Machine Learning

4771

Instructor: Tony Jebara

Topic 6

- Review: Support Vector Machines
- Primal & Dual Solution
- Non-separable SVMs
- Kernels
- SVM Demo

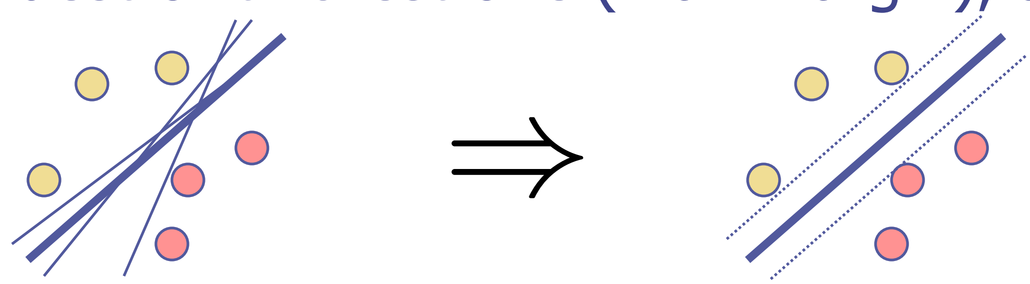
Review: SVM

- Support vector machines are (in the simplest case) linear classifiers that do structural risk minimization (SRM)
- Directly maximize margin to reduce guaranteed risk J(θ)
- Assume first the 2-class data is linearly separable:

$$have\ \left\{\left(x_1,y_1\right),\ldots,\left(x_N,y_N\right)\right\}\ \ where\ x_i \in \mathbb{R}^D\ and\ y_i \in \left\{-1,1\right\}$$

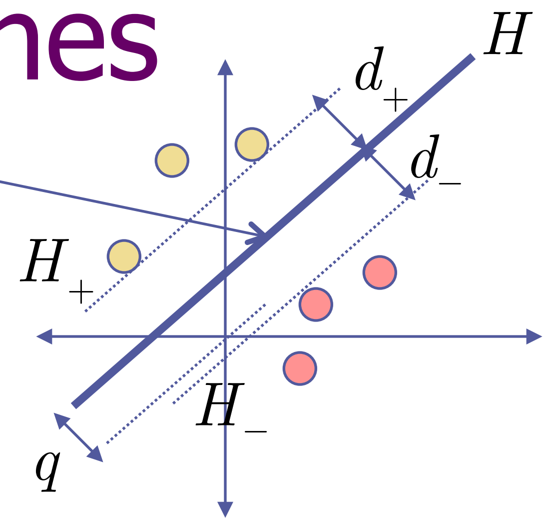
$$f\left(x;\theta\right) = sign\left(w^T x + b\right)$$

- Decision boundary or hyperplane given by $w^T x + b = 0$
- Note: can scale w & b while keeping same boundary
- Many solutions exist which have empirical error $R_{emp}(\theta)=0$
- Want widest or thickest one (max margin), also it's unique!



Support Vector Machines

- Define:



$$w^T x + b = 0$$

H_+=positive margin hyperplane
H_- =negative margin hyperplane
q =distance from decision plane to origin

$$q = \min_x \left\| \vec{x} - \vec{0} \right\| \quad subject\,to \quad w^T x + b = 0$$

$$\min_x \tfrac{1}{2} \left\| \vec{x} - \vec{0} \right\|^2 - \lambda \left(w^T x + b \right)$$

1) grad $\frac{\partial}{\partial x} \left(\frac{1}{2} x^T x - \lambda \left(w^T x + b \right) \right) = 0$

$$x - \lambda w = 0$$

$$x = \lambda w$$

2) plug into constraint

$$w^T x + b = 0$$

$$w^T \left(\lambda w \right) + b = 0$$

$$\lambda = -\frac{b}{w^T w}$$

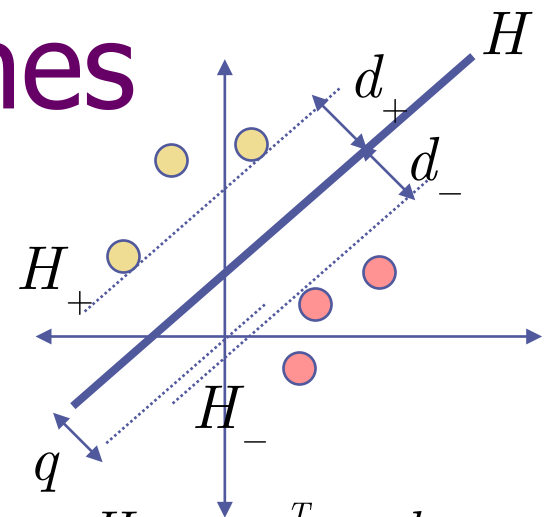
3) Sol'n $\hat{x} = -\left(\frac{b}{w^T w} \right) w$

4) distance $q = \left\| \hat{x} - \vec{0} \right\| = \left\| -\frac{b}{w^T w} w \right\| = \frac{|b|}{w^T w} \sqrt{w^T w} = \frac{|b|}{\|w\|}$

5) Define without loss of generality since can scale b & w

$$H \rightarrow w^T x + b = 0$$
$$H_+ \rightarrow w^T x + b = +1$$
$$H_- \rightarrow w^T x + b = -1$$

Support Vector Machines



- The constraints on the SVM for $R_{emp}(\theta)=0$ are thus:

$$w^T x_i + b \geq +1 \quad \forall y_i = +1$$
$$w^T x_i + b \leq -1 \quad \forall y_i = -1$$

- Or more simply: $y_i\left(w^T x_i + b\right) - 1 \geq 0$

$$H_+ \to w^T x + b = +1$$
$$H_- \to w^T x + b = -1$$

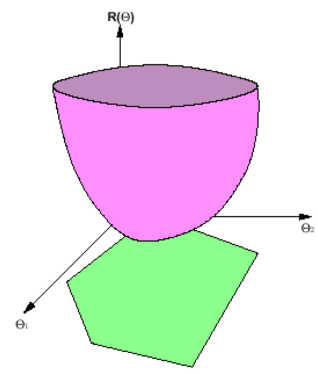
- The margin of the SVM is:

$$m = d_+ + d_-$$

- Distance to origin: $H \to q = \frac{|b|}{\|w\|}$ $\quad H_+ \to q_+ = \frac{|b-1|}{\|w\|}$ $\quad H_- \to q_- = \frac{|-1-b|}{\|w\|}$
- Therefore: $d_+ = d_- = \frac{1}{\|w\|}$ and margin $m = \frac{2}{\|w\|}$
- Want to max margin, or equivalently minimize: $\|w\|$ or $\|w\|^2$
- SVM Problem: $\min \frac{1}{2}\|w\|^2 \quad subject\, to \quad y_i\left(w^T x_i + b\right) - 1 \geq 0$
- This is a quadratic program!
- Can plug this into a matlab function called "qp()", done!

SVM in Dual Form

- We can also solve the problem via convex duality
- Primal SVM problem L_P: $\min \frac{1}{2}\|w\|^2 \;\; subject\,to \;\; y_i\left(w^T x_i + b\right) - 1 \geq 0$
- This is a quadratic program, quadratic cost function with multiple linear inequalities (these carve out a convex hull)

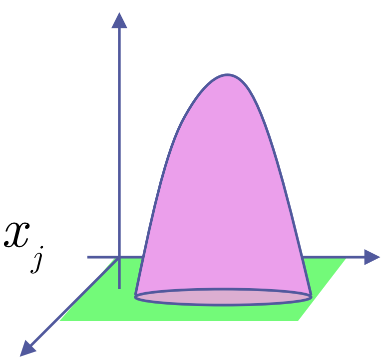


- Subtract from cost each inequality times an α Lagrange multiplier, take derivatives of w & b:

$$L_P = \min_{w,b} \max_{\alpha \geq 0} \frac{1}{2}\|w\|^2 - \sum_i \alpha_i \left(y_i\left(w^T x_i + b\right) - 1\right)$$

$$\frac{\partial}{\partial w} L_P = w - \sum_i \alpha_i y_i x_i = 0 \rightarrow w = \sum_i \alpha_i y_i x_i$$

$$\frac{\partial}{\partial b} L_P = -\sum_i \alpha_i y_i = 0$$



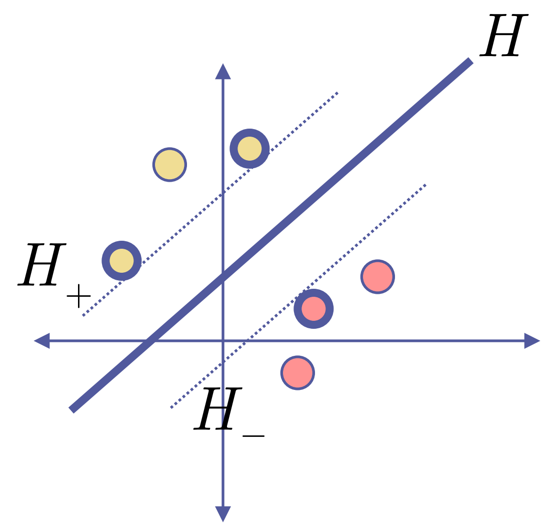
- Plug back in, dual: $L_D = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j$
- Also have constraints: $\sum_i \alpha_i y_i = 0 \;\; \& \;\; \alpha_i \geq 0$
- Above L_D must be maximized! convex duality... also qp()

SVM Dual Solution Properties

- We have dual convex program:

$$\sum_i \alpha_i - \tfrac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \quad \textit{subject to} \ \sum_i \alpha_i y_i = 0 \ \ \& \ \ \alpha_i \geq 0$$

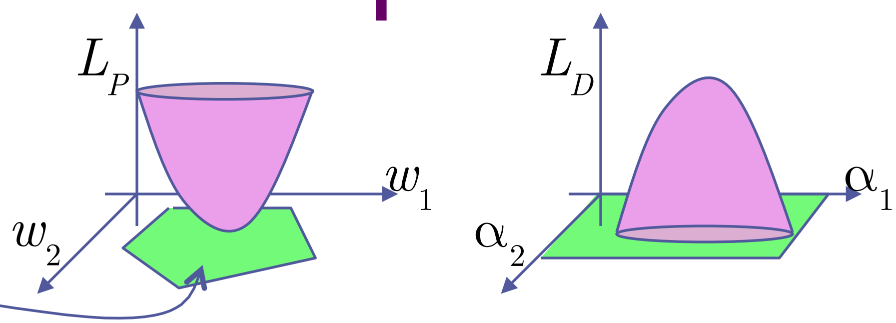
- Solve for N alphas (one per data point) instead of D w's
- Still convex (qp) so unique solution, gives alphas
- Alphas can be used to get w: $w = \sum_i \alpha_i y_i x_i$
- Support Vectors: have non-zero alphas shown with thicker circles, all live on the margin: $w^T x_i + b = \pm 1$
- Solution is sparse, most alphas=0 these are *non-support vectors* SVM ignores them if they move (without crossing margin) or if they are deleted, SVM doesn't change (stays robust)



SVM Dual Solution Properties

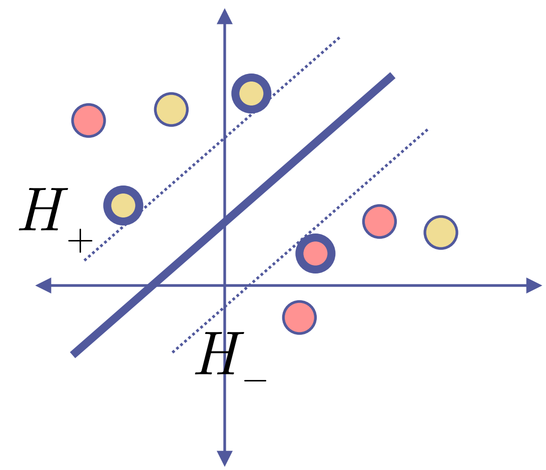
- Primal & Dual Illustration:

$w^T x_i + b \geq \pm 1$

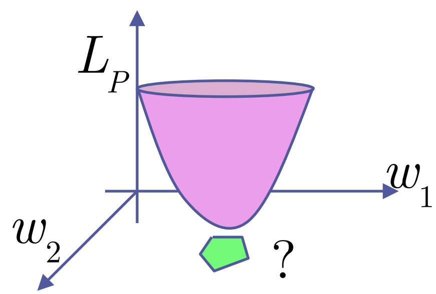


- Recall we could get w from alphas: $w = \sum_i \alpha_i y_i x_i$
- Or could use as is: $f(x) = sign(x^T w + b) = sign\left(\sum_i \alpha_i y_i x^T x_i + b\right)$
- Karush-Kuhn-Tucker Conditions (KKT): solve value of b
  on margin (for nonzero alphas) have: $w^T x_i + b = y_i$
  using known w, compute b for each support vector
  $\tilde{b}_i = y_i - w^T x_i \quad \forall i : \alpha_i > 0$ then… $b = average(\tilde{b}_i)$
- Sparsity (few nonzero alphas) is useful for several reasons
- Means SVM only uses some of training data to learn
- Should help improve its ability to generalize to test data
- Computationally faster when using final learned classifier

Non-Separable SVMs



- What happens when non-separable?
- There is no solution and convex hull shrinks to nothing



- Not all constraints can be resolved, their alphas go to ∞
- Instead of perfectly classifying each point: $y_i\left(w^T x_i + b\right) \geq 1$ we "Relax" the problem with (positive) slack variables xi's allow data to (sometimes) fall on wrong side, for example:

$$w^T x_i + b \geq -0.03 \quad if \;\; y_i = +1$$

- New constraints:

$$w^T x_i + b \geq +1 - \xi_i \quad if \;\; y_i = +1 \quad where\, \xi_i \geq 0$$
$$w^T x_i + b \leq -1 + \xi_i \quad if \;\; y_i = -1 \quad where\, \xi_i \geq 0$$

- But too much xi's means too much slack, so penalize them

$$L_P : \min \tfrac{1}{2}\left\|w\right\|^2 + C\sum\nolimits_i \xi_i \quad subject\,to \;\; y_i\left(w^T x_i + b\right) - 1 + \xi_i \geq 0$$

Non-Separable SVMs

- This new problem is still convex, still qp()!
- User chooses scalar C (or cross-validates) which controls how much slack xi to use (how non-separable) and how robust to outliers or bad points on the wrong side

Large margin → **Low slack** → **On right side** → **For xi positivity** →

$$L_P : \min \tfrac{1}{2}\left\|w\right\|^2 + C\sum_i \xi_i - \sum_i \alpha_i \left(y_i\left(w^T x_i + b\right) - 1 + \xi_i\right) - \sum_i \beta_i \xi_i$$

$$\tfrac{\partial}{\partial b} L_P \; and \; \tfrac{\partial}{\partial w} L_P \; as \, before...$$

$$\tfrac{\partial}{\partial \xi_i} L_P = C - \alpha_i - \beta_i = 0$$

$$\alpha_i = C - \beta_i \;\; but... \; \alpha_i \,\&\, \beta_i \geq 0$$

$$\therefore 0 \leq \alpha_i \leq C$$

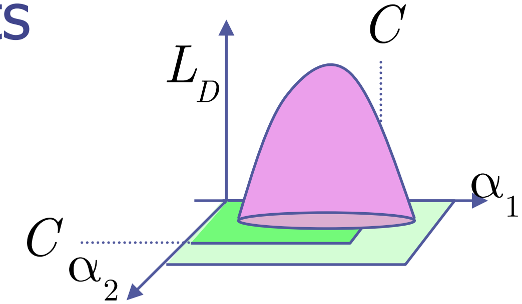
- Can now write dual problem (to maximize):

$$L_D : \max \sum_i \alpha_i - \tfrac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \;\; subject\, to \sum_i \alpha_i y_i = 0 \;\; and \;\; \alpha_i \in \left[0, C\right]$$

- Same dual as before but alphas can't grow beyond C

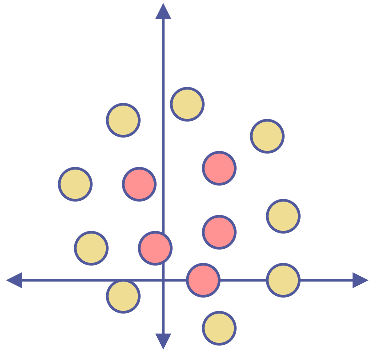
Non-Separable SVMs

- As we try to enforce a classification for a data point its Lagrange multiplier alpha keeps growing endlessly
- Clamping alpha to stop growing at C makes SVM "give up" on those non-separable points
- The dual program is now:
- Solve as before with extra constraints that alphas positive AND less than C… gives alphas… from alphas get $w = \sum_i \alpha_i y_i x_i$



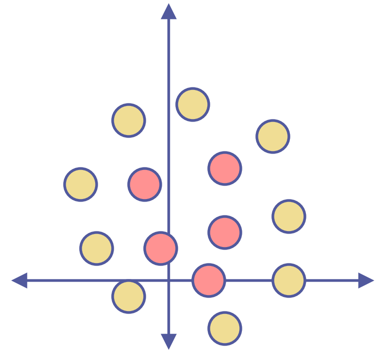
- Karush-Kuhn-Tucker Conditions (KKT): solve value of b on margin for not=zero alphas AND not=C alphas for all others have support vectors, assume $\xi_i = 0$ and use formula $y_i\left(w^T x_i + \tilde{b}_i\right) - 1 + \xi_i = 0$ to get $\tilde{b}_i$ and $b = average\left(\tilde{b}_i\right)$
- Mechanical analogy: support vector forces & torques

Nonlinear SVMs



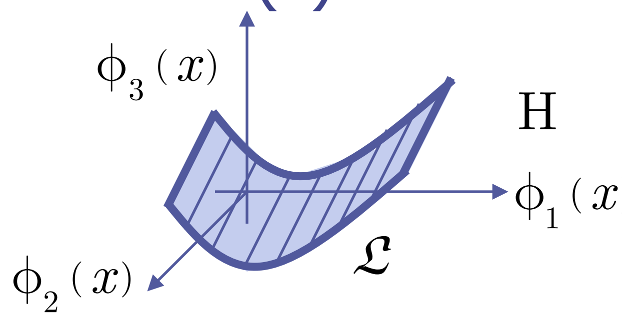
- What if the problem is not linear?

Nonlinear SVMs



- What if the problem is not linear?
- We can use our old trick…
- Map d-dimensional x data from L-space to high dimensional H (Hilbert) feature-space via basis functions $\Phi(x)$
- For example, quadratic classifier:

$$x_i \rightarrow \Phi(x_i) \quad via \quad \Phi(\vec{x}) = \begin{bmatrix} \vec{x} \\ vec\left(\vec{x}\vec{x}^T\right) \end{bmatrix}$$



- Call phi's feature vectors computed from original x inputs
- Replace all x's in the SVM equations with phi's
- Now solve the following learning problem:

$$L_D : \max \sum_i \alpha_i - \tfrac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \quad s.t. \ \alpha_i \in [0,C], \sum_i \alpha_i y_i = 0$$

- Which gives a nonlinear classifier in original space:

$$f(x) = sign\left(\sum_i \alpha_i y_i \phi(x)^T \phi(x_i) + b\right)$$

Kernels (see http://www.youtube.com/watch?v=3liCbRZPrZA)

- One important aspect of SVMs: all math involves only the *inner products* between the phi features!

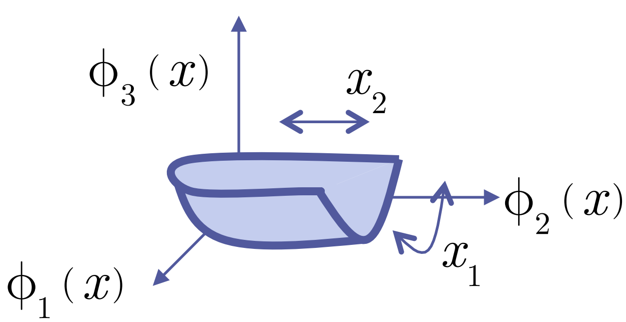
$$f(x) = sign\left(\sum_i \alpha_i y_i \phi(x)^T \phi(x_i) + b\right)$$

- Replace all inner products with a general kernel function
- Mercer kernel: accepts 2 inputs and outputs a scalar via:

$$k(x,\tilde{x}) = \left\langle \phi(x), \phi(\tilde{x}) \right\rangle = \begin{cases} \phi(x)^T \phi(\tilde{x}) & \textit{for finite } \phi \\ \int_t \phi(x,t)\phi(\tilde{x},t)\,dt & \textit{otherwise} \end{cases}$$

- Example: quadratic polynomial

$$\phi(x) = \begin{bmatrix} x_1^2 & \sqrt{2}x_1x_2 & x_2^2 \end{bmatrix}^T$$



$$\begin{aligned} k(x,\tilde{x}) &= \phi(x)^T \phi(\tilde{x}) \\ &= x_1^2\tilde{x}_1^2 + 2x_1\tilde{x}_1x_2\tilde{x}_2 + x_2^2\tilde{x}_2^2 \\ &= \left(x_1\tilde{x}_1 + x_2\tilde{x}_2\right)^2 \end{aligned}$$

Kernels

- Sometimes, many $\Phi(x)$ will produce the same k(x,x')
- Sometimes k(x,x') computable but features huge or infinite!
- Example: polynomials

If explicit polynomial mapping, feature space $\Phi(x)$ is huge

d-dimensional data, p-th order polynomial, $\dim(\mathrm{H}) = \binom{d+p-1}{p}$

images of size 16x16 with p=4 have dim(H)=183million

but can equivalently just use kernel: $k(x,y) = (x^T y)^p$

$$k(x,\tilde{x}) = (x^T\tilde{x})^p = \left(\sum_i x_i \tilde{x}_i\right)^p$$

Multinomial Theorem

$$\propto \sum_r \frac{p!}{r_1!r_2!r_3!\ldots(p-r_1-r_2-\ldots)!} x_1^{r_1}x_2^{r_2}\cdots x_d^{r_d}\tilde{x}_1^{r_1}\tilde{x}_2^{r_2}\cdots\tilde{x}_d^{r_d}$$

w=weight on term

$$\propto \sum_r \left(\sqrt{w_r}\, x_1^{r_1}x_2^{r_2}\cdots x_d^{r_d}\right)\left(\sqrt{w_r}\, \tilde{x}_1^{r_1}\tilde{x}_2^{r_2}\cdots\tilde{x}_d^{r_d}\right)$$

$$\propto \phi(x)\phi(\tilde{x})$$

Equivalent!

Kernels

- Replace each $x_i^T x_j \rightarrow k\left(x_i, x_j\right)$, for example:

  P-th Order Polynomial Kernel: $k\left(x, \tilde{x}\right) = \left(x^T \tilde{x} + 1\right)^p$

  RBF Kernel (infinite!): $k\left(x, \tilde{x}\right) = \exp\left(-\frac{1}{2\sigma^2}\left\|x - \tilde{x}\right\|^2\right)$

  Sigmoid (hyperbolic tan) Kernel: $k\left(x, \tilde{x}\right) = \tanh\left(\kappa x^T \tilde{x} - \delta\right)$

- Using kernels we get generalized inner product SVM:

$$L_D : \max \sum_i \alpha_i - \tfrac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j k\left(x_i, x_j\right) \quad s.t. \;\; \alpha_i \in \left[0, C\right], \sum_i \alpha_i y_i = 0$$

$$f\left(x\right) = sign\left(\sum_i \alpha_i y_i k\left(x_i, x\right) + b\right)$$

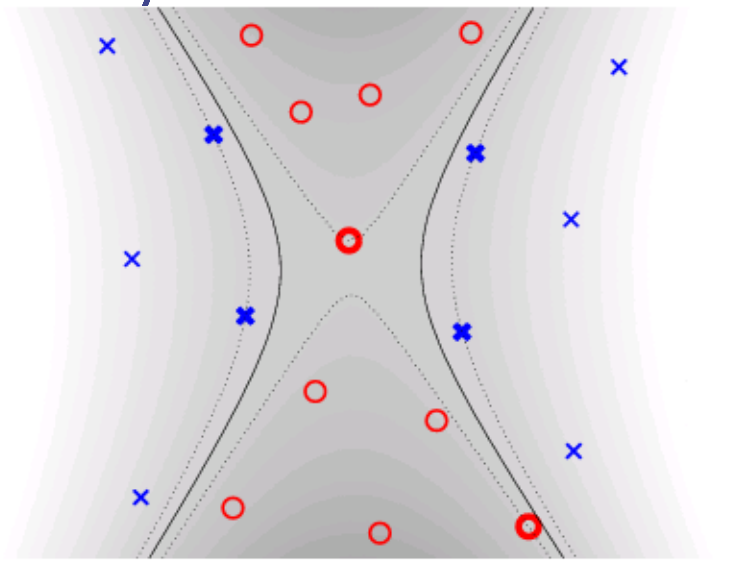
- Still qp solver, just use Gram matrix K (positive definite)

$$K = \begin{bmatrix} k\left(x_1, x_1\right) & k\left(x_1, x_2\right) & k\left(x_1, x_3\right) \\ k\left(x_1, x_2\right) & k\left(x_2, x_2\right) & k\left(x_2, x_3\right) \\ k\left(x_1, x_3\right) & k\left(x_2, x_3\right) & k\left(x_3, x_3\right) \end{bmatrix} \qquad K_{ij} = k\left(x_i, x_j\right)$$

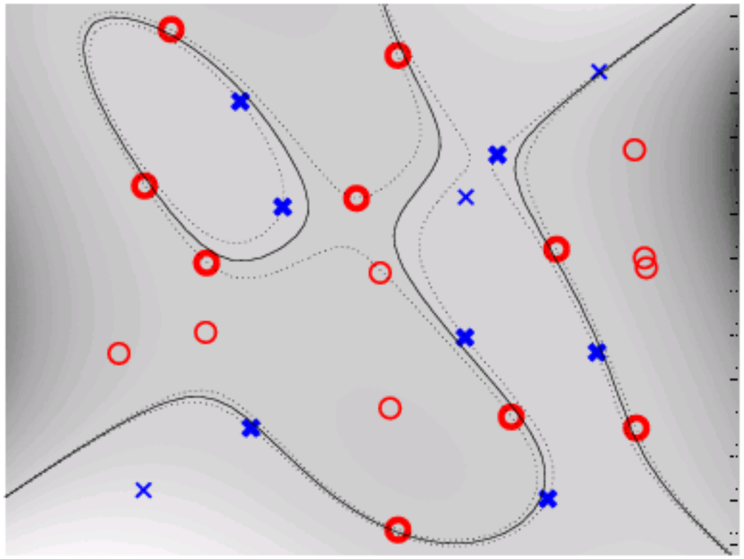
Kernelized SVMs

- Polynomial kernel: $k\left(x_i, x_j\right) = \left(x_i^T x_j + 1\right)^p$
- Radial basis function kernel: $k\left(x_i, x_j\right) = \exp\left(-\frac{1}{2\sigma^2}\left\|x_i - x_j\right\|^2\right)$

Polynomial Kernel



RBF kernel



- Least-squares, logistic-regression, perceptron are also kernelizable

SVM Demo

- SVM Demo by Steve Gunn:
  http://www.isis.ecs.soton.ac.uk/resources/svminfo/
- In svc.m replace
  [alpha lambda how] = qp(…);
  with
  [alpha lambda how] = quadprog(H,c,[],[],A,b,vlb,vub,x0);

This replaces the old Matlab command qp (quadratic programming) with the new one for more recent versions