DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

October 18 2018

Logistics

- HW2 is due on Friday, Oct. 19 at midnight
- Project proposal is due on Oct 22 (1 page on Gradescope)
  - Project Title
  - Problem Description
    - What is the machine learning problem you are trying to solve?
  - Dataset
    - Link to data, brief description, number of records, feature dimensionality
    - At least 10,000 records
  - Approach
    - Data exploration
    - Normalization if any
    - Feature selection if any
    - Machine learning models (several) you will try for your problem
    - Methodology for splitting into training and testing, cross validation
    - Language and packages you plan to use
    - Metrics, how you will evaluate your models

Review

- Ensemble learning are powerful learning methods
- Bagging uses bootstrapping (with replacement), trains T models, and averages their prediction
  - Random forests vary training data and feature set at each split
- Boosting is an ensemble of weak learners that emphasizes mis-predicted examples
  - AdaBoost has great theoretical and experimental performance
  - Can be used with linear models or simple decision trees

Outline

- Quick review on ensemble learning
- SVM
  – Linearly separable data
    - Separating hyperplanes
    - Maximum margin classifier
  – Non-separable data
    - Support vector classifier
- Non-linear decision boundaries
  – Kernels and Radial SVM

Ensemble Learning

Consider a set of classifiers $h_1, \ldots, h_L$

Idea: construct a classifier $H(\mathbf{x})$ that combines the individual decisions of $h_1, \ldots, h_L$

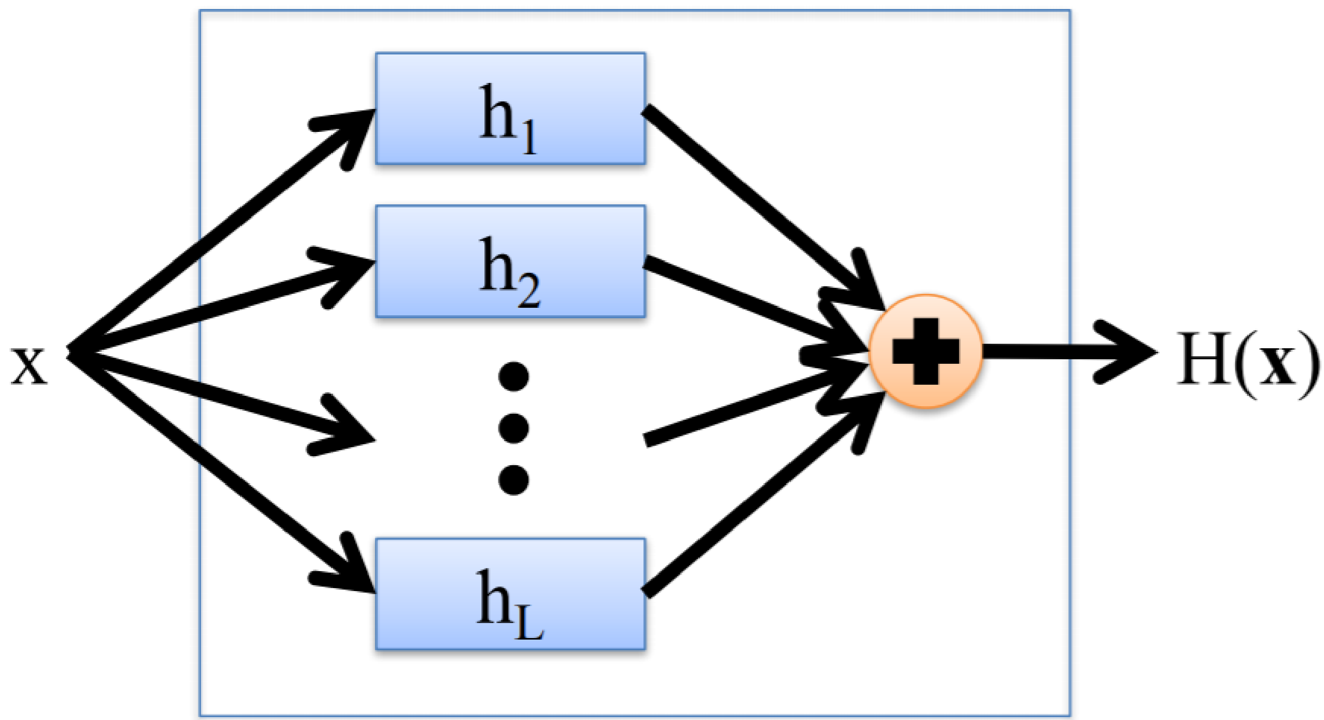
- e.g., could have the member classifiers vote, or
- e.g., could use different members for different regions of the instance space

Successful ensembles require **diversity**

- Classifiers should make different mistakes
- Can have different types of base learners

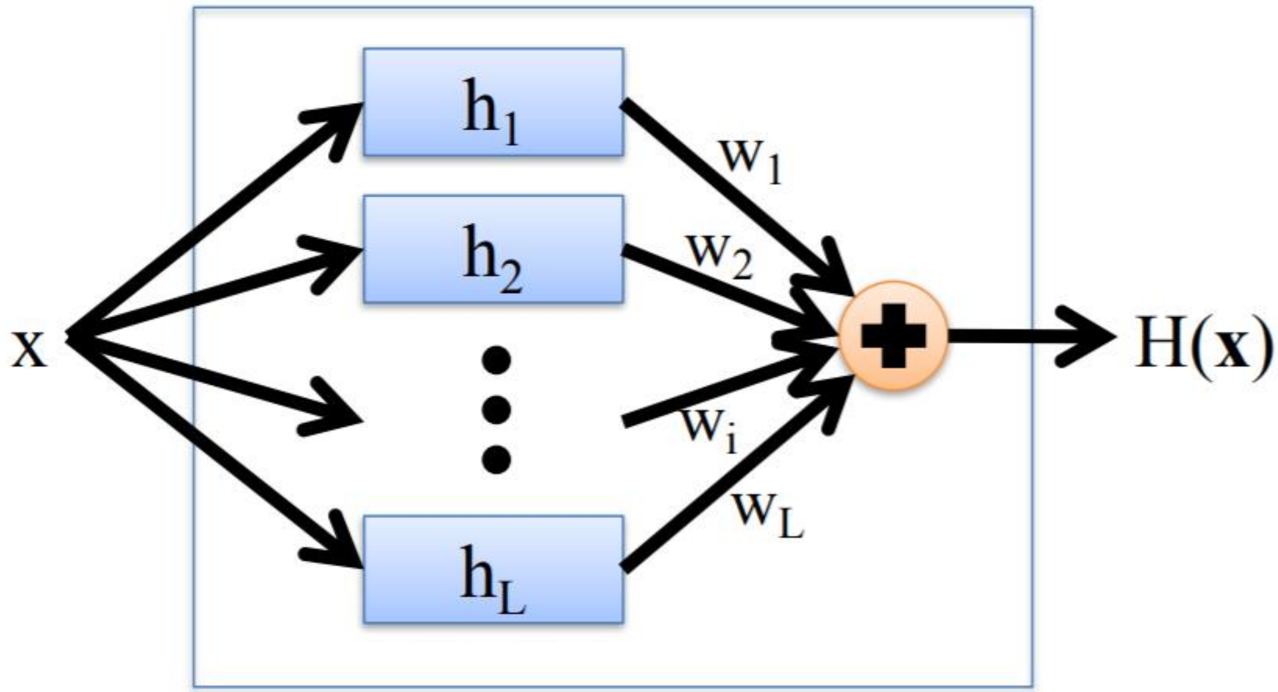
- Bagging
- Boosting

Combining Classifiers: Averaging



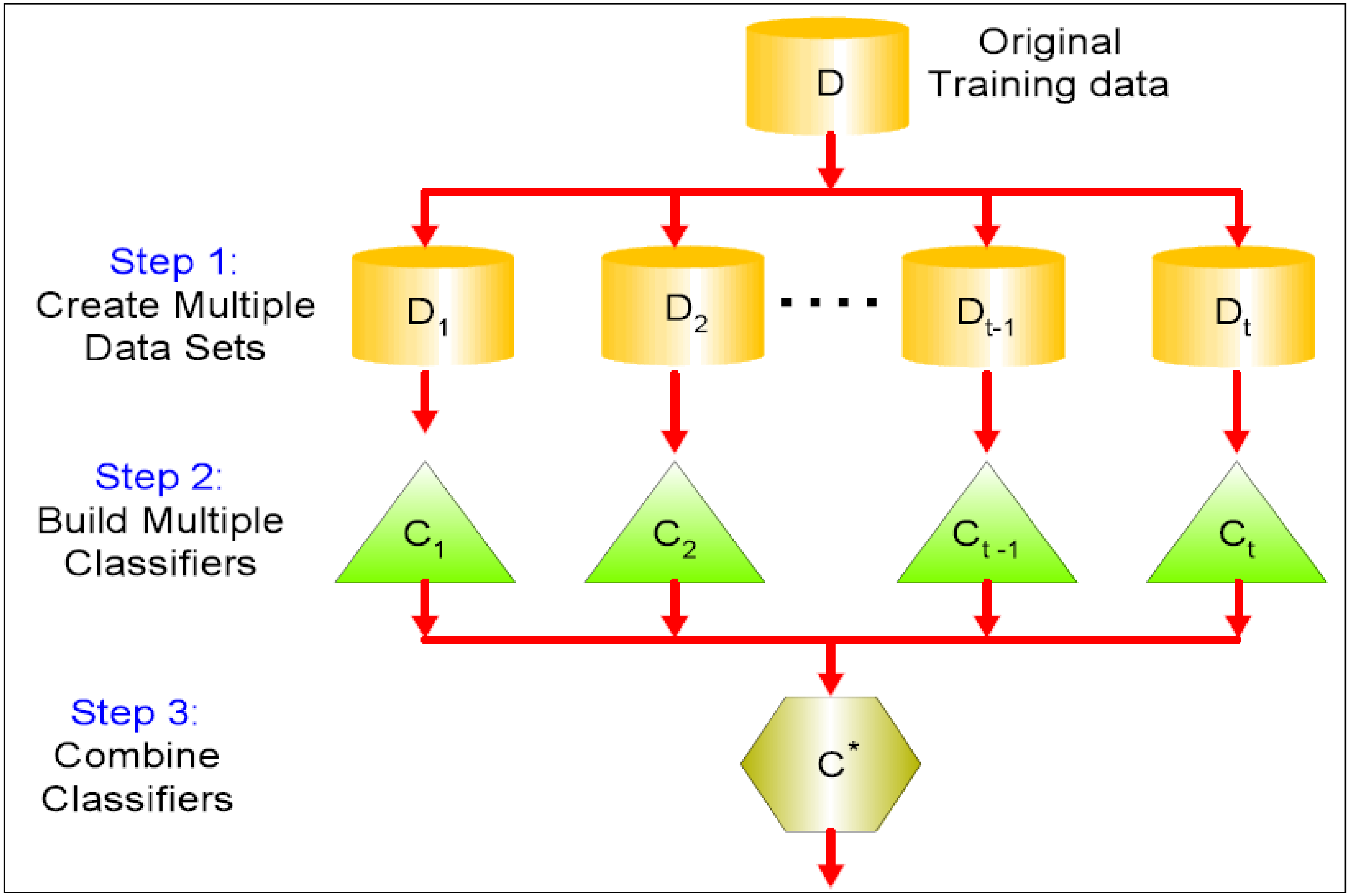
- Final hypothesis is a simple vote of the members

Combining Classifiers: Weighted Averaging



- Coefficients of individual members are trained using a validation set

Bagging



Bootstrap samples

RF: subset of features at each split

Majority Votes

Evaluating Bagging

- Sampling with replacement

Training Data

Data ID

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Sample each training point with probability 1/n
- Out-Of-Bag (OOB) observation: point not in sample
  - For each point: prob $(1-1/n)^n$
  - About 1/3 of data
  - OOB error: error on OOB samples
- OOB average error
  - Compute across all models in Ensemble
  - Use instead of Cross-Validation error

AdaBoost

- A meta-learning algorithm with great theoretical and empirical performance
- Turns a base learner (i.e., a "weak hypothesis") into a high performance classifier
- Creates an ensemble of weak hypotheses by repeatedly emphasizing mispredicted instances

Adaptive Boosting
Freund and Schapire 1997

AdaBoost

INPUT: training data $X, y = \{(x^{(i)}, y^{(i)})\}, \mathrm{i} = 1 \ldots \mathrm{n}$
the number of iterations T

1: Initialize a vector of n uniform weights $\mathbf{w}_1 = \left[\frac{1}{n}, \ldots, \frac{1}{n}\right]$
2: **for** $t = 1, \ldots, T$
3: Train model h_t on X, y with instance weights $\mathbf{w}_t$
4: Compute the weighted training error rate of h_t:

$$\epsilon_t = \sum_{i: y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$

5: Choose $\beta_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y^{(i)} h_t(x^{(i)})), i = 1, \ldots, n$$

7: Normalize $\mathbf{w}_{t+1}$ to be a distribution:

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^{n} w_{t+1,j}} \quad \forall i = 1, \ldots, n$$

8: **end for**
9: **Return** the hypothesis

$$H(\mathbf{x}) = \mathrm{sign}\left(\sum_{t=1}^{T} \beta_t h_t(\mathbf{x})\right)$$

Base Learner Requirements

- AdaBoost works best with "weak" learners
  - Should not be complex
  - Typically high bias classifiers
  - Works even when weak learner has an error rate just slightly under 0.5 (i.e., just slightly better than random)
    - Can prove training error goes to 0 in $O(\log n)$ iterations

- Examples:
  - Decision stumps (1 level decision trees)
  - Depth-limited decision trees
  - Linear classifiers

Outline

- Quick review on ensemble learning
- SVM
  - Linearly separable data
    - Separating hyperplanes
    - Maximum margin classifier
  - Non-separable data
    - Support vector classifier
- Non-linear decision boundaries
  - Kernels and Radial SVM

Hyperplane

- Line (2-dimensions): $\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$
- Hyperplane (d-dimensions): $\theta_0 + \theta_1 x_1 + \cdots \theta_d x_d = 0$

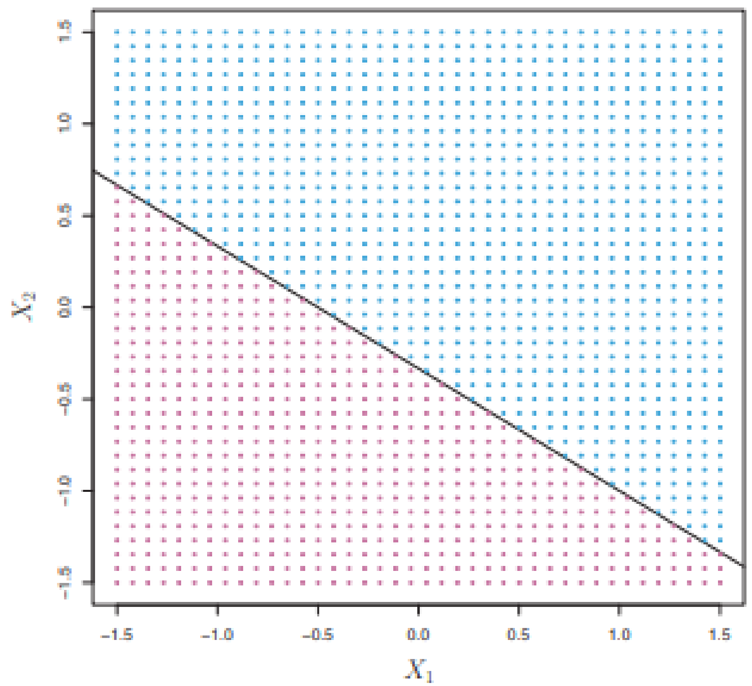


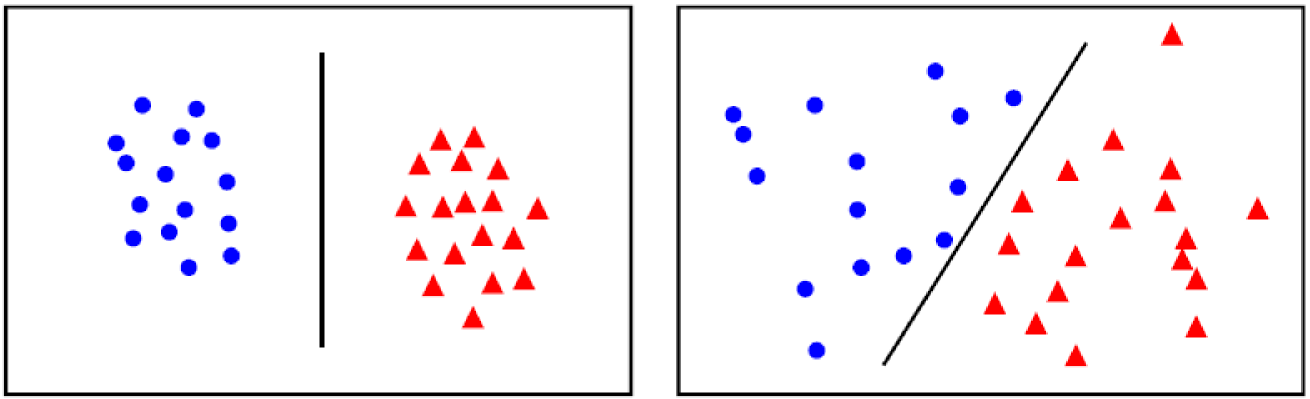
FIGURE 9.1. *The hyperplane* $1 + 2X_1 + 3X_2 = 0$ *is shown. The blue region is the set of points for which* $1 + 2X_1 + 3X_2 > 0$*, and the purple region is the set of points for which* $1 + 2X_1 + 3X_2 < 0$.

Notation

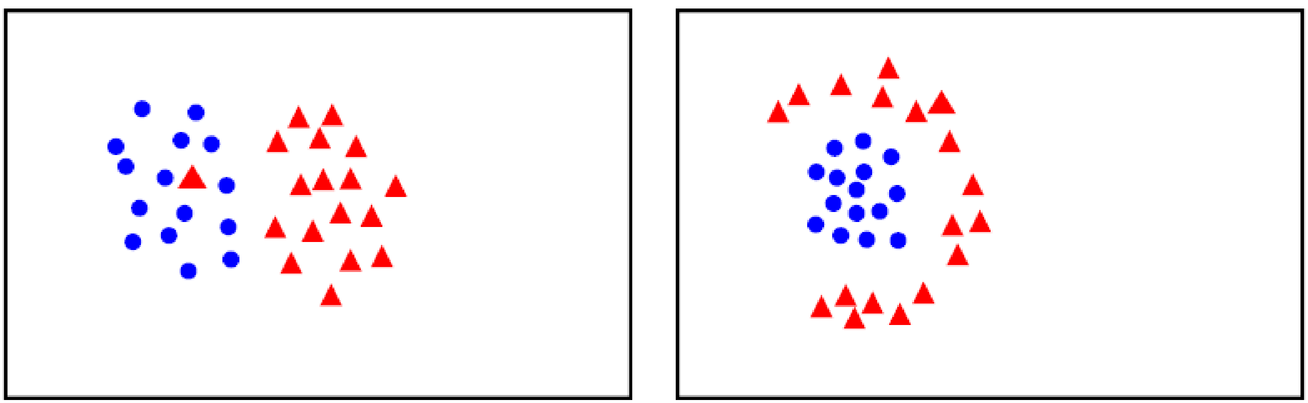
- Training data $x^{(1)}, \dots, x^{(n)}$ with $x^{(i)} = \left(x_1^{(i)}, \dots, x_d^{(i)}\right)^{\mathrm{T}}$
- Labels are from 2 classes: $\mathrm{y}^{(\mathrm{i})} \in \{-1,1\}$
- Goal:
  - Build a model to classify training data
  - Test it on new data $x_1', \dots, x_n'$ to predict labels $y_1', \dots, y_n'$

Linear separability

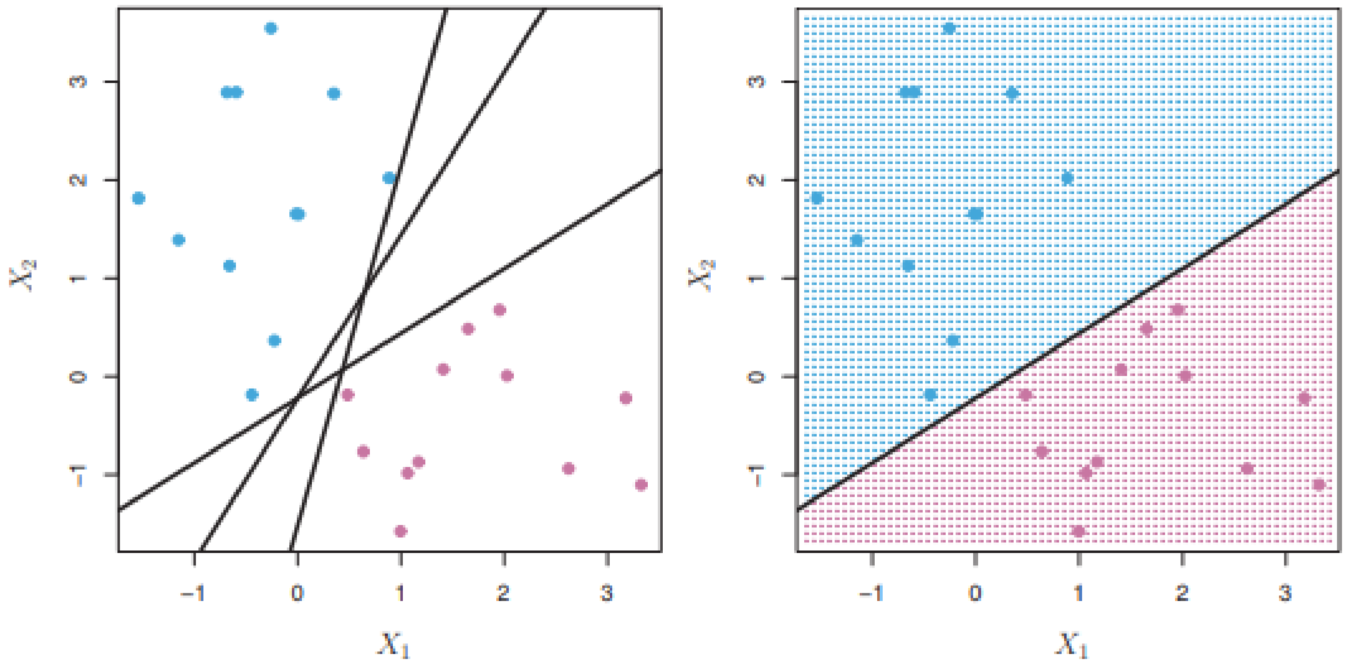
linearly separable



not linearly separable



Separating hyperplane



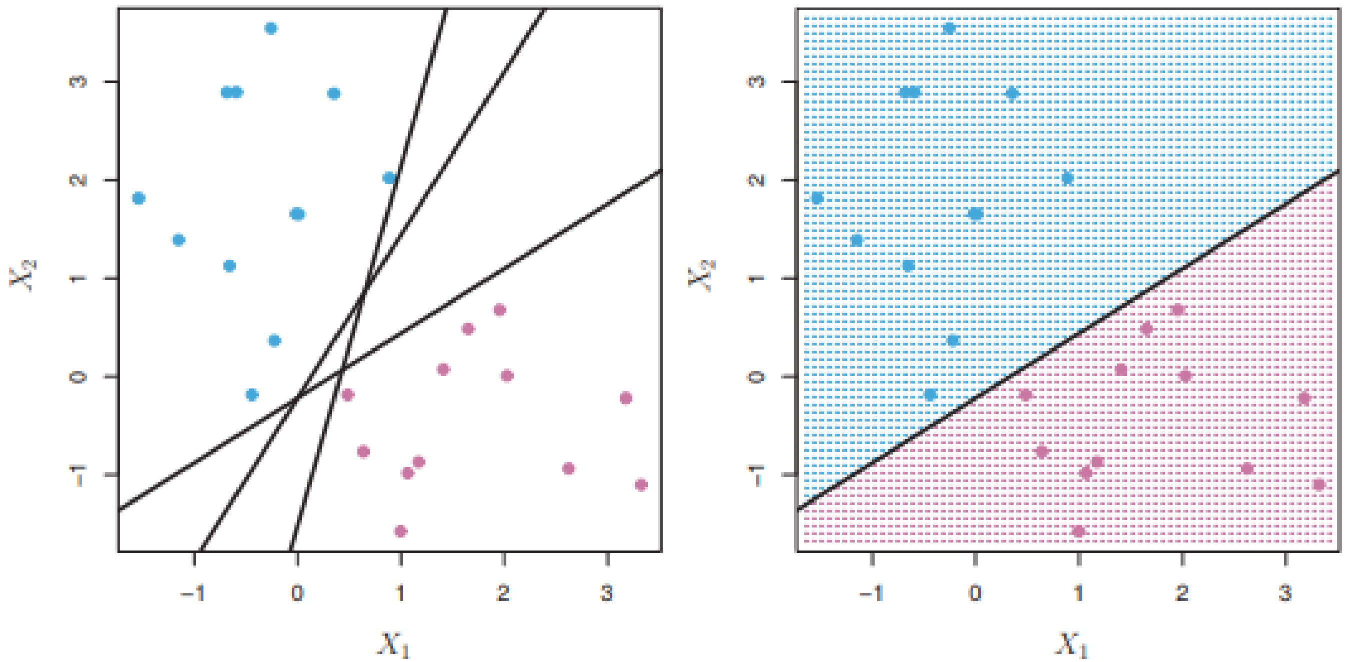
$$\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)} > 0 \ \text{ if } y^{(i)} = 1$$

$$\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)} < 0 \ \text{ if } y^{(i)} = -1$$

For all training data $x^{(i)}, y^{(i)}$ $i \in \{1, \ldots, n\}$

Perfect separation between the 2 classes

Separating hyperplane



$$y^{(i)}(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}) > 0$$

For all training data $x^{(i)}, y^{(i)}$, $i \in \{1, \ldots, n\}$

From separating hyperplane to classifier

- Training data $x^{(1)}, \dots, x^{(n)}$ with $x^{(i)} = \left(x_1^{(i)}, \dots, x_d^{(i)}\right)^{\mathrm{T}}$
- Labels are from 2 classes: $y^{(i)} \in \{-1,1\}$
- Let $\theta_1, \dots, \theta_d$ such that:

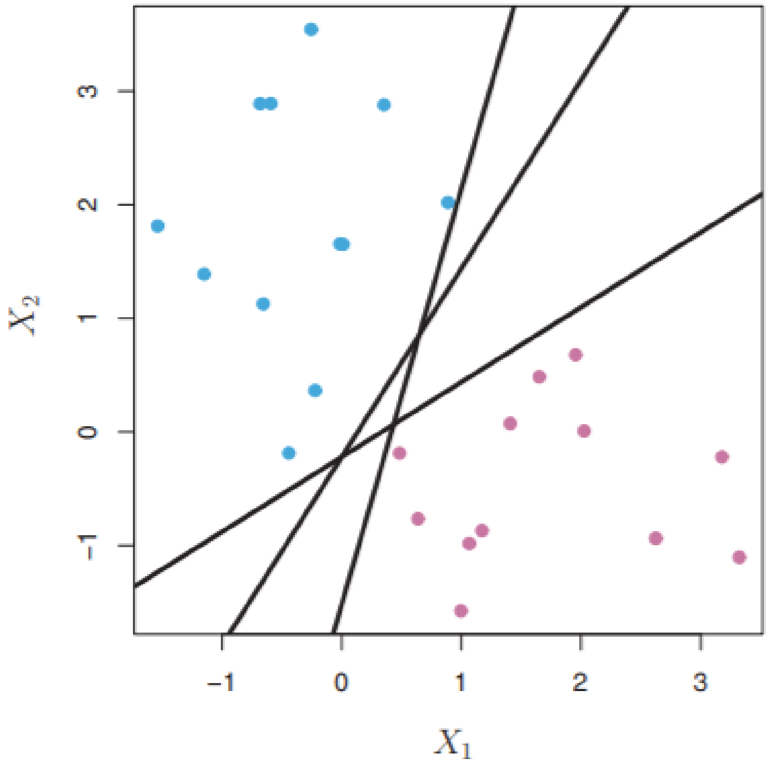
$$y^{(i)}(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}) > 0$$

- Classifier

$$f(z) = \mathrm{sign}(\theta_0 + \theta_1 z_1 + \cdots \theta_d z_d) = \mathrm{sign}(\theta^T \mathrm{z})$$

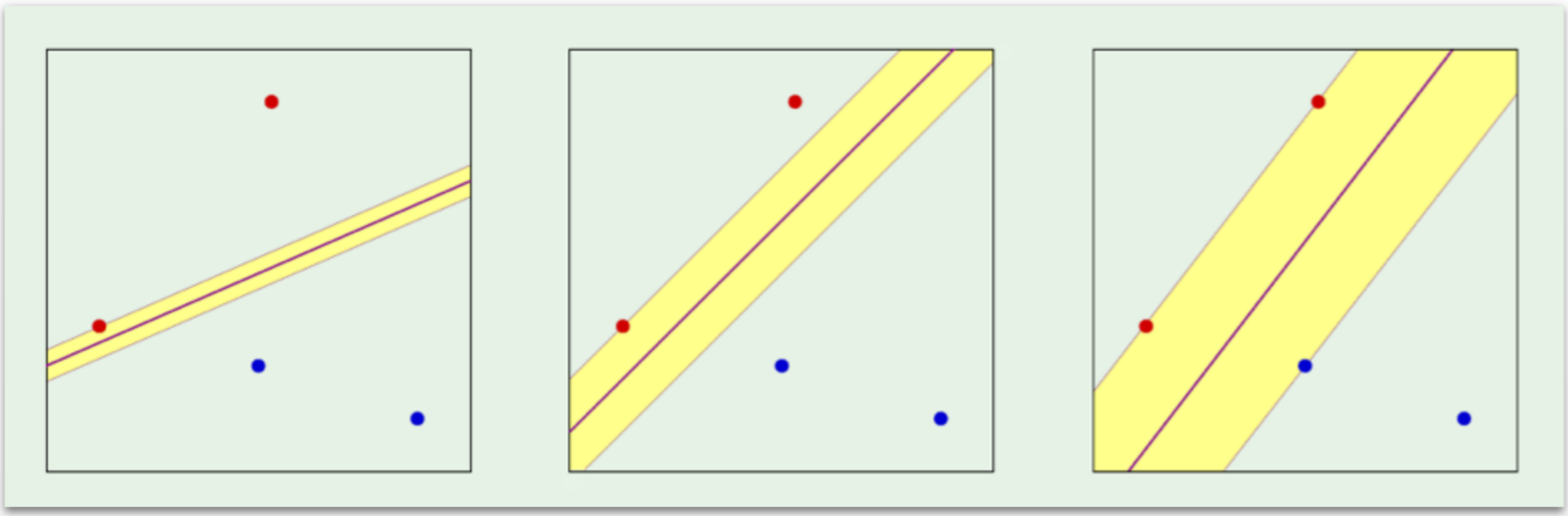
- Test on new point x'
  - If $f(x') > 0$ predict y'$= 1$
  - Otherwise predict y'$= -1$

Separating hyperplane



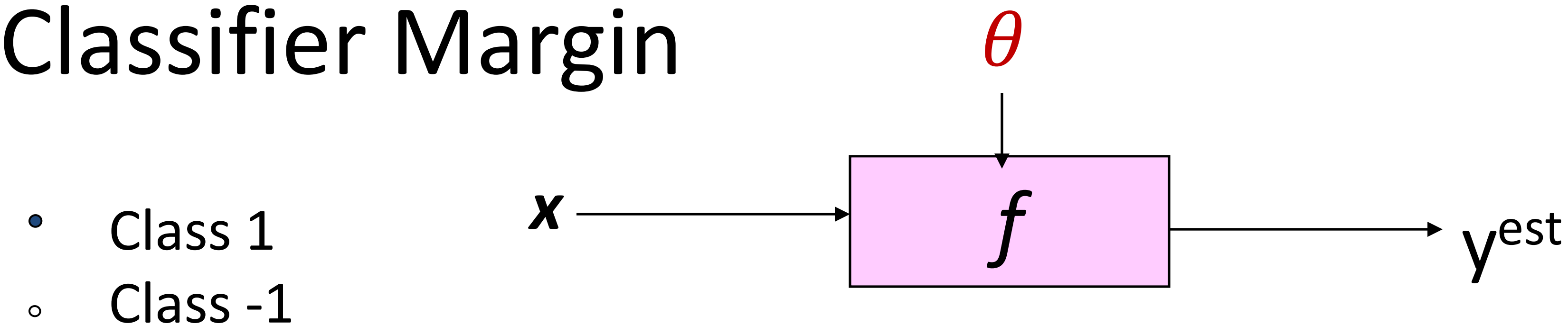
- If a separating hyperplane exists, there are infinitely many
- Which one should we choose?

Intuition

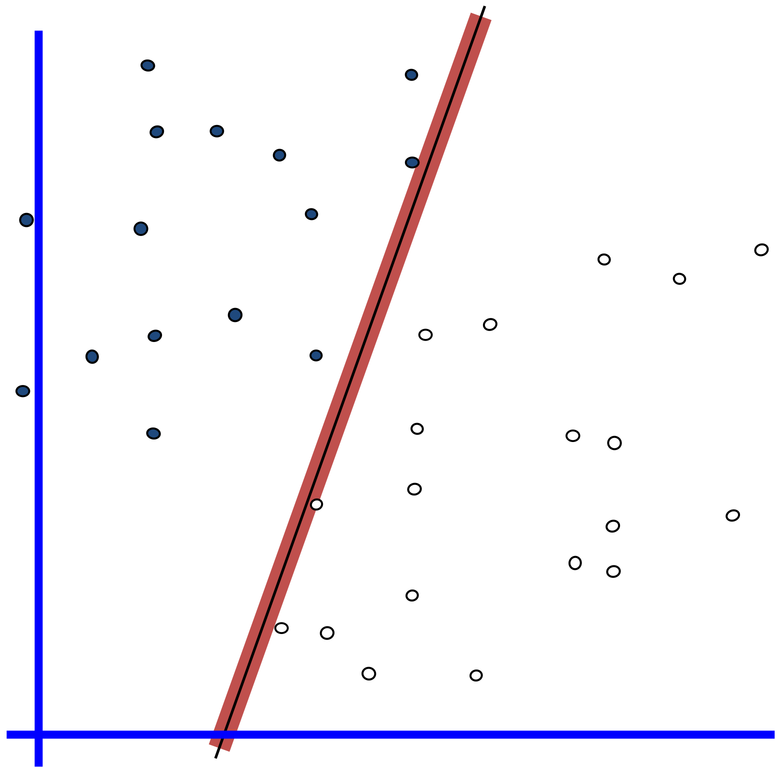


Which of these linear classifiers is the best?

Classifier Margin

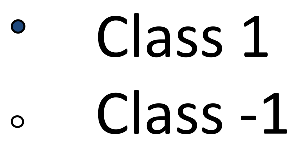


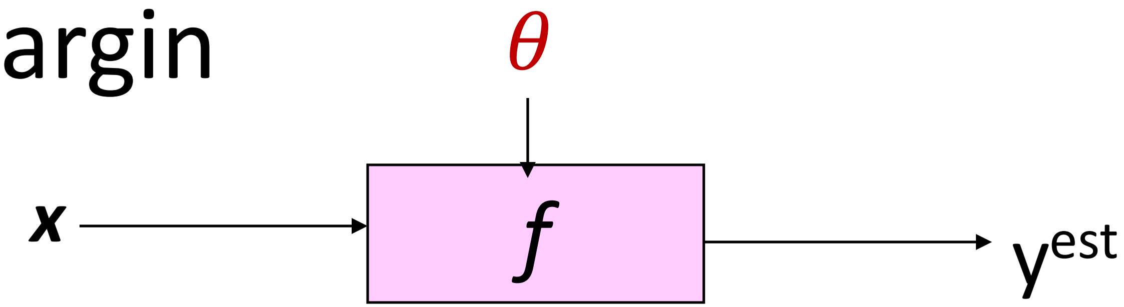
$f(\boldsymbol{x}, \theta) = sign(\theta^T x)$



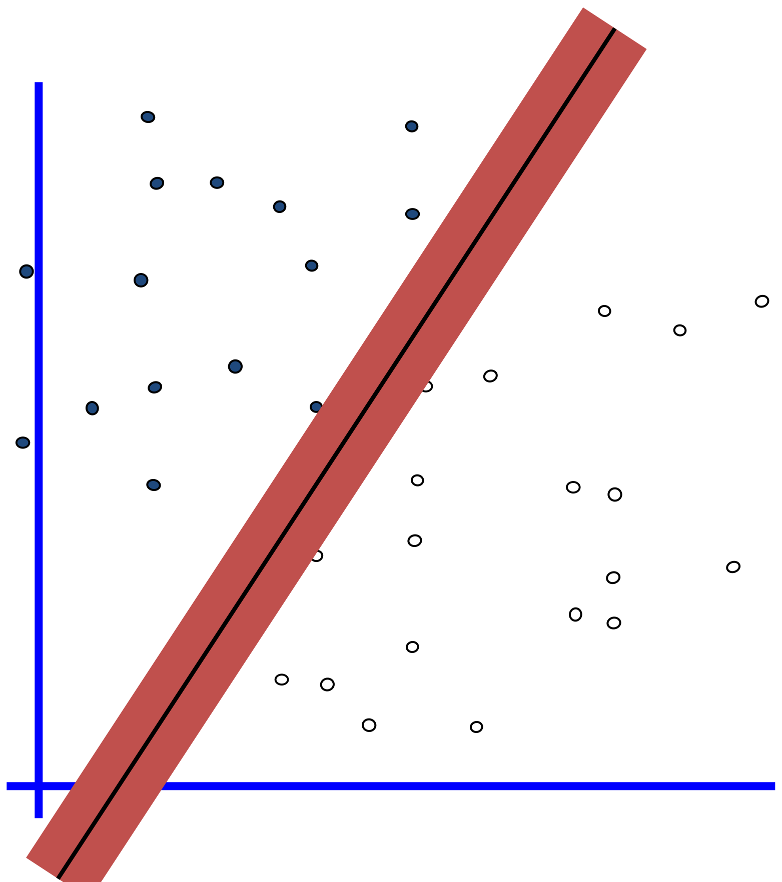
Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



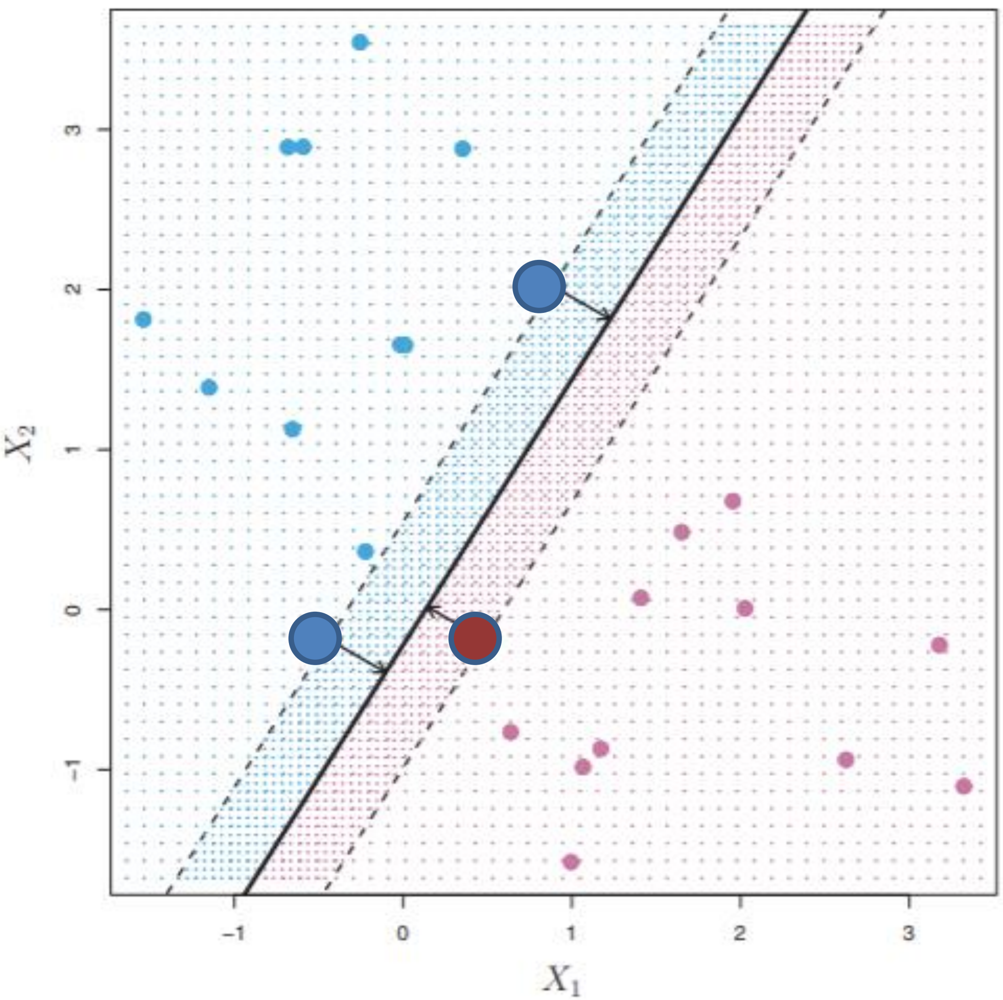


$f(x, \theta) = sign(\theta^T x)$



The maximum margin linear classifier is the linear classifier with the maximum margin!

Classifier margin



- Support vectors are "closest" to hyperplane
- If support vectors change, classifier changes

Finding the maximum margin classifier

- Training data $x^{(1)}, \dots, x^{(n)}$ with $x^{(i)} = \left(x_1^{(i)}, \dots, x_d^{(i)}\right)^{\mathrm{T}}$
- Labels are from 2 classes: $y_i \in \{-1,1\}$

max M

$$y^{(i)}\left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right) \geq M \ \forall i$$

$$\left|\left|\theta\right|\right|_2 = 1$$

Normalization constraint

Each point is on the right side of hyper-plane at distance $\geq M$

Equivalent formulation

- Min $||\theta||^2$
- $y^{(i)}\left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right) \geq 1 \ \forall i$

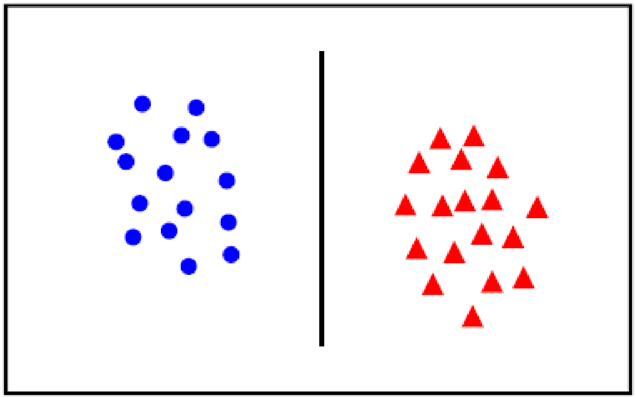
- Can be solved with quadratic optimization techniques
- It's easier to optimize the dual problem
- Maximum margin classifier – given by solution θ to this optimization problem

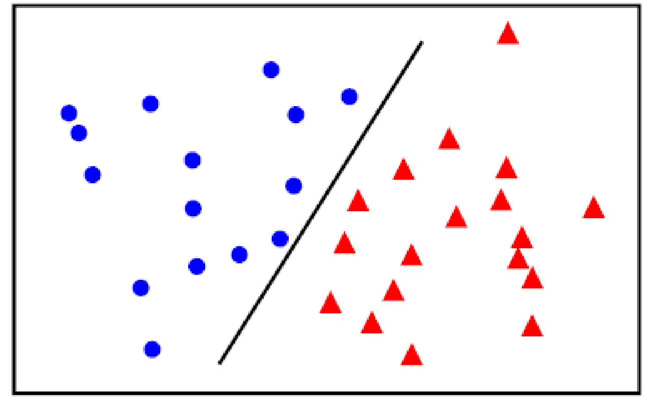
Outline

- Quick review on ensemble learning
- SVM
  - Linearly separable data
    - Separating hyperplanes
    - Maximum margin classifier
  - Non-separable data
    - Support vector classifier
- Non-linear decision boundaries
  - Kernels and Radial SVM

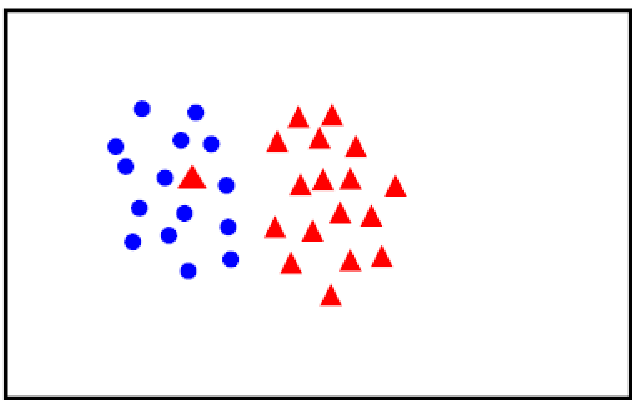
Linear separability

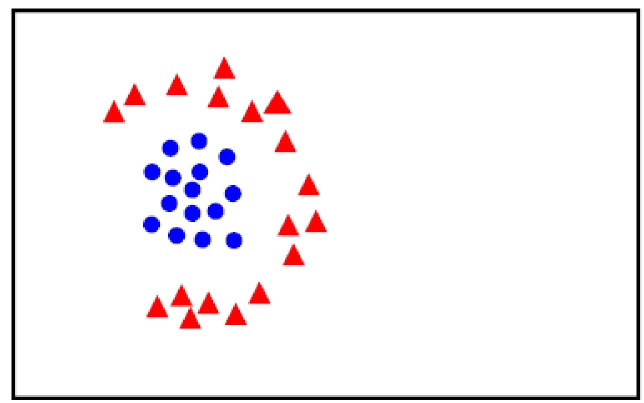
linearly separable



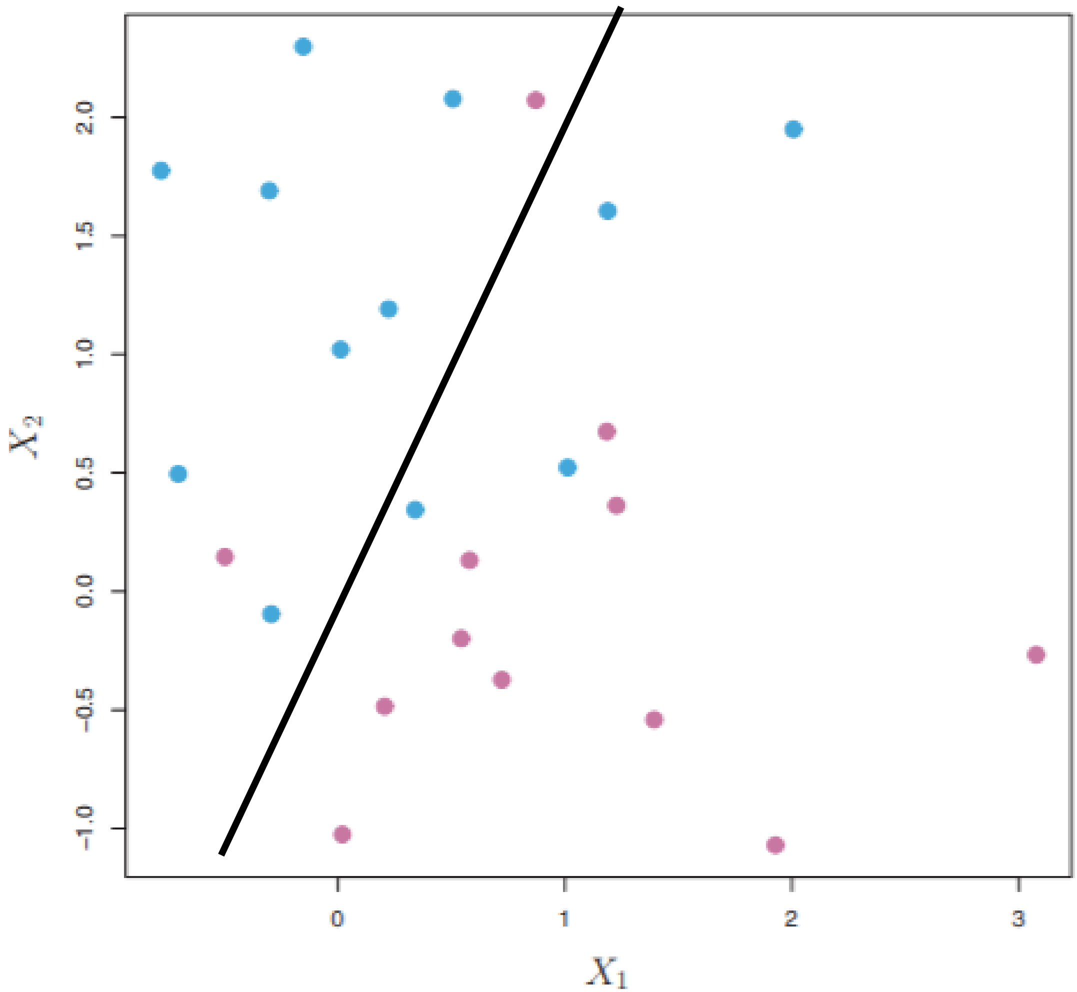


not linearly separable



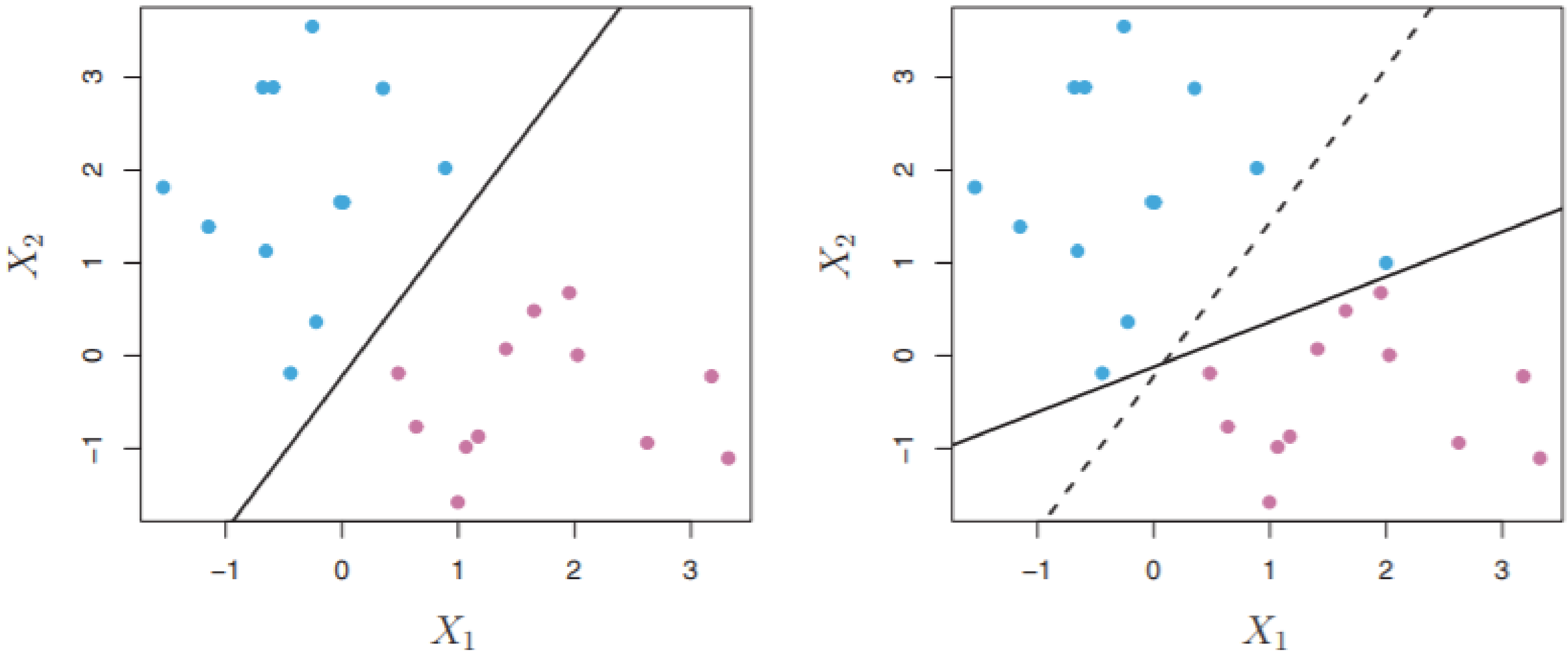


Non-separable case



Optimization problem has no solution!

Maximum margin is not always the best!



- Overfits to training data
- Sensitive to small modification (high variance)

Support vector classifier

- Allow for small number of mistakes on training data
- Obtain a more robust model

max M

$$y^{(i)}\left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right) \geq M(1 - \epsilon_i)\forall i$$

$$||\theta||_2 = 1$$

$$\epsilon_i \geq 0, \sum_i \epsilon_i = C$$

Slack

Error Budget (Hyper-parameter)

max M

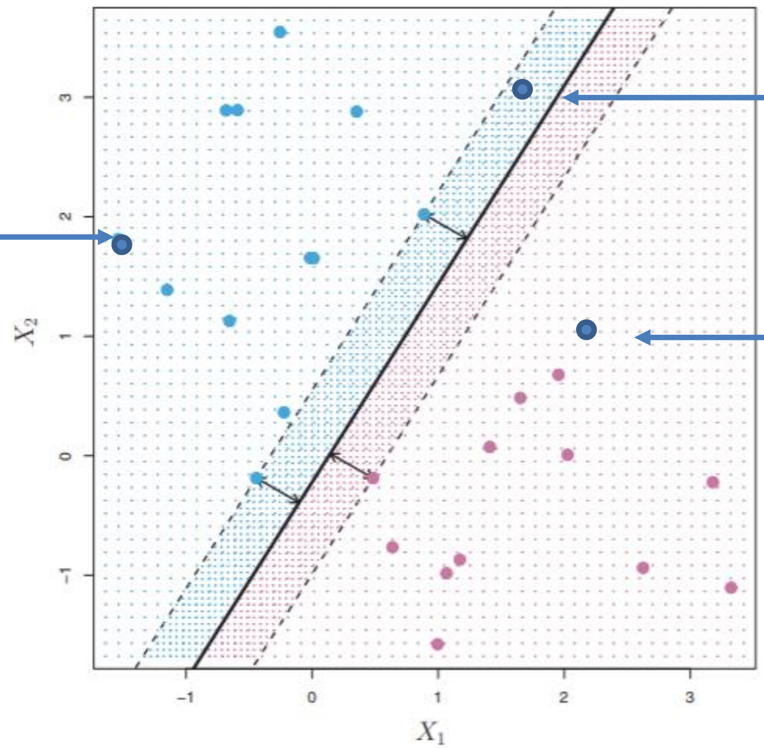
$$y^{(i)}\left(\theta_0+\theta_1 x_1^{(i)}+\cdots\theta_d x_d^{(i)}\right)\geq M(1-\epsilon_i\,)\;\forall i$$

$$||\theta||_2 = 1$$

$$\epsilon_i \geq 0, \sum_i \epsilon_i = C$$

Error Budget

Slack

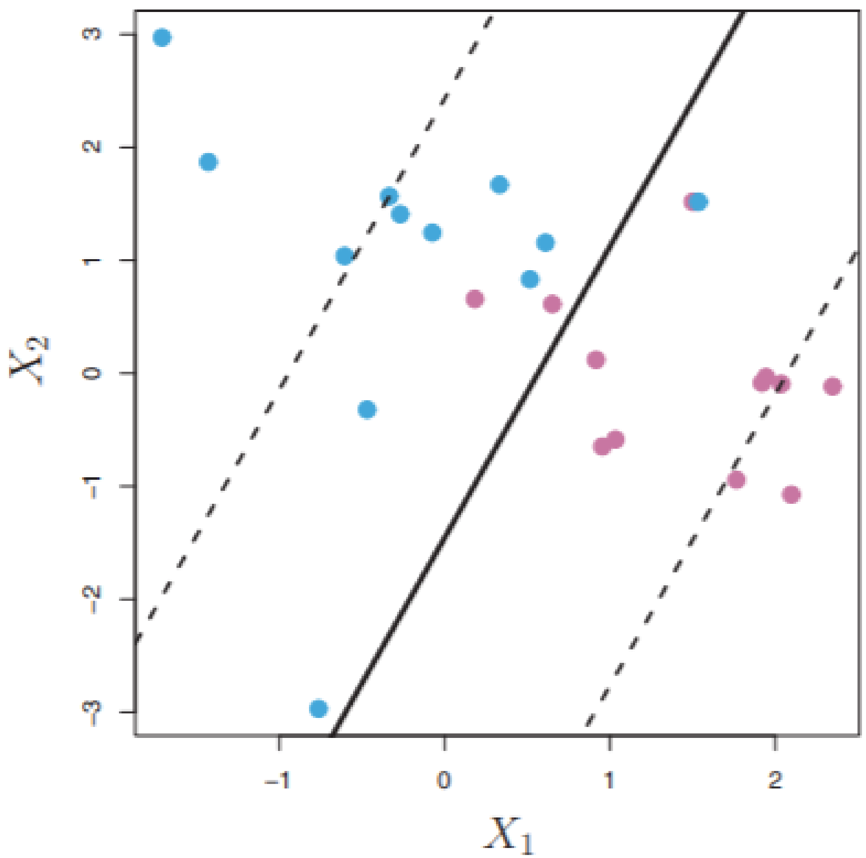


$\epsilon_i = 0$
Correct side of margin

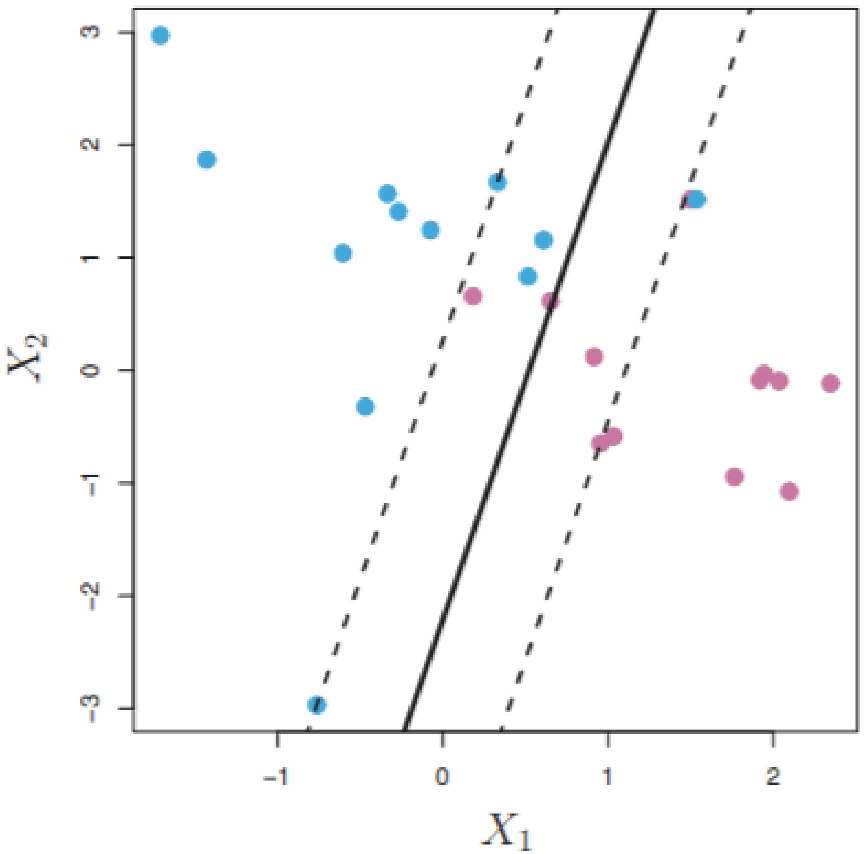
$0 < \epsilon_i < 1$
Violates margin
Correct label

$\epsilon_i > 1$
Incorrect label
At most C data points

Error Budget and Margin



Larger C
Low variance



Smaller C
Over-fitting

Find best hyper-parameter C by cross-validation

Equivalent formulation

- Min $||\theta||^2 + C\sum_i \epsilon_i$
- $y^{(i)}\left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right) \geq 1 - \epsilon_i \; \forall i$
- $\epsilon_i \geq 0$

- Inner product of 2 vectors $a = (a_1, \ldots, a_d)$ and $b = (b_1, \ldots, b_d)$ is $< a, b > = \sum_i a_i b_i$
- Solution is Support Vector Classifier
  - $f(z) = \theta_0 + \sum_i \alpha_i < z, x^{(i)} >$
  - Where $\alpha_i \neq 0$ only for support vectors (for all other training points $\alpha_i = 0$)
  - Linear SVM

Properties

- Maximum margin classifier
  - Classifier of maximum margin
  - For linearly separable data
- Support vector classifier
  - Allows some slack and sets a total error budget (hyper-parameter)
  - Final classifier on a point is a linear combination of inner product of point with support vectors
  - Efficient to evaluate

Objective for Logistic Regression

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y^{(i)}\log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1-y^{(i)}\right)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)\right]$$

- Cost of a single instance:

$$\operatorname{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1-h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

- Can re-write objective function as

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n}\underbrace{\operatorname{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}), y^{(i)}\right)}_{\text{Cross-entropy loss}}$$

Regularized Logistic Regression

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y^{(i)}\log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1-y^{(i)}\right)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)\right]$$

- We can regularize logistic regression exactly as before:

$$\begin{aligned} J_{\text{regularized}}(\boldsymbol{\theta}) &= J(\boldsymbol{\theta}) + \lambda\sum_{j=1}^{d}\theta_j^2 \\ &= J(\boldsymbol{\theta}) + \lambda\|\boldsymbol{\theta}_{[1:d]}\|_2^2 \end{aligned}$$

L2 regularization

Connection to Logistic Regression

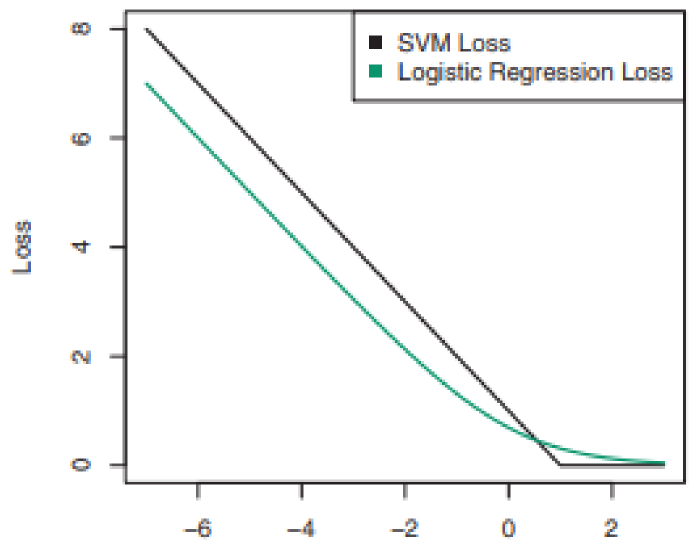
- $J(\theta) = \sum_{i=0}^{n} \max\left(0, 1 - y^{(i)} f\left(x^{(i)}\right)\right) + \lambda \sum_{j=1}^{d} \theta_j^2$

Hinge loss

$f\left(x^{(i)}\right) = \theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}$

- $J(\theta) = C \sum_{i=0}^{n} \max\left(0, 1 - y^{(i)} f\left(x^{(i)}\right)\right) + \sum_{j=1}^{d} \theta_j^2$

C = regularization cost



$y^{(i)}\left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right)$

Resilience to outliers

- LDA is very sensitive to outliers
  - Estimates mean and co-variance using all training data
- SVM is resilient to outliers
  - Decision hyper-plane mainly depends on support vectors
- Logistic regression is also resilient to points far from decision boundary
  - Cross-entropy uses logs in the loss function

Acknowledgements

- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
  - Andrew Moore
- Thanks!