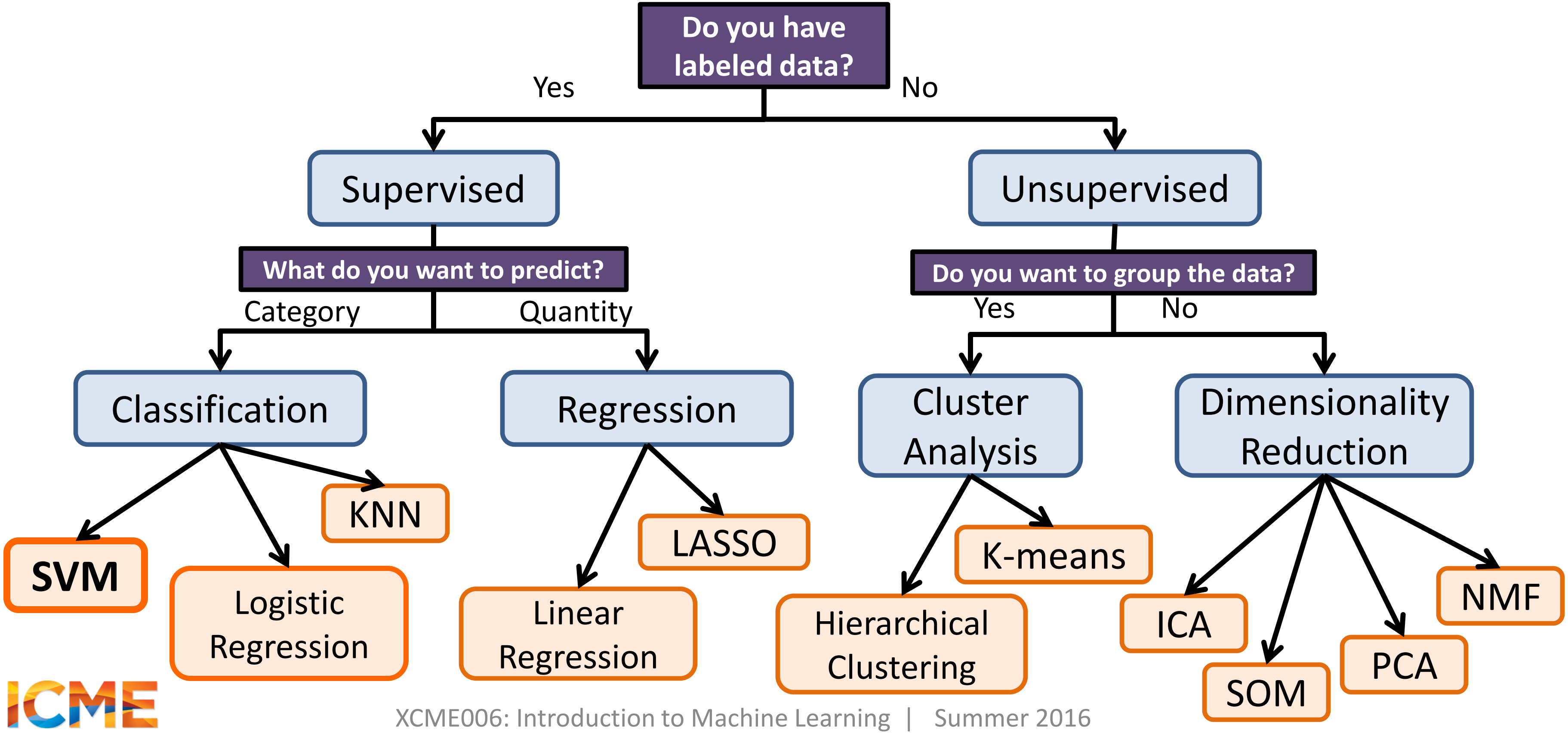
Support Vector Machines (SVM)

Gabriel Maher
gdmaher@stanford.edu

Institute for Computational and Mathematical Engineering,
Stanford University

Adapted from slides by
K. Bergen and A. Ioannidis

Types of Algorithms



Support Vector Machines

- Method for supervised classification
  - Binary classification (two class)

- Generalization of *maximal margin classifier*
- *Support vector classifier*: can be applied to data that is not linearly separable
- *Support vector machine*: non-linear decision boundary

Maximal Margin Classifier

- *Maximal margin classifier*
  - Key assumption: two classes are separable by linear decision boundary

- First, we need to review *hyperplanes*…

Hyperplanes

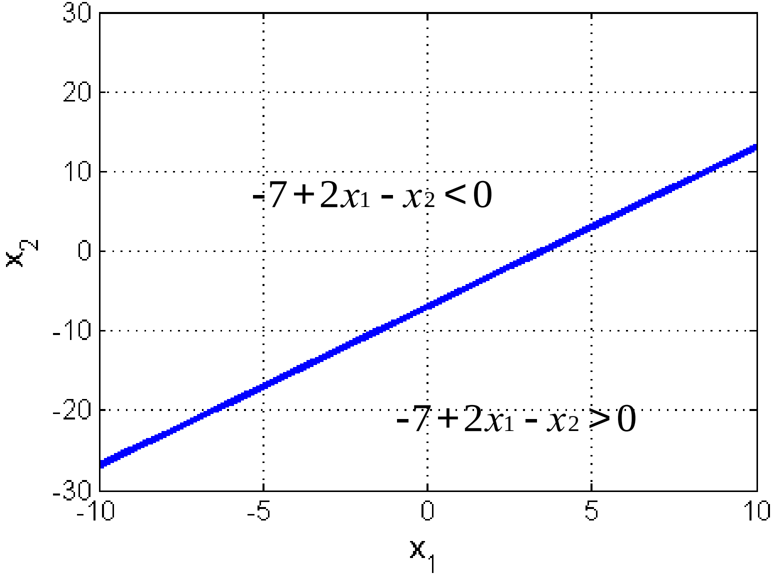
- What is a hyperplane?
  - In d-dimensional space, a $(d-1)$-dimensional affine subspace
    - e.g. line in 2D, plane in 3D
  - Hyperplane in d-dimensional space:

$$(\star)\ \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_d x_d = 0$$

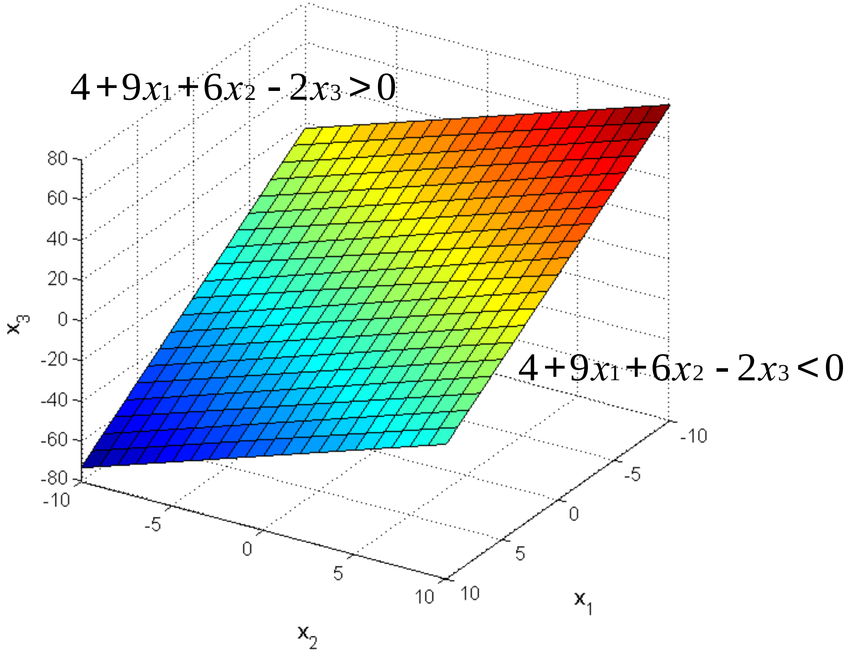
  - Separates space into two half-spaces

Hyperplanes

$-7 + 2x_1 - x_2 = 0$



$4 + 9x_1 + 6x_2 - 2x_3 = 0$



Separating Hyperplane Classifier

Idea:
Use a separating hyperplane for binary classification

Key assumption:
Classes can be separated by a linear decision boundary

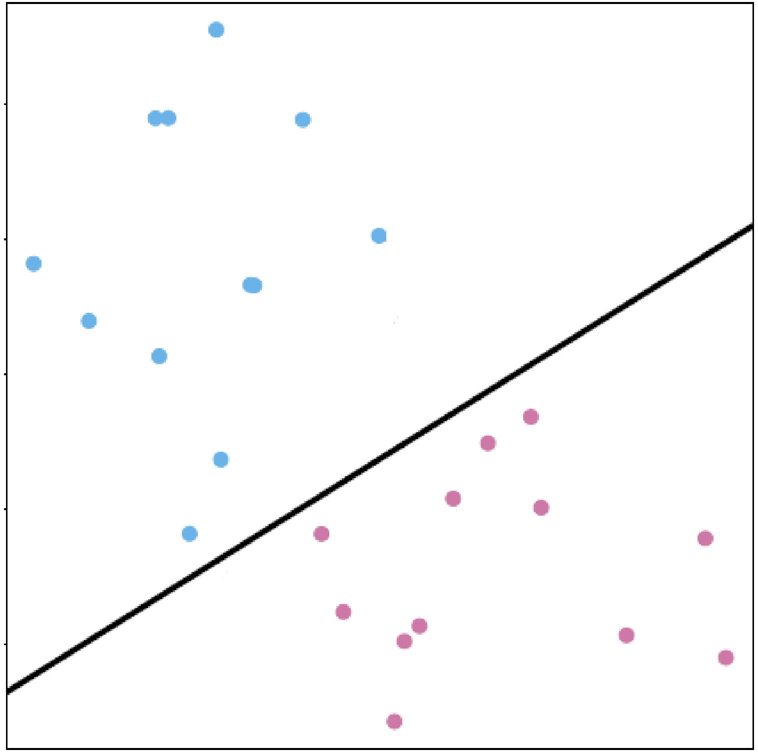


Figure 9.2 , ISL 2013

Separating Hyperplane Classifier

To classify new data points:

Assign class by location of new data point with respect to hyperplane:

$$\hat{Y} = \text{sign}\left(\beta_0 + \beta_1 X_1 + \cdots + \beta_d X_d\right)$$

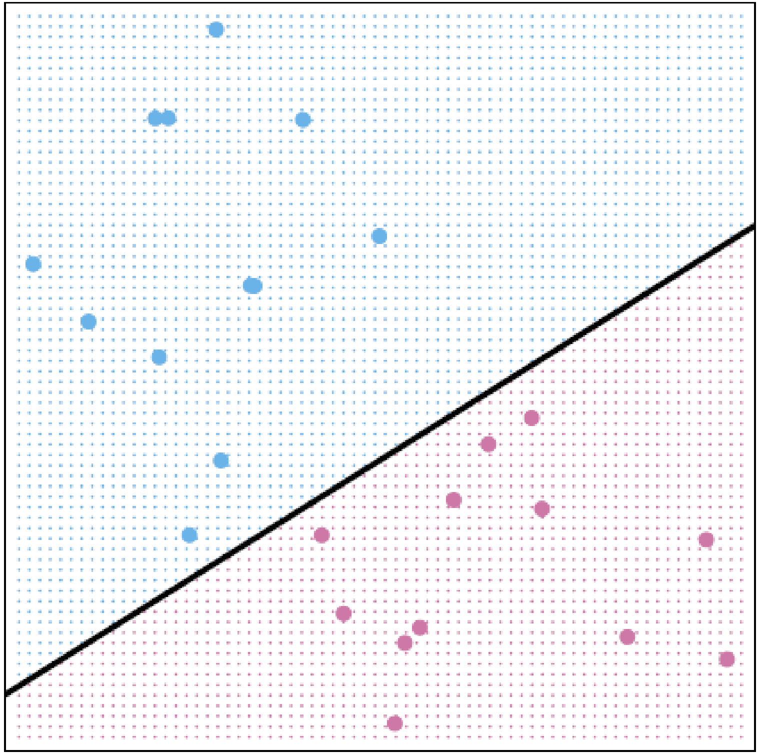


Figure 9.2 , ISL 2013

Separating Hyperplane Classifier

Key assumption:
Classes can be separated by a linear decision boundary

→ Many possible separating hyperplanes…

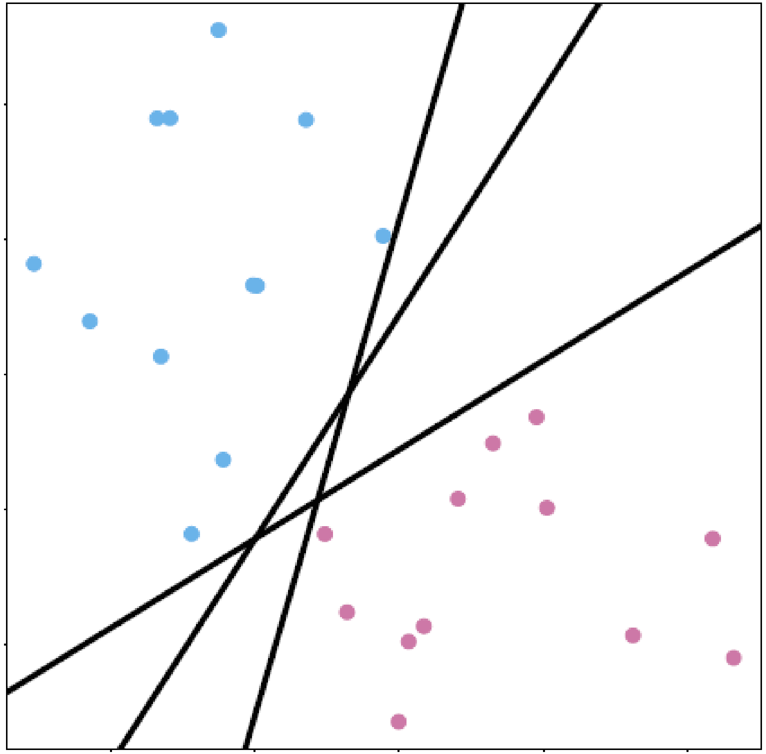


Figure 9.2 , ISL 2013

Mini Quiz:

Which linear decision boundary?

What criteria would you use to choose?

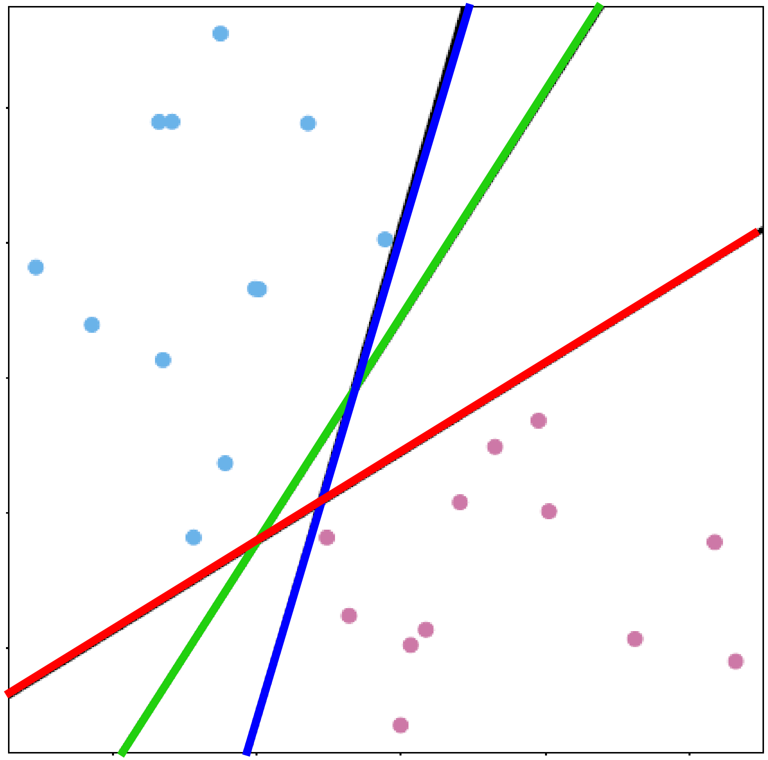


Figure 9.2 , ISL 2013

Separating Hyperplane Classifier

Which linear decision boundary?

Separating hyperplane "farthest" from training data

→ "Maximal Margin Classifier"

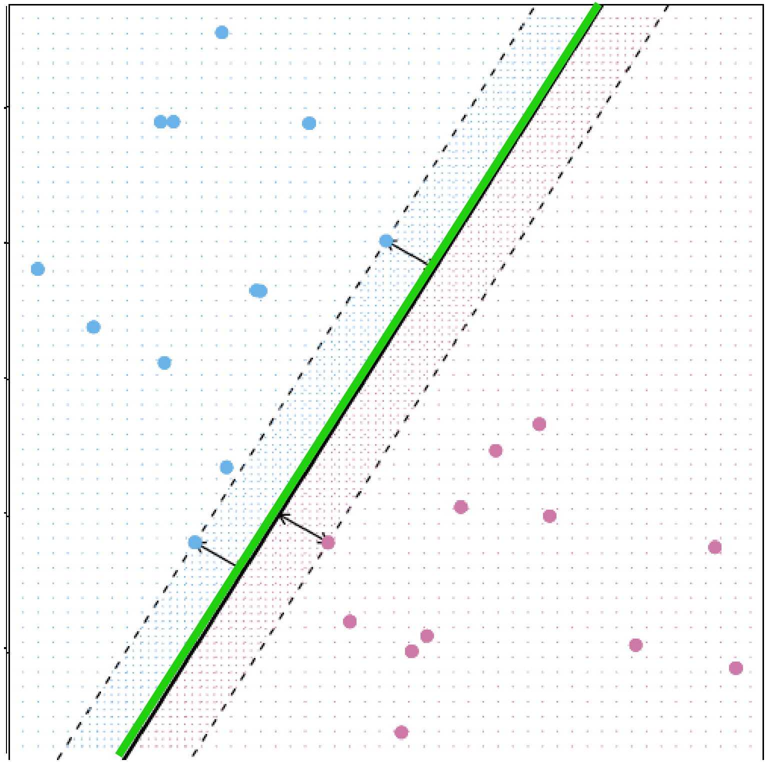


Figure 9.2, 9.3 , ISL 2013

Maximal Margin Classifier

Maximal margin hyperplane

Hyperplane "farthest" from training data → maximizes margin

Margin: smallest distance between any training observation and the hyperplane

Support vectors: the training observations equidistant from the hyperplane

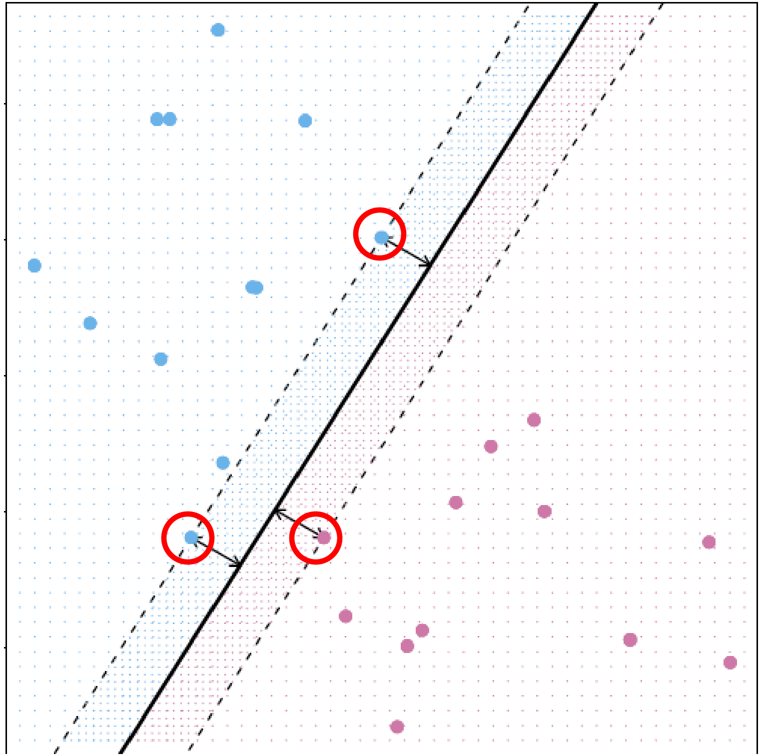


Figure 9.3 , ISL 2013

Maximal Margin Classifier

- *Support vectors*
  - The training observations equidistant from the maximal margin (MM) hyperplane
  - “Support”: MM hyperplane only depends on these observations
    - If support vectors are perturbed, then MM hyperplane will change
    - If any other training observations are perturbed, MM hyperlane not effected

Maximal Margin Classifier

- To find maximal margin hyperplane, solve:

$$\underset{(\beta_0,\ldots,\beta_d)}{\text{maximize}} \quad M$$

maximize the margin, M

$$\text{subject to} \quad \sum_{j=0}^{d} \beta_j^2 = 1$$

constraint necessary for well-defined optimization problem

$$Y^{(i)}\left(\beta_0 + \beta_1 X_1^{(i)} + \cdots + \beta_d X_d^{(i)}\right) \geq M, \ \forall i$$

all training points must be distance at least M from hyperplane

$Y^{(i)} \in \{-1, 1\}$ are the class labels

Maximal Margin Classifier

Recall our assumption:

Classes can be separated by a linear decision boundary

What if there's no separating hyperplane?

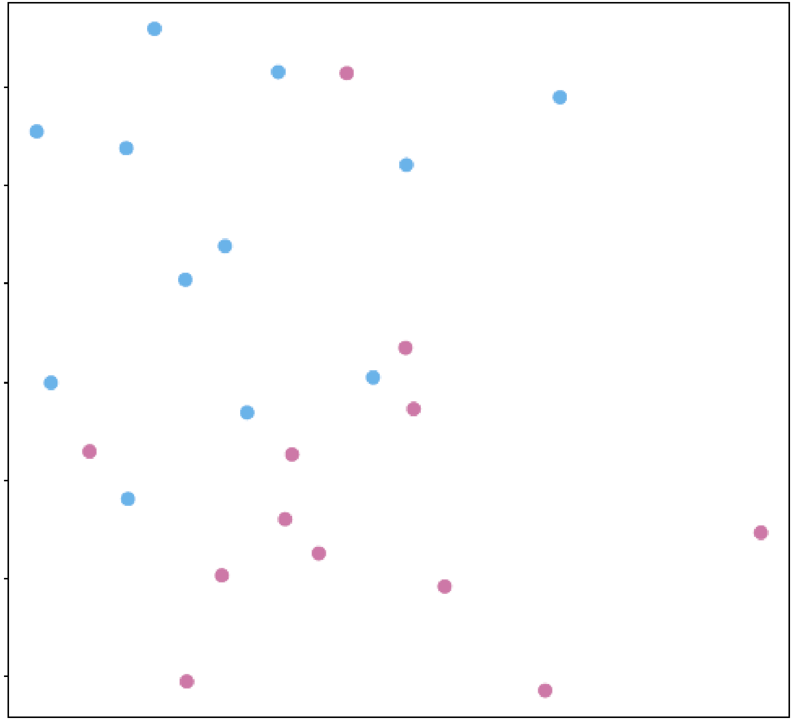


Figure 9.4 , ISL 2013

Maximal Margin Classifier

Disadvantage:

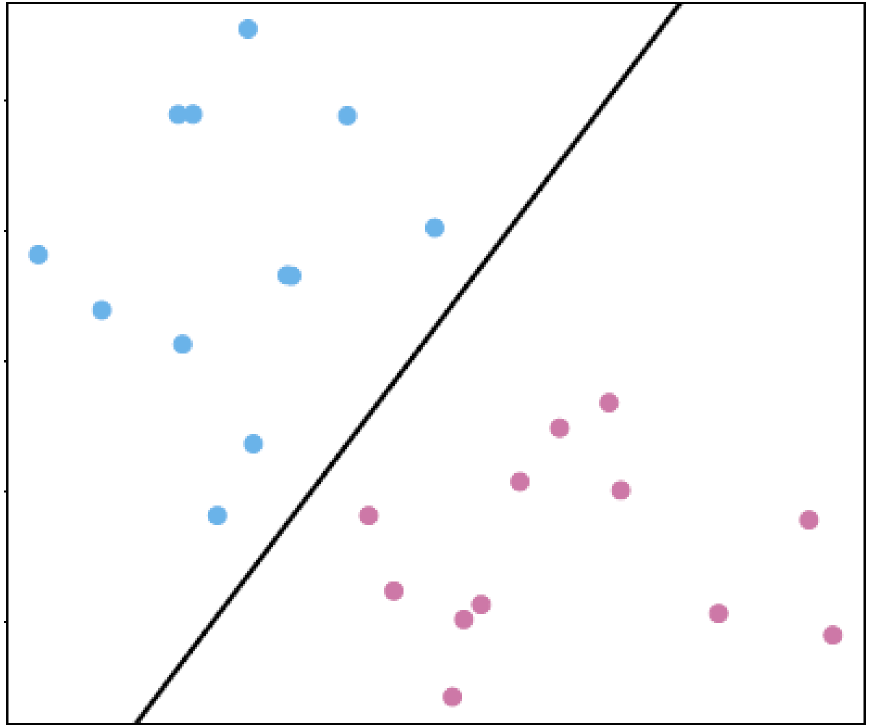


Figure 9.5 , ISL 2013

Maximal Margin Classifier

Disadvantage:

Can be sensitive to individual observations

May overfit training data

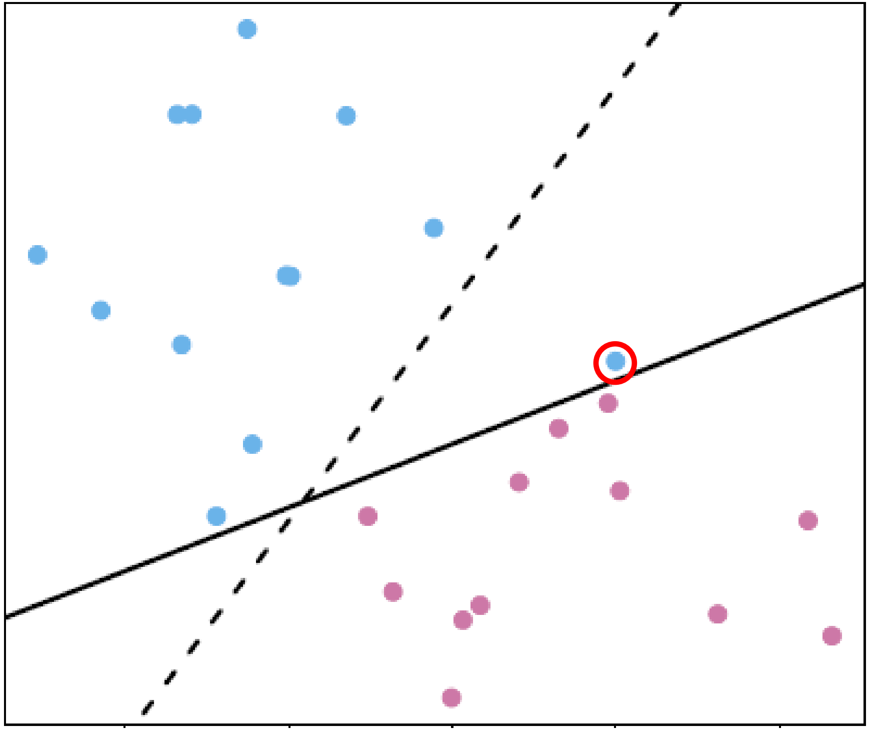


Figure 9.5 , ISL 2013

Support Vector Classifier

What if there's no separating hyperplane?

Support Vector Classifier: allows training samples on the "wrong side" of the margin or hyperplane

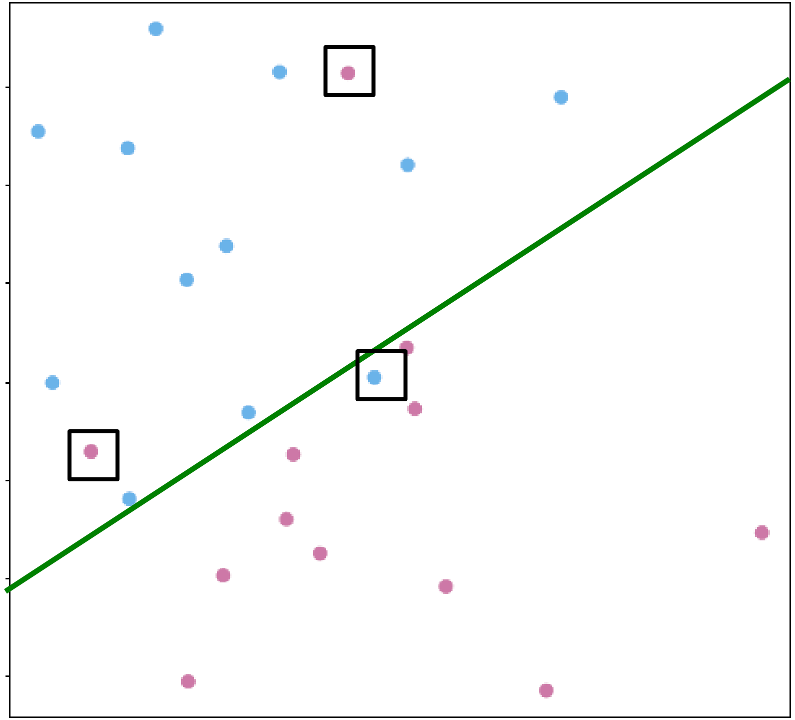


Figure 9.4 , ISL 2013

Support Vector Classifier

- *Support Vector Classifier*
  - Hyperplane-based classifier
  - Allows *some* training samples on “wrong side” of margin/hyperplane
  - *Soft margin*: margin is not a hard boundary

Support Vector Classifier

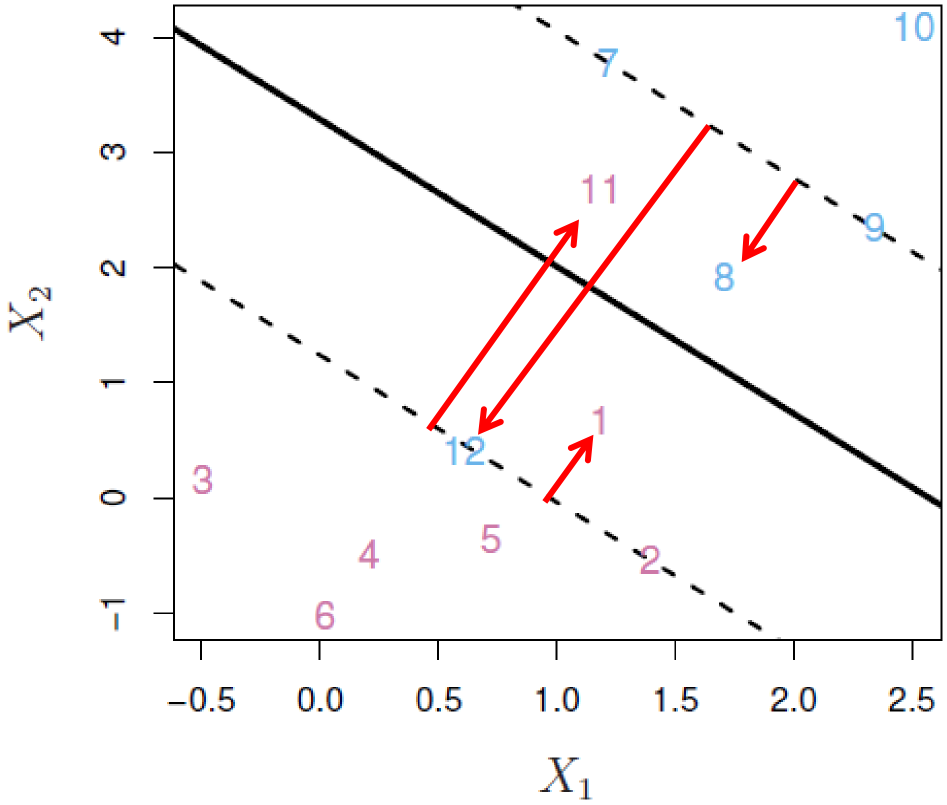


Figure 9.6 , ISL 2013

Support Vector Classifier

- *Support Vector Classifier*
  - Hyperplane-based classifier
  - Allows *some* training samples on “wrong side” of margin/hyperplane
  - *Soft margin*: margin is not a hard boundary

- Idea: solve maximal margin problem, but allow violations of the margin
  - Impose penalty to limits number/degree of violations

Support Vector Classifier

- To find hyperplane for the SV classifier, solve:

maximize the margin, M

constraint necessary for well-defined optimization problem

training points less than distance M from hyperplane *pay a penalty ε$_i$*

$$\underset{(\beta_0,\ldots,\beta_d,\epsilon_1,\ldots,\epsilon_n)}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=0}^{d} \beta_j^2 = 1$$

limit on total penalties

$$Y^{(i)}\left(\beta_0 + \beta_1 X_1^{(i)} + \cdots + \beta_d X_d^{(i)}\right) \geq M(1-\epsilon_i),$$

$$\sum_{i=1}^{n} \epsilon_i \leq C, \qquad \epsilon_i \geq 0,\ \forall i$$

"slack" variable ε$_i$

Support Vector Classifier

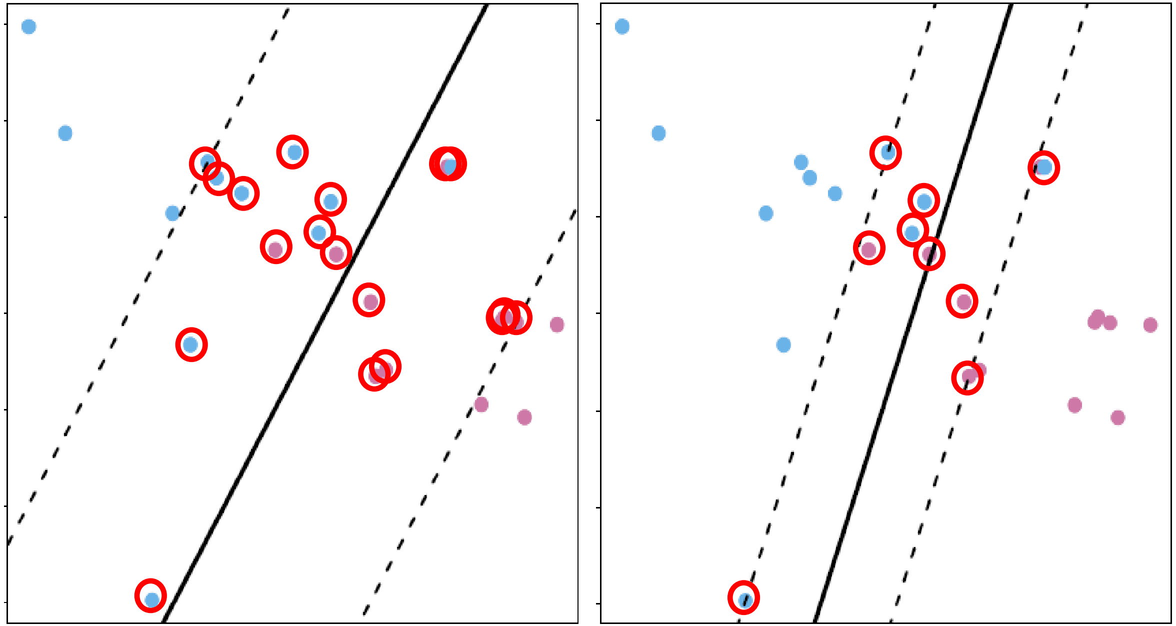
- Slack variables ε_i allow for violations of the margin
  - $\varepsilon_i = 0$: training point $X^{(i)}$ is on correct side of margin
  - $\varepsilon_i > 0$: $X^{(i)}$ violates the margin
  - $\varepsilon_i > 1$: $X^{(i)}$ is misclassified (wrong side of hyperplane)
- Penalty parameter C – "budget" for violations
  - Allows at most C misclassifications on training set

Support Vector Classifier

“Misclassification budget” parameter C is selected by *cross-validation*

* controls bias-variance trade-off *

Support vectors: observations on margin or violating margin



Large budget
many Support Vectors

Small budget
fewer Support Vectors

Figure 9.7 , ISL 2013

Support Vector Classifier

Disadvantage:

Linear decision boundary

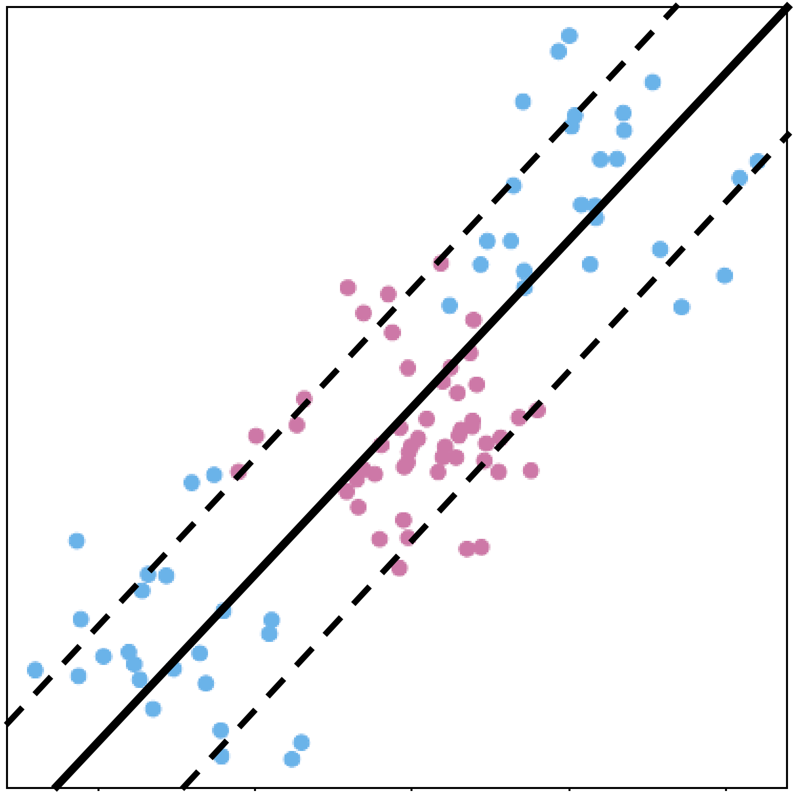
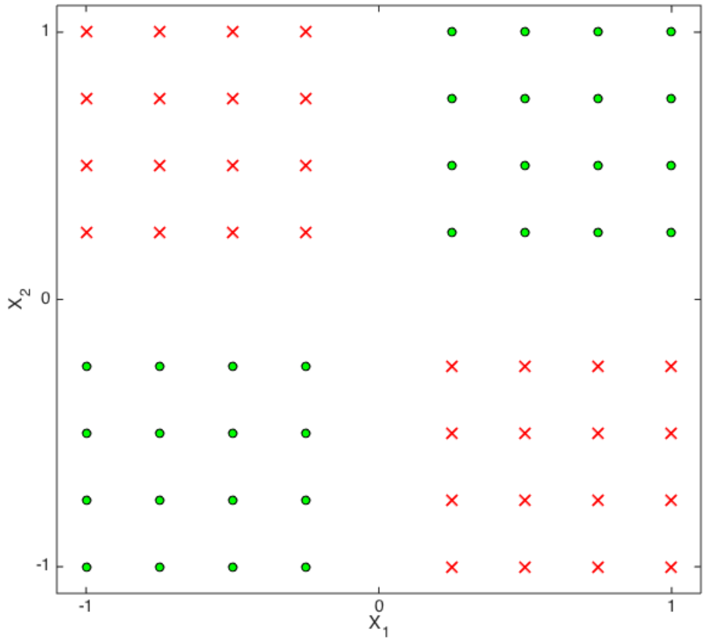


Figure 9.8 , ISL 2013

Expanding Feature Space

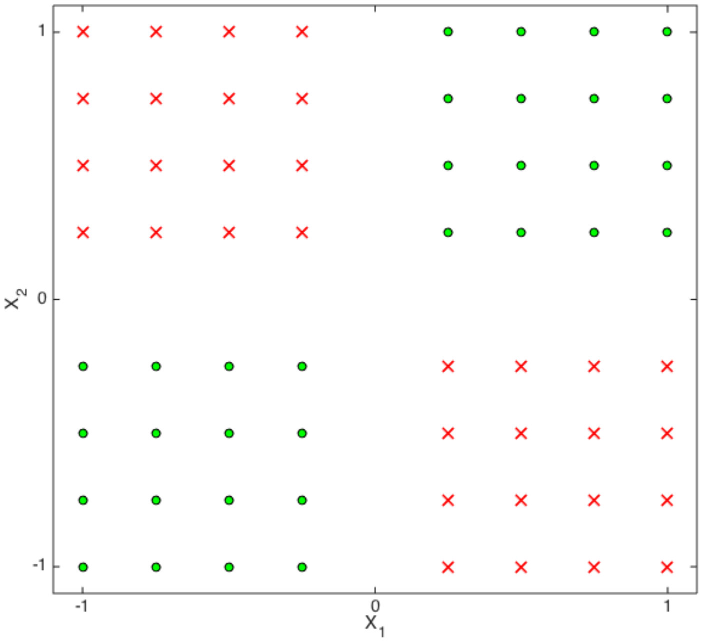


variables: X_1 , X_2

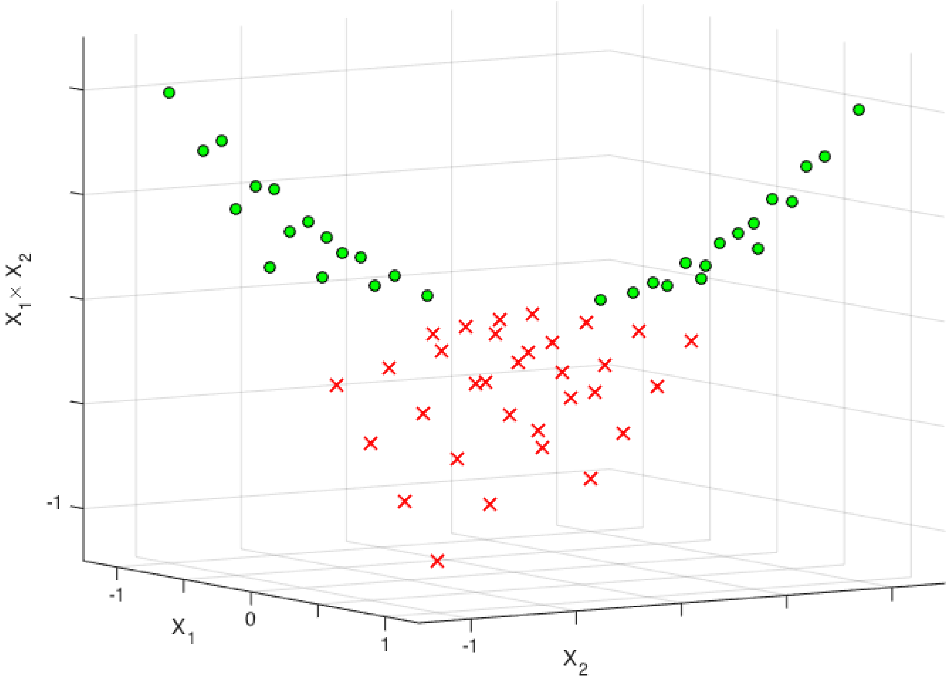
Some data sets are not linearly separable…

But they *become* linearly separable when transformed into a *higher* dimensional space

Expanding Feature Space



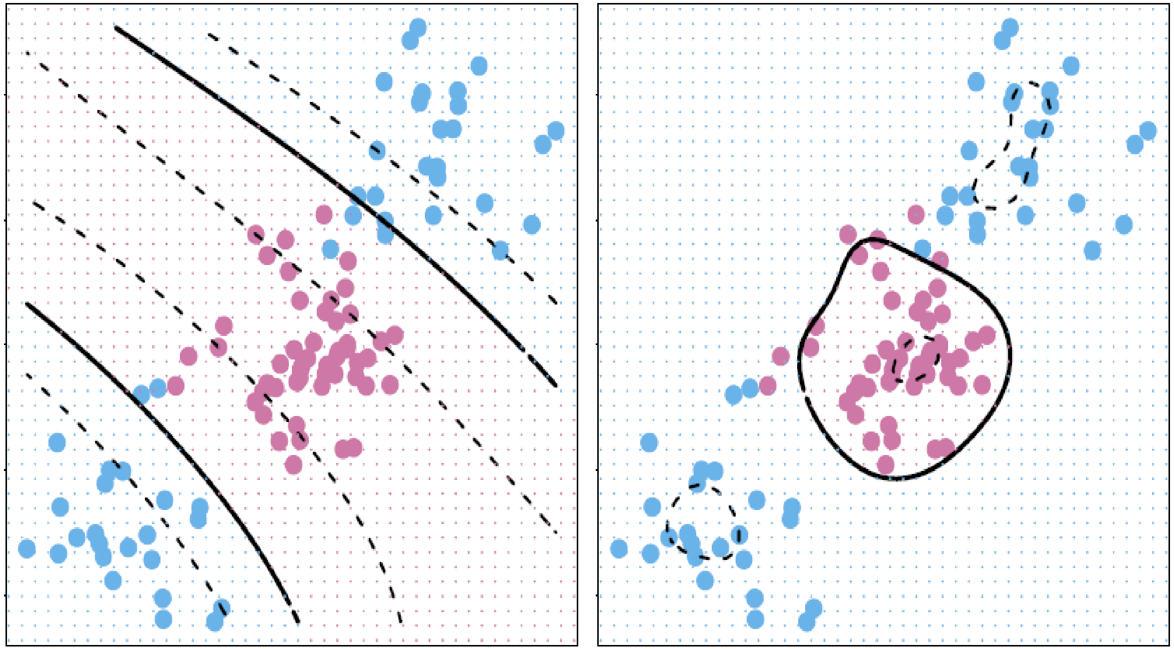
variables: X_1 , X_2



variables: X_1 , X_2, X_1X_2

Support Vector Machine (SVM)

Support Vector Machine: extension that uses *kernels* to achieve *non-linear decision boundary*



Support Vector Machine

- *Kernel*: generalization of inner product

- Kernels (implicitly) map data into higher-dimensional space
  - Apply support vector classifier in high-dimensional space with hyperplane (linear) decision boundary

Support Vector Machine

- Computations in support vector classifier requires only inner products of training data

$$f(X) = \beta + \sum_{i \in S} \alpha_i \langle X^{(i)}, X \rangle$$

- In SVM we replace inner product with kernel function

$$f(X) = \beta + \sum_{i \in S} \alpha_i K\left(X^{(i)}, X\right)$$

Support Vector Machine

- Properties of kernels $K(X, X')$:
  - Generalization of inner product

$$K(X, X') = \langle \phi(X), \phi(X') \rangle, \quad \phi \text{ feature mapping}$$

  - Symmetric: $K(X, X') = K(X', X)$
  - Gives a measure of similarity between X and X'
    - If X and X' close together, then $K(X, X')$ large
    - If X and X' far apart, then $K(X, X')$ small

Support Vector Machine

- Linear kernel

$$K(X, X') = \langle X, X' \rangle$$

- Polynomial kernel (degree p)

$$K(X, X') = (1 + \langle X, X' \rangle)^p$$

- Radial basis kernel

$$K(X, X') = \exp\left(-\gamma \|X - X'\|^2\right)$$

Support Vector Machine

- Why use kernels instead of explicitly constructing larger feature space?
  - Computational advantage

$$\phi : \mathbb{R}^d \to \mathbb{R}^D, \quad d << D$$

$$K(X, X') = \langle \phi(X), \phi(X') \rangle \quad \text{in } O(d)$$

- Other machine learning methods use kernels
  - e.g. kernel PCA

- Example: polynomial kernel, p = 2, d = 2:

$$K(X,Y) = (1 + \langle X, Y \rangle)^2 \qquad X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

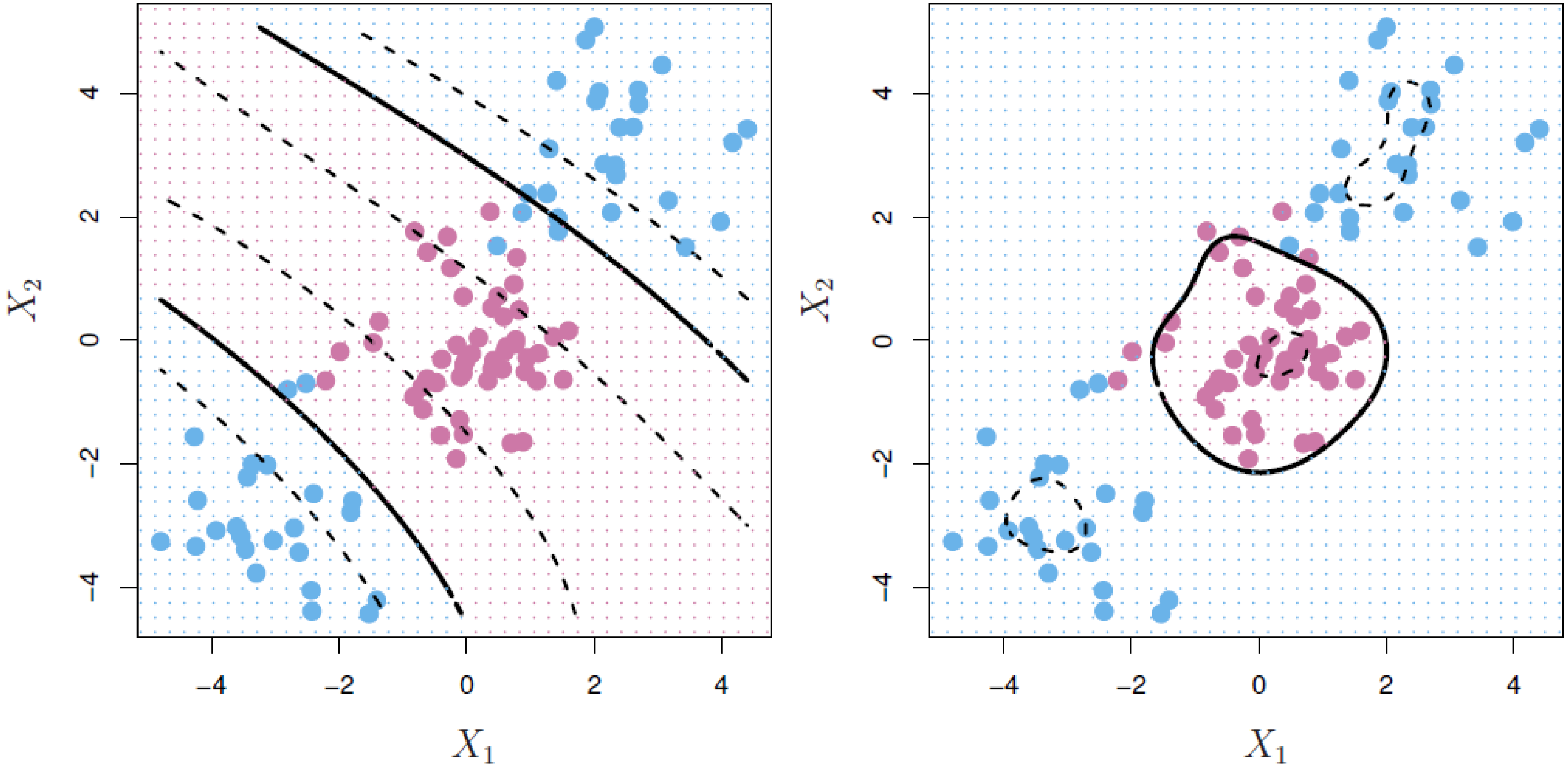
Then

$$K(X,Y) = 1 + 2X_1Y_1 + 2X_2Y_2 + X_1^2Y_1^2 + X_2^2Y_2^2 + 2X_1Y_1X_2Y_2$$

$$= \langle \phi(X), \phi(Y) \rangle$$

$$\text{where} \quad \phi(X) = \begin{bmatrix} 1 \\ \sqrt{2}X_1 \\ \sqrt{2}X_2 \\ \sqrt{2}X_1X_2 \\ X_1^2 \\ X_2^2 \end{bmatrix}$$

Support Vector Machine



Support Vector Machine

- Advantages
  - Regularization parameter C to avoid overfitting
  - Use of kernel gives flexibility in form of decision boundary
  - Optimization problem convex – unique solution
- Disadvantages
  - Must tune hyperparameters (e.g. C, kernel function)
    - Poor performance if not well-chosen
  - Must formulate as binary classification
  - Difficult to interpret

Questions?

Expanding Feature Space

- Linear regression → non-linear model
  - Create new features that are functions of predictors
- Apply same technique to support vector classifier
  - Consider polynomial functions of predictors:

Non-linear Decision Boundary

- Suppose our original data has d features:

$$X = [X_1, X_2, \cdots, X_d]$$

- Expand feature space to include 2d features:

$$\tilde{X} = [\underbrace{X_1}_{\tilde{X}_1}, \underbrace{(X_1)^2}_{\tilde{X}_2}, \underbrace{X_2}_{\tilde{X}_3}, \underbrace{(X_2)^2}_{\tilde{X}_4}, \cdots, \underbrace{X_d}_{\tilde{X}_{2d-1}}, \underbrace{(X_d)^2}_{\tilde{X}_{2d}}]$$

  – Decision boundary will be non-linear in original feature space

Non-linear decision boundary

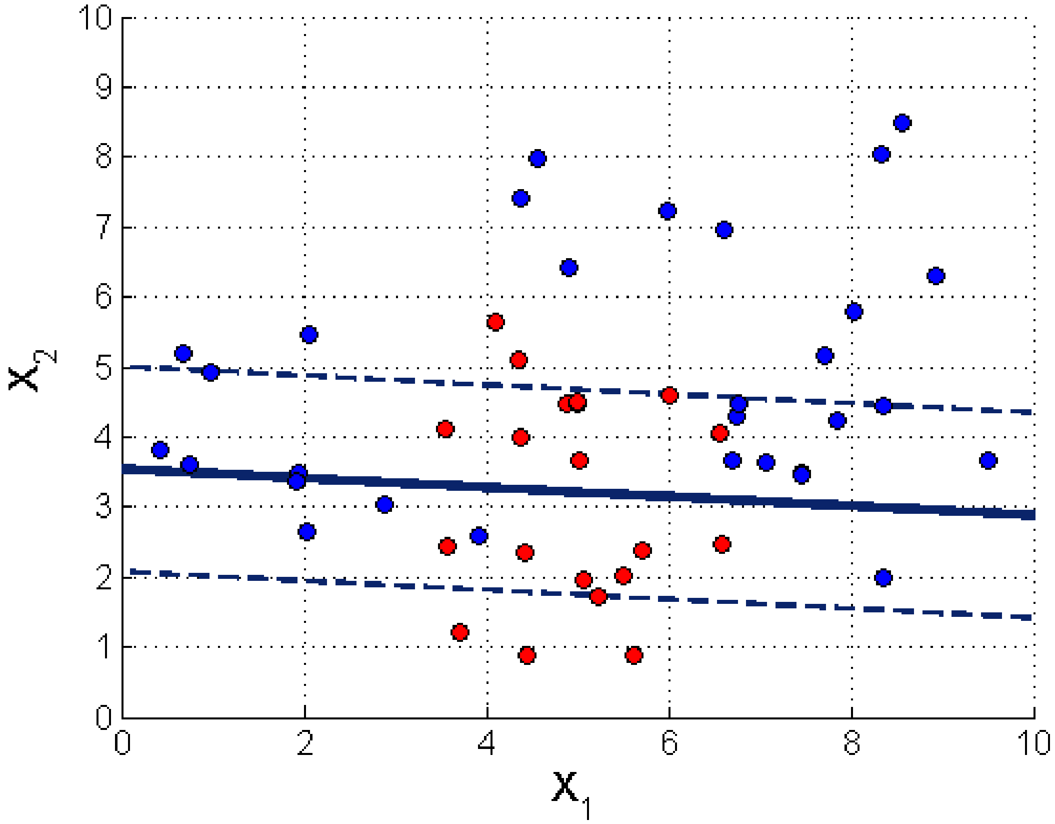
- Decision boundary in enlarged features space is linear:

$$\beta_0 + \beta_1 \tilde{X}_1 + \beta_2 \tilde{X}_2 + \cdots + \beta_{2d-1} \tilde{X}_{2d-1} + \beta_{2d} \tilde{X}_{2d} = 0$$

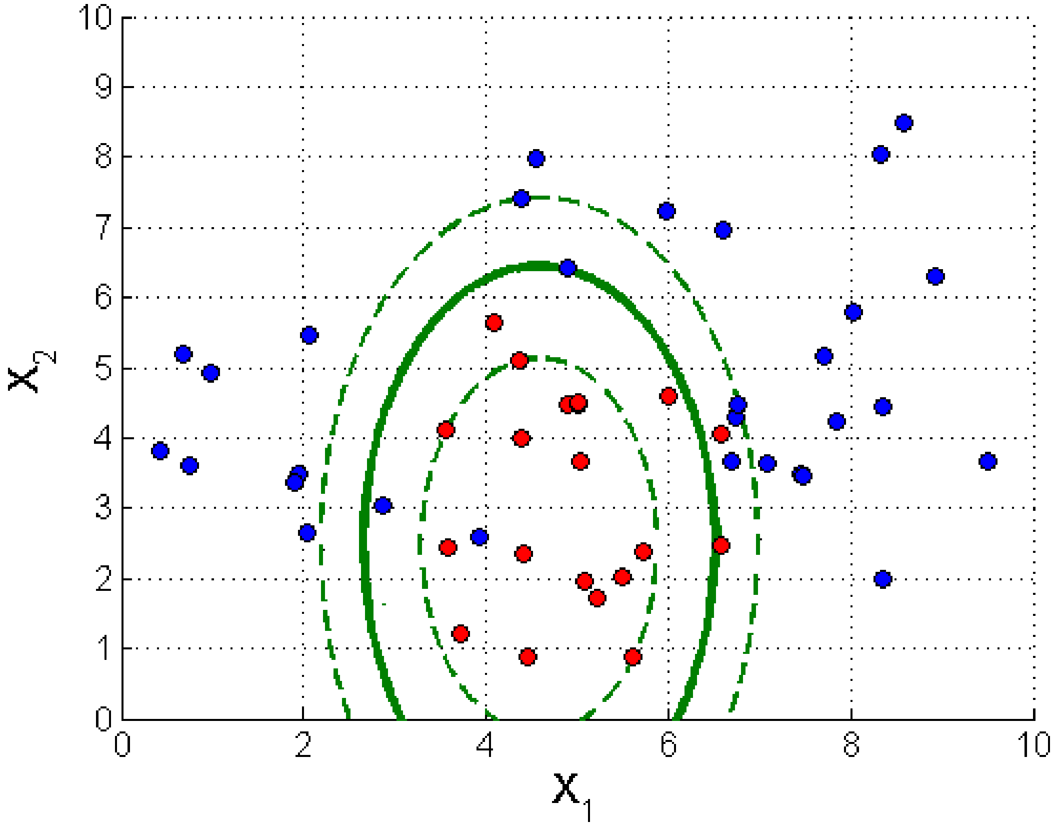
- Decision boundary in enlarged features space is an ellipse *in the original features space*:

$$\beta_0 + \beta_1 X_1 + \beta_2 (X_1)^2 + \cdots + \beta_{2d-1} X_d + \beta_{2d} (X_d)^2 = 0$$

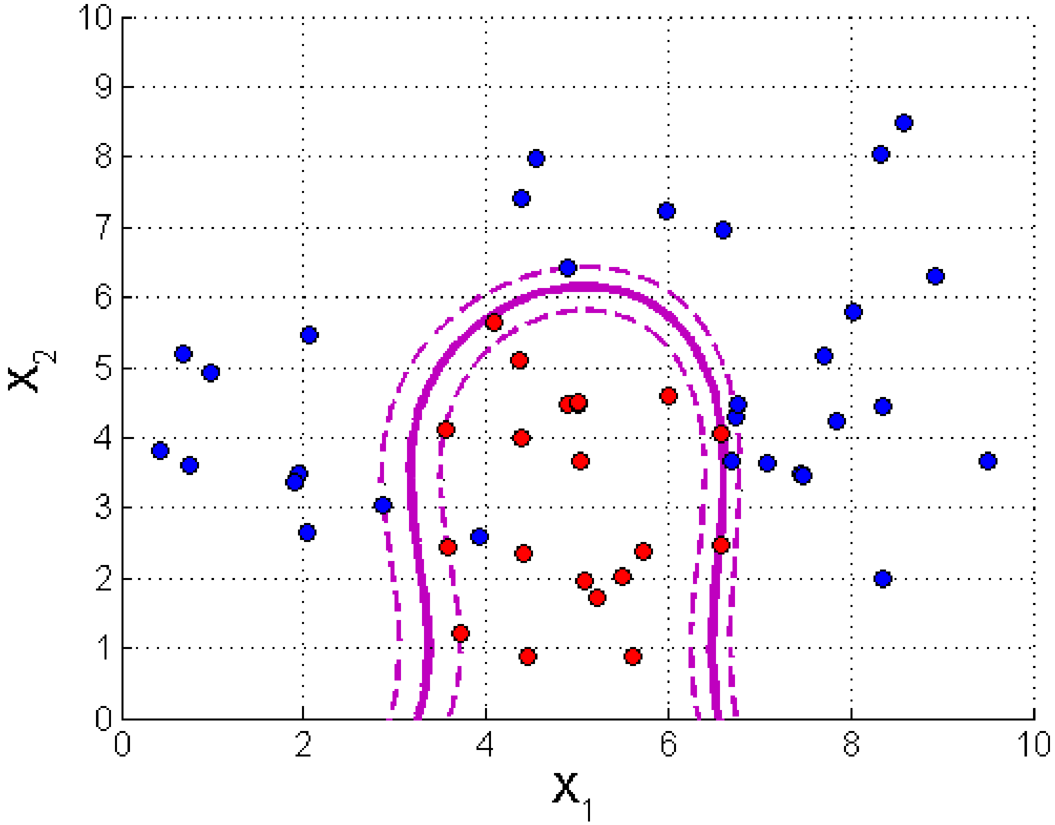
Non-linear Decision Boundary



Non-linear Decision Boundary



Non-linear Decision Boundary



Non-linear Decision Boundary

- Add higher order polynomial terms to expanded features set
  → number of features grows quickly
  - Large number of features becomes computationally challenging
  - We need an efficient way to work with large number of features

SVM with 3+ classes

- SVMs are designed for binary classification
  - Separating hyperplane naturally separates data into two classes

- How do we handle the case when the data belong to more than two classes?

- Popular approaches:
  1. One-versus-one
  2. One-versus-all

SVM with 3+ classes

- *One-versus-one* classification
  - Construct an SVM for each pair of classes
  - For K classes, this requires training $\frac{K(K-1)}{2}$ SVMs
  - To classify a new observation, apply all $\frac{K(K-1)}{2}$ SVMs to the observation – take the most frequent class among pairwise results as predicted class
  - Disadvantage: computationally expensive for large values of K

SVM with 3+ classes

The image part with relationship ID rId2 was not found in the file.

Questions?

Measuring Classifier Performance

- We can show the performance of the classifier in a table called a *confusion matrix*:
  - "Good performance": TP, TN large and FP, FN small

Predicted class

		+	**–**
True class	**+**	True Positive (TP)	False Negative (FN) Type II error
	–	False positive (FP) Type I error	True Negative (TN)

Measuring Classifier Performance

True positive rate (TPR)

(recall, sensitivity)

		Predicted class +	Predicted class −
True class	+	**TP**	**FN**
	−	FP	TN

$$TPR = \frac{TP}{TP + FN}$$

Positive predictive value (PPV) *(precision)*

		Predicted class +	Predicted class −
True class	+	**TP**	FN
	−	**FP**	TN

$$PPV = \frac{TP}{TP + FP}$$

False positive rate (FPR)

		Predicted class +	Predicted class −
True class	+	TP	FN
	−	**FP**	**TN**

$$FPR = \frac{FP}{FP + TN}$$

True negative rate (SPC)

(specificity)

		Predicted class +	Predicted class −
True class	+	TP	FN
	−	**FP**	**TN**

$$SPC = \frac{TN}{FP + TN}$$

Measuring Classifier Performance

True positive rate (TPR)

(recall, sensitivity)

Predicted class

True class	+	−
+	**TP**	**FN**
−	FP	TN

$$TPR = \frac{TP}{TP + FN}$$

Positive predictive value (PPV) *(precision)*

Predicted class

True class	+	−
+	**TP**	FN
−	**FP**	TN

$$PPV = \frac{TP}{TP + FP}$$

Precision/recall

False positive rate (FPR)

Predicted class

True class	+	−
+	TP	FN
−	**FP**	**TN**

$$FPR = \frac{FP}{FP + TN}$$

True negative rate (SPC)

(specificity)

Predicted class

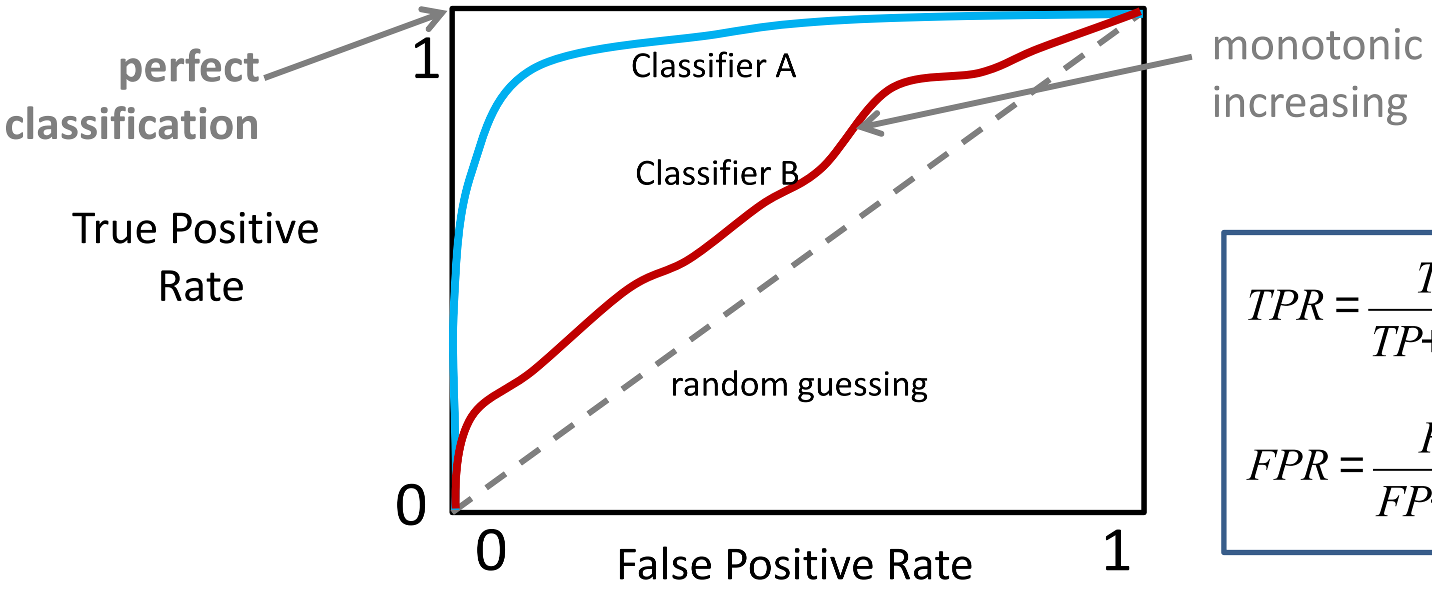
True class	+	−
+	TP	FN
−	**FP**	**TN**

$$SPC = \frac{TN}{FP + TN}$$

ROC curve

Measuring Classifier Performance

- ROC (receiver operating characteristic) curve



$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

Measuring Classifier Performance

- Disadvantage of ROC curve
  - ROC curve doesn't capture imbalances in number of observations in true classes
  - e.g. Consider data set in which 1% belongs to class "+" and 99% to class "–"
    - Suppose we obtain the classification results below then, TPR = 0.9, and TNR = 0.12
    - TPR and TNR do not capture that 13x as many FP as TP

		Predicted class +	Predicted class –
True class	+	90	10
True class	–	1188	8712

Measuring Classifier Performance

- Precision/recall
  - Precision (Positive predictive value): $PPV = \frac{TP}{TP+FP}$
    - Fraction of samples predicted as (+) that are truly (+)
  - Recall (True positive rate): $TPR = \frac{TP}{TP+FN} = \frac{TP}{P}$
    - Fraction of (+) samples correctly classified as (+)
  - Recall and precision inversely related
  - In perfect classifier, Recall = 1, Precision = 1
  - Imbalanced class example: Recall = 0.9, Precision = 0.07

Predicted class

True class		+	−
	+	90	10
	−	1188	8712