University of Colorado **Boulder**

Department of Computer Science

CSCI 5622: Machine Learning

Chris Ketelsen

Lecture 14: Support Vector Machines
The Kernel Trick

Our Roadmap

Last Last Time:

- Talk linear SVM when data is linearly separable

Last Time:

- Talk linear SVM when data is not linearly separable
- Look at efficient algorithm for finding weights

This Time:

- Look at SVM as a nonlinear classifier
- Learn the Kernel Trick

Soft-Margin SVMs

Primal Optimization Problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i^p \\ \text{s.t.} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{for } i = 1, \ldots, m \end{aligned}$$

Dual Optimization Problem:

$$\begin{aligned} \max_{\alpha} \quad & \textstyle\sum_i \alpha_i - \tfrac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_j^T\mathbf{x}_i \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \text{ for } i = 1, \ldots, m \\ & \textstyle\sum_i \alpha_i y_i = 0 \end{aligned}$$

Soft-Margin SVMs

Dual Optimization Problem:

$$\begin{aligned} \max_{\alpha} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_j^T \mathbf{x}_i \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \text{ for } \quad i = 1, \ldots, m \\ & \sum_i \alpha_i y_i = 0 \end{aligned}$$

- Feature vectors only appear in dot products
- Solve with SMO Algorithm

Non-Linearly Separable Case

What can we do if data is clearly not linearly separable?

Non-Linearly Separable Case

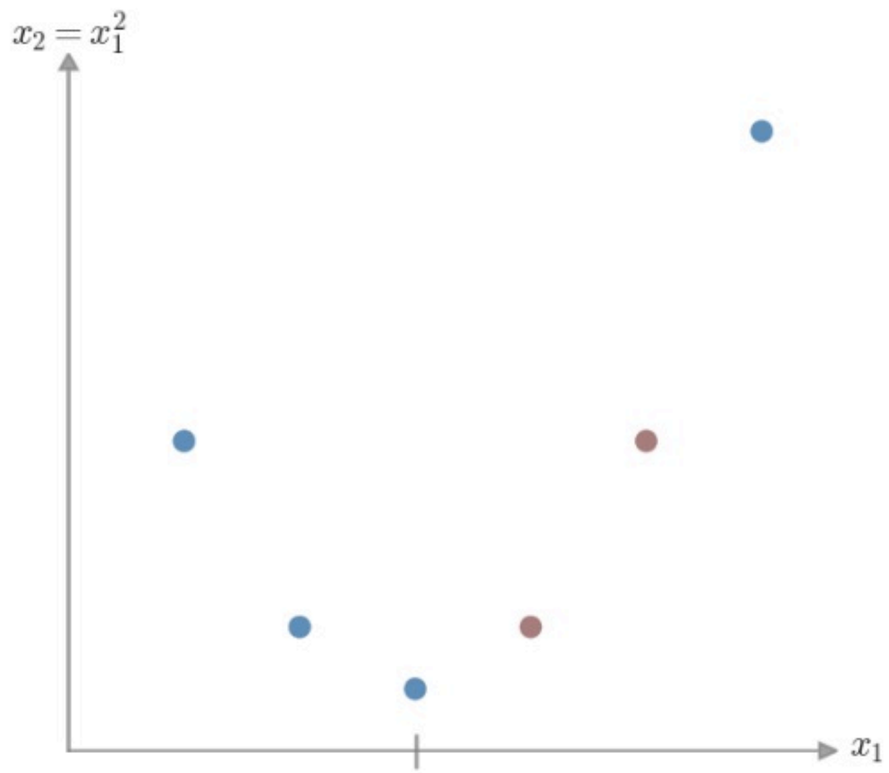
What can we do if data is clearly not linearly separable?

Non-Linearly Separable Case

Do we just have to accept some misclassification?

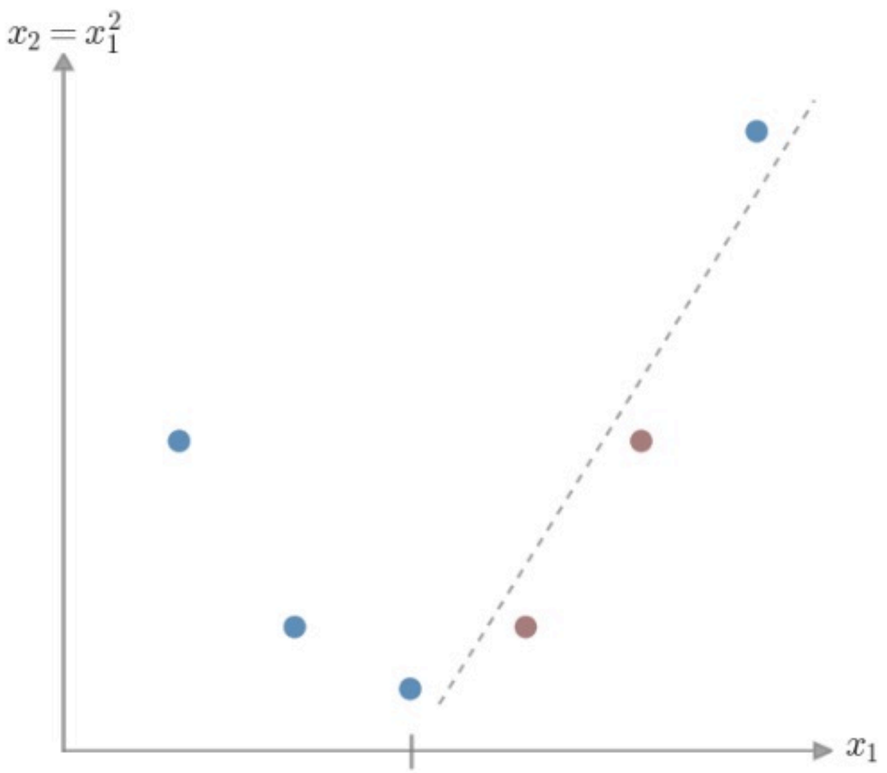
Non-Linearly Separable Case

Idea: Project into higher dimensional space



Non-Linearly Separable Case

Idea: Here, data **is** linearly separable



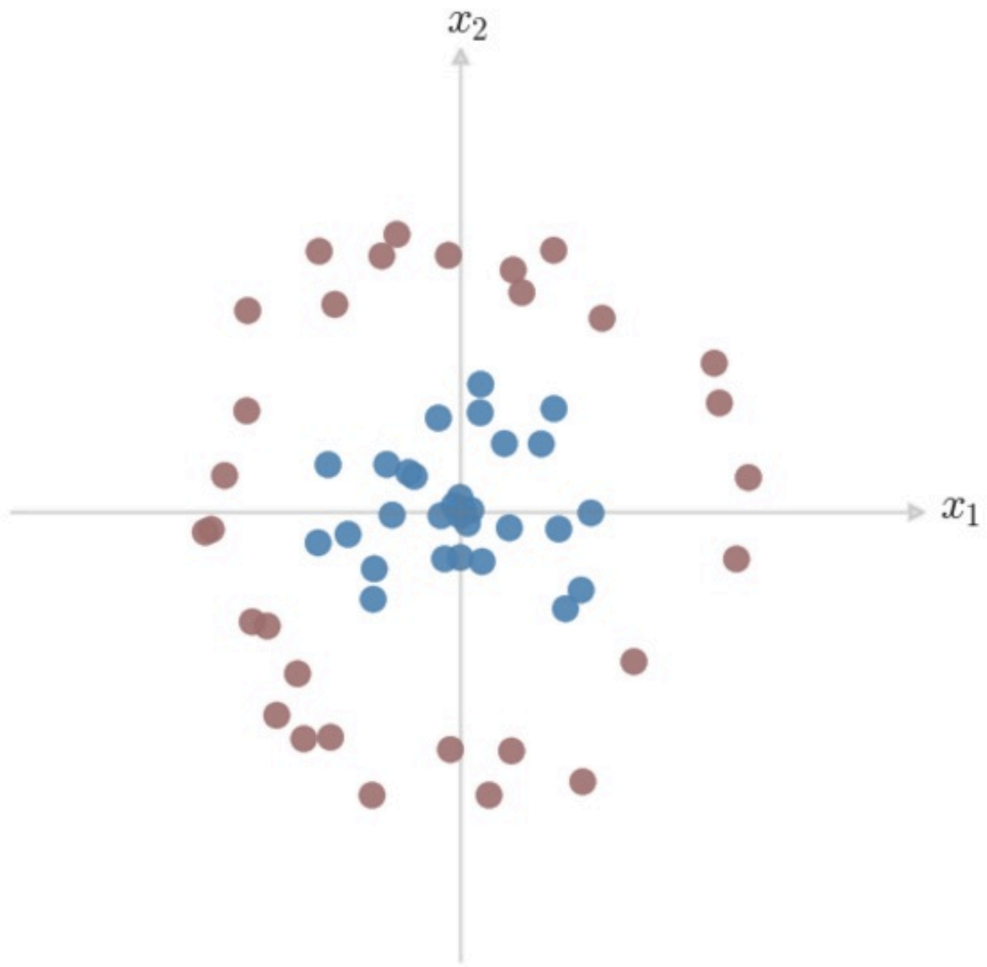
Derived Features

Started with original feature vector $\mathbf{x} = [\, x_1 \,]$

Created a new derived feature vector $\phi(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_1^2 \end{bmatrix}$

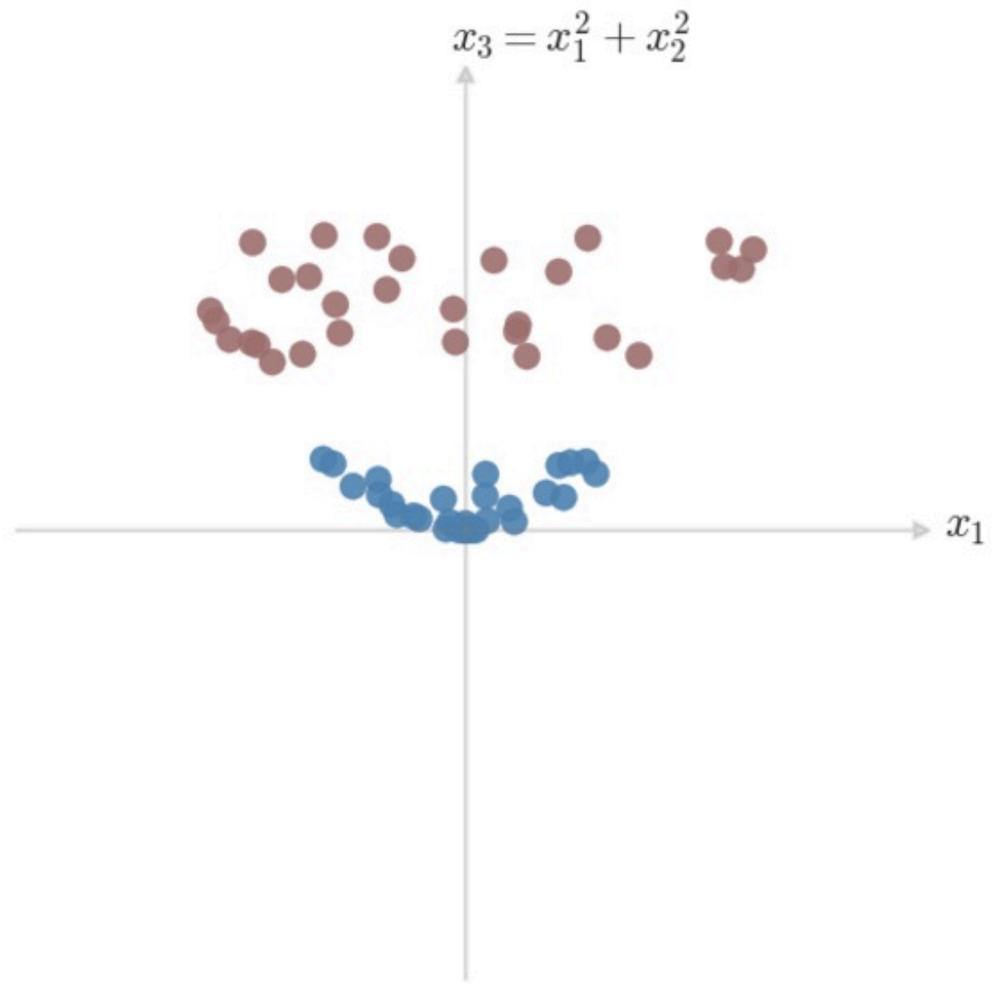
Non-Linearly Separable Case

Definitely not linearly separable in two dimensions



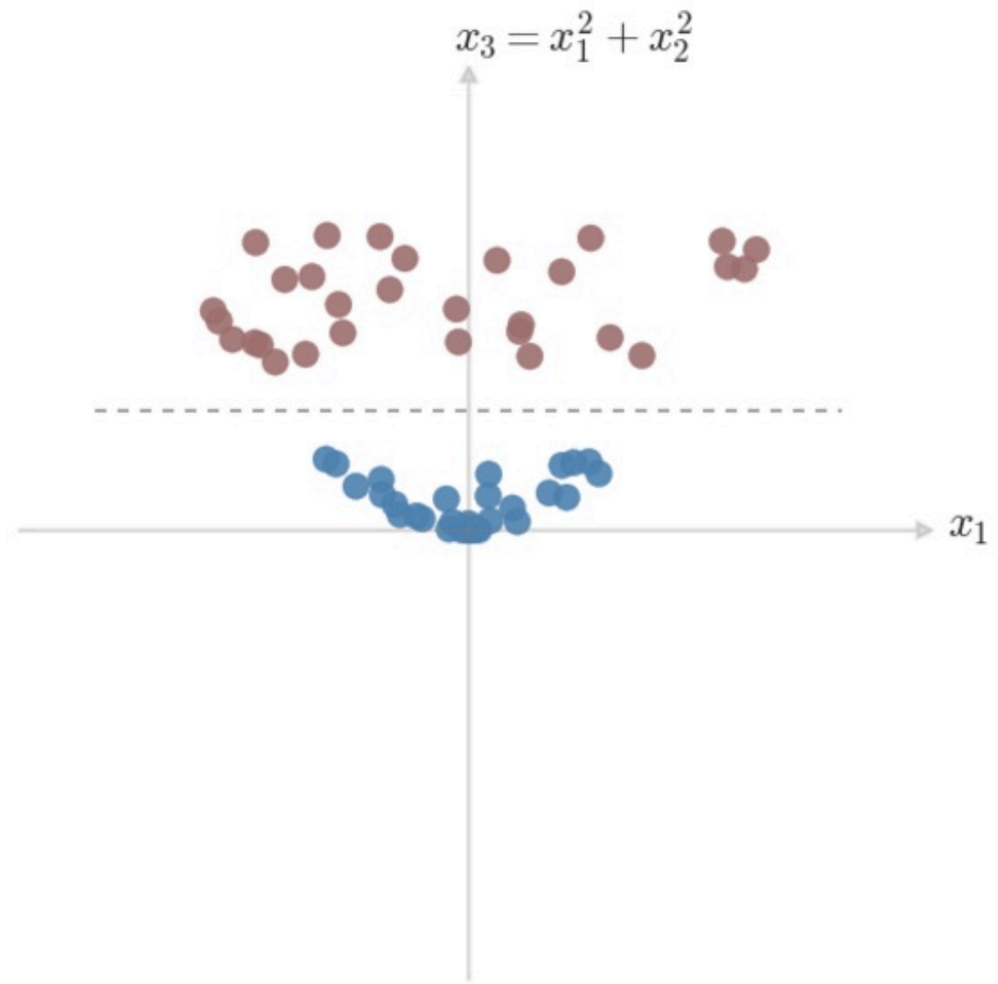
Non-Linearly Separable Case

But in three dimensions ...



Non-Linearly Separable Case

Easily linearly separable



Derived Features

Started with original feature vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

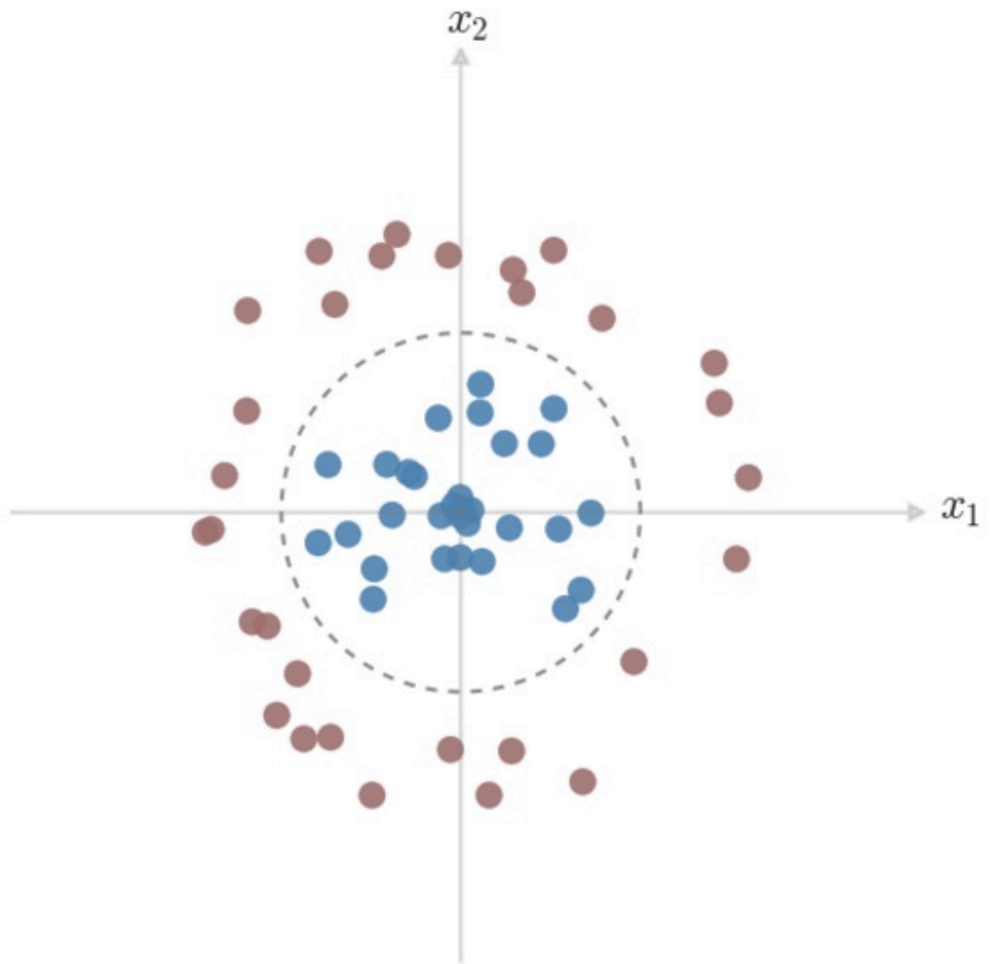
Created a new derived feature vector $\phi(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 + x_2^2 \end{bmatrix}$

Decision boundary is a plane in $\mathbb{R}^3$

In terms of the 2D features: $x_1^2 + x_2^2 = \texttt{constant}$

Derived Features

a.k.a. a circle



Derived Features and SVMs

Linear DB in Higher Dim $\Leftrightarrow$ Nonlinear DB in Lower Dim

Can actually use this trick for any linear classifier (NB, LogReg)

But something nice happens with SVMs

Remember the SVM dual problem only depends on dot products

$$\max_{\alpha} \sum_{i} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_j^T \mathbf{x}_i$$

Derived Features and SVMs

Remember the SVM dual problem only depends on dot products

$$\max_{\alpha} \sum_{i} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

Replace with dot products of derived feature vectors

$$\max_{\alpha} \sum_{i} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Replace dot products with **kernel** function $K(\mathbf{x}, \mathbf{z})$

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

Kernels and SVMs

The kernel $K(\mathbf{x}, \mathbf{z})$ is a generalization of $\phi(\mathbf{x})^T \phi(\mathbf{z})$

Think up a K that says something about relationship b/w $\mathbf{x}$ and $\mathbf{z}$

Under certain conditions on K there exists relationship

$$K(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{n} \phi_i(\mathbf{x})^T \phi_i(\mathbf{z})$$

Kernels and SVMs

Replace dot products with **kernel** function $K(\mathbf{x}, \mathbf{z})$

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

Do this everywhere in the SVM formulation, e.g.

$$\mathbf{w}^T \mathbf{x} = \sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x})$$

Can learn an SVM classifier in high dimension w/o forming $\phi(\mathbf{x})$

Kernels and SVMs

OK sounds cool, but how much does it really save us?

Example: Suppose $\mathbf{x}, \mathbf{z} \in \mathbb{R}^D$ and $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})^2$

$$\begin{aligned} K(\mathbf{x}, \mathbf{z}) &= \left(\sum_{i=1}^D x_i z_i\right)\left(\sum_{j=1}^D x_j z_j\right) \\ &= \sum_{i=1}^D \sum_{j=1}^D x_i x_j z_i z_j \\ &= \sum_{i,j=1}^D (x_i x_j)(z_i z_j) \\ &= \phi(\mathbf{x})^T \phi(\mathbf{z}) \end{aligned}$$

What do these feature vectors look like?

Kernels and SVMs

Example: Suppose $\mathbf{x}, \mathbf{z} \in \mathbb{R}^D$ and $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})^2$

For $D = 3$:

$$\phi(\mathbf{x}) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \end{bmatrix} \in \mathbb{R}^{D^2}$$

Kernels and SVMs

Example: Suppose $\mathbf{x}, \mathbf{z} \in \mathbb{R}^D$ and $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})^2$

Just forming $\phi(\mathbf{x})$ would be $\mathcal{O}(D^2)$ in time and space

But equivalent kernel evaluation is just $\mathcal{O}(D)$ in time

Around $2D$ flops for dot product and 1 flop to square

That's a lot cheaper!

Kernel Intuition

We're replacing $\mathbf{x}^T\mathbf{z}$ with $K(\mathbf{x}, \mathbf{z})$

Usual dot product $\mathbf{x}^T\mathbf{z}$ tells you about **similarity** of $\mathbf{x}$ and $\mathbf{z}$

Think about text classification

If documents $\mathbf{x}$ and $\mathbf{z}$ share many terms then $\mathbf{x}^T\mathbf{z} \gg 0$

If documents $\mathbf{x}$ and $\mathbf{z}$ share no terms then $\mathbf{x}^T\mathbf{z} = 0$

Think of a kernel function K that does something similar, but also includes nonlinear combinations of the features

Kernel Intuition

Example - Polynomial Kernel: $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T\mathbf{z} + c)^p$

Gives same idea but also includes linear and quadratic combinations of features

Example - RBF Kernel: $K(\mathbf{x}, \mathbf{z}) = \exp\left[-\gamma\|\mathbf{x} - \mathbf{z}\|^2\right]$

If $\mathbf{x} \approx \mathbf{z}$ then $K(\mathbf{x}, \mathbf{z}) \approx 1$

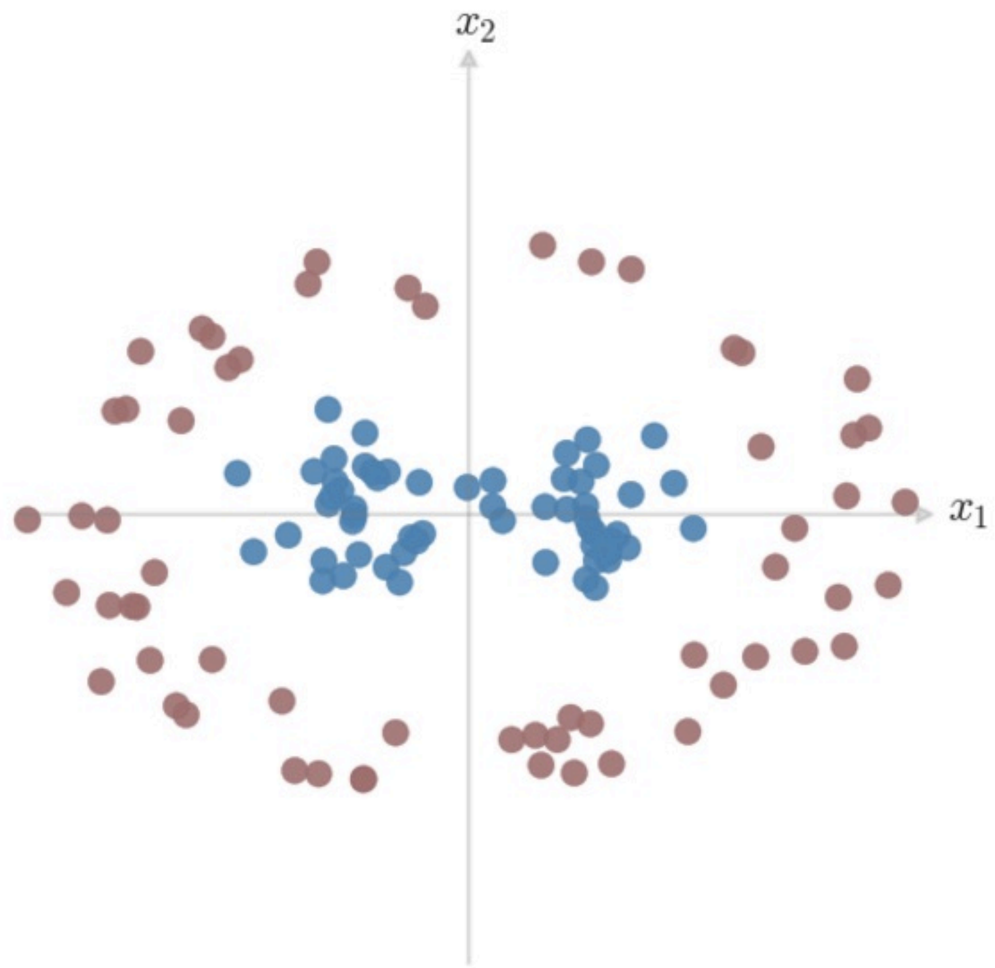
If $\mathbf{x}$ very different from $\mathbf{z}$ then $K(\mathbf{x}, \mathbf{z}) \approx 0$

γ is a tuning parameter that determines how fast similarity falls off

Incidentally: RBF Kernel is like Polynomial Kernel of infinite degree

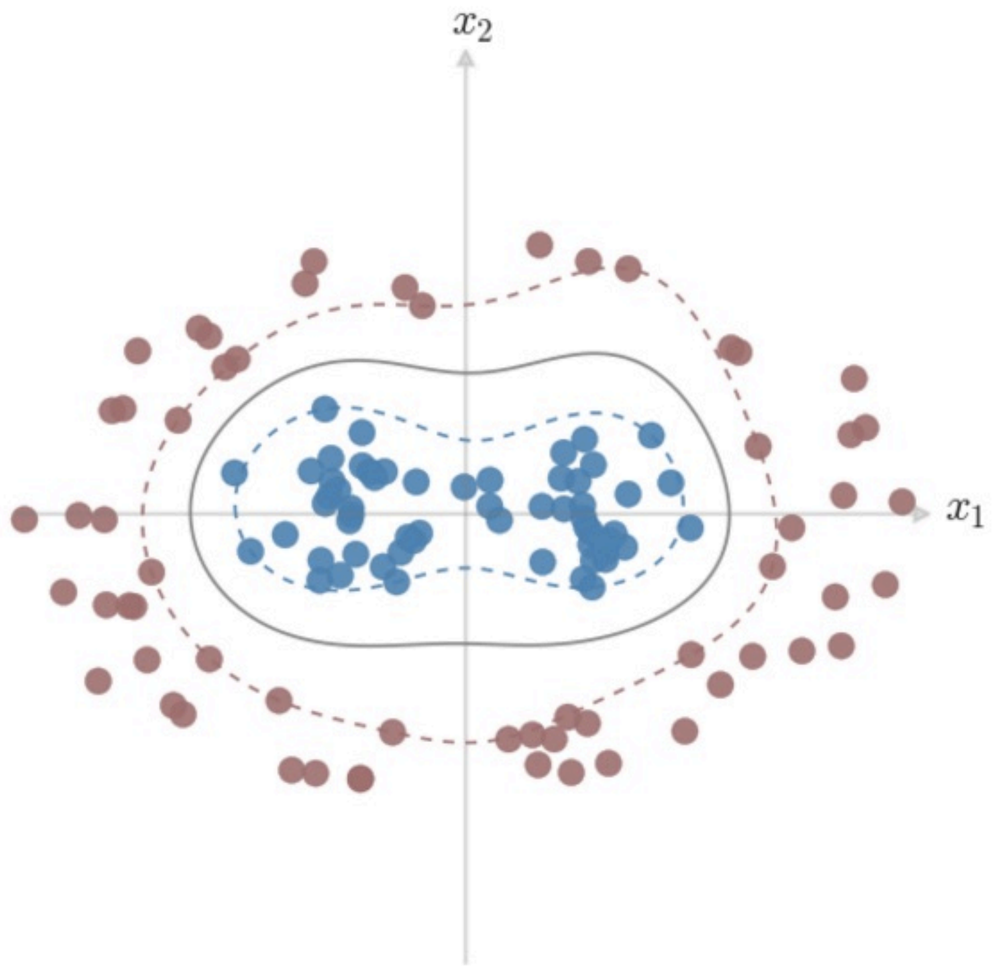
RBF Example

Suppose you have the following data



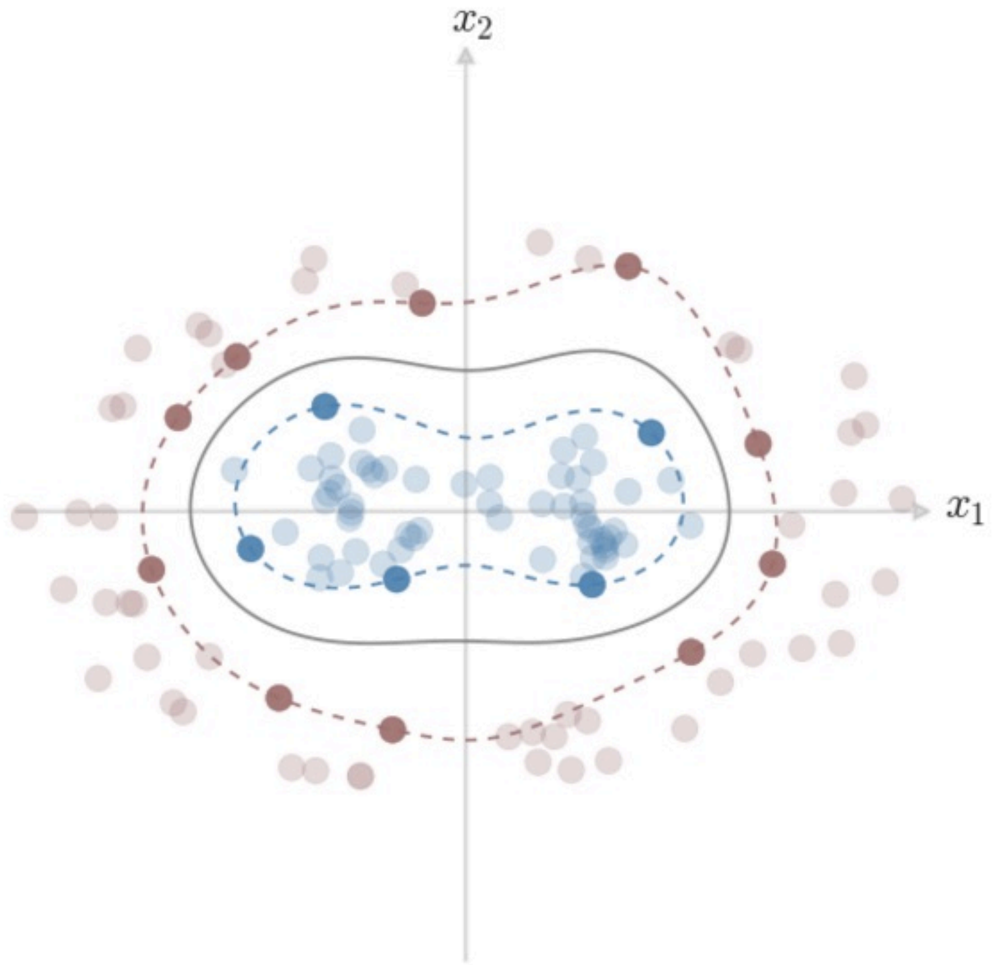
RBF Example

Result with $\gamma = 3$ and $C = 5$:



RBF Example

Can also visualize support vectors



Kernel Validity

So what makes a function $K(\mathbf{x}, \mathbf{z})$ a valid kernel?

Symmetry: $K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z}) = \phi(\mathbf{z})^T \phi(\mathbf{x}) = K(\mathbf{z}, \mathbf{x})$

Positive Semi-Definiteness:

Let K be the matrix s.t. $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$

For any vector $\mathbf{z}$, must have $\mathbf{z}^T K \mathbf{z} \geq 0$

Mercer's Theorem (Lite): K is a valid kernel if for any set of vectors $\mathbf{x}_i$ for $i = 1, \ldots, m < \infty$ the corresponding kernel matrix K is symmetric and positive semi-definite.

Implementation Challenges

- Yes, we can compute $K(\mathbf{x}, \mathbf{z})$ faster than $\phi(\mathbf{x})^T \phi(\mathbf{z})$, but $K(\mathbf{x}, \mathbf{z})$ is still more expensive than $\mathbf{x}^T \mathbf{z}$
- For complicated kernels, precompute and store matrix $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ using some symmetric storage format
- The specific weights in decision function $\mathbf{w}^T \mathbf{x} + b$ are no longer computable. Predict based on dual expression

$$\mathbf{w}^T \mathbf{x} + b = \sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

- **Note**: Only depends on support vectors

Support Vector Machine Summary

- Great Out-of-the-Box Classifier
- Convex objective guarantees globally optimal solution
- Solutions are in a sense **sparse** b/c of dependence on SVs
- Kernel trick gives flexibility

- QP problem can be slow
- Have to choose the kernel yourself

Generalization Bounds for SVM

$$\forall h \in H \quad R(h) \leq \hat{R}(h) + \sqrt{\frac{2d\ln(em/d)}{m}} + \sqrt{\frac{\ln(1/\delta)}{2m}}$$

Recap: For $\mathbf{x} \in \mathbb{R}^n$ the VC dim of a linear classifier is $d = n + 1$

Recall that Kernel SVM are really linear classifiers in enhanced feature space

Linear Kernel: $d = n + 1$

Polynomial Kernel: $d = \binom{n+p-1}{p} + 1$

RBF Kernel: $d = \infty$

Generalization Bounds for SVM

It looks like the generalization error depends heavily on the kernel and the size of the feature space.

But remember, we haven't assumed a thing about the data distribution $\mathcal{D}$

Vapnik proved that if you make some assumptions about the distribution, then generalization bounds for SVM (any margin-based classifier, actually) can be made a lot better, and even independent of the number of features.

Generalization Bounds for SVM

Vapnik introduced a sort of *effective* VC dimension

Assume the data $\{\mathbf{x}_i\}_{i=1}^m$ lives in a ball of radius R

$$\|\mathbf{x}_i\| \leq R \text{ for all } i = 1, \ldots, m$$

Let M be the margin of our classifier

Let d be the traditional VC dimension

Define the *effective* VC dimension $d^\star$ by

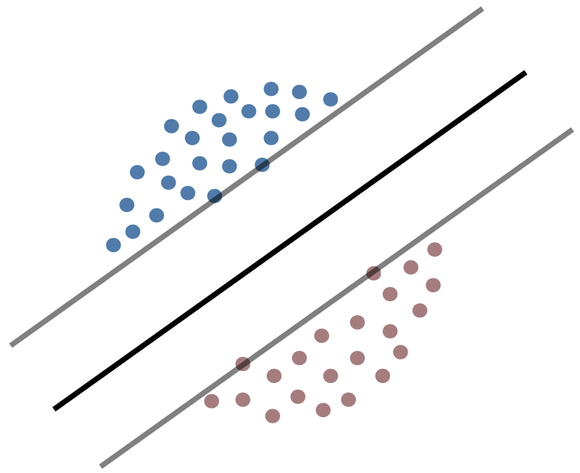
$$d^\star = \min\left(d, \frac{4R^2}{M^2}\right)$$

Generalization Bounds for SVM

We then have a similar generalization bound

$$\forall h \in H \quad R(h) \leq \hat{R}(h) + \sqrt{\frac{2d^{\star} \ln(em/d^{\star})}{m}} + \sqrt{\frac{\ln(1/\delta)}{2m}}$$

Intuition: Good bounds if R/M is not too large

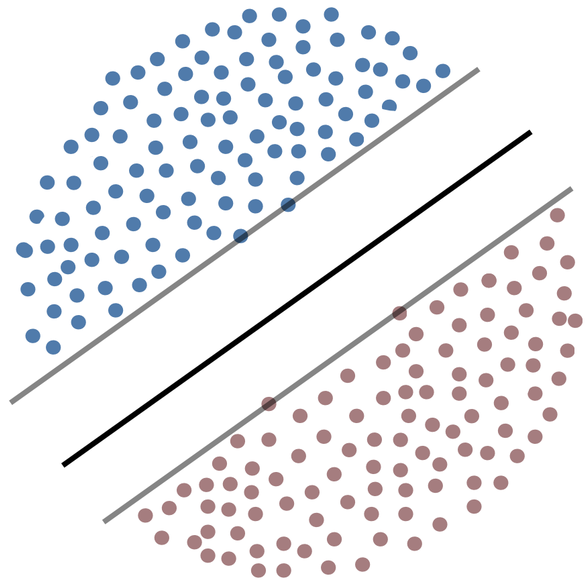


Generalization Bounds for SVM

We then have a similar generalization bound

$$\forall h \in H \quad R(h) \leq \hat{R}(h) + \sqrt{\frac{2d^{\star} \ln(em/d^{\star})}{m}} + \sqrt{\frac{\ln(1/\delta)}{2m}}$$

Intuition: Good bounds if R/M is not too large



Generalization Bounds for SVM

Take-Aways:

- If model induces a large margin compared to the spread of the data, effective complexity of model is reduced
- Because Kernel SVMs induce a margin they can have good generalization bounds even if the kernel is complex
- If you have simple kernel that induces a margin then **use it**!

In Class

- **Your Questions!**
- Work some simple examples
- Revisit cross-validation for hyperparameter tuning

Coming Up

Next Time

- Decision Trees
- Boosting

Acknowledgments

Many of these slides were adopted from Jordan Boyd-Graber

In Class

In Class

In Class

In Class

In Class