Machine Learning

4771

Instructors:
Adrian Weller and Ilia Vovsha

Lecture 13: Support Vector Machines

- Dual Forms
- Non-Separable Data
- Support Vector Machines (Bishop 7.1, Burges Tutorial)
- Kernels

Dual Form Derivation

- Recall optimal hyperplane problem in primal space:

$$\min_{w} \frac{1}{2}\|w\|^2 \quad s.t \;\; \forall i: \; y_i(w^T x_i + b) \geq 1$$

- This is a convex program, define the Lagrangian and find stationary point:

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{\ell} \alpha_i \left[y_i(w^T x_i + b) - 1 \right], \; \alpha_i \geq 0$$

- Minimize L over {w,b}, maximize over {alphas}:

$$\frac{\partial L(w,b,\alpha)}{\partial w} = w - \sum_{i=1}^{\ell} \alpha_i y_i x_i = 0 \;\Rightarrow w = \sum_{i=1}^{\ell} y_i \alpha_i x_i$$

$$\frac{\partial L(w,b,\alpha)}{\partial b} = -\sum_{i=1}^{\ell} \alpha_i y_i = 0 \;\Rightarrow \sum_{i=1}^{\ell} y_i \alpha_i = 0$$

Dual Form

This is a convex program, define the Lagrangian and find stationary point:

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{\ell} \alpha_i \left[y_i (w^T x_i + b) - 1 \right], \ \alpha_i \geq 0$$

- Minimize L over {w,b}, maximize over {alphas}:

$$\frac{\partial L(w,b,\alpha)}{\partial w} \Rightarrow w = \sum_{i=1}^{\ell} y_i \alpha_i x_i, \quad \frac{\partial L(w,b,\alpha)}{\partial b} \Rightarrow \sum_{i=1}^{\ell} y_i \alpha_i = 0, \quad \alpha_i \geq 0$$

- Plug back into the Lagrangian and get the dual form:

$$\max_{\alpha} D(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j \left(x_i \cdot x_j \right)$$

$$s.t: \ \sum_{i=1}^{\ell} y_i \alpha_i = 0, \ \alpha_i \geq 0$$

Why Solve in Dual Space?

- QP runs in cubic polynomial time (in terms of # of variables)
- QP in primal space has complexity $O(d^3)$, where d is the dimensionality of the input vectors (weight vector)
- QP in dual space has complexity $O(N^3)$, where N is the number of examples
- More importantly: dual space yields "deeper results"

$$\min_{w} \frac{1}{2}\|w\|^2$$

$$s.t \;\; \forall i: \; y_i(w^T x_i + b) \geq 1$$

$$\max_{\alpha} D(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j)$$

$$s.t: \; \sum_{i=1}^{\ell} y_i \alpha_i = 0, \; \alpha_i \geq 0$$

$$\max_{\alpha} D(\alpha) \Rightarrow \alpha^* \Rightarrow w^* = \sum_{i=1}^{\ell} y_i \alpha_i^* x_i$$

Dual Form Properties

1. We obtain the solution vector w* from the alphas. The vector is a weighted sum of the examples, if weight (apha) is zero, the example is not "relevant"
2. Both the hyperplane and the objective of the optimization problem do not explicitly depend on the dimensionality of the vectors (only on the inner product)
3. The "contribution" from each class is equal (regardless of class size)

$$(1) \quad w^* = \sum_{i=1}^{\ell} y_i \alpha_i^* x_i$$

$$(2a) \quad f(x) = (w^*)^T x + b^* = \sum_{i=1}^{\ell} y_i \alpha_i^* (x_i \cdot x) + b^*$$

$$(2b) \quad \max_{\alpha} D(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j)$$

$$(3) \quad \sum_{i=1}^{\ell} y_i \alpha_i = 0 \quad \Rightarrow \sum_{y_i=1} \alpha_i = \sum_{y_i=-1} \alpha_i$$

Support Vectors

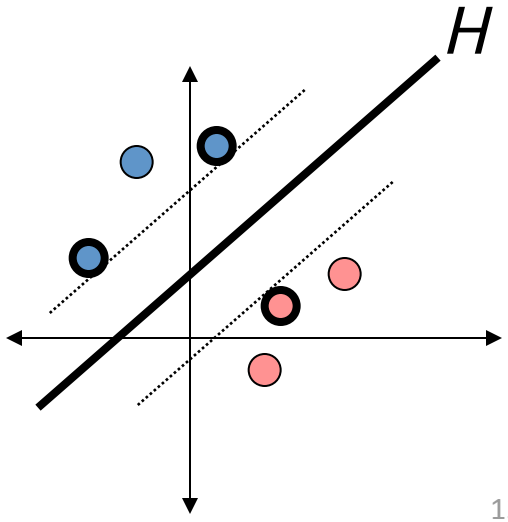
1. The value of b* is chosen to maximize the margin. The optimal values of {w*,b*} must satisfy the KKT conditions
2. From (1), we can conclude that nonzero alphas correspond to vectors that satisfy the equality constraint and hence are the closest to the optimal hyperplane. We call them *support vectors* (SVs)
3. The optimal hyperplane is unique, but the expansion on the SVs is not
4. To compute the optimal value of b: consider b for each SV and average the values to smooth out numerical errors

$$(1) \quad \forall i: \alpha_i^* \left[y_i((w^*)^T x_i + b^*) - 1 \right] = 0$$

$$(2) \quad \alpha_i^* > 0 \quad \Rightarrow \quad y_i((w^*)^T x_i + b^*) = 1$$

$$(4) \quad \forall i: \alpha_i > 0, \ (w^T x_i + b) = y_i$$

$$\Rightarrow \quad \hat{b}_i = y_i - w^T x_i \Rightarrow \ b = avg\{\hat{b}_i\}$$

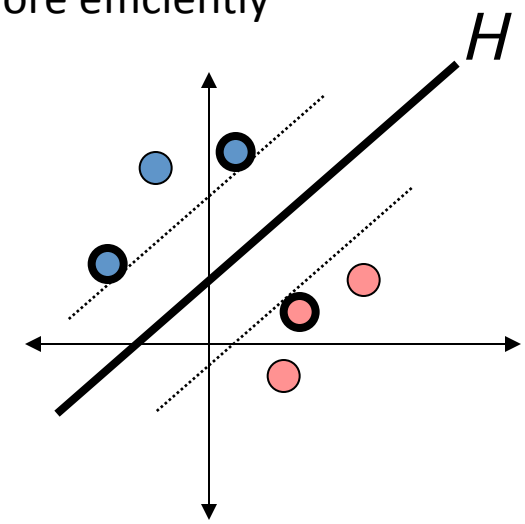


Sparse Solution

- If most alphas are zero, the solution is "sparse"
- Sparsity is useful for several reasons:
    1. Examples with zero alphas are non-support vectors that can be ignored at test time (computationally faster)
    2. Only some of the data is "relevant" to the learning machine
    3. Few SVs → easier problem → better generalization
    4. Given large amounts of data, can do optimization more efficiently

$$w^* = \sum_{i=1}^{\ell} y_i \alpha_i^* x_i$$

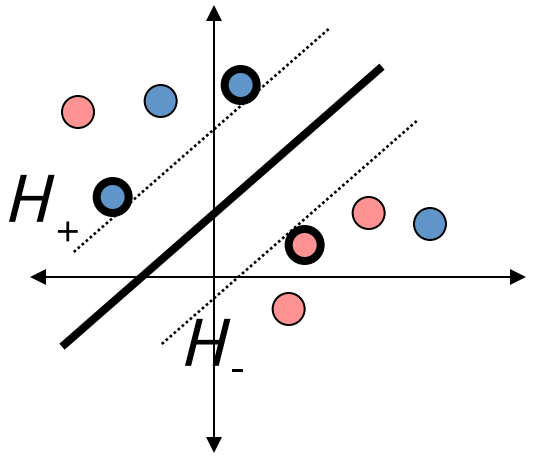
$$f(x) = (w^*)^T x + b^* = \sum_{i=1}^{\ell} y_i \alpha_i^* (x_i \cdot x) + b^*$$



Non-Separable Sets

- What happens if the data is (linearly) non-separable?
- There is no solution, since not all constraints can be resolved, the corresponding alphas go to infinity
- Idea: instead of perfectly classifying each point, "relax" the problem by introducing non-negative *slack* variables xi's to allow mistakes (but minimize mistakes)
- Instead of hard margin, we get *soft-margin formulation*
- New set of constraints:

$$\forall i:\ y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$



Δ-Margin Separating Hyperplanes

• To construct the delta-margin separating hyperplane for linearly non-separable case we consider the following problem:

$$\min F(\xi) = \sum_{i=1}^{\ell} \xi_i$$

$$s.t \ \ \forall i: \ y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

$$\|w\|^2 \leq \frac{1}{\Delta^2}$$

• Why this particular form? Recall:

$$\text{margin} = \Delta = \frac{|f(x) = \pm 1|}{\|w\|} = \frac{1}{\|w\|}$$

$$h \leq \min\left\{ \left\lceil \frac{r^2}{\Delta^2} \right\rceil, \ N \right\} + 1, \quad r = \max_i \|x_i\|$$

SOCP Primal Form (soft-margin)

• Note: we are doing SRM. Effectively we are constructing a structure (each element includes all hyperplanes with margin of at least some value delta)

• We need to specify delta (or a range of deltas) and somehow pick the best one

• For each delta we would solve a *second order cone program* (SOCP) since the additional constraint on the norm of w is a *second order cone constraint*

• It might be simpler computationally to consider an equivalent QP (actually its not clear if the SOCP is more "expensive")

$$\min F(\xi) = \sum_{i=1}^{\ell} \xi_i$$

$$s.t \ \ \forall i: \ y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

$$\|w\|^2 \leq \frac{1}{\Delta^2}$$

QP Primal Form (soft-margin)

• Another approach: modify the primal form for the linearly-separable case by adding slack variables

$$\min_{w,\xi} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}\xi_i$$

$$s.t \quad \forall i: \ y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

• The penalty parameter (C) penalizes each mistake and balances empirical error and capacity (*regularization parameter*)

•We need to specify C (or a range of Cs) and somehow pick the best one. For each C we need to solve a QP

Soft-Margin Dual Form

• To obtain the dual from the primal, we follow the same derivation procedure we used in the separable (hard-margin) case (define Lagrangian, find saddle point, plug back in):

$$\min_{w,\xi} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}\xi_i \qquad \max_{\alpha} D(\alpha) = \sum_{i=1}^{\ell}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j \left(x_i \cdot x_j\right)$$

$$s.t \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \qquad s.t: \ \sum_{i=1}^{\ell} y_i \alpha_i = 0, \ \ 0 \leq \alpha_i \leq C$$

$$\xi_i \geq 0$$

• The dual is almost identical to the one for the separable case, but now the alphas cannot grow beyond the upper bound C

Soft-Margin Properties

• As we try to enforce a classification for a data point its Lagrange multiplier alpha keeps growing.

• Clamping alpha to stop growing at C makes the machine "give up" on those non-separable points (mistakes)

• Mechanical analogy: support vector forces & torques

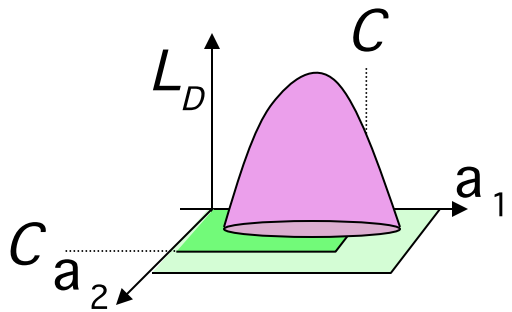
• The optimal values of {w*,b*} must satisfy the KKT conditions.

• To compute the optimal value of b: consider b for each SV whose alpha value < C (*unbounded support vectors*) and average the values to smooth out numerical errors

$$0 < \alpha_i^* < C \quad \Rightarrow \quad \xi_i = 0$$

$$\Rightarrow y_i((w^*)^T x_i + b^*) = 1$$

$$\Rightarrow \hat{b}_i = y_i - w^T x_i \Rightarrow \; b = avg\left\{\hat{b}_i\right\}$$



QP vs. SOCP

- Why is the QP form (primal & dual) solved by default in practice?
- Obvious answer: computational reasons (but also a great example of herd mentality)
- From optimization perspective: QPs are easier (faster) to solve than SOCPs
- However this is not necessarily the case if we use *decomposition methods* (break the problems into parts, take many small steps instead of few large ones)
- Optimal solutions sets coincide (for each delta can find a C which yields the same H)
- Observe that delta is directly related to the VC dimension. That is (potentially) delta is a more intuitive parameter to set

$$\min F(\xi) = \sum_{i=1}^{\ell} \xi_i$$

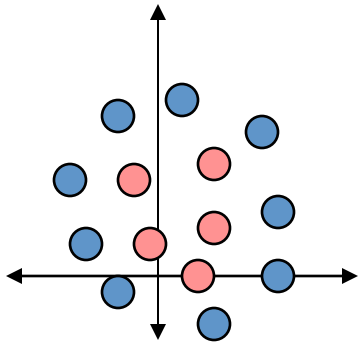
$$s.t \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \ \xi_i \geq 0$$

$$\|w\|^2 \leq \frac{1}{\Delta^2}$$

$$\min_{w,\xi} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell} \xi_i$$

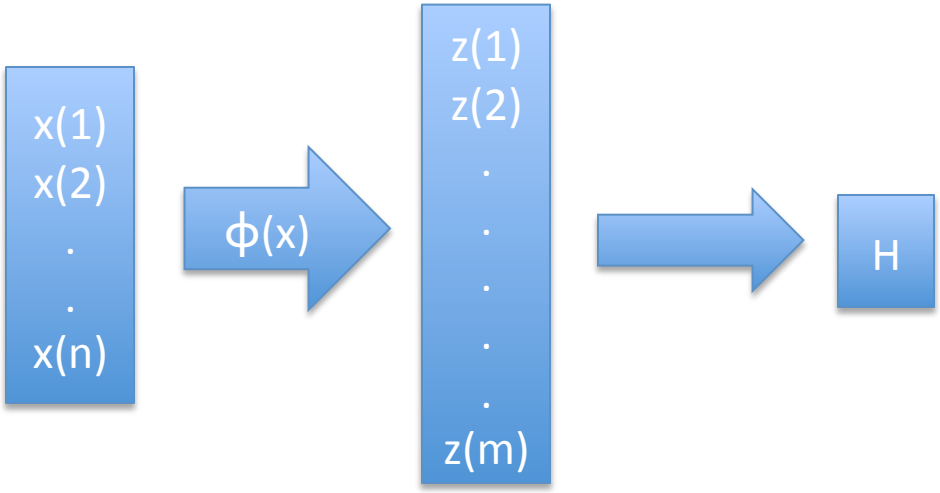
$$s.t \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \ \xi_i \geq 0$$

Support Vector Machines (idea)



- What happens if the problem is nonlinear?
- We can only use linear decision rules (problem setting)
- But using them in input space would give poor performance!
- Idea:

1. Map input vectors {x} into some high-dimensional *feature space* (Z) through some nonlinear mapping (ϕ) chosen a priori
2. In the feature space, construct an optimal (linear) hyperplane



Support Vector Machines

- Note: we have seen this idea before when we discussed regression
- The nonlinear mapping is akin to using basis functions
- Vectors in feature space are called *feature vectors*

$$x_i \rightarrow \Phi(x_i), \quad (x_i \cdot x_j) \rightarrow (\Phi(x_i) \cdot \Phi(x_j))$$

- To generalize the original problem, we replace all input vectors with feature vectors
- We obtain a nonlinear classifier in original space

$$D(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j (\Phi(x_i) \cdot \Phi(x_j))$$

$$w^* = \sum_{i=1}^{\ell} y_i \alpha_i^* \Phi(x_i)$$

$$f(x) = \sum_{i=1}^{\ell} y_i \alpha_i^* (\Phi(x_i) \cdot \Phi(x)) + b^*$$

Kernels (idea)

- So far we assumed that the nonlinear mapping is explicit i.e. we had to specify how each element of the feature vector is obtained from the input vector
- Example: quadratic polynomial $$\Phi(x) = \left[x_1^2,\ \sqrt{2}x_1x_2,\ x_2^2 \right]$$
- *Curse of dimensionality*: what if our feature space has dimensionality of one billion?
- Example: polynomials
  - Consider d-dimensional data and p-order polynomials $$\dim(H) = \binom{d+p-1}{p}$$
  - Explicit mapping yields a huge feature space
  - Example: images of size 16x16 with p=4 have dim(H)=183million
- Observe that the algorithm depends on data only *through dot products*
- We can define a *kernel function* which considers the feature space *implicitly*

Kernels

- The concept of kernel functions is old (1950's), powerful & general
- For any algorithm that depends on data only through dot (inner) products, we can replace each inner product with a general kernel function
- The kernel function defines a *similarity measure* according to which we compare each pair of examples
- An arbitrary function can be used as a kernel if it satisfies *Mercer's condition*

$$D(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j K(x_i, x_j)$$

$$w^* = \sum_{i=1}^{\ell} y_i \alpha_i^* \Phi(x_i)$$

$$f(x) = \sum_{i=1}^{\ell} y_i \alpha_i^* K(x_i, x) + b^*$$

$$x_i \rightarrow \Phi(x_i)$$

$$(x_i \cdot x_j) \rightarrow K(x_i, x_j)$$

Mercer's Condition

• ***Theorem (Mercer)***: A continuous symmetric function K(u,v) has an expansion of the form (1) below, if and only if, condition (2) is valid. We say that K(u,v) describes an inner product in some feature space

$$(1) \quad K(u,v) = \sum_{k=1}^{\infty} a(k) z_u(k) z_v(k), \ \forall k : a(k) > 0$$

$$(2) \quad \int\int K(u,v) g(u) g(v) \, du \, dv \geq 0, \ \forall g$$

• We assume that g is a "reasonable" (finite norm) function defined on the same domain as K

Example: Quadratic Polynomial

• To solve the optimization problem we need to evaluate the kernel for each pair of examples in the data set…recall objective contains a sum over all {i,j}

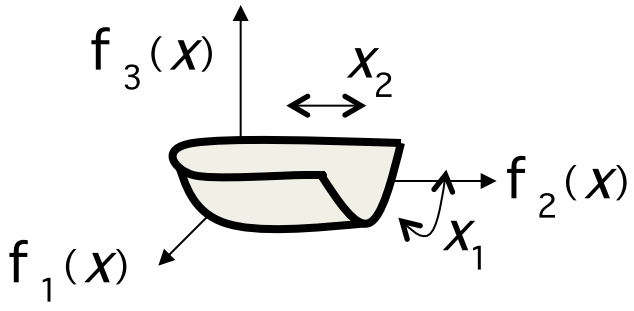
• Mercer's condition requires that the *Gram Matrix* (matrix of all pairs) is *positive semi-definite* (PSD)

• Concrete example:

$$K = \begin{bmatrix} K(x_1,x_1) & K(x_1,x_2) & K(x_1,x_3) \\ K(x_2,x_1) & K(x_2,x_2) & K(x_2,x_3) \\ K(x_3,x_1) & K(x_3,x_2) & K(x_3,x_3) \end{bmatrix}$$

$$\Phi(x) = \left[x_1^2,\ \sqrt{2}x_1x_2,\ x_2^2 \right]$$

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

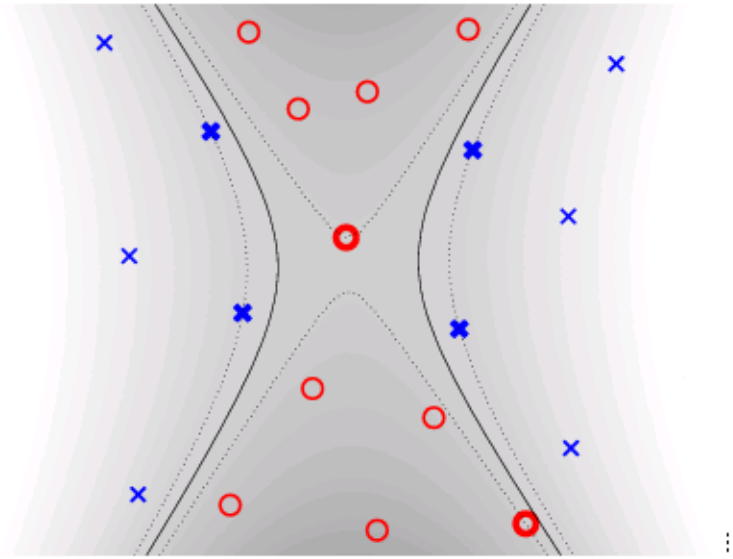


Typical Kernels

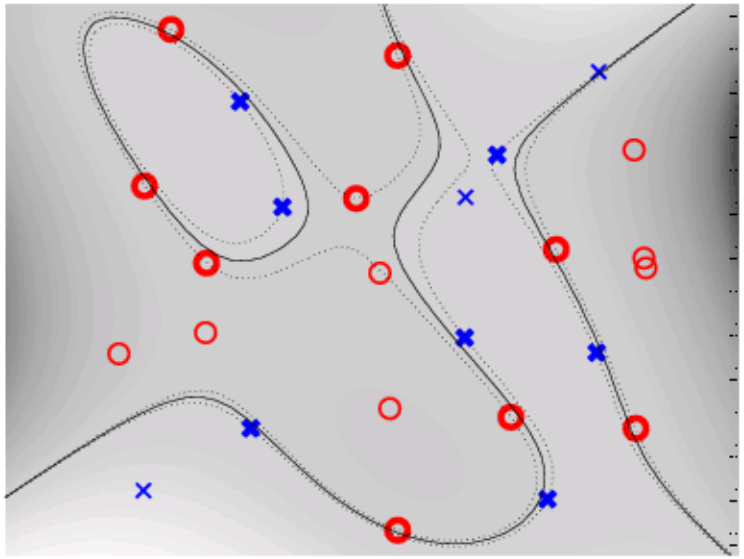
$$\text{Polynomial}: \quad K(x_i, x_j) = (x_i^T x_j + 1)^p$$

$$\text{RBF}: \quad K(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2}\|x_i - x_j\|^2\right)$$

Polynomial



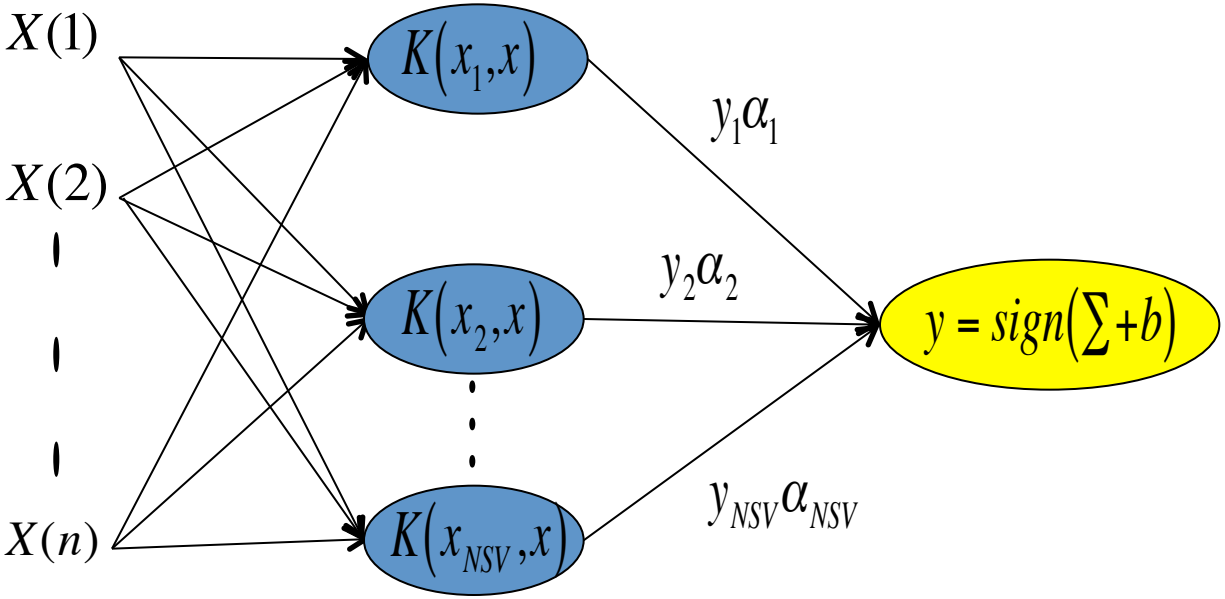
RBF



SVMs as Neural Nets

- Two-layer feed-forward neural network:
  - First layer selects the "basis" (similarity measure w.r.t to each SV)
  - Second layer constructs a linear function in the space selected by the first layer

$$f(x) = \sum_{i=1}^{\ell} y_i \alpha_i^* K(x_i, x) + b^*$$



SVMs in Practice

- Most real-world problems are non-trivial (non-separable): we must use a soft-margin formulation
- Most real-world problems are nonlinear (at least globally): we need to use kernels (radial basis function is the most popular choice)
- Soft-margin form with kernels → two parameters to tune (C & kernel parameter)
- Parameter search problem: we must specify a grid of parameters, solve a QP for each and select the best pair. We often use cross-validation for this purpose
- Note: leave-one-out CV is very efficient if we have few support vectors!
- Large scale data requires special treatment (can't store the kernel matrix in memory)
- Free software implementing decomposition methods: LIBSVM, SVM-Light
- Feature selection is not obvious when using kernels (hyperplane defined implicitly):

$$w^* = \sum_{i=1}^{\ell} y_i \alpha_i^* \Phi(x_i)$$