

Intermediate Topics in Machine Learning & Deep Learning

Alexander Ioannidis

ioannidis@stanford.edu

Institute for Computational and Mathematical Engineering,
Stanford University

Schedule

Day 1

Session 1: Intermediate Topics in ML (SVMs and Kernels)

Session 2: Transfer Learning & Generating Labels for Deep Learning

Day 2

Session 3: Dimensionality Reduction and Variational Autoencoders (VAEs)

Session 4: Representation Learning, Weakly Supervised, Semi-supervised,
and self-supervised learning

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

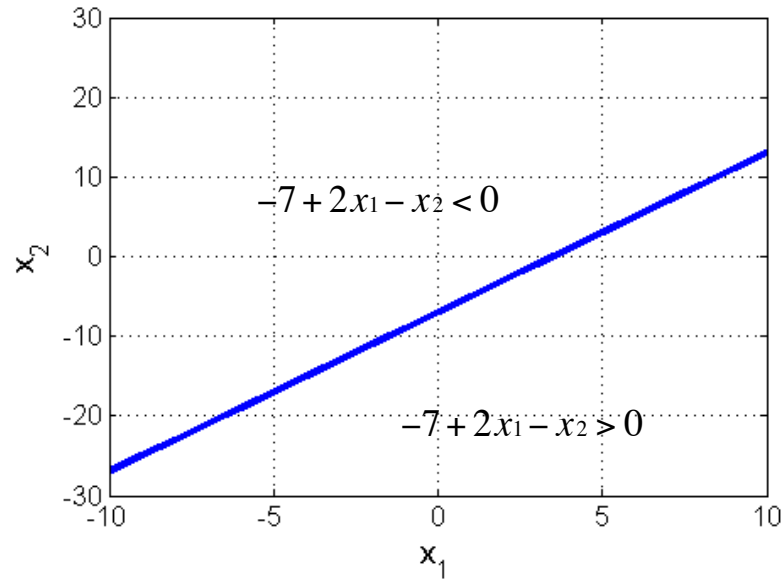
- What is a hyperplane?
 - In d -dimensional space, a $(d-1)$ -dimensional affine subspace
 - e.g. line in 2D, plane in 3D
 - Hyperplane in d -dimensional space:

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

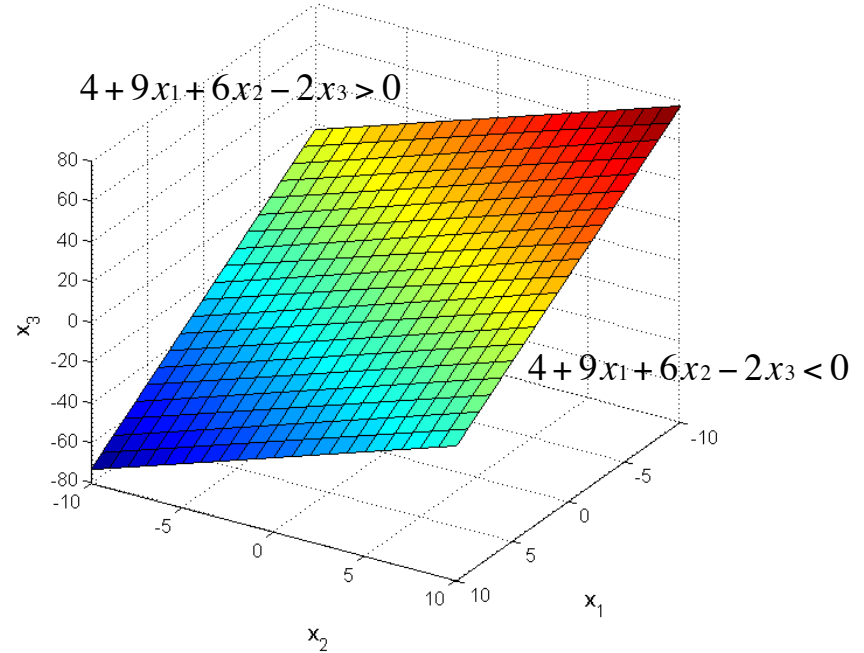
- Separates space into two half-spaces

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

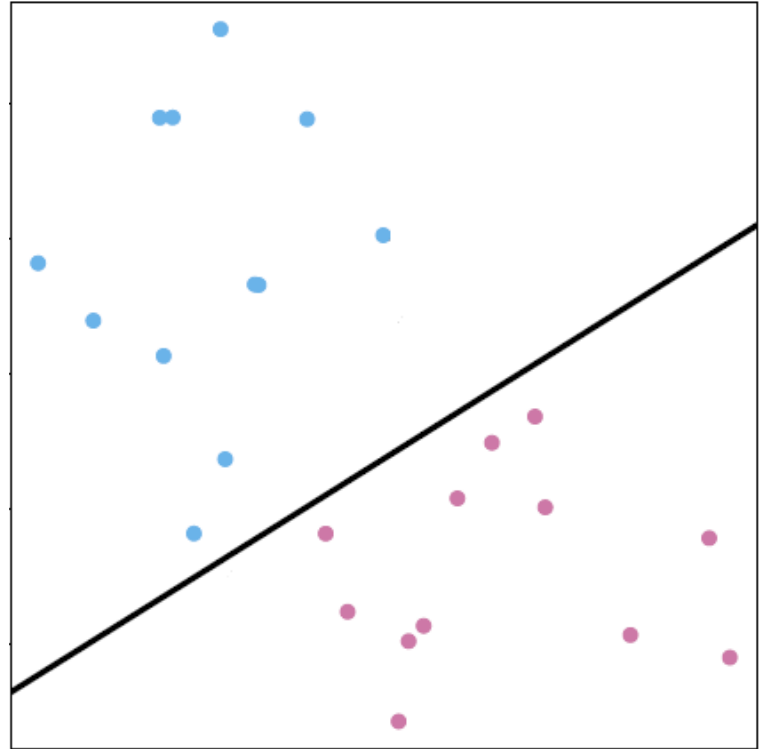


Figure 9.2 , ISL 2013

Separating Hyperplane Classifier

To classify new data points:

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

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

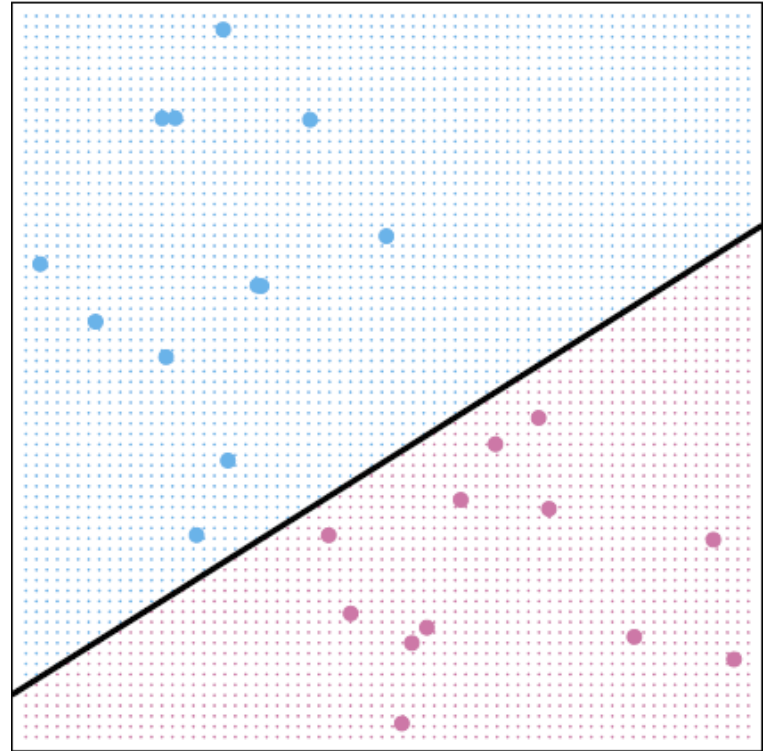


Figure 9.2 , ISL 2013

Separating Hyperplane Classifier

Key assumption:

Classes can be separated by a linear decision boundary

→ Many possible separating hyperplanes...

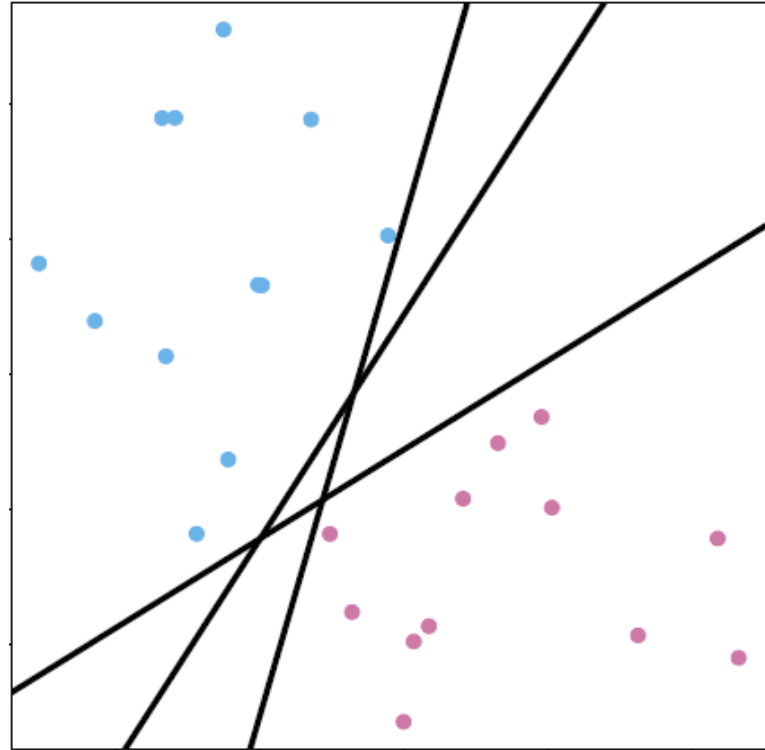


Figure 9.2 , ISL 2013

Mini Quiz:

Which linear decision boundary?

What criteria would you use to choose?

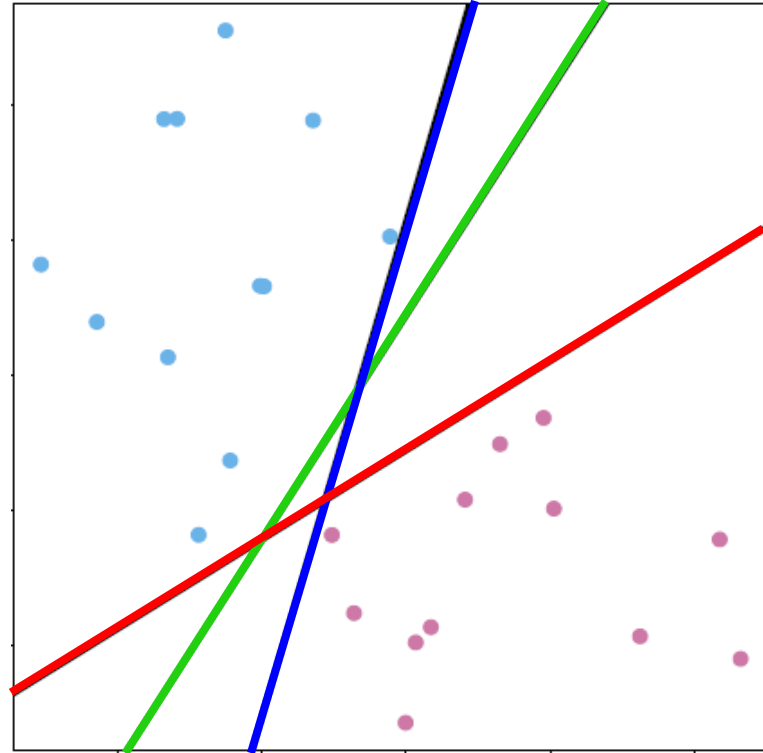


Figure 9.2 , ISL 2013

Separating Hyperplane Classifier

Which linear decision boundary?

Separating hyperplane
“farthest” from training data

→ “Maximal Margin Classifier”

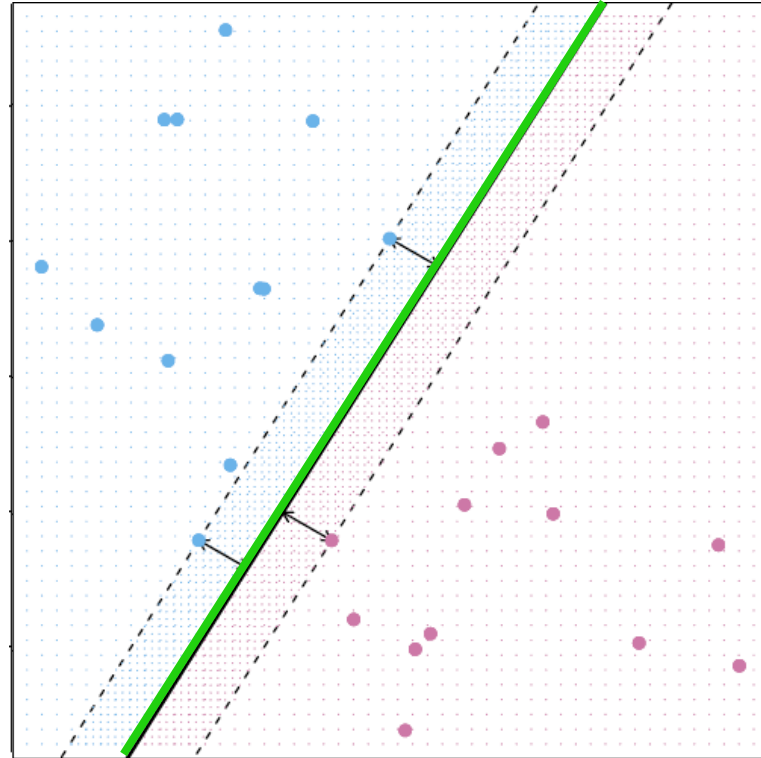


Figure 9.2, 9.3 , ISL 2013

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

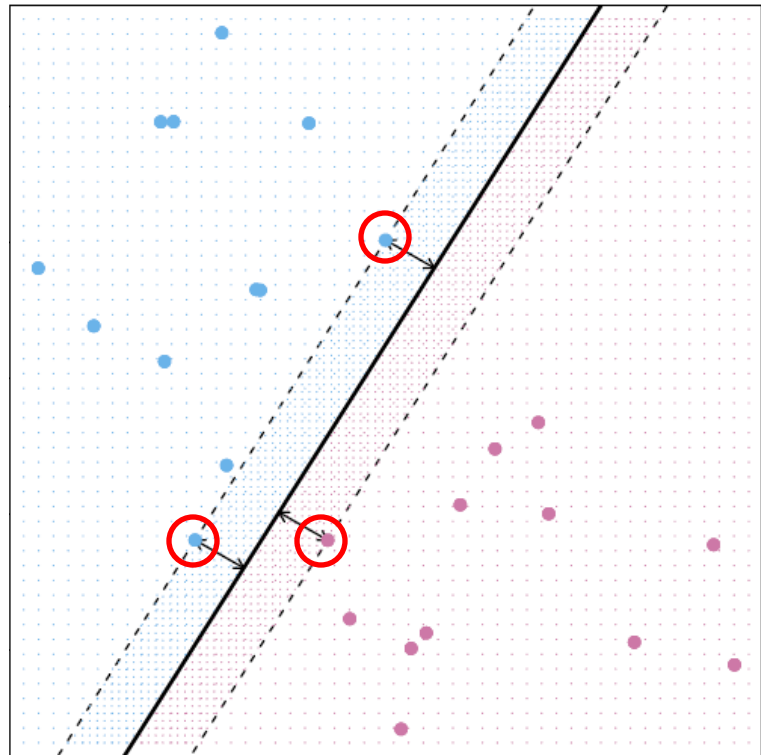


Figure 9.3 , ISL 2013

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 hyperplane not effected

Maximal Margin Classifier

- To find maximal margin hyperplane, solve:

The diagram shows the optimization problem for a Maximal Margin Classifier. It consists of three parts: an objective function to maximize, a constraint to ensure the problem is well-defined, and a set of constraints to ensure all training points are at a distance of at least M from the hyperplane. Blue arrows point from callout boxes to the corresponding parts of the equations.

$$\begin{aligned} &\underset{(\beta_0, \dots, \beta_d)}{\text{maximize}} && M && \leftarrow \text{maximize the margin, } M \\ &\text{subject to} && \sum_{j=0}^d \beta_j^2 = 1 && \leftarrow \text{constraint necessary for well-defined optimization problem} \\ &&& Y^{(i)} \left(\beta_0 + \beta_1 X_1^{(i)} + \dots + \beta_d X_d^{(i)} \right) \geq M, \quad \forall i && \leftarrow \text{all training points must be distance at least } M \text{ from hyperplane} \end{aligned}$$

$Y^{(i)} \in \{-1, 1\}$ 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?

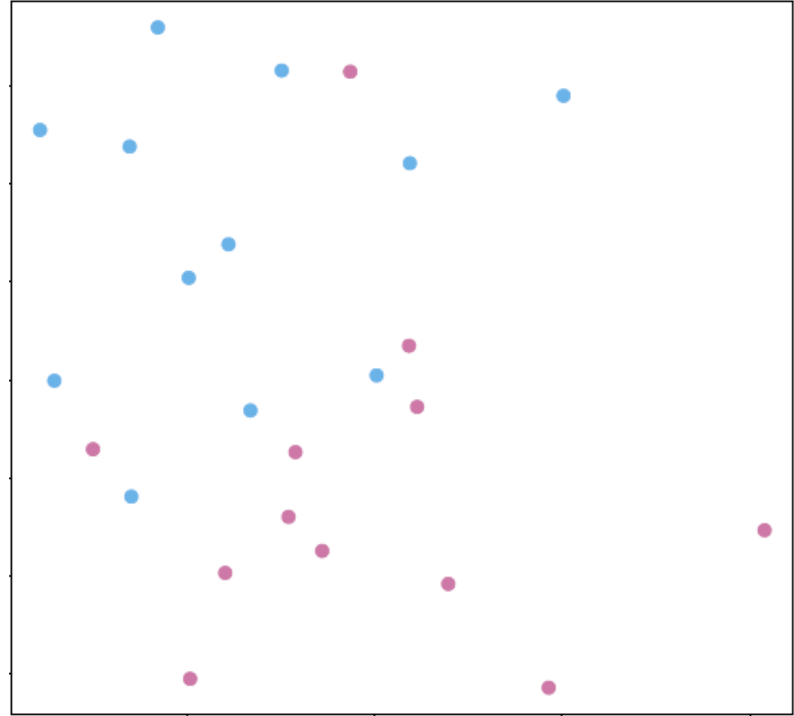


Figure 9.4 , ISL 2013

Maximal Margin Classifier

Disadvantage:

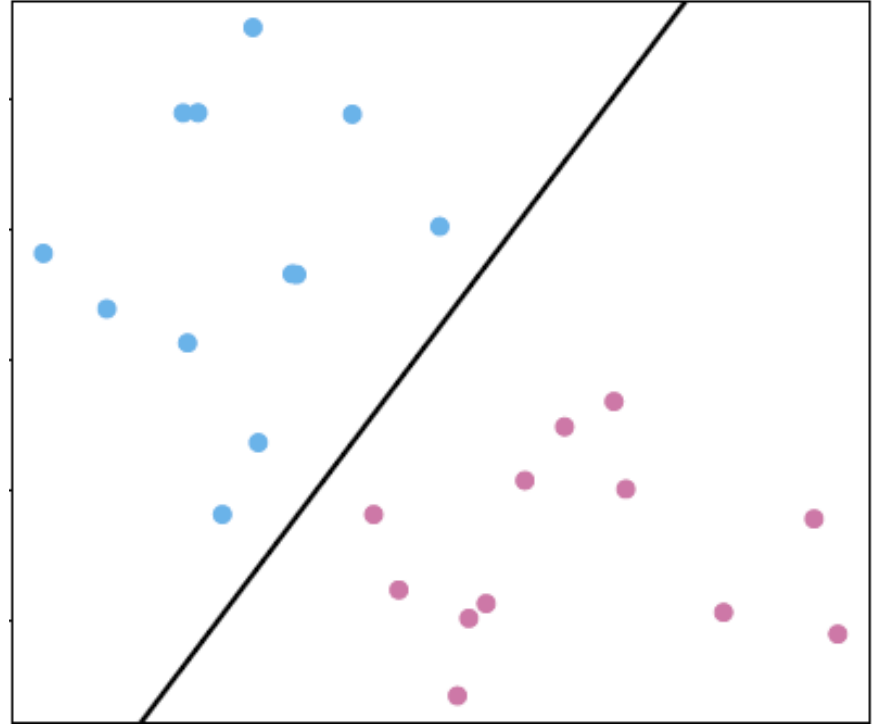


Figure 9.5 , ISL 2013

Maximal Margin Classifier

Disadvantage:

Can be sensitive to individual observations

May overfit training data

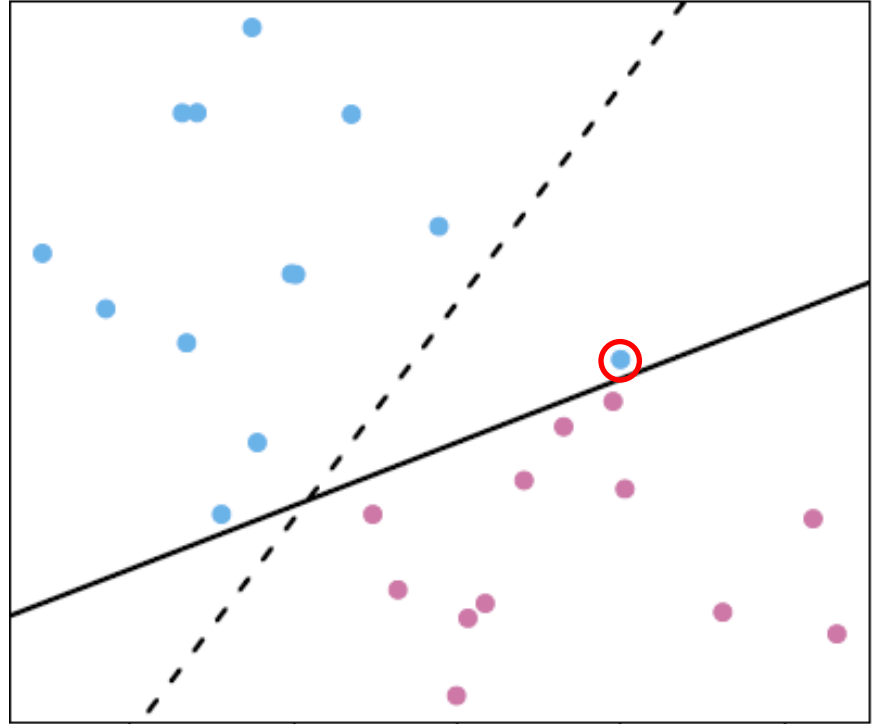


Figure 9.5 , ISL 2013

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

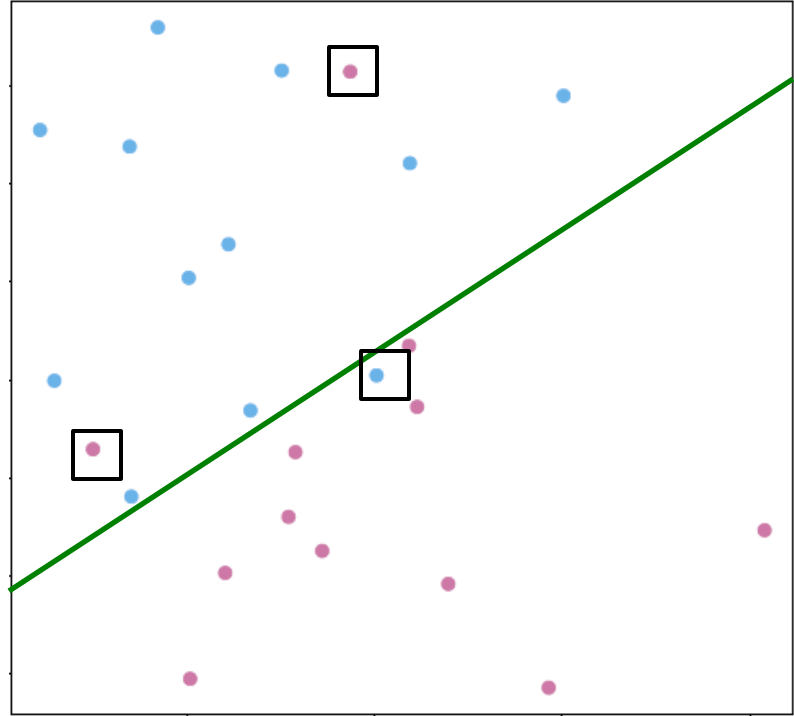


Figure 9.4 , ISL 2013

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

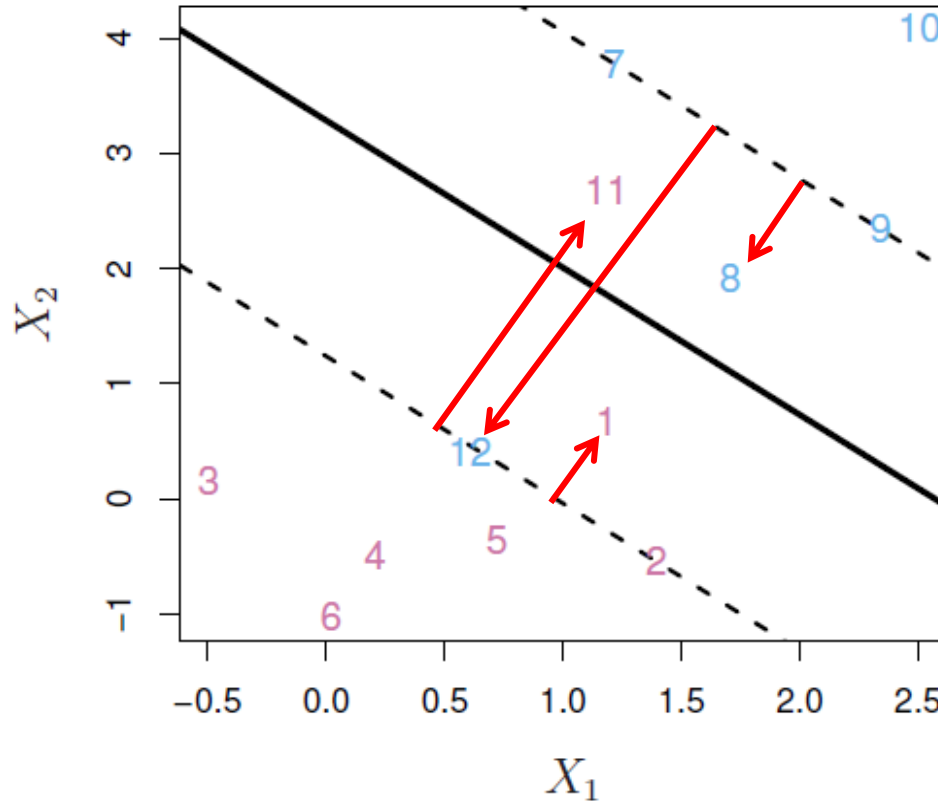


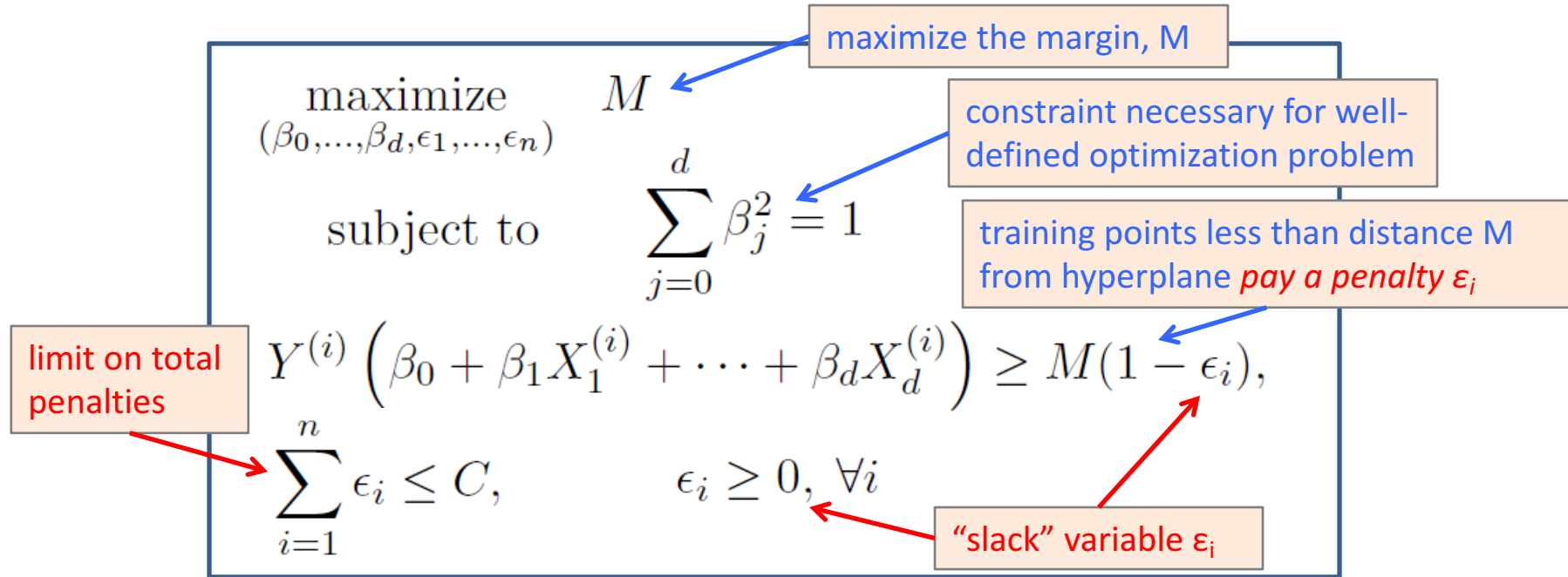
Figure 9.6 , ISL 2013

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:



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

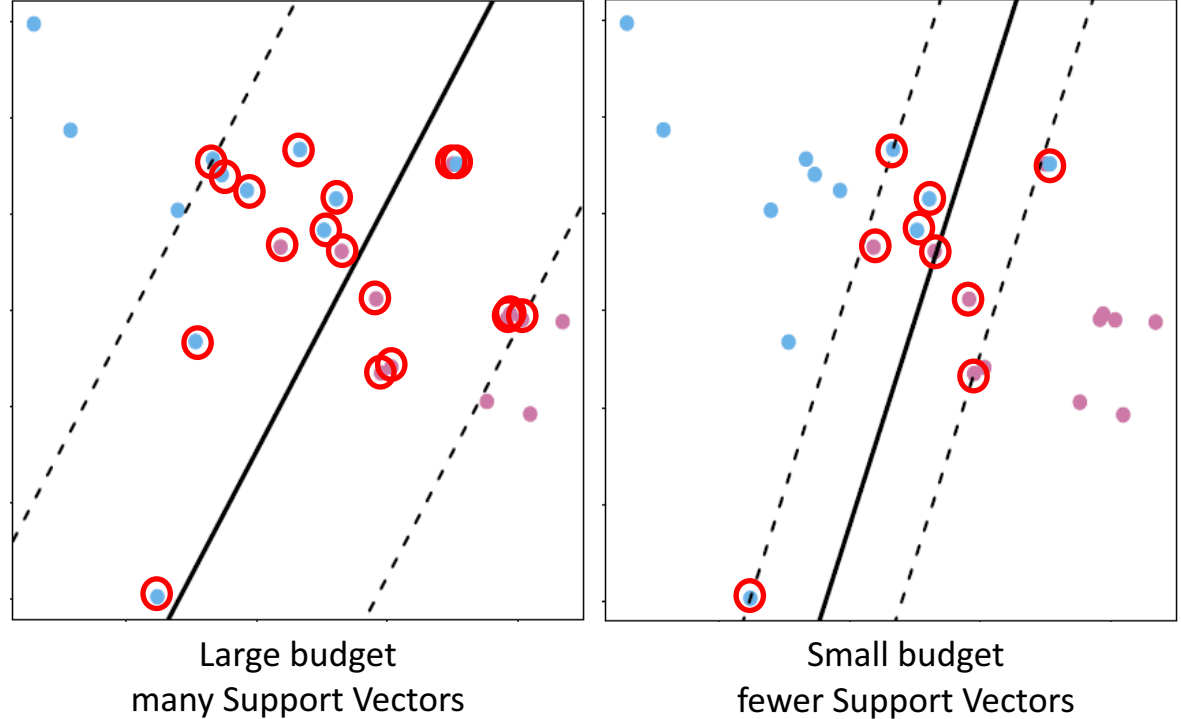


Figure 9.7 , ISL 2013

Support Vector Classifier

Disadvantage:

Linear decision boundary

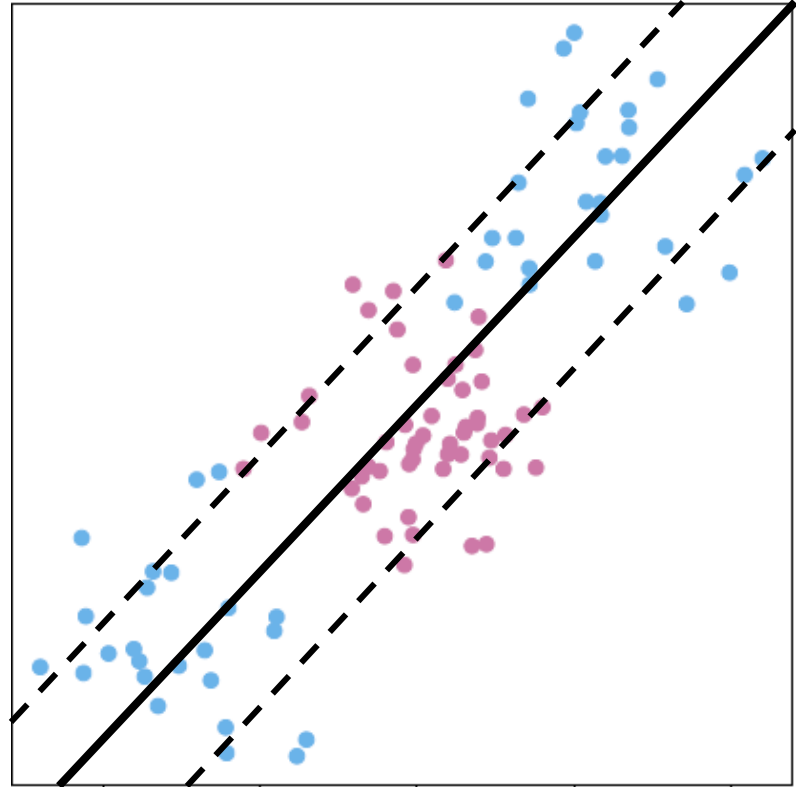
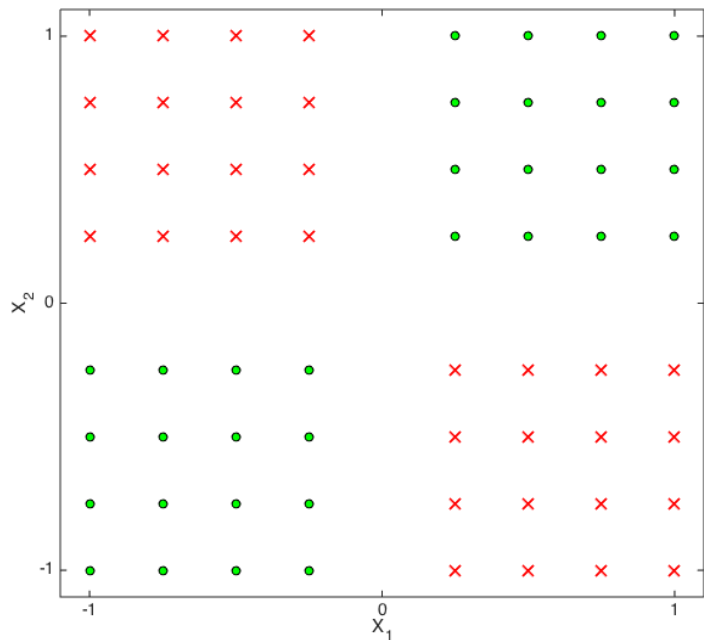


Figure 9.8 , ISL 2013

Expanding Feature Space

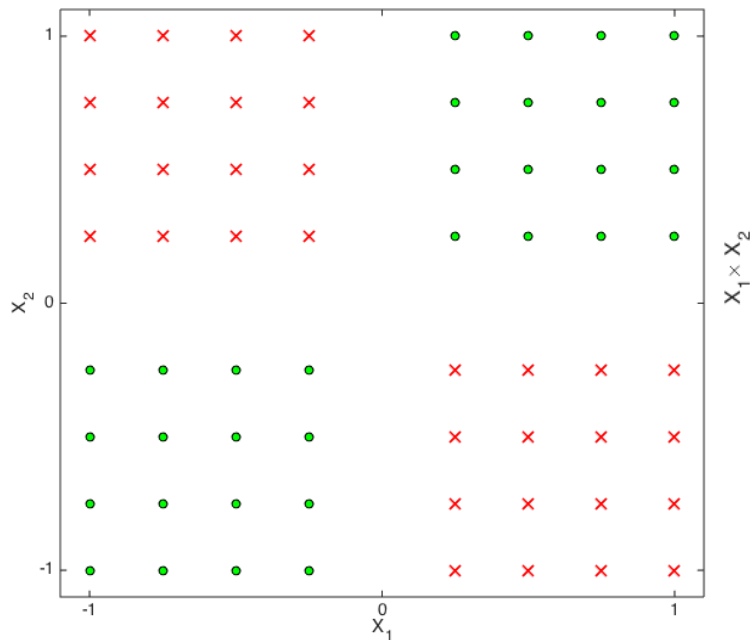


variables: X_1 , X_2

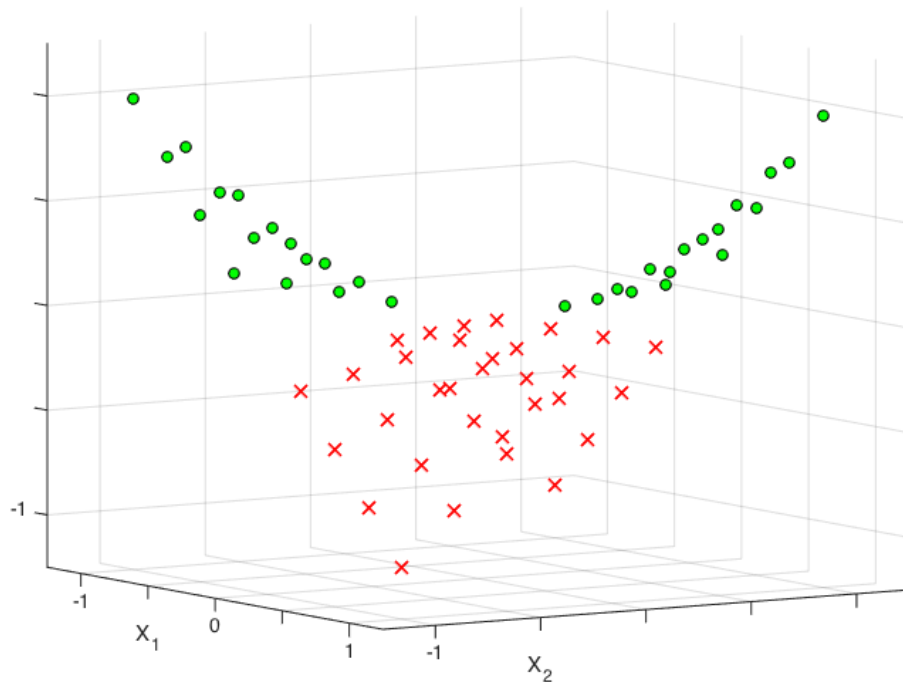
Some data sets are not linearly separable...

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

Expanding Feature Space

- Linear regression \rightarrow 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:

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

- Expand feature space to include $2d$ features:

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

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

Non-linear decision boundary

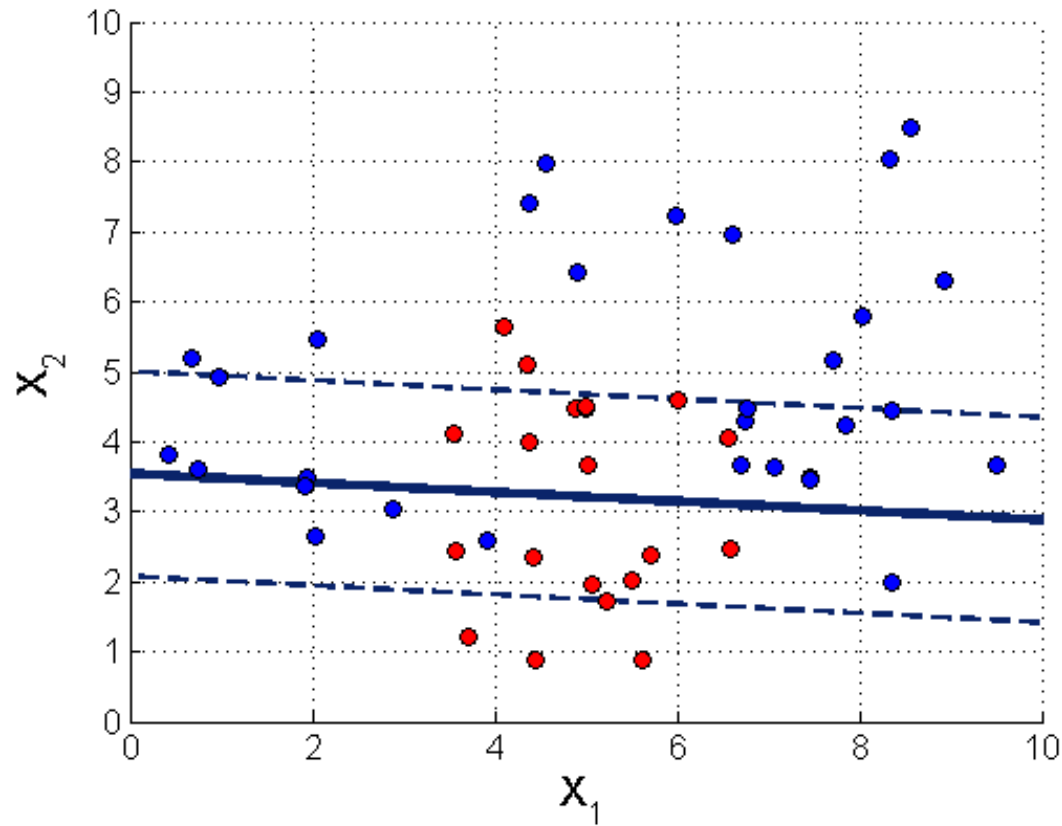
- Decision boundary in enlarged features space is linear:

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

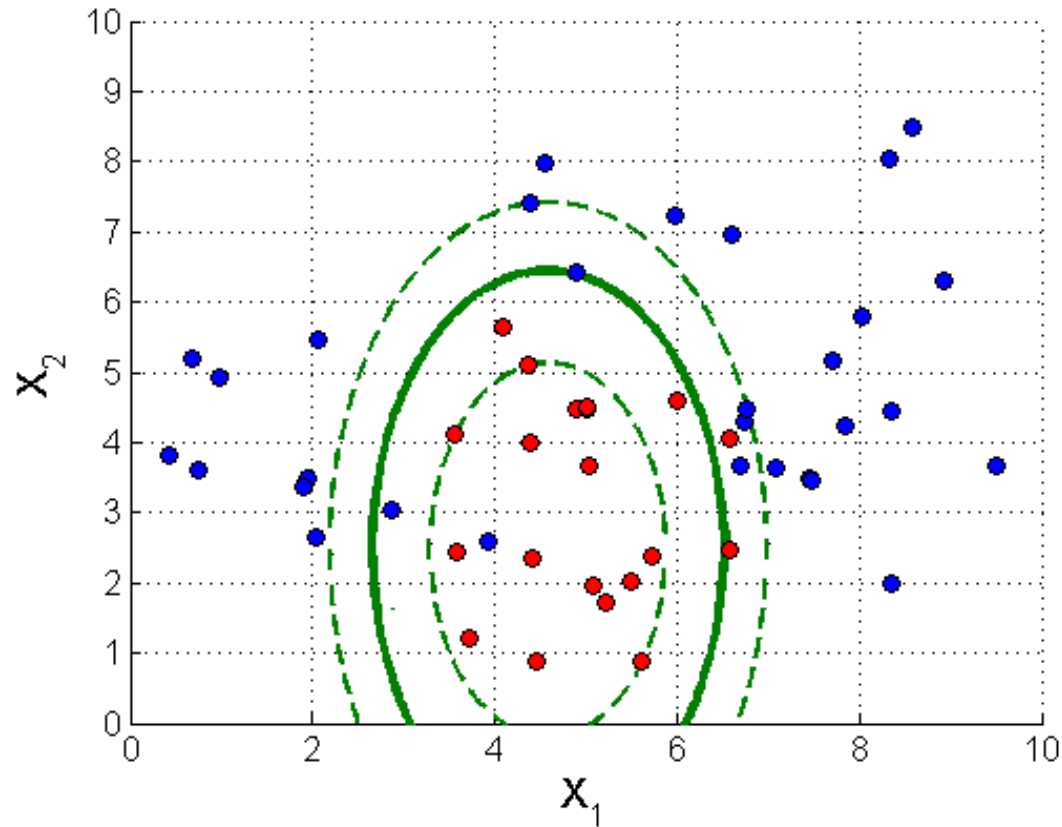
- Decision boundary in enlarged features space is an ellipse *in the original features space*:

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

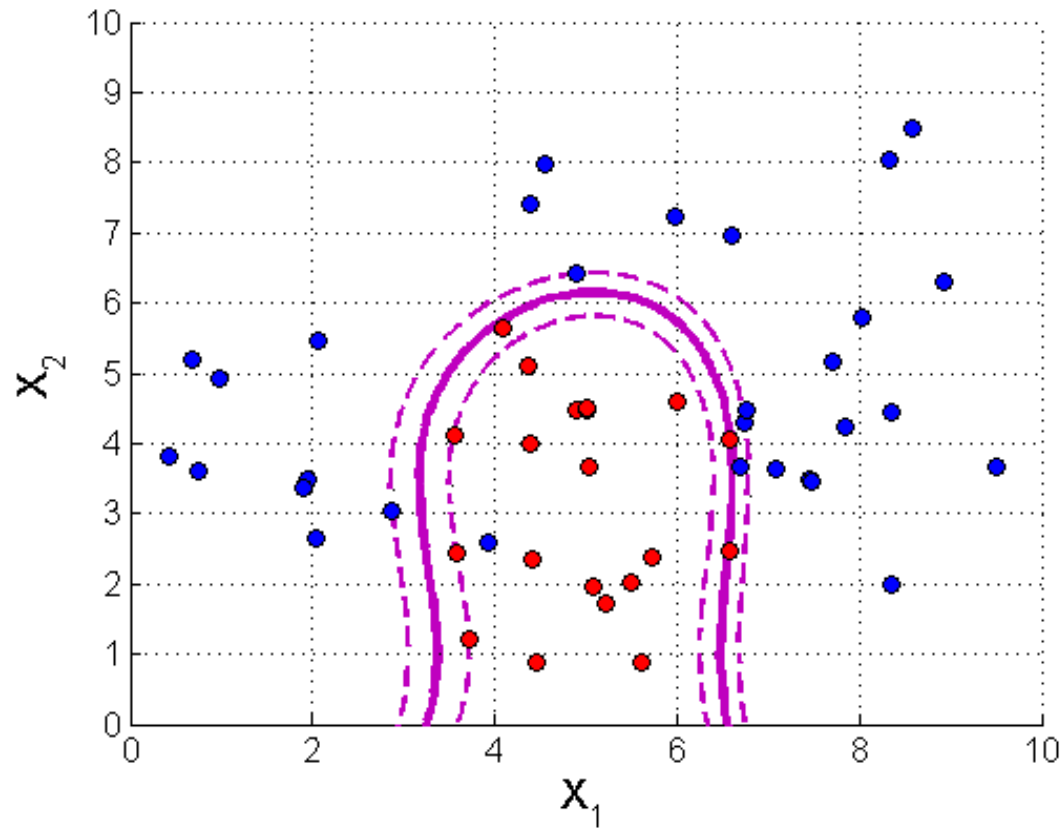
Non-linear Decision Boundary



Non-linear Decision Boundary



Non-linear Decision Boundary

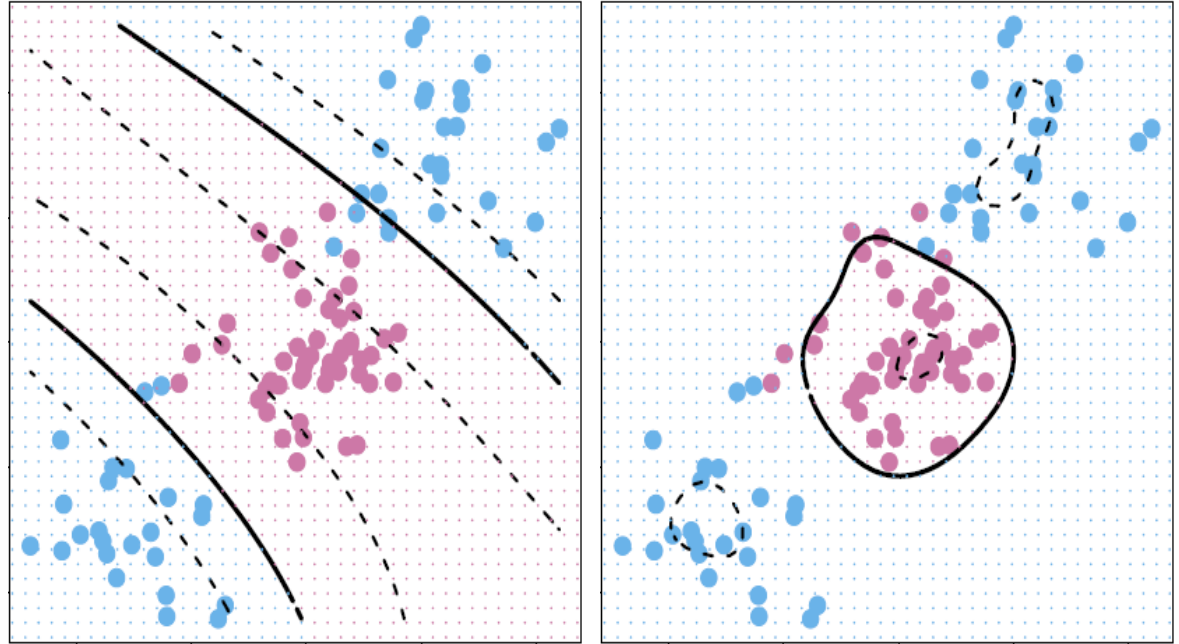


Non-linear Decision Boundary

- 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

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

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

Support Vector Machine

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

$$\phi : \mathbb{R}^d \rightarrow \mathbb{R}^D, \quad d \ll D$$

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

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

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

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

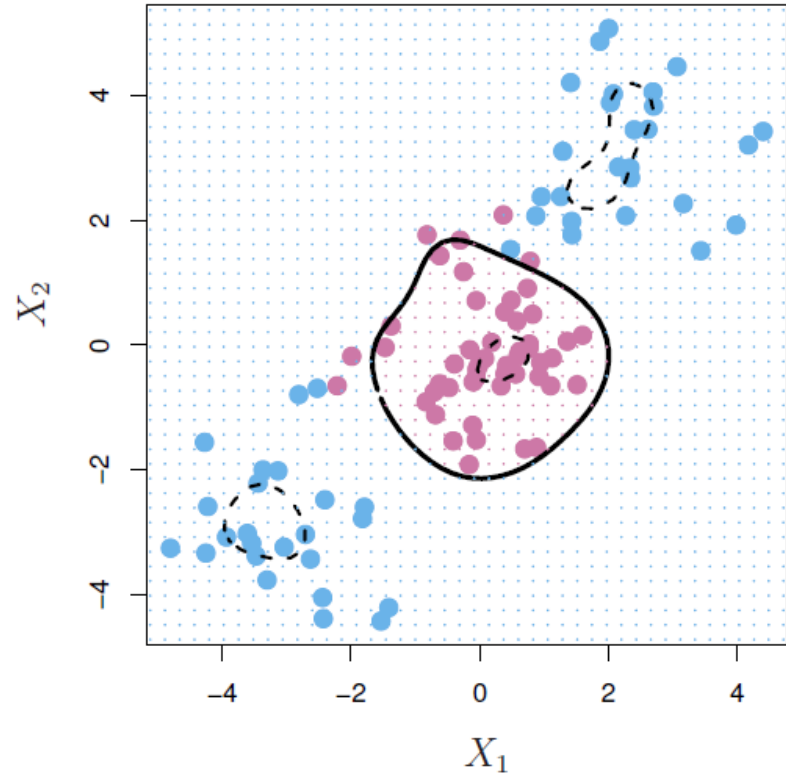
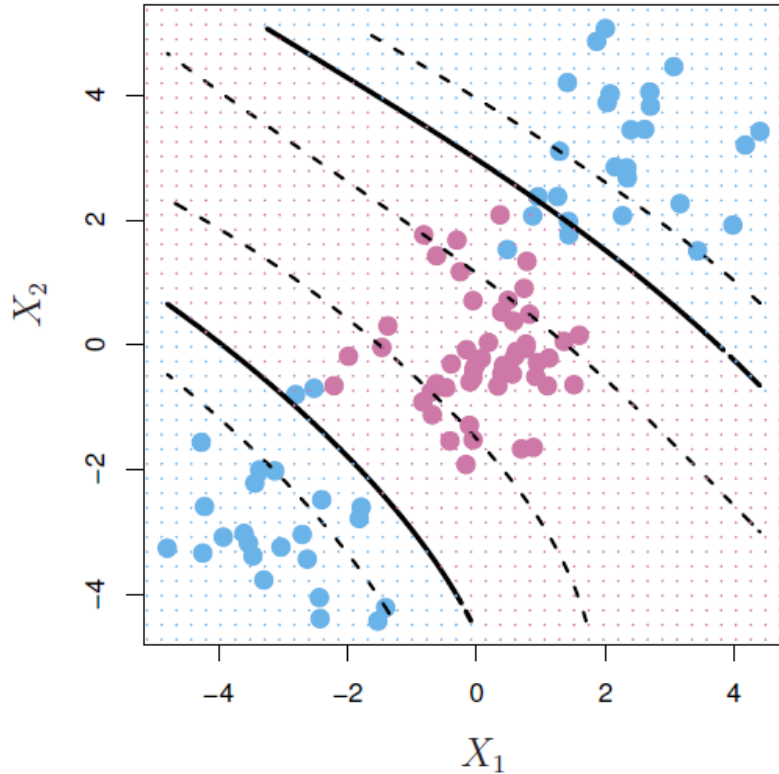
Then

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

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

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

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?

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

- *One-versus-all* classification
 - Fit K SVMs, in which class k represents one class, and the remaining $K - 1$ classes are combined to form the second class
 - Distance to separating hyperplane is a proxy for confidence of the classification
 - For new observation, choose “highest confidence” class to make prediction

Questions?

Imbalanced Classes

- Imbalanced classes: one class (+) occurs significantly more frequently in training set than the other (—)
 - e.g. fraud detection, medical database
- Why is this a problem?
 - Algorithms perform best when trained on roughly even numbers of observations in each class
 - Poor performance on underrepresented class

Imbalanced Classes

- How can we improve performance when we have imbalanced classes?
 - Collect more data for underrepresented class
 - Weighting of classes
 - Sampling methods

Imbalanced Classes

- Weighting of classes: applying different weights to false negatives in cost function
 - e.g. in SVM, larger weights to penalties for violations of margin for class (−) than for class (+):

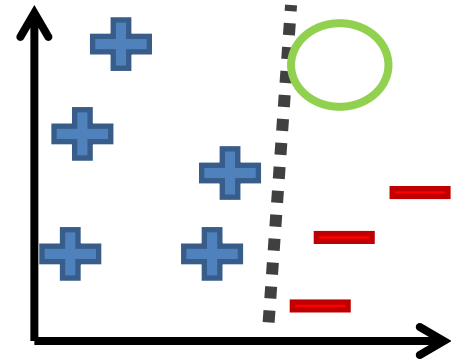
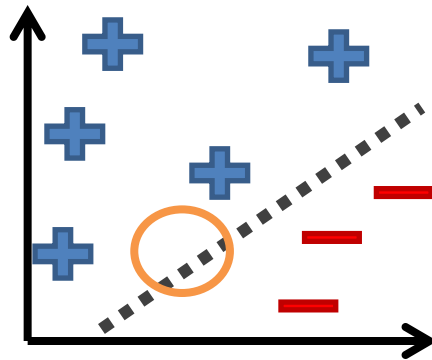
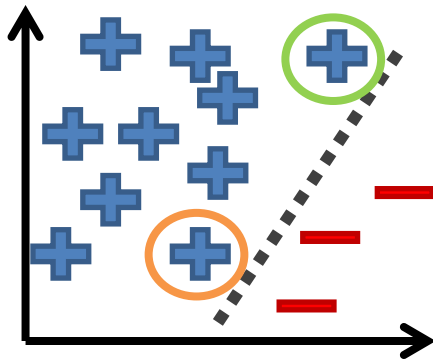
$$\sum_{i=1}^n \epsilon_i < C \quad \text{becomes}$$
$$\left(\sum_{i: Y^{(i)}=(+)} \epsilon_i \right) + \omega \left(\sum_{i: Y^{(i)}=(-)} \epsilon_i \right) < C$$

Imbalanced Classes

- Sampling Methods – modify set of training observations to make classes more even
- Balance class labels by
 - Undersampling class (+)
 - Oversampling class (–)

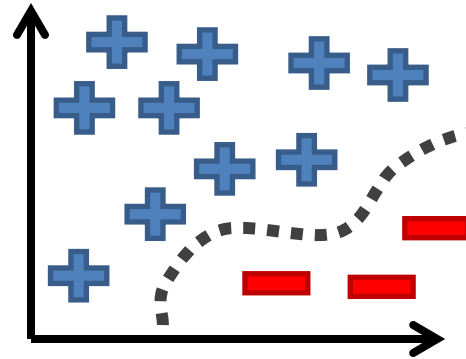
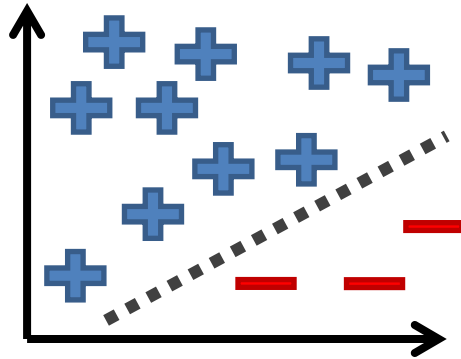
Imbalanced Classes

- Disadvantages of over/under sampling
 - Undersampling class (+) may remove important training observations



Imbalanced Classes

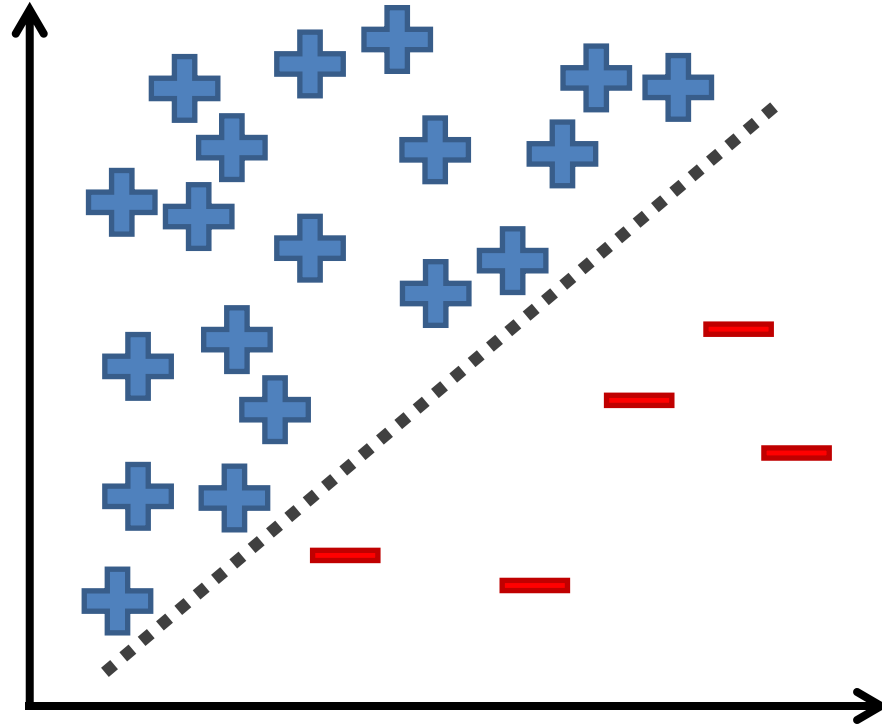
- Disadvantages of over/under sampling
 - Oversampling class (—) may result in over fitting



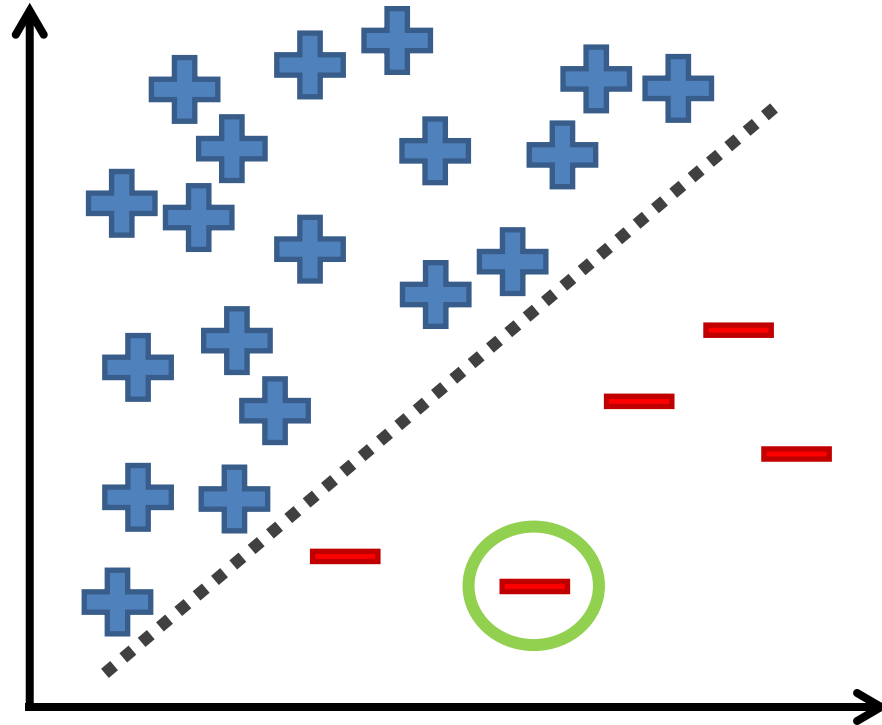
Imbalanced Classes

- Synthetic Minority Oversampling
 - Method for oversampling class $(-)$ that generates new minority observations by perturbing existing minority observations:
 1. Selects observation $X^{(-)}$ in class $(-)$ at random
 2. Finds k nearest neighbors of $X^{(-)}$ selects one of the neighbors $X_{nn}^{(-)}$ at random
 3. New sample $X_{new}^{(-)}$ is a perturbation of $X^{(-)}$ along the direction $X_{nn}^{(-)} - X^{(-)}$

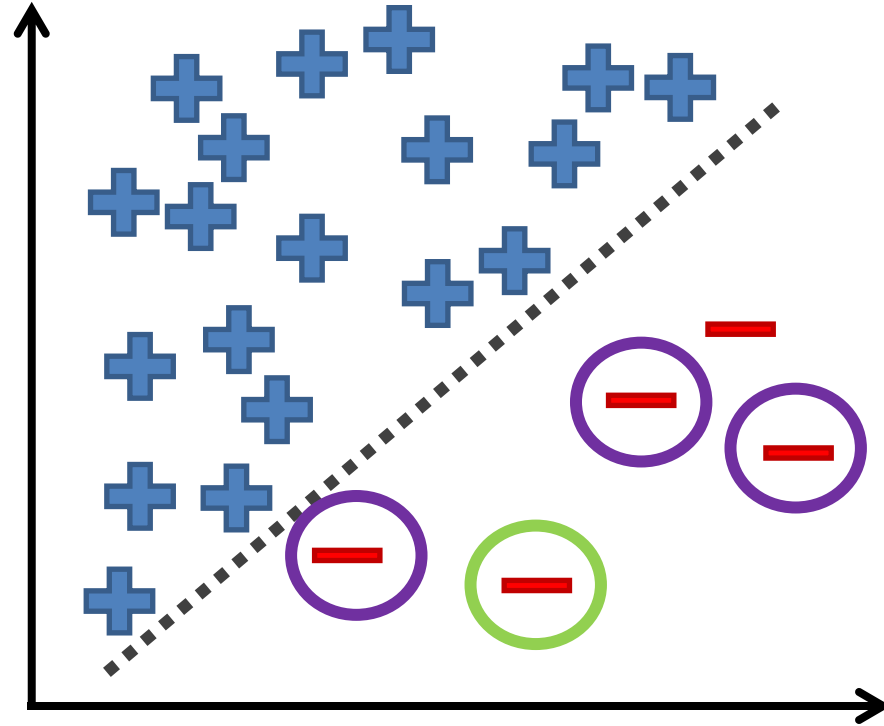
Imbalanced Classes



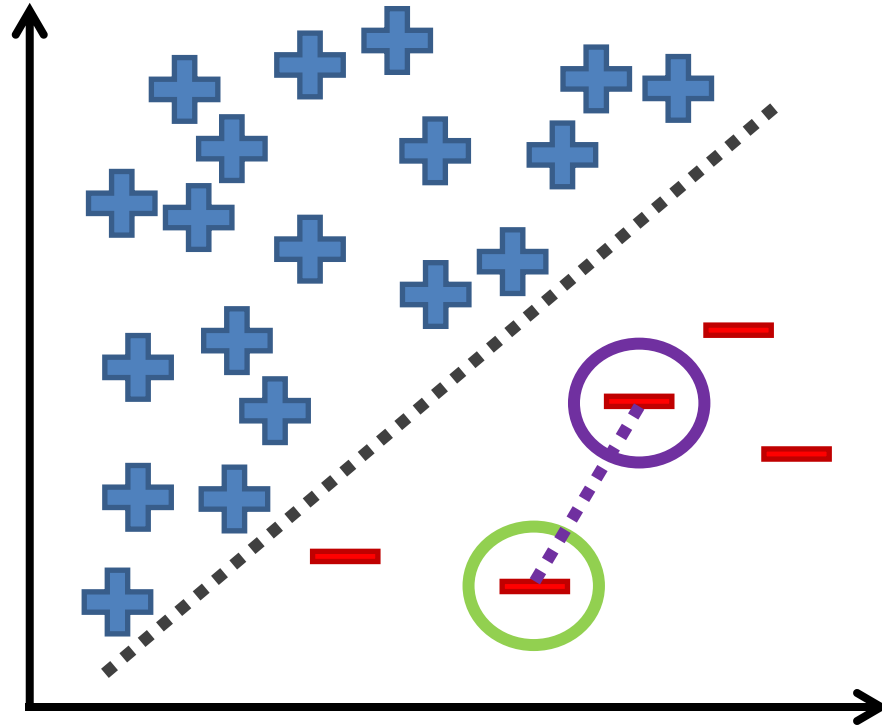
Imbalanced Classes



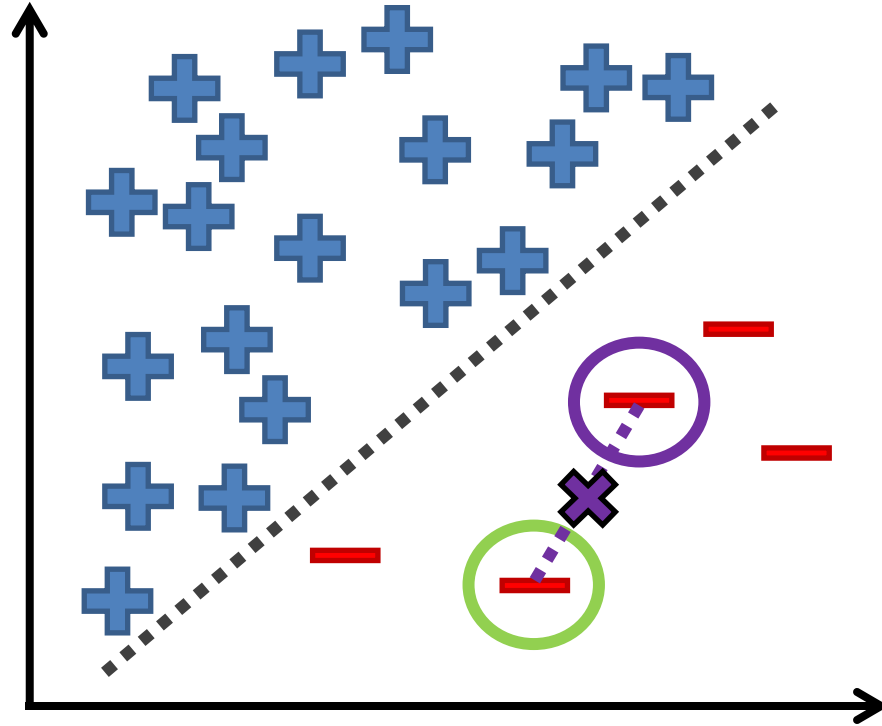
Imbalanced Classes



Imbalanced Classes



Imbalanced Classes



Imbalanced Classes

