Kernel Methods

Which classifier is the best in practice?

- Extensive experimentation by Caruana et al 2006

- Low dimensions (9-200)

1. Boosted decision trees
2. Random forests
3. Bagged decision trees
4. SVM
5. Neural nets
6. K nearest neighbors
7. Boosted stumps
8. Decision tree
9. Logistic regression
10. Naïve Bayes

- High dimensions (500-100K)

1. HMC MLP
2. Boosted MLP
3. Bagged MLP
4. Boosted trees
5. Random forests

- We still have some distance to cover !

Classification using K nearest neighbors

- Classification Rule: Find K nearest instances; take majority label

$$p(y_i = c|x_i, D, K) = \frac{1}{K} \sum_{j \in N_k(x_j, D)} I(y_j = c)$$

- Nearness: Euclidean distance given feature vector

- Memory based learning: No training !
- Non parametric model (#params grows with data size)

KNN Decision Boundary Example

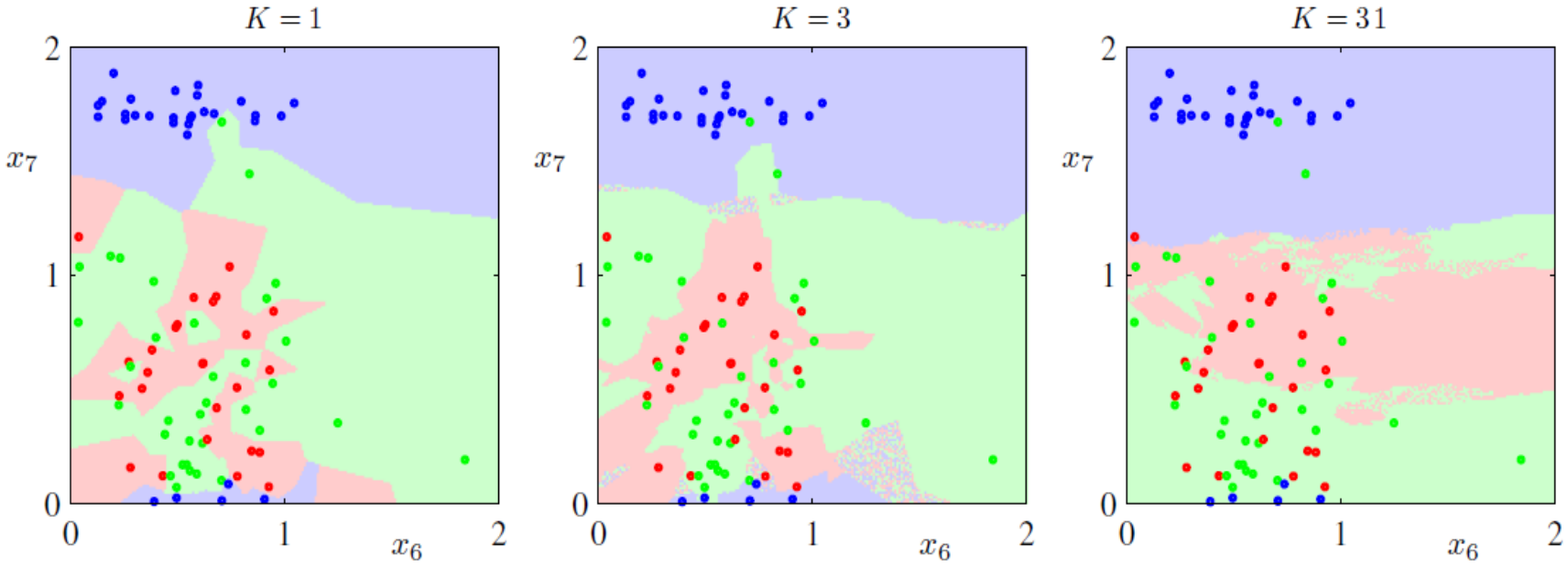


Figure from Bishop

Classification using K nearest neighbors

- Classification Rule: Find K nearest instances; take majority label

$$p(y_i = c|x_i, D, K) = \frac{1}{K} \sum_{j \in N_k(x_j, D)} I(y_j = c)$$

- Nearness: Euclidean distance given feature vector

- Possible to avoid using any representation if pairwise distances / similarities are given

Kernels: Motivation

- Assumed fixed length vector representation $\mathrm{x_i} \in R^d$
- For unstructured variable length input?
  - Text, speech, other sequences
  - Tree, molecular structure and other structured objects

1. Use domain expertise
2. Avoid feature representation and use kernels
3. Learn representation
   - Unsupervised learning, deep learning, …

Kernel: Definition

K:X×X →R

- Intuition: similarity between any inputs
- Often positive and symmetric

$$K(x, x') \geq 0 \text{ and } K(x, x') = K(x', x)$$

Kernel: Examples

- Linear Kernel $K(x, x') = x_i x_i'^T$
- RBF Kernel

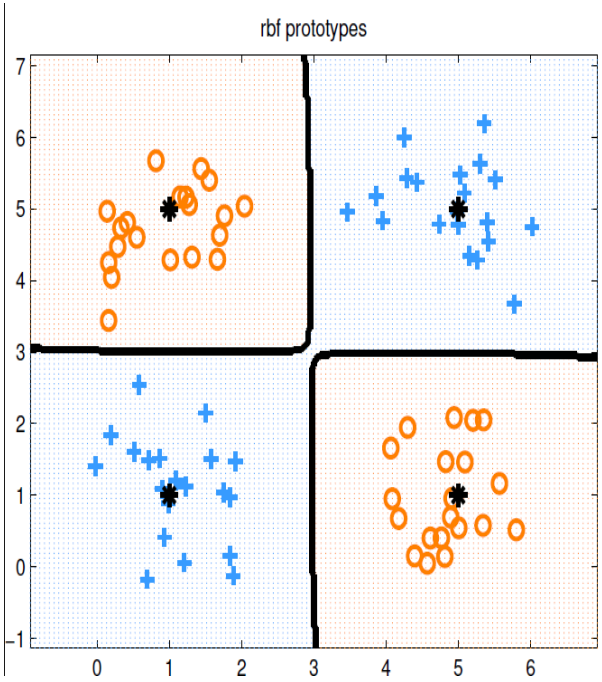
$$K(x, x') = \exp(-\frac{1}{2}(x - x')^T \Sigma^{-1}(x - x'))$$

- Document kernels: Cosine sim, TF-IDF

$$K(x, x') = \frac{x_i x'^T_i}{||x_i||_2 ||x'_i||_2}$$

- String Kernels
  - #shared substrings

Non-linear decision boundaries



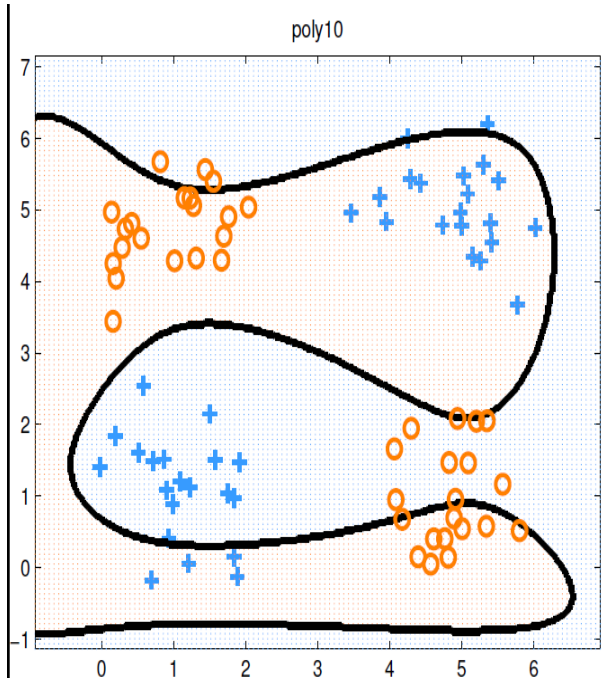


Figure from Murphy

Theory: Mercer Kernels

- Definition: Gram matrix is positive definite
- Examples: Polynomial, Gaussian, String …

  Mercer Theorem: For a Mercer Kernel K, there exists $\phi: X \to R^d$ such that $K(x, x') = \phi(x)^T \phi(x')$ for all $x, x' \in X$

- $K(x, x') = (x^T x')^2 = \phi(x)^T \phi(x')$ with $\phi(x) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)$

- Not all Kernels are Mercer Kernels …

Kernel Method: One definition

- Assumes definition of kernel $K(x, x')$
- Does not consider feature representation $\phi(x)$
- Accesses x only via $K(x, x')$

- Note $K(x, x') = \phi(x)^T \phi(x)$ for some ϕ if K is Mercer

Kernel Trick

- Simple trick to kernelize existing algorithms that are based on inner products
- Replace inner products with Kernel calls
- Explicit feature representation by-passed

K NN as a Kernel Method

1. Represent Euclidean distance computation with inner products
2. Replace inner products with kernel calls

$$\left|\left|x - x'\right|\right|_2^2 = \langle x, x\rangle + \langle x', x'\rangle - 2\langle x, x'\rangle$$
$$= K(x, x) + K(x', x') - 2\,K(x, x')$$

- Now applicable for general objects for which kernels are defined

Kernelized K Means algorithm

- Future Assignment

Kernelized Ridge Regression

- Regularized empirical risk function

$$J(w,\lambda) = \sum_i (y - \hat{y}_i)^2 + \lambda \left|\left|w\right|\right|^2$$

- Where $\hat{y}_i = w^T x_i + w_0$ $\quad \widehat{w} = X^T(XX^T + \lambda I_D)^{-1} y$

- Optimal solution

Kernelized Ridge Regression

- Regularized empirical risk function

$$J(w,\lambda) = \sum_i (y - \hat{y}_i)^2 + \lambda \left|\left|w\right|\right|^2$$

- Where $\hat{y}_i = w^T x_i + w_0$

- Optimal solution $\widehat{w} = X^T(XX^T + \lambda I_D)^{-1} y$

- Define dual variables $\alpha = (XX^T + \lambda I_D)^{-1} y$

- Learning $w = X^T \alpha = \sum_i \alpha_i x_i$

- Prediction $\hat{f}(x) = w^T x = \sum_i \alpha_i x_i^T x = \sum_i \alpha_i K(x_i, x)$

Kernel Method: Another definition

- Kernel $\phi(x) = [K(x, \mu_1), K(x, \mu_2), \ldots, K(x, \mu_k)]$

- Centroids: $\mu_1, \ldots, \mu_k \in X$

- Way to get non-linear decision boundaries

1. What is $K(x, \mu_i)$? RBF, …

2. What are the centroids?

Centroids for Kernel Machines

1. Uniformly tile space, only for low dimensions
2. Use clustering to get centroids
3. U $\phi(x) = [K(x, x_1), K(x, x_2), \ldots, K(x, x_N)]$

- Problem: High dimension -> Too many parameters
  1. Use regularization
  2. Support Vector Machines

SVM: Classical Definition

- Which of many separating hyperplanes?

- Large margin principle: Go with the largest margin

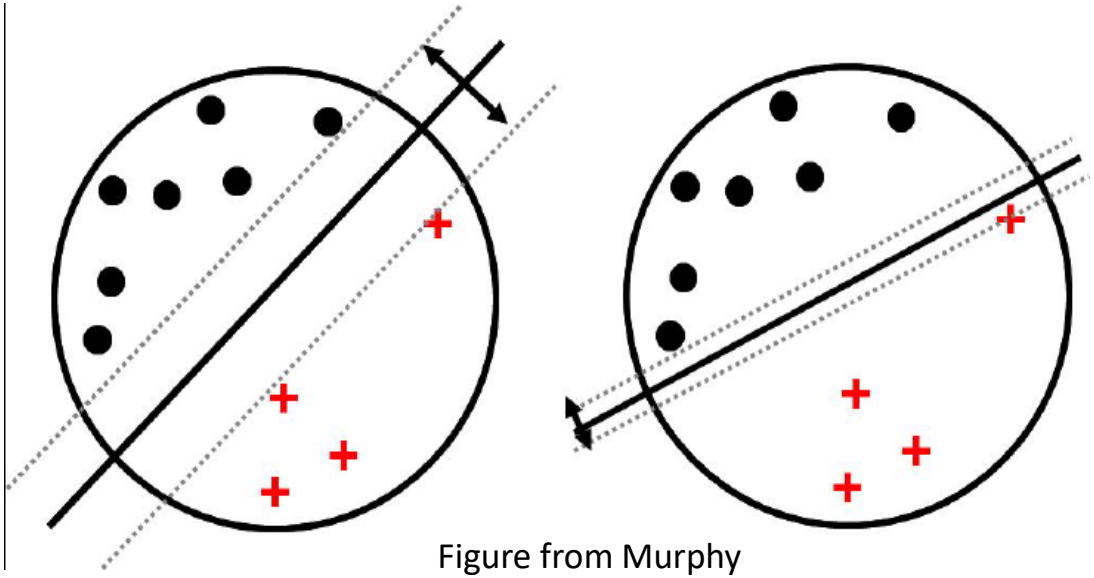


Figure from Murphy

Basic Max Margin Definition

- $y_i \in \{+1, -1\}$

1. Margin: min perpendicular distance to any instance

$$\max_{w,w_0} \min_i \frac{y_i(w^T x_i + w_0)}{||w||}, s.t. y_i(w^T x_i + w_0) > 0, \forall i = 1, \dots, N$$

2. Hyperplane must classify training data correctly

$$\min_{w,w_0} \frac{1}{2}||w||^2, s.t. y_i(w^T x_i + w_0) \geq 1 \forall i = 1, \dots, N$$

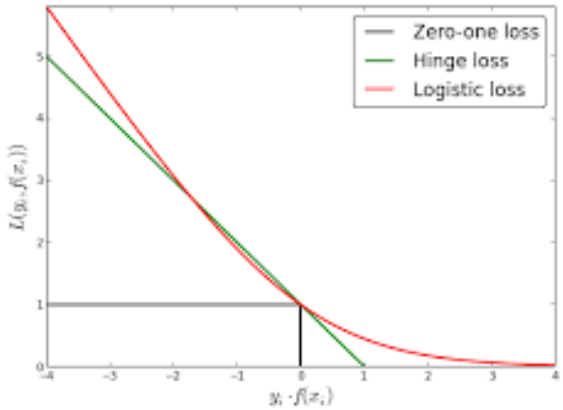
- Rescaling s.t. smallest $y_i f_i = 1$

Soft Margin Constraints

- No feasible solution if data is not linearly separable
- Trade off margin vs misclassification error

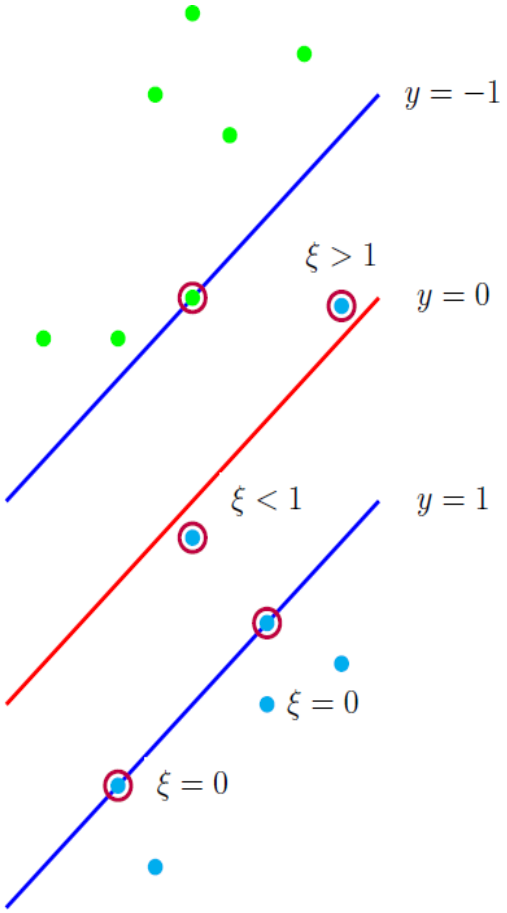
$$\min_{w,w_0} \frac{1}{2}||w||^2 + C\sum_i \left(1 - y_i f(x_i)\right)_+$$

- Hinge loss



- Difficult to optimize because of max

Soft Margin Constraints



- Relax constraints using slack variables

$$y_i f_i \geq 1 - \xi_i, s.t. 0 \leq \xi_i$$

Soft Margin Constraints

- Relax constraints using slack variables

$$y_i f_i \geq 1 - \xi_i, s.t. 0 \leq \xi_i$$

$$\min_{w, w_0, \xi} \frac{1}{2}\left|\left|w\right|\right|^2 + C\sum_i \xi_i \, , s.t. \xi_i \geq 0, y_i(w^T x_i + w_0) \geq 1 - \xi_i$$

- Quadratic program
- $O(N^3)$ solution in general

Role of Kernel

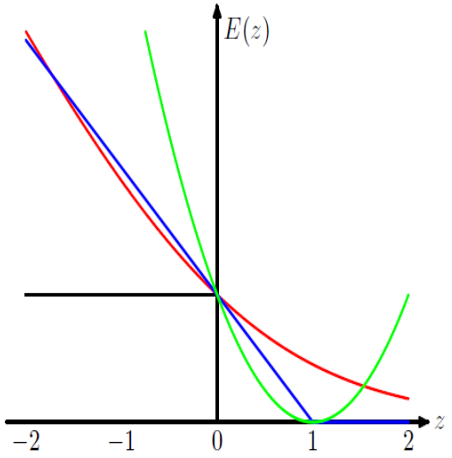
- Solution: $\hat{w} = \sum_i \alpha_i x_i$
- Where $\alpha_i = \lambda_i y_i$, where λ_i are lagrangian multiplier for constraints
- α_i is sparse because of Hinge Loss

- $\alpha_i \geq 0$: support vectors
  - Incorrectly classified
  - Correctly classified but inside the margin
- Prediction rule

$$\hat{y}(x) = sgn(f(x)) = sgn(\hat{w}_0 + \hat{w}^T x)$$
$$= sgn(\hat{w}_0 + \sum_i \alpha_i K(x, x_i))$$

SVM as Sparse Kernelized Logistic Regression

- Logistic regression uses negative log likelihood loss
- Replace with a sparsity promoting Hinge Loss



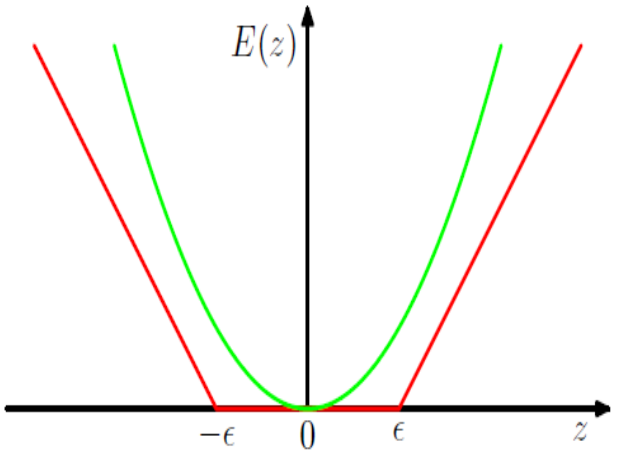
- Results in sparse solution and sparse prediction

SVM as Sparse Kernelized Ridge Regression

- Rec $\hat{f}(x) = w^T x = \sum_i \alpha_i x_i^T x = \sum_i \alpha_i K(x_i, x)$

- Kernelized but not sparse

- Use sparsity promoting loss function
- Epsilon insensitive loss
- Ensures alpha is sparse



Pros and Cons of SVMs

- Very popular
- Better accuracy than LR, NB, …
- Easily available implementations
- Many extensions

- Does not generate probabilities
- Better alternatives exist
  - Relevance vector machines produce probabilities and sparser solutions