernel trick**"!!!

Separable using higher-order features



What if data is not linearly separable?



Use features of features of features of features….

$\Phi(\mathbf{x}) = (x_1^2, x_2^2, x_1x_2, \ldots., \exp(x_1))$

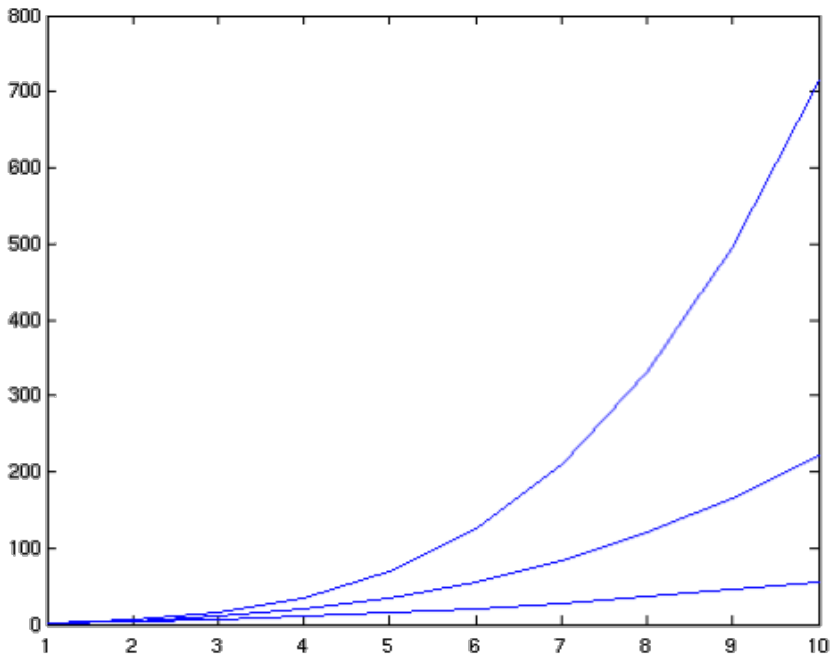
Feature space becomes really large very quickly!

Higher Order Polynomials

m – input features

d – degree of polynomial

$$\text{num. terms} = \left(\begin{array}{c} d+m-1 \\ d \end{array} \right) = \frac{(d+m-1)!}{d!(m-1)!} \sim m^d$$



grows fast!
d = 6, m = 100
about 1.6 billion terms

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

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

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

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

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$
$$\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

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

$$= (x_1 z_1 + x_2 z_2)^2$$

$$= (\mathbf{x} \cdot \mathbf{z})^2$$

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

Finally: The Kernel Trick!

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

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

- Never represent features explicitly
  - Compute dot products in closed form
- Constant-time high-dimensional dot-products for many classes of features

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

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

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

Common Kernels

- 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)

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

- 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 $\varphi(\mathbf{x})$?

Answer: Mercer kernels K

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

Overfitting

- Huge feature space with kernels, what about overfitting???
  - Maximizing margin leads to sparse set of support vectors
  - 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($\mathbf{w}.\Phi(\mathbf{x})$+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 $C > \alpha_k > 0$

- 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:

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

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

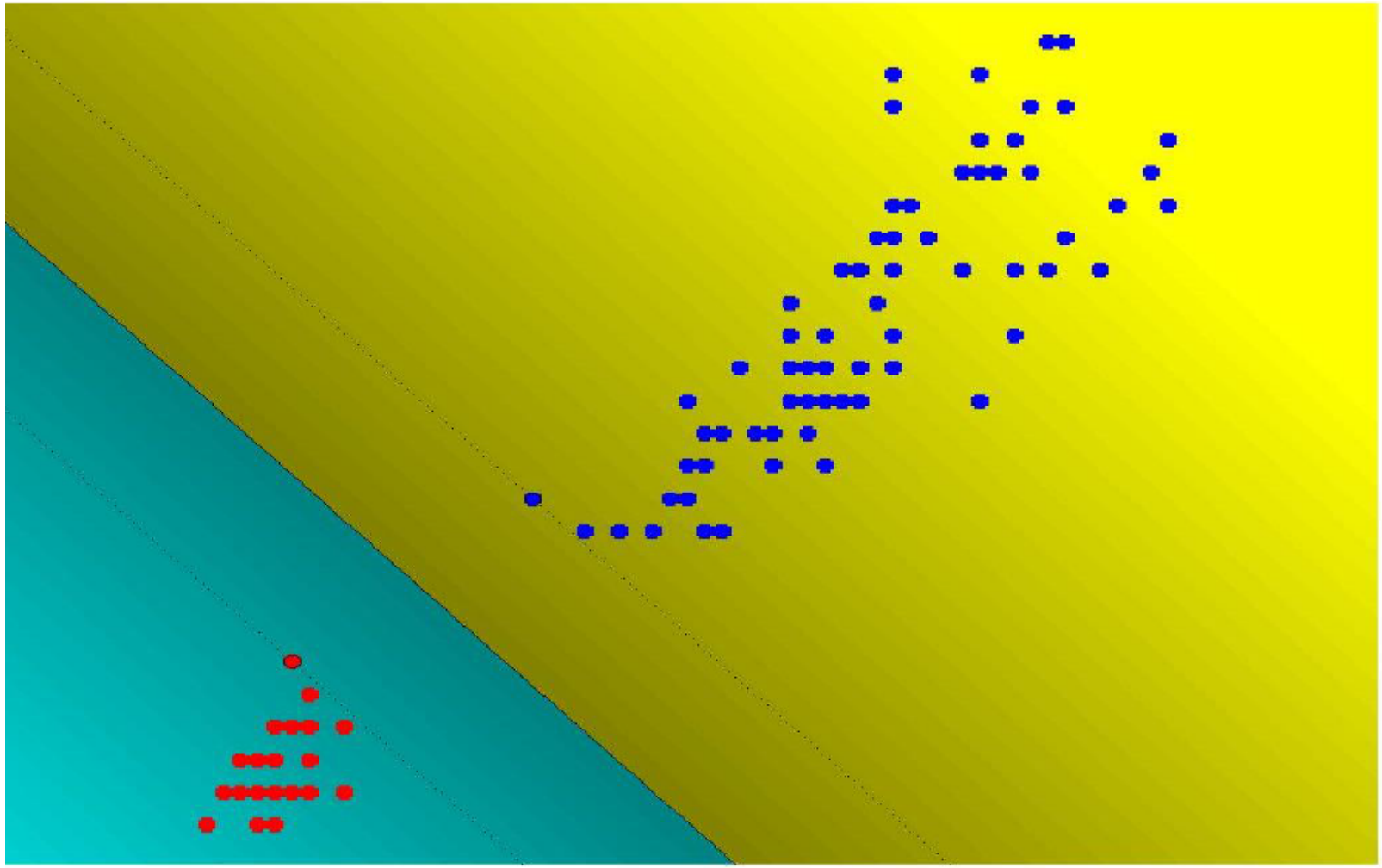
for any k where $C > \alpha_k > 0$

Classify as

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

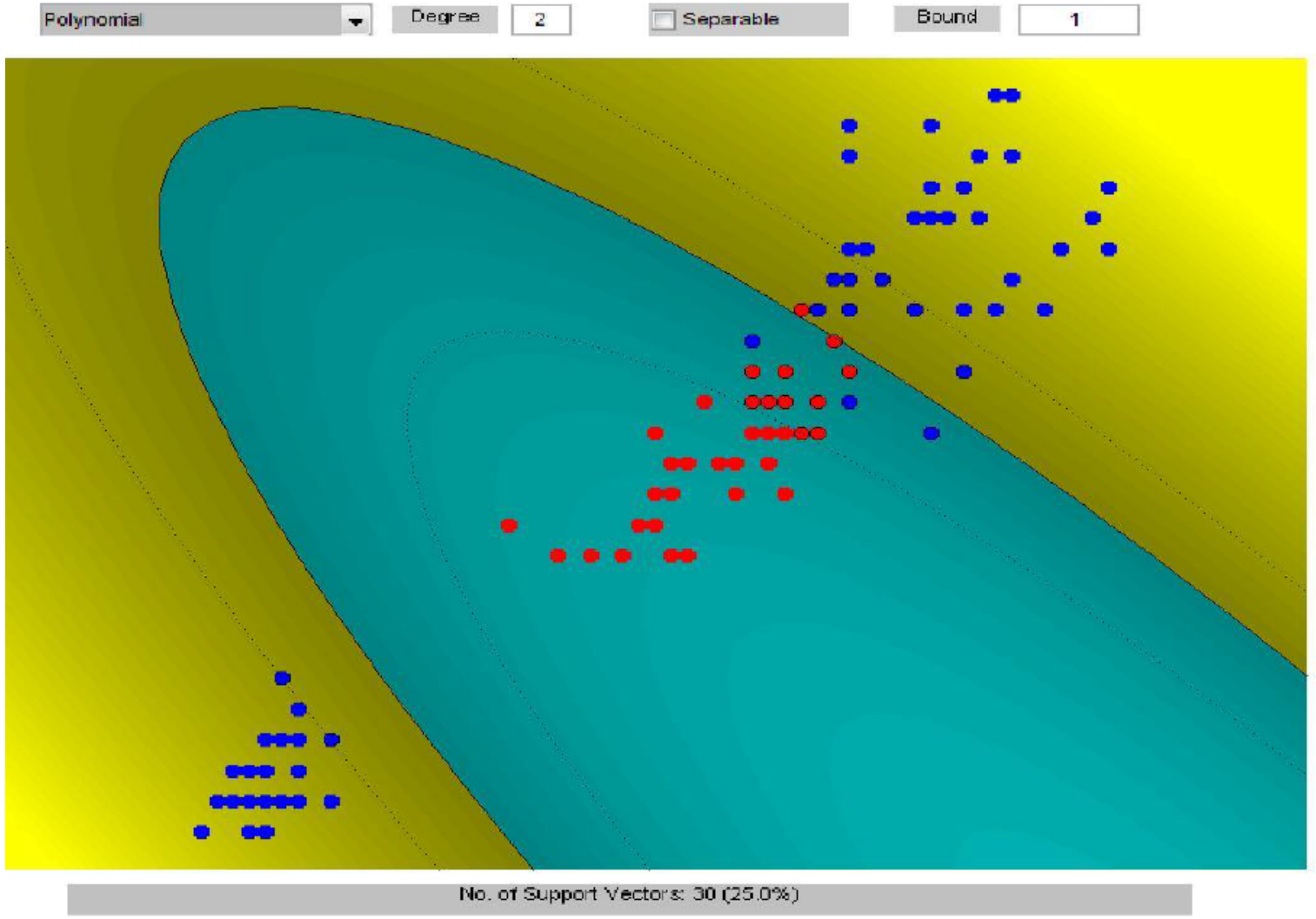
SVMs with Kernels

- Iris dataset, 2 vs 13, Linear Kernel



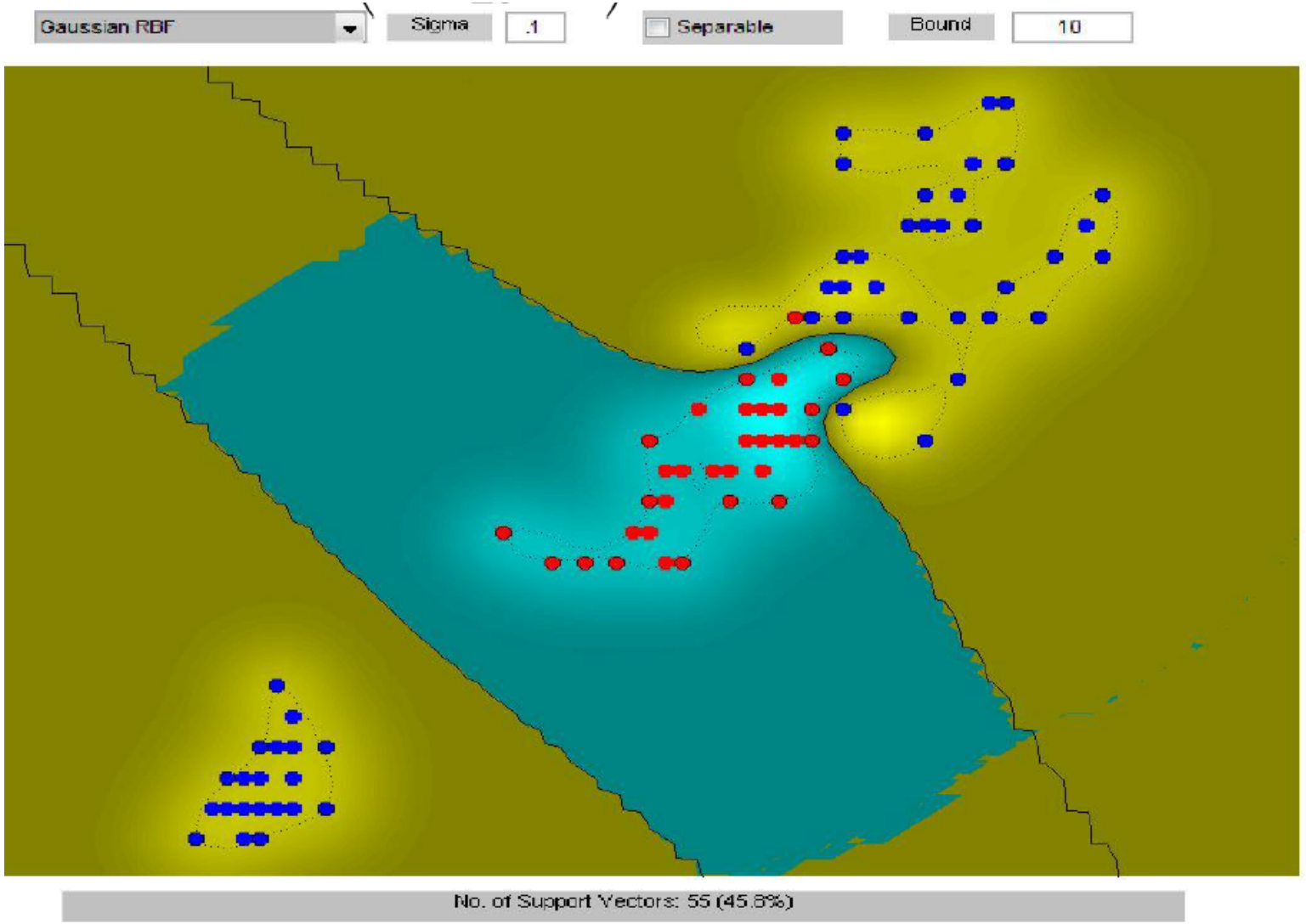
SVMs with Kernels

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



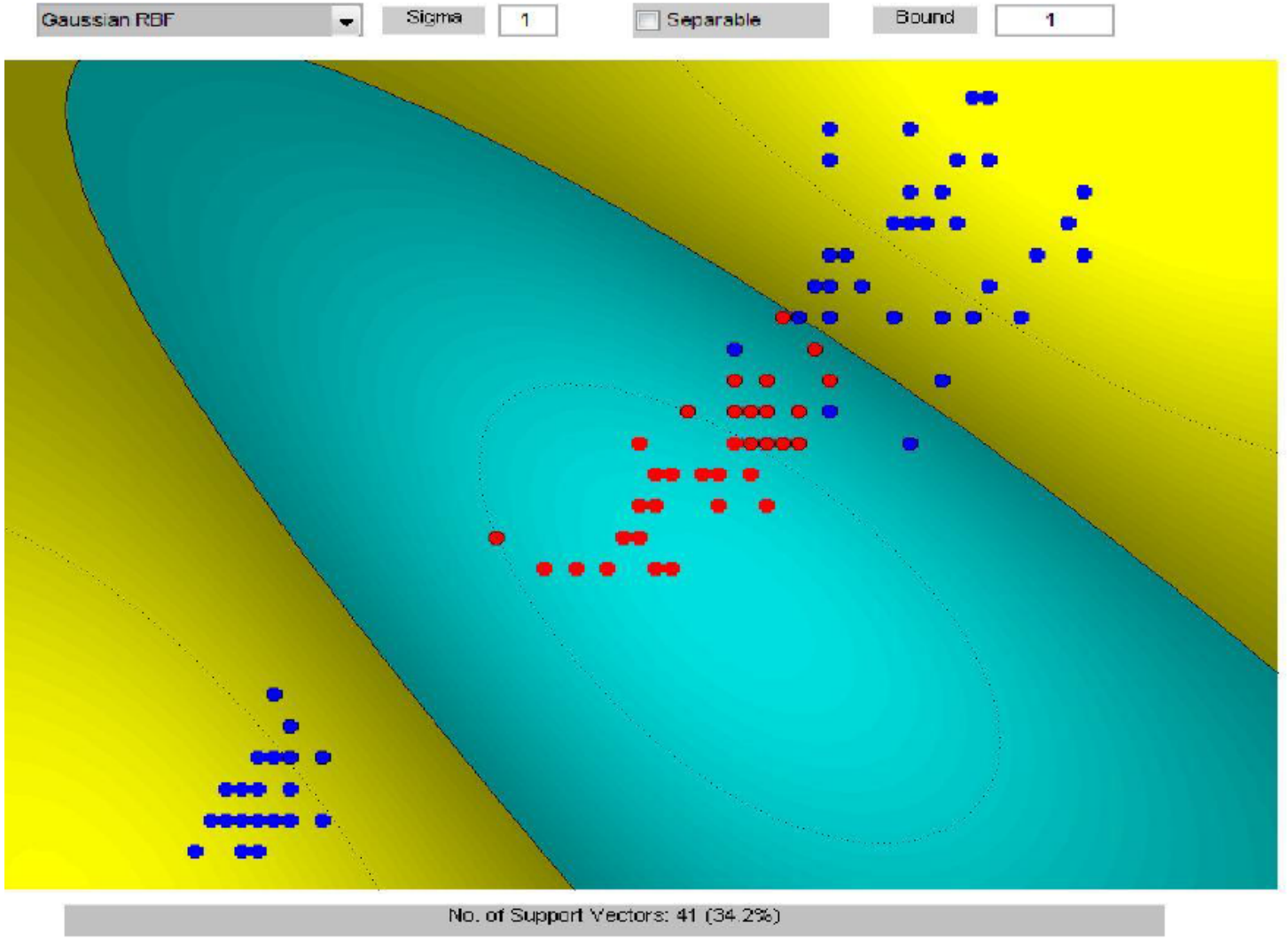
SVMs with Kernels

- Iris dataset, 1 vs 23, Gaussian RBF kernel



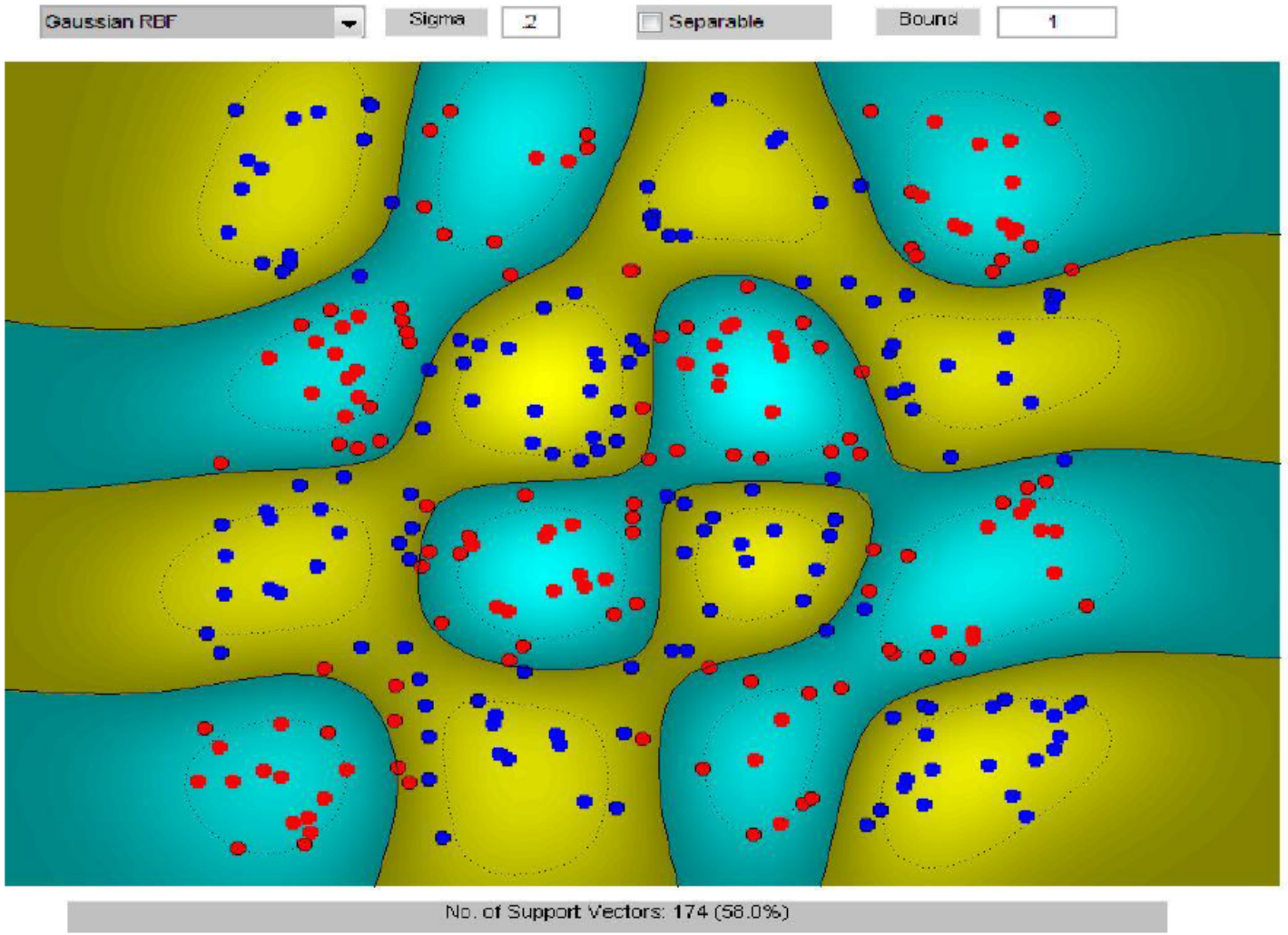
SVMs with Kernels

- Iris dataset, 1 vs 23, Gaussian RBF kernel



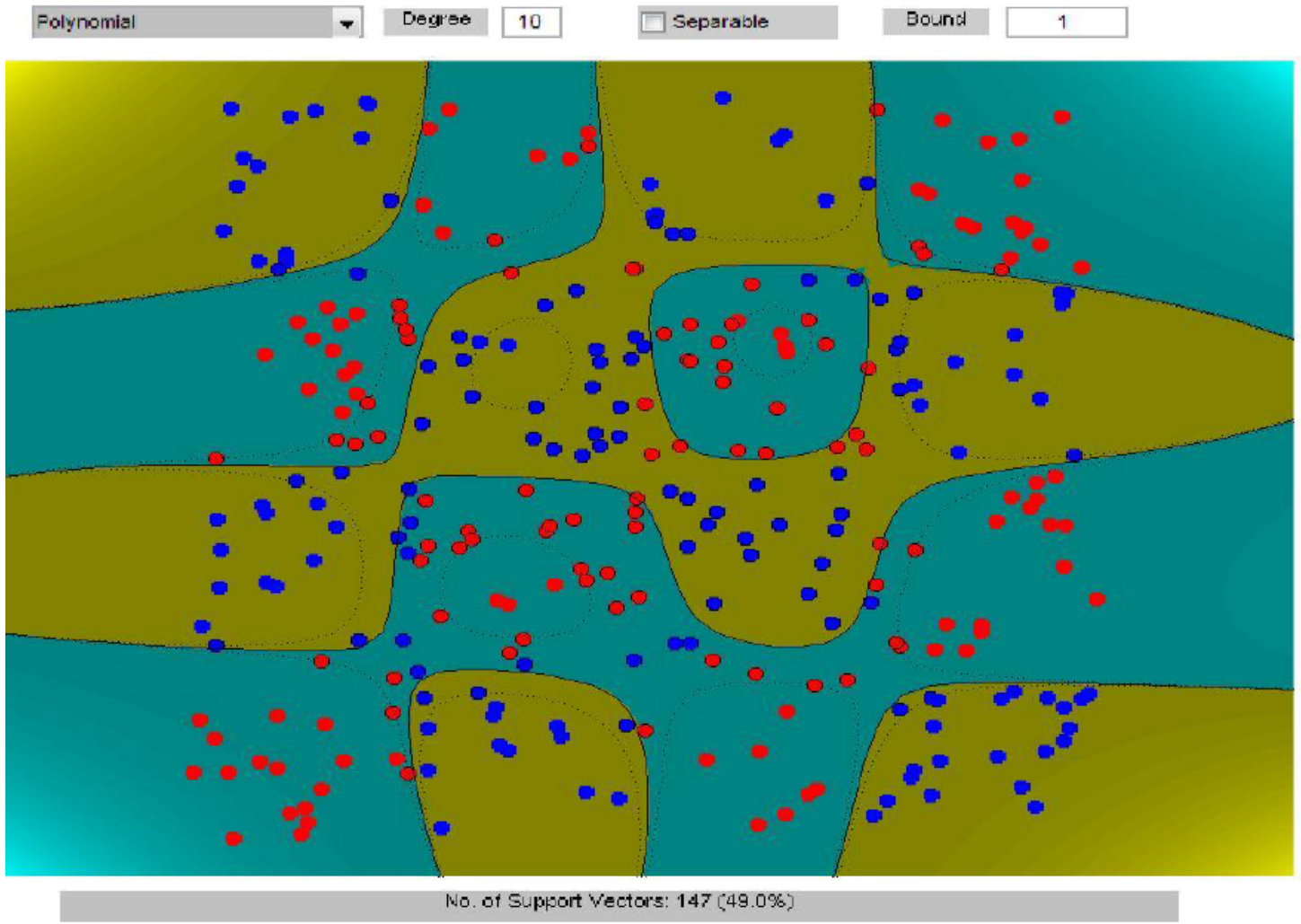
SVMs with Kernels

- Chessboard dataset, Gaussian RBF kernel



SVMs with Kernels

- Chessboard dataset, Polynomial kernel



Corel Dataset

Air shows

Bears

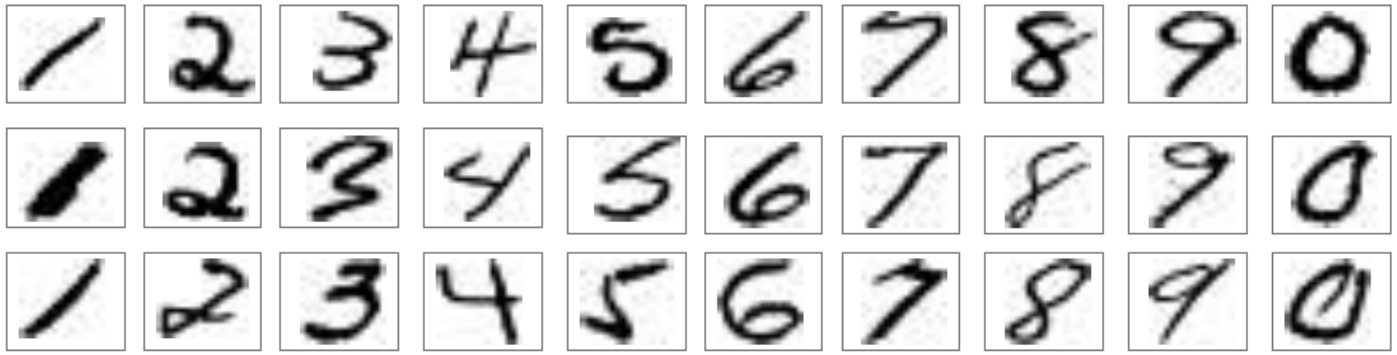
Horses

Corel Dataset

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
air-shows	1	31		1							1	1			
bears	2		26	2	2		2	1		1					
elephants	3		1	27				3						3	
tigers	4			1	32			1							
horses	5					34									
polar-bears	6						30					1		2	1
african-animals	7			1	1			30		1				1	
cheetahs	8						1		32			1			
eagles	9	1								33					
mountains	10	3								1	24	3	3		
fields	11			1				1			2	27	3		
deserts	12						2	1	1	2	1	3	24		
sunsets	13													34	
night scenes	14	1												2	31

Olivier Chapelle 1998

USPS Handwritten digits



- 1000 training and 1000 test instances

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

SVMs vs. Kernel Regression

SVMs

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

or

$$sign\left(\sum_i \alpha_i y_i K(\mathbf{x},\mathbf{x}_i)+b\right)$$

Kernel Regression

$$sign\left(\frac{\sum_i y_i K(\mathbf{x},\mathbf{x}_i)}{\sum_j K(\mathbf{x},\mathbf{x}_j)}\right)$$

Differences:

- SVMs:
  - Learn weights α_i (and bandwidth)
  - Often sparse solution
- KR:
  - Fixed "weights", learn bandwidth
  - Solution may not be sparse
  - Much simpler to implement

SVMs vs. Logistic Regression

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

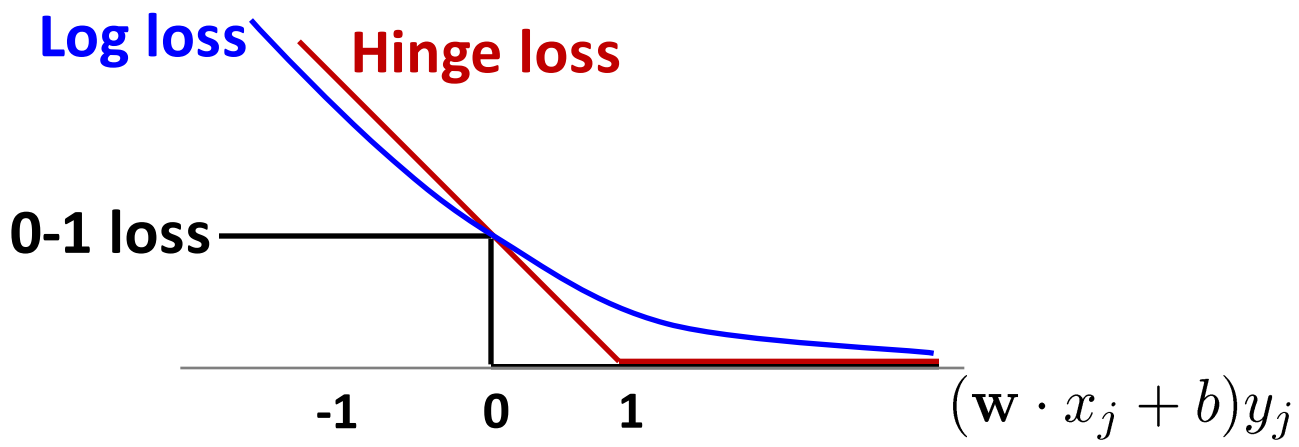
SVMs vs. Logistic Regression

SVM : **Hinge loss**

$$\text{loss}(f(x_j), y_j) = (1 - (\mathbf{w} \cdot x_j + b)y_j))_+$$

Logistic Regression : **Log loss** (-ve log conditional likelihood)

$$\text{loss}(f(x_j), y_j) = -\log P(y_j \mid x_j, \mathbf{w}, b) = \log(1 + e^{-(\mathbf{w} \cdot x_j + b)y_j})$$



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:

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

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

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

- Derive simple gradient descent rule on α_i

Can we use kernels in regression?

Ridge regression

$$\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2 \qquad \widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y}$$

Recall

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

Hence $\mathbf{A}^T\mathbf{A}$ 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)

Using dual formulation, we will write the solution in terms of $\mathbf{A}\mathbf{A}^T$

Ridge regression

$$\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2 \qquad \widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y}$$

Similarity with SVMs

Primal problem:

$$\min_{\beta, z_i} \sum_{i=1}^{n} z_i^2 + \lambda\|\beta\|_2^2$$
$$\text{s.t. } z_i = Y_i - X_i\beta$$

SVM Primal problem:

$$\min_{w, \xi_i} C\sum_{i=1}^{n} \xi_i + \frac{1}{2}\|w\|_2^2$$
$$\text{s.t. } \xi_i = \max(1 - Y_i X_i\, w, 0)$$

Lagrangian:

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

α_i – Lagrange parameter, one per training point

Ridge regression (dual)

$$\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2 \qquad \widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y}$$

Dual problem:

$$\max_{\alpha} \min_{\beta, z_i} \sum_{i=1}^{n} z_i^2 + \lambda\|\beta\|^2 + \sum_{i=1}^{n} \alpha_i(z_i - Y_i + X_i\beta)$$

$\alpha = \{\alpha_i\}$ for i = 1,…, n

Taking derivatives of Lagrangian wrt β and z_i we get:

$$\beta = -\frac{1}{2\lambda}\mathbf{A}^\top\alpha \qquad z_i = -\frac{\alpha_i}{2}$$

Dual problem: $$\max_{\alpha} \; -\frac{\alpha^\top\alpha}{4} - \frac{1}{4\lambda}\alpha^\top\mathbf{A}\mathbf{A}^\top\alpha - \alpha^\top\mathbf{Y}$$

n-dimensional optimization problem

Ridge regression (dual)

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

$$\widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y}$$
$$= \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I})^{-1}\mathbf{Y}$$

Dual problem:

$$\max_{\alpha} \; -\frac{\alpha^\top\alpha}{4} - \frac{1}{4\lambda}\alpha^\top\mathbf{A}\mathbf{A}^\top\alpha - \alpha^\top\mathbf{Y} \qquad \Rightarrow \widehat{\alpha} = -\left(\frac{\mathbf{A}\mathbf{A}^\top}{\lambda} + \mathbf{I}\right)^{-1}_{2}\mathbf{Y}$$

can get back $\hat{\beta} = -\frac{1}{2\lambda}\mathbf{A}^\top\hat{\alpha} = \mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top + \lambda\mathbf{I})^{-1}\mathbf{Y}$

Weighted average of training points

Weight of each training point (but typically not sparse)

Kernelized ridge regression

$$\widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y}$$

Using dual, can re-write solution as:

$$\widehat{\beta} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I})^{-1}\mathbf{Y}$$

How does this help?

- Only need to invert n x n matrix (instead of p x p or m x m)
- More importantly, kernel trick!

  AA^T involves only inner products between the training points
  BUT still have an extra **A**^{T}

  Recall the predicted label is $\widehat{f}_n(X) = \mathbf{X}\widehat{\beta}$

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

XA^T contains inner products between test point **X** and training points!

Kernelized ridge regression

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

Using dual, can re-write solution as:

$$\widehat{\beta} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I})^{-1}\mathbf{Y}$$

How does this help?

- Only need to invert n x n matrix (instead of p x p or m x m)
- More importantly, kernel trick!

$$\widehat{f}_n(X) = \mathbf{K}_X(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{Y} \quad \text{where} \quad \begin{aligned} \mathbf{K}_X(i) &= \boldsymbol{\phi}(X) \cdot \boldsymbol{\phi}(X_i) \\ \mathbf{K}(i,j) &= \boldsymbol{\phi}(X_i) \cdot \boldsymbol{\phi}(X_j) \end{aligned}$$

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

Kernelized ridge regression

$$\widehat{f}_n(X) = \mathbf{K}_X(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{Y} \quad \text{where} \quad \begin{aligned} \mathbf{K}_X(i) &= \boldsymbol{\phi}(X) \cdot \boldsymbol{\phi}(X_i) \\ \mathbf{K}(i,j) &= \boldsymbol{\phi}(X_i) \cdot \boldsymbol{\phi}(X_j) \end{aligned}$$

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

Examples of kernels:

Polynomials of degree exactly 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 $K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||^2}{2\sigma^2}\right)$

Ridge Regression with (implicit) nonlinear features $\boldsymbol{\phi}(X)$! $f(X) = \phi(X)\beta$