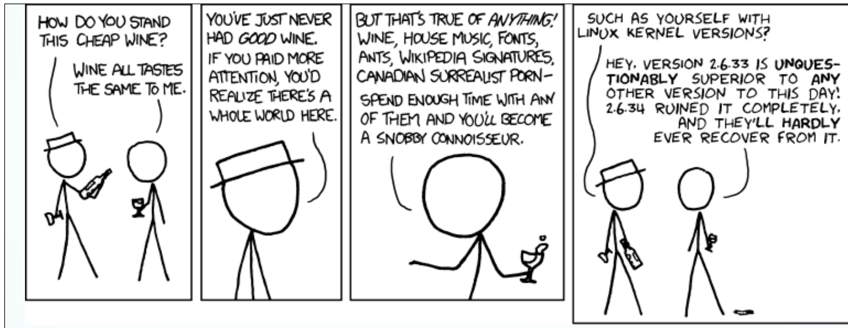


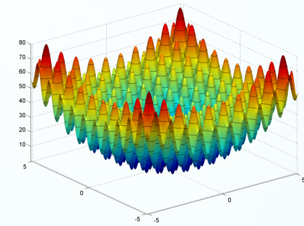
Kernel Methods

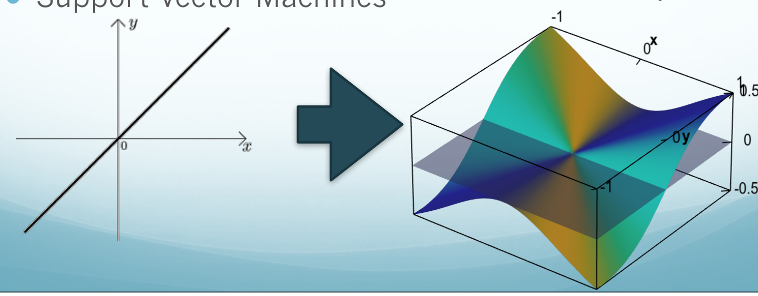
Mark Dredze

Machine Learning
CS 601.475

Linear Classification Methods

- Linear Regression $\hat{y} = \text{sign}(w \cdot x)$
- Logistic Regression
- Perceptron
- Support Vector Machines

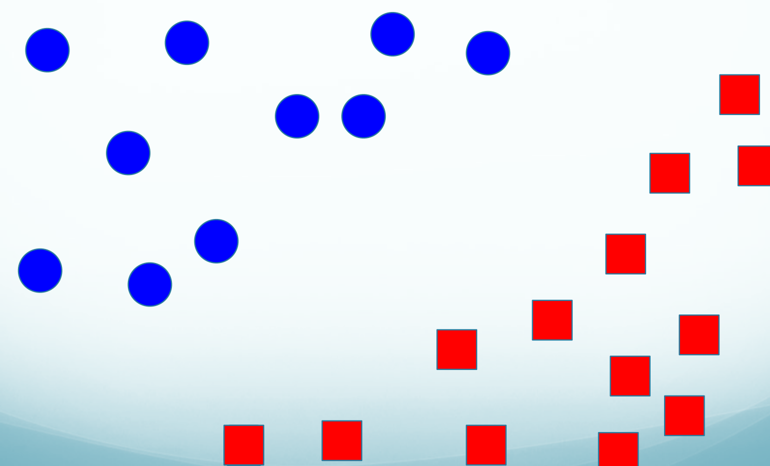




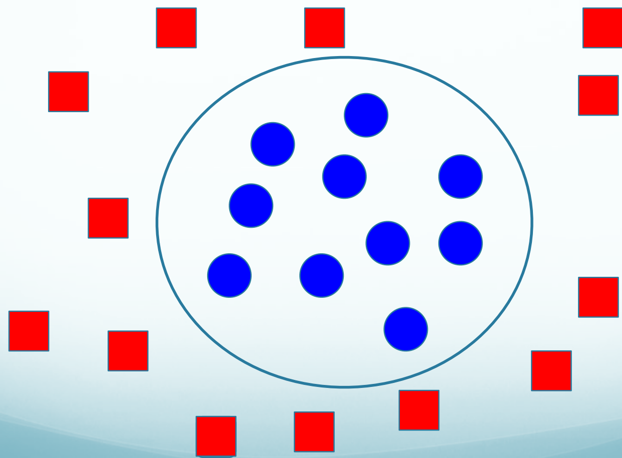
Review: Lingering Questions

- What would we do if we saw all of the data (batch)?
  - We'd pick the best separating hyperplane!
- Which separating hyperplane is the best?
  - The maximum margin separator
  - Use a quadratic regularizer on the weights
- What can we do for non-linear data?
  - It's not separable, use slack variables
  - Can we do better?

Linearly Separable



Not Linearly Separable



Handling Non-Linear Data

- Option 1: Add features by hand that make the data separable
  - Requires feature engineering
- Option 2: Learn a small number of additional features that will suffice
  - We'll see this eventually
- Option 3: Kernel trick
  - Today

Feature Mapping Functions

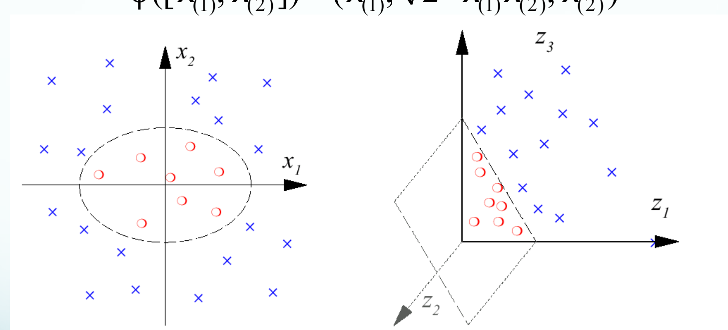
- Assuming a two dimensional vector x = [x(1),x(2)]
  - x(i) is the ith position of x
- Let's apply a feature mapping function: 2nd order polynomial function

$$\phi([x_{(1)}, x_{(2)}]) = (x_{(1)}^2, \sqrt{2} \cdot x_{(1)} x_{(2)}, x_{(2)}^2)$$

- Why is this useful?
  - Elliptical decision boundary: $\phi(x)_{(1)} + 2\phi(x)_{(3)} < 3$
  - Not linear in x, but linear in Φ(x)
  - Boundaries defined by linear combinations of feature function are ellipses, parabolas, and hyperbolas in the original space.

Geometric Interpretation

$$\phi([x_{(1)}, x_{(2)}]) = (x_{(1)}^2, \sqrt{2} \cdot x_{(1)} x_{(2)}, x_{(2)}^2)$$



$x_{(1)}^2 + 2x_{(2)}^2 < 3$ $\phi(x)_{(1)} + 2\phi(x)_{(3)} < 3$

Non-linear in x Linear in ϕ(x)

Why Feature Mapping Functions?

- Recall that to make something linearly separable I can just add a unique feature to every example
- Any dataset is linearly separable if we use enough dimensions
  - In an n-dimensional space, almost any set of up to n+1 labeled points is linearly separable!
- We can obtain linear separability by projecting data into higher dimensional spaces
  - Use smarter techniques to obtain generalizeable separability

Feature Functions + SVM

- Replace x with a feature mapping function

$$\arg\min_{w} ||w||_2^2$$
$$s.t. \quad y_i(w \cdot \phi(x_i)) \geq 1 \quad \forall i$$

- The dot product is now taken over a higher dimensional feature space
  - If ϕ is quadratic then the feature space is a quadratic space in terms of the inputs

Limitations

- We still have to learn w
  - w will grow in size of the feature space
  - e.g. quadratic kernel: |x| = 100 → |phi(x)| = 10000
- Feature functions just increase the feature space in a non-linear way
- Too limiting

SVMs and w

- Wait a minute, there is no w!

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\phi(x_i)\phi(x_j^T))$$

- There is no modeling constraint that prevents us from making $\phi(x)$ very large
- αs do not grow in the size of $\phi(x)$
- Thank you dual!

Kernels

- Let's replace $\phi(x_i)\phi(x_j^T)$ with a kernel function K

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

- where

$$x^T \cdot w = x^T \cdot \sum_{i=1}^{n}[\alpha_i y_i x_i] = \sum_{i=1}^{n}\alpha_i y_i K(x, x_i)$$

$$K(x, x') = (\phi(x)\phi(x')^T)$$

Why?

- We have removed all dependencies in the SVM on the size of the feature space
  - The feature space $\phi(x)$ appears only in the kernel
- As long as the Kernel function does the work, we can handle any feature space

Intuition About Over-Fitting

- Wait a minute!
- Assuming we project features then even using the simple projection shown so far, we'd have way to many features!
- Didn't we learn that too many features means over-fitting?

Saved by the Dual

- We aren't free to choose a parameter for each feature
- w is a linear combination of the inputs
  - We can only choose the parameters for αs
  - There are only n αs, no matter how large our feature space projection
- The inputs x put a constraint on our flexibility in high dimensional space

The Kernel Trick

- Take a linear SVM
- Substitute a non-linear kernel
- Optimize objective in the dual
- We get non-linear classification!
- Without
  - Over-fitting
  - Learning too many parameters
  - Computing a large feature space

What is a Kernel?

- A kernel is a scalar product between two high dimensional feature vectors
  - $K(x,x') = (\phi(x)\phi(x')^T)$
- A proposed kernel function can be written in this form
- We can define any mapping function and then compute the kernel

Quadractic Kernel

- Let's take the cross product of all features (quadratic)
  - $K(x,x')=(x\ x')^2$
- Why is the quadratic a valid kernel?
  - It's actually just a scalar product of the two vectors

$$K(x,x') = (x \cdot x')^2$$
$$= (x_1 x'_1 + x_2 x'_2)^2$$
$$= (x_1^2 x'^2_1 + x_2^2 x'^2_2 + 2 x_1\ x_2 x'_1\ x'_2)$$
$$= (x_1^2, x_2^2, \sqrt{2} x_1\ x_2) \cdot (x'^2_1, x'^2_2, \sqrt{2} x'_1\ x'_2)$$
$$= \Phi(x) \cdot \Phi(x')$$

- $\phi(x)$ is the feature mapping function used for the ellipse example
  - This is true for arbitrary dimensions of x

Polynomial Kernel

- In fact, this is true of any exponent p

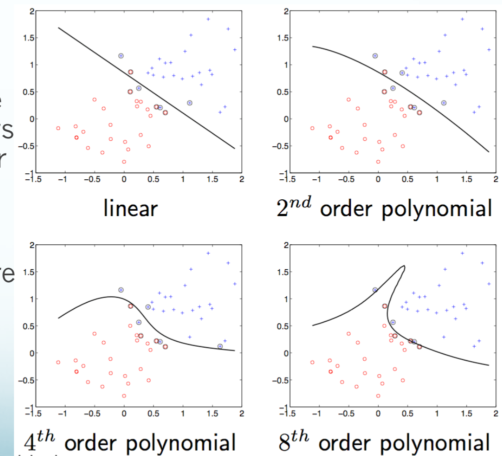
$$K(x,x') = (1 + (x^T x'))^P$$

- This is the polynomial kernel
  - To get the feature vectors we would concatenate all elements up to the *p*th order polynomial terms of the components of x (weighted appropriately)

- http://www.youtube.com/watch?v=3liCbRZPrZA

Polynomial Kernel

- In the given feature space the separators are non-linear
- In the high dimensional space they are linear
- Support vectors are circled



Pictures by Tommi Jaakkola

Decision Boundary

- How does the kernel influence the decision boundary?
- Recall prediction given by

$$x^T \cdot w = \sum_{i=1}^{n} \alpha_i y_i K(x, x_i)$$

- The larger $K(x, x_i)$ the more x_i contributes to the decision for x
  - x receives a label based on those support vectors (examples with large α) with highest $K(x, x_i)$

Similarity Function?

- Does that mean $K(x, x_i)$ is a similarity function?
  - Give same label as most similar examples
- Sort of
  - Recall: $\cos\theta = \frac{x \cdot x'}{\|x\|\|x'\|}$
  - Therefore $x \cdot x' = \|x\|\|x'\|\cos\theta$
  - So $\alpha K(x, x') = \alpha\phi(x) \cdot \phi(x') = \alpha\|\phi(x)\|\|\phi(x')\|\cos\theta$

Similarity Function?

$$\alpha K(x, x') = \alpha\phi(x)\cdot\phi(x') = \alpha\|\phi(x)\|\|\phi(x')\|\cos\theta$$

- Note
  - $\phi(x)$: constant across all x' in the prediction
  - $\alpha\phi(x')$: α is scaled per x' so this just weighs importance
  - cos θ: the angle between the vectors
    - When θ is 0, this is 1 so larger values for more similar vectors

Kernel Definitions

- A scalar product of two vectors in high dimensional space
- OR
- Mercer's theorem

Mercer's Theorem

- Suppose K is a valid kernel
- Define Kernel matrix (Gram matrix) as

$$K_{ij} = K(x_i, x_j)$$

- K must be symmetric

$$K_{ij} = K(x_i, x_j) = \phi(x_i)^T\phi(x_j) = \phi(x_j)^T\phi(x_i) = K(x_j, x_i) = K_{ji}$$

Mercer's Theorem

- $\phi_k(x)$ kth position of vector
- $$\begin{aligned} z^T K z &= \textstyle\sum_i \sum_j z_i K_{ij} z_j \\ &= \textstyle\sum_i \sum_j z_i \phi(x_i)^T \phi(x_j) z_j \\ &= \textstyle\sum_i \sum_j z_i \sum_k \phi_k(x_i)\phi_k(x_j) z_j \\ &= \textstyle\sum_k \sum_i \sum_j z_i \phi_k(x_i)\phi_k(x_j) z_j \\ &= \textstyle\sum_k \left(\sum_i z_i \phi_k(x_i)\right)^2 \\ &\geq 0 \end{aligned}$$

Mercer's Theorem

- Let K : R^M x R^M -> R be given. Then for K to be a valid (Mercer) kernel, it is necessary and sufficient that for any finite data set, the corresponding kernel matrix is symmetric positive semi-definite.

Kernel Definitions

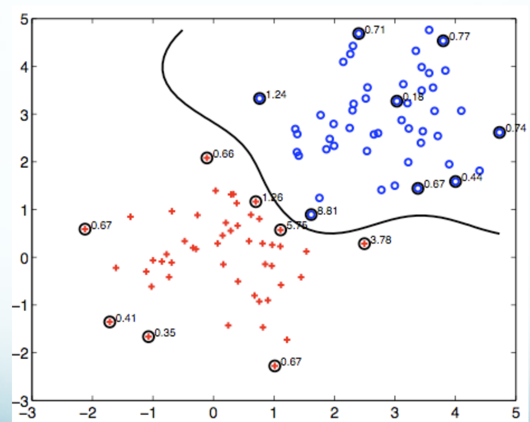
- A kernel is
  - A scalar product of two vectors in high dimensional space
  - Mercer's theorem
- How do we test a kernel without writing $\phi(x)$ explicitly?
- Equivalent definition
  - The Gram matrix **K** should be positive semidefinite for all x
    - Gram matrix **K** : $\mathbf{K}_{ij} = K(x_i,x_j)$
    - Positive semidefinite: $x^T \mathbf{M} x \geq 0$

Example of a Kernel

- Polynomial kernel $K(x,x') = (1 + (x^T x'))^P$
- Radial Basis Function (RBF) kernel
  - Gaussian version
  - Infinite dimensional function

$$K(x,x') = \exp\left(-\frac{1}{2}\|x - x'\|^2\right)$$

Radial Basis Function



Pictures by Tommi Jaakkola

Building Kernels

- How do we build a kernel?
  - Decide on a projection that is meaningful for data
- How do we know something is valid?
  - Show it's a scalar product
  - Show positive semidefinite kernel matrix
  - Best: compose new kernels from old kernels

Kernel Operations

- Many operations over kernels yield new kernels

$$K(x,x') = cK_1(x,x')$$
$$K(x,x') = f(x)K_1(x,x')f(x')$$
$$K(x,x') = \exp(K_1(x,x'))$$
$$K(x,x') = K_1(x,x') + K_2(x,x')$$
$$K(x,x') = K_1(x,x')K_2(x,x')$$

  - More examples in the book

Gaussian Kernel

- Use a Gaussian to define a kernel
  - Since this is not a probability drop the normalization

$$K(x,x') = \exp(-\|x-x'\|^2/2\sigma^2)$$

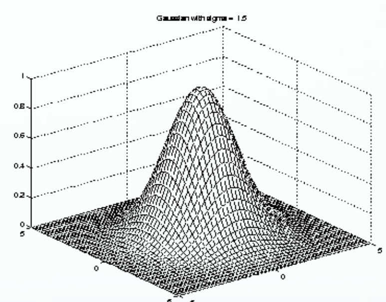
  - Why is this a valid kernel?
  - Expand the square

$$\|x-x'\|^2 = x^Tx + x'^Tx' - 2x^Tx'$$

  - Substitute

$$K(x,x') = \exp(-x^Tx/2\sigma^2)\exp(-x'^Tx'/2\sigma^2)\exp(2x^Tx'/2\sigma^2)$$
$$K(x,x') = f(x)K_1(x,x')f(x')$$
$$K(x,x') = \exp(K_1(x,x'))$$

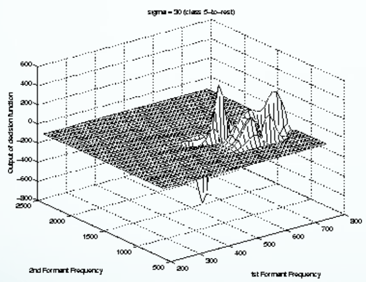
Gaussian Kernel



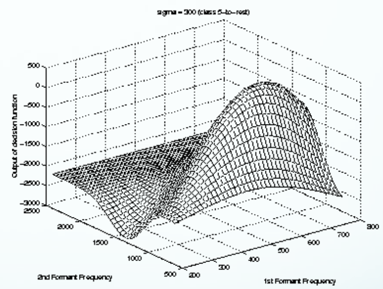
- Three dimensional Gaussian
  - σ determines the smoothness of the function
  - Large σ means the support vector has greater influence area
    - Less support vectors needed to cover boundaries

http://mi.eng.cam.ac.uk/~kkc21/thesis_main/node31.html

Decision Boundary



Smaller σ



Larger σ

Kernels for Objects

- We've talked about kernels as operating over $x \in \Re^M$
- However, we can define x as anything
  - As long as we can compute $K(x, x')$
- Kernels for
  - Strings
  - Trees/Graphs
  - Images

Kernels for Strings

- Represent a document as a feature vector
  - Each feature corresponds to a word in the document
  - Classify document based on the words
- Even better: each feature corresponds to a sub-string in the document
  - Include non-contiguous sub-strings
  - Value of feature is dependent on frequency of where it appears
- For sub-strings of size > 4 cannot compute this feature space
  - Way too many features!

Kernels for Strings

- String Subsequence Kernel

$$K(x, x') = \sum_{u \in \Sigma^d} \sum_{i:u=x[i]} \sum_{j:u=x'[j]} \lambda^{|i|+|j|}$$

  - For all string u of length d
  - For all substrings of x
  - For all substrings of x'
  - λ to the power of the size of the combined lengths
- Computing features would take $O(|\Sigma|^d)$ time
- Can compute the kernel for this feature representation using dynamic programming

Lodhi, et al. 2002

Biology: Splice Site Recognition

- Find the boundary between exons and introns in eukaryotes (complex organism)
  - What part of DNA codes for genes
- Input is a sequence of DNA base pairs
- Normally each feature indicates a substring of base pairs appearing in the sequence

Biology: Splice Site Recognition

- Each possible substring of DNA is a new feature
- Use the kernel approach as for strings
- Problem for DNA: long substrings unlikely but still informative
- Solution: a kernel from many weighted spectrum kernels

$$K_\ell(x,x') = \sum_{d=1}^{\ell} \beta_d K_d^{spectrum}(x,x')$$

Other Kernel Methods

Everything Can be Non-Linear

Kernel Perceptron

- We showed a derivation for dual Perceptron

$$\hat{y} = sign(\sum_{i=1}^{n} \alpha_i y_i x_i x)$$

  - α_i=1 if we made a mistake on round i
- Replace the dot product with a kernel

$$\hat{y} = sign(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x))$$

Kernel Linear Regression

- We can define linear regression with quadratic regularization using a linear combination of x

$$w = -\frac{1}{\lambda}\sum_{i=1}^{N}\{w^T\phi(x_i) - y_i\}\phi(x_i)$$
$$= \sum_{i=1}^{N}\alpha_i\phi(x_i)$$

- So we can get a kernel version

$$w^T\phi(x) = k(x)^T(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{Y}$$
$$k_i(x)^T = K(x_i, x)$$

Kernel Logistic Regression

- We can do the same trick with logistic regression
- Represent w in terms of x and α

$$w = \sum_{i=1}^{N}\alpha_i\phi(x_i)$$

- Insert a kernel in place of a dot product in the model

$$P(y=1 \mid x, w) = \frac{1}{1 + \exp\{-(\sum_{i=1}^{n}\alpha_i K(x, x_i) + b)\}}$$

- Derive new gradient descent rule on α

Summary

- The good
  - Arbitrarily high dimensionality
  - Extensions to other data types
  - Non-linearity in a parametric linear framework
- The bad
  - What is a good kernel?
    - Whole field on designing kernels, learning kernels
  - Cannot handle large data
    - Kernel matrix grows quadratic in N