

EE 4146 Data Engineering and Learning Systems

Lecture 9: Linear classifier

Semester A, 2021-2022

Schedules

Week	Date	Topics
1	Sep. 1	Introduction
2	Sep. 8	Data exploration
3	Sep. 15	Feature reduction and selection (HW1 out)
4	Sep. 22	Mid-Autumn Festival
5	Sep. 29	Clustering I: Kmeans based models (HW1 due in this weekend)
6	Oct. 6	Clustering II: Hierarchical/density based/fuzzing clustering
7	Oct. 13	Adverse Weather
8	Oct. 20	Midterm (no tutorials this week)
9	Oct. 27	Linear classifiers
10	Nov. 3	Classification based on decision tree (Tutorial on project) (HW2 out)
11	Nov. 10	Bayes based classifier (Tutorial on codes) (HW2 due in this weekend)
12	Nov. 17	KNN and classifier ensemble
13	Nov. 24	Deep learning based models (Quiz)
14	Make-up Video recording	Summary

■ Options for the last lecture

- Make up video recording
 - Make up lecture on Nov. 29 at 3:00 PM-6:00 PM
 - Make up lecture on Dec. 1 at 3:00 PM-6:00 PM
 - Will do poll
-
- Feedback
 - We did poll in the class and around 80% students choose the make-up video recording. Therefore, I will upload the last lecture for summary on or before Nov. 29.

Outline

- Basic Introduction
- The Perceptron Algorithm
- Logistic Classifiers
- Support Vector Machines
- Evaluation for Classification/Imbalanced Issues
- Midterm paper illustration
 - Results will be released before Monday
 - You can find TA at G2325, AC1 in the open hour to get the examination paper
 - Open hour: Monday 1:00-7:00 PM

Supervised vs. Unsupervised Learning

- Unsupervised learning (clustering)
 - The class labels of training data is unknown
 - Given a set of measurements, observations, etc. with the aim of establishing clusters in the data
- Supervised learning (classification)
 - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
 - New data is classified based on the training set

Prediction Problems

■ Classification

- predicts categorical class labels (discrete or nominal)
- classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute, and uses it in classifying new data

■ Typical applications

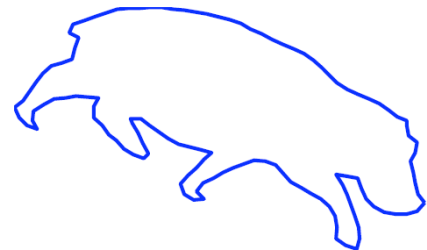
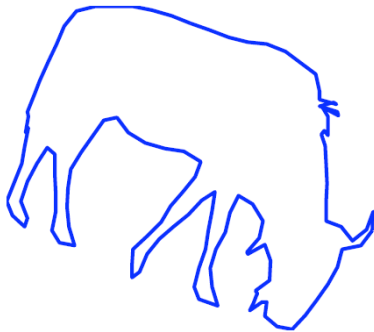
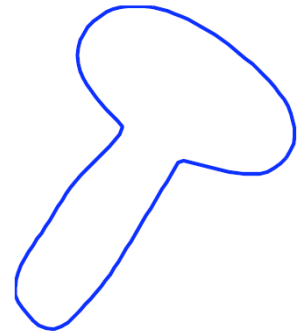
- Credit/loan approval:
- Medical diagnosis: if a tumor is cancerous or benign
- Fraud detection: if a transaction is fraudulent
- Web page categorization: which category it is

Classification—A Two-Step Process

- **Model construction:** describing a set of predetermined classes
 - Each sample is assumed to belong to a predefined class, as determined by the class label attribute
 - The set of samples used for model construction is training set
 - The model is represented as classification rules, decision trees, or mathematical formula, etc.
- **Model usage:** for classifying future or unknown objects
 - Estimate accuracy of the model
 - The known label of test sample is compared with the classified result from the model
 - Accuracy rate is the percentage of test set samples that are correctly classified by the model
 - Test set is independent of training set (otherwise overfitting)

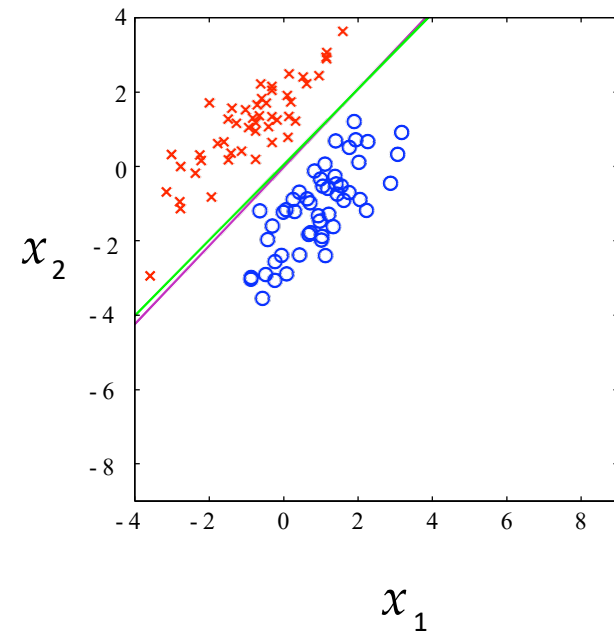
Example Problem

- Animal or Vegetable?



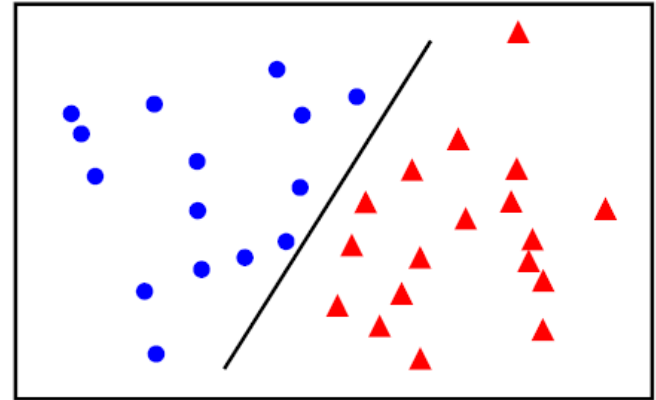
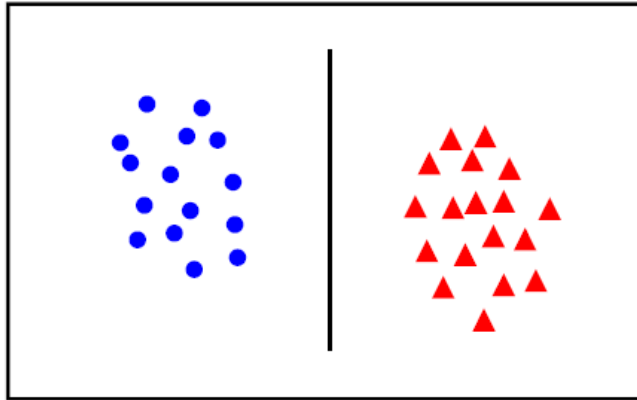
Linear Models for Classification

- Linear models for classification separate input vectors into classes using linear (hyperplane) decision boundaries.
- Example:
 - 2D Input vector x
 - Two discrete classes C_1 and C_2

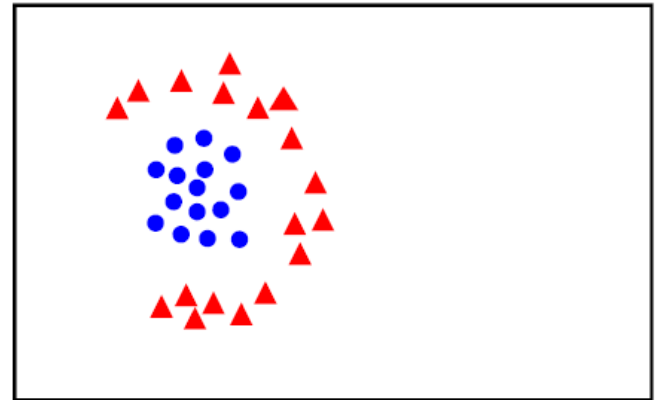
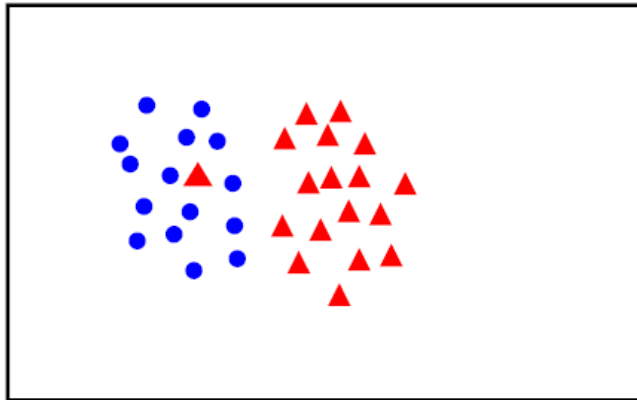


Linear separability

linearly
separable



not
linearly
separable

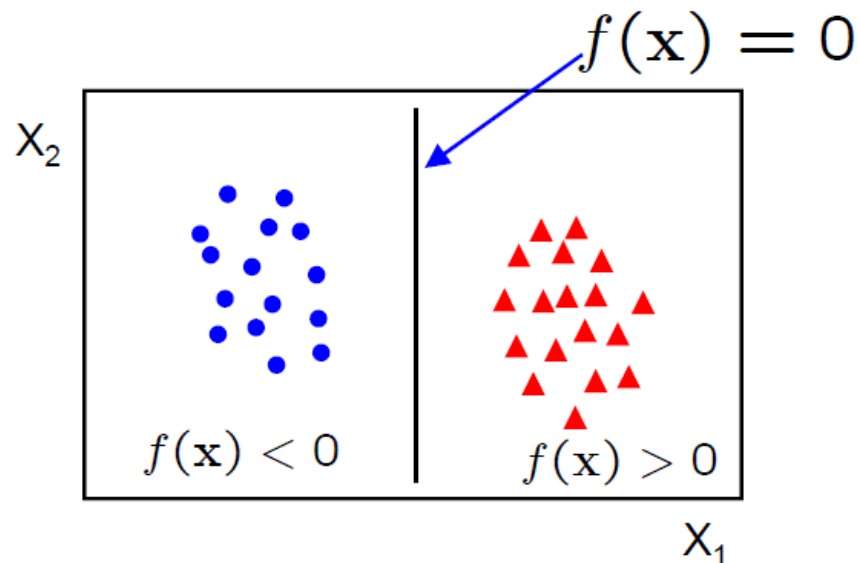


Linear classifiers

- A linear classifier has the form

$$y(\mathbf{x}) = f(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0$$

- in 2D the discriminant is a line
- \mathbf{w} is known as the weight vector
- w_0 the bias

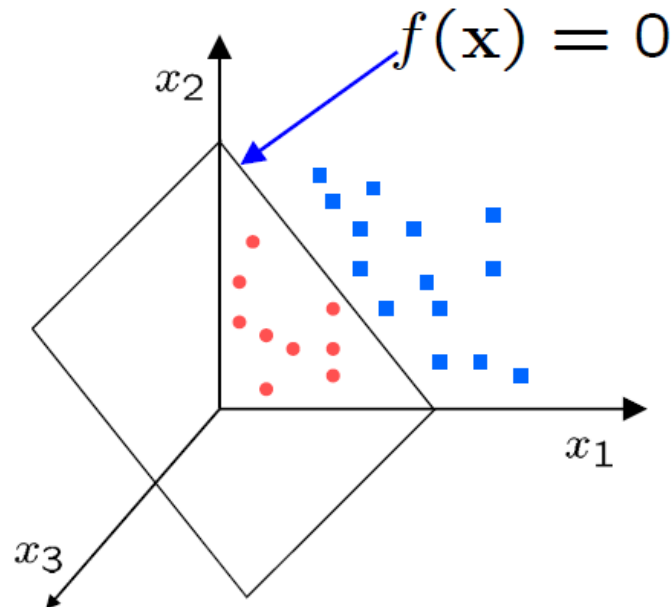


Linear classifiers

- A linear classifier has the form

$$y(\mathbf{x}) = f(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0$$

- in 3D the discriminant is a plane
- For a linear classifier, the training data is used to learn \mathbf{w}



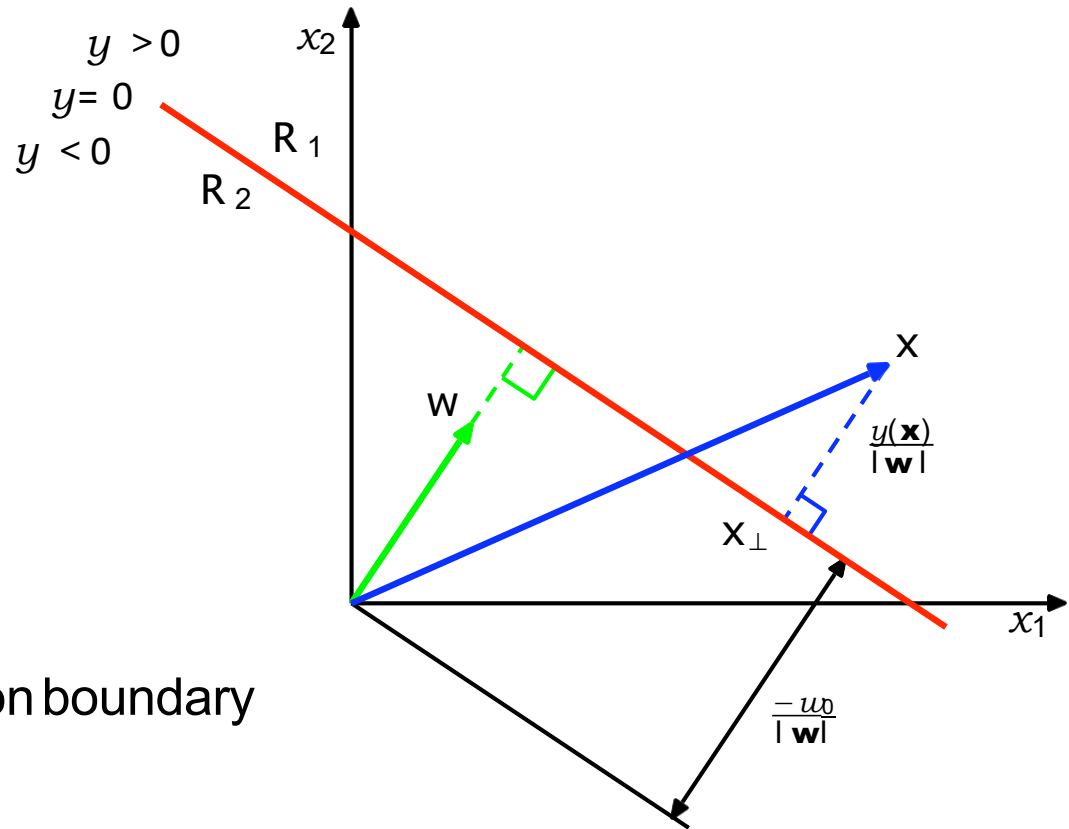
Two-Class Discriminant Function

$$y(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0$$

$y(\mathbf{x}) > 0 \rightarrow \mathbf{x}$ assigned to \mathbb{C}_1

$y(\mathbf{x}) < 0 \rightarrow \mathbf{x}$ assigned to \mathbb{C}_2

Thus $y(\mathbf{x}) = 0$ defines the decision boundary



Two-Class Discriminant Function

$$y(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0$$

$y(\mathbf{x}) > 0 \rightarrow \mathbf{x}$ assigned to \mathbb{C}_1

$y(\mathbf{x}) < 0 \rightarrow \mathbf{x}$ assigned to \mathbb{C}_2

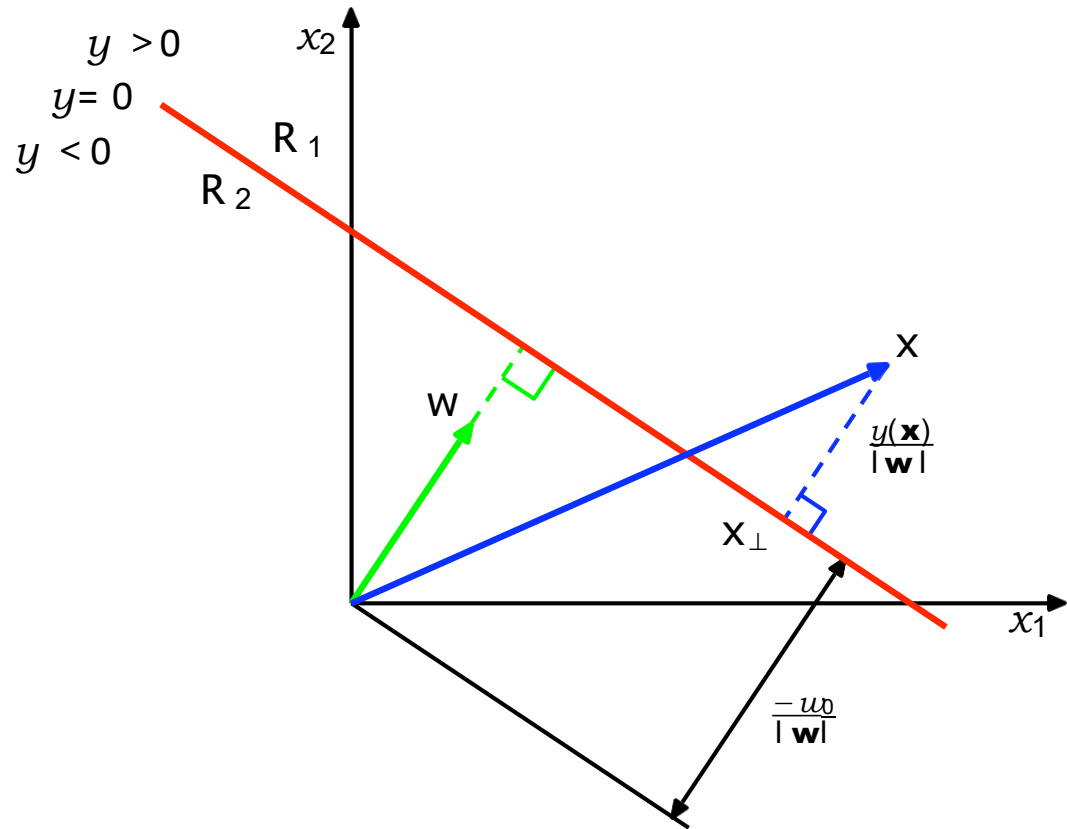
For convenience, let

$$\mathbf{w} = [w_1 \dots w_M]^t \Rightarrow [w_0 \ w_1 \dots w_M]^t$$

and

$$\mathbf{x} = [x_1 \dots x_M]^t \Rightarrow [1 \ x_1 \dots x_M]^t$$

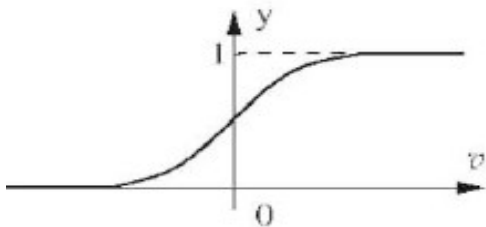
So we can express $y(\mathbf{x}) = \mathbf{w}^t \mathbf{x}$



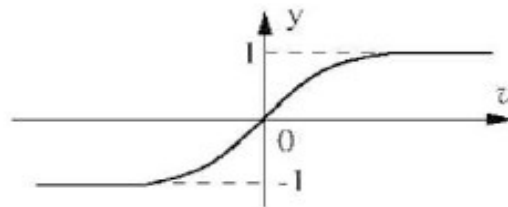
Generalized Linear Models

- For classification problems, we want y to be a predictor of x . In other words, we wish to map the input vector into one of a number of discrete classes, lie between 0 and 1.
- For this purpose, it is useful to elaborate the linear model by **introducing a nonlinear activation function f** , which typically will constrain y to lie between -1 and 1 or between 0 and 1.

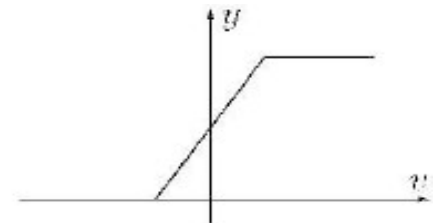
$$y(\mathbf{x}) = f(\mathbf{w}^t \mathbf{x} + w_0)$$



Log-sigmoid function



Tan-sigmoid function

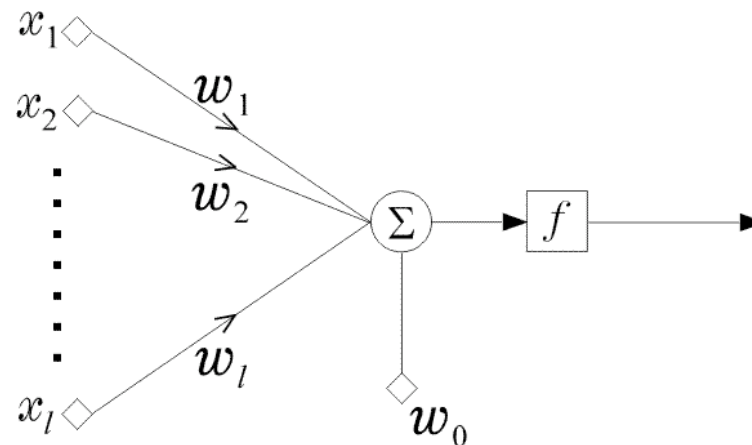


Linear function

The Perceptron

$$y(\mathbf{x}) = f(\mathbf{w}^t \mathbf{x} + w_0) \quad \begin{array}{l} y(\mathbf{x}) \geq 0 \rightarrow \mathbf{x} \text{ assigned to } C_1 \\ y(\mathbf{x}) < 0 \rightarrow \mathbf{x} \text{ assigned to } C_2 \end{array}$$

- A classifier based upon this simple generalized linear model is called a (single layer) **perceptron**.
- It can also be identified with an abstracted model of a neuron called the McCulloch Pitts model.

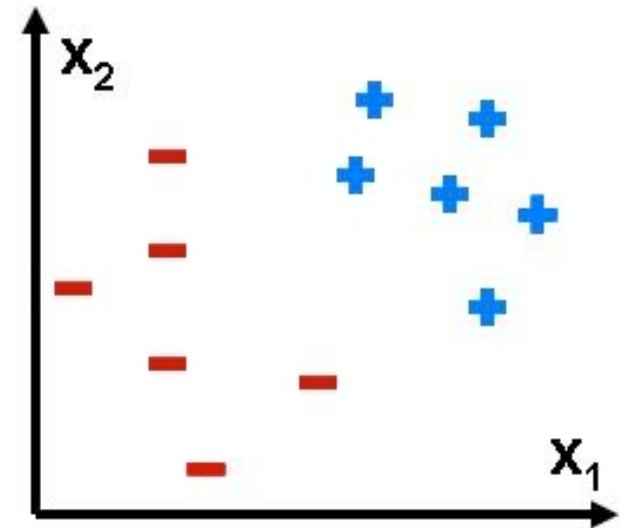


Outline

- Basic Introduction
- The Perceptron Algorithm
- Logistic Classifiers
- Support Vector Machines
- Evaluation for Classification/Imbalanced Issues

Linearly Separable Inputs

- For starters, let's assume that the training data is in fact perfectly linearly separable.
- In other words, there exists at least one hyperplane (one set of weights) that yields 0 classification error.
- We seek an algorithm that can automatically find such a hyperplane.

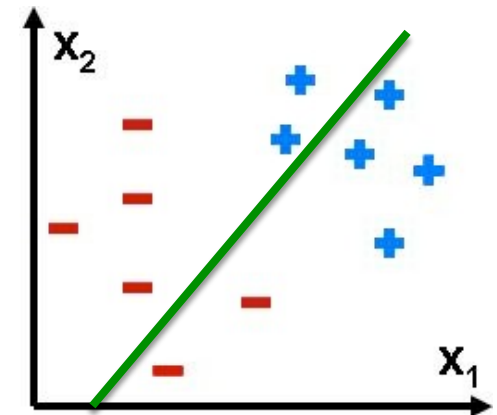


The Perceptron Algorithm

- The perceptron algorithm was invented by Frank Rosenblatt (1962).
- The algorithm is iterative.
- The strategy is to start with a random **guess** at the weights \mathbf{w} , and to then **iteratively** change the weights to move the hyperplane in a direction that lowers the classification error.

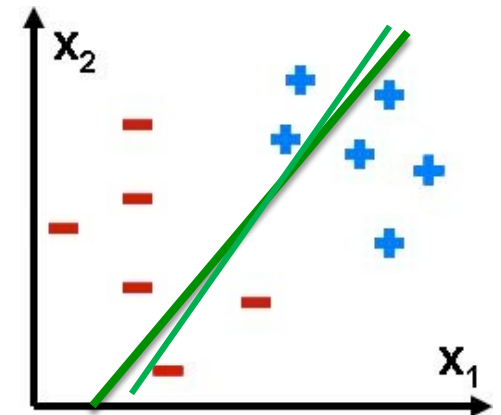


Frank Rosenblatt (1928 – 1971)



The Perceptron Algorithm

- Note that as we change the weights continuously, the **classification error changes in discontinuous**, piecewise constant fashion.
- Thus we cannot use the classification error as our objective function to minimize.



The Perceptron Criterion

- Note that we seek \mathbf{w} such that
- To be noted here, w^t is the weight, t/t_n corresponding to the output y , or $f(x)$

$$\mathbf{w}^t \mathbf{x} \geq 0 \text{ when } t = +1$$

$$\mathbf{w}^t \mathbf{x} < 0 \text{ when } t = -1$$

- In other words, we would like

$$\mathbf{w}^t \mathbf{x}_n t_n \geq 0 \quad \forall n$$

- Thus we seek to minimize $E_P(\mathbf{w}) = -\sum_{n \in \mathcal{M}} \mathbf{w}^t \mathbf{x}_n t_n$

where M is the set of misclassified inputs.

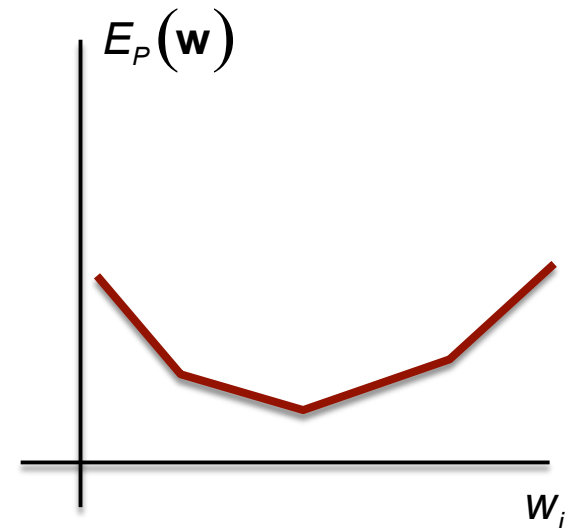
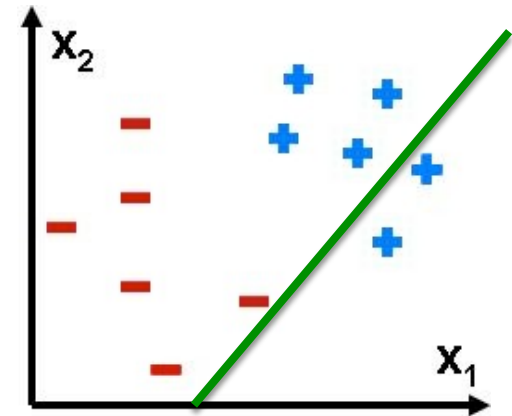
The Perceptron Criterion

$$E_p(\mathbf{w}) = - \sum_{n \in \mathcal{M}} \mathbf{w}^t \mathbf{x}_n t_n$$

where \mathcal{M} is the set of misclassified inputs.

■ Observations:

- $E_p(\mathbf{w})$ is always non-negative.
- $E_p(\mathbf{w})$ is continuous and piecewise linear, and thus easier to minimize.



The Perceptron Algorithm

$$E_P(\mathbf{w}) = - \sum_{n \in \mathcal{M}} \mathbf{w}^t \mathbf{x}_n t_n$$

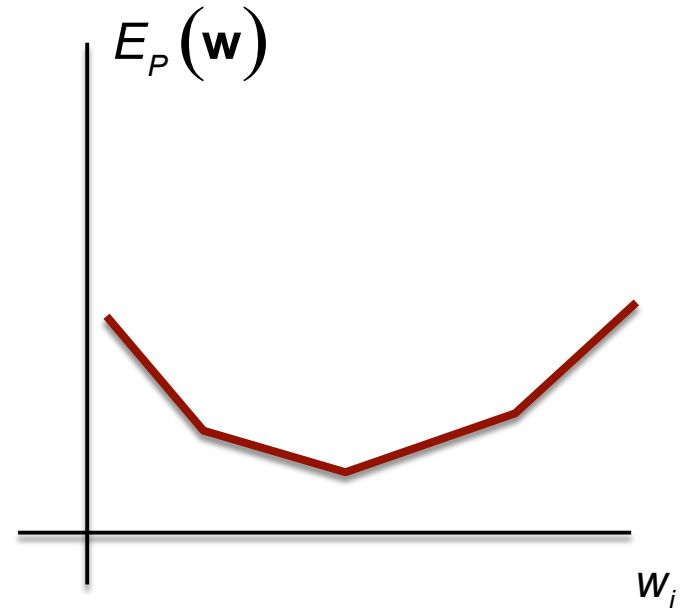
where \mathcal{M} is the set of misclassified inputs.

$$\frac{dE_P(\mathbf{w})}{d\mathbf{w}} = - \sum_{n \in \mathcal{M}} \mathbf{x}_n t_n$$

where the derivative exists.

■ Gradient descent

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^{\tau} + \eta \sum_{n \in \mathcal{M}} \mathbf{x}_n t_n$$



The Perceptron Algorithm

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^t + \eta \sum_{n \in \mathcal{M}} \mathbf{x}_n t_n$$

- Why does this make sense?
 - If an input from C_1 ($t = +1$) is misclassified, we need to make \mathbf{w} more positive.
 - If an input from C_2 ($t = -1$) is misclassified, we need to \mathbf{w} more negative.

The Perceptron Algorithm

- The algorithm can be implemented sequentially:
- Repeat until convergence:
- For each input (\mathbf{x}_n, t_n) :
 - If it is correctly classified, do nothing
 - If it is misclassified, update the weight vector to be

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} + \eta \mathbf{x}_n t_n$$

- Note that this will lower the contribution of input n to the objective function:

$$-\left(\mathbf{w}^{(\tau)}\right)^t \mathbf{x}_n t_n \rightarrow -\left(\mathbf{w}^{(\tau+1)}\right)^t \mathbf{x}_n t_n = -\left(\mathbf{w}^{(\tau)}\right)^t \mathbf{x}_n t_n - \eta \left(\mathbf{x}_n t_n\right)^t \mathbf{x}_n t_n < -\left(\mathbf{w}^{(\tau)}\right)^t \mathbf{x}_n t_n.$$

Not Monotonic

- While updating with respect to a misclassified input n will lower the error for that input, the error for other misclassified inputs may increase.
- Also, new inputs that had been classified correctly may now be misclassified.
- The result is that the perceptron algorithm is **not guaranteed to reduce the total error monotonically** at each stage.

The Perceptron Convergence Theorem

- Despite this non-monotonicity, if in fact the data are linearly separable, then the algorithm is guaranteed to find an exact solution in a finite number of steps (Rosenblatt, 1962).

Practical Limitations

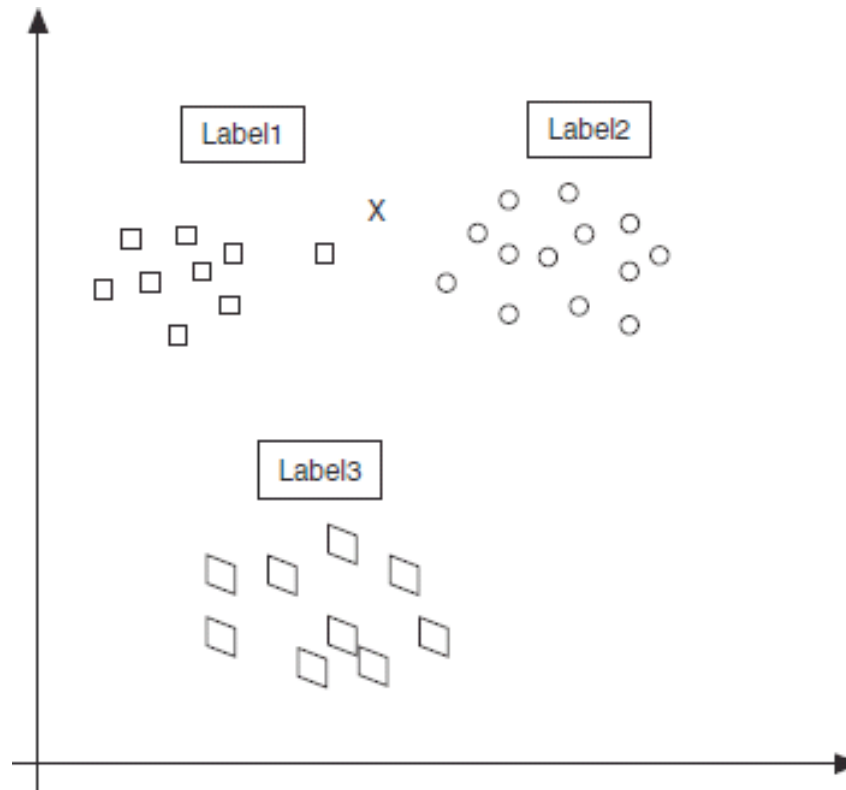
- The Perceptron Convergence Theorem is an important result. However, there are practical limitations:
 - Convergence may be slow
 - If the data are not separable, the algorithm will not converge.
 - We will only know that the data are separable once the algorithm converges.
 - The solution is in general not unique, and will depend upon initialization, scheduling of input vectors, and the learning rate η .

Generalization to inputs that are not linearly separable

- The single-layer perceptron can be generalized to yield good linear solutions to problems that are not linearly separable.
- Example: The Pocket Algorithm (Gal 1990)
- Idea:
 - Run the perceptron algorithm
 - Keep track of the weight vector w^* that has produced the best classification error achieved so far.
 - It can be shown that w^* will converge to an optimal solution with probability 1.

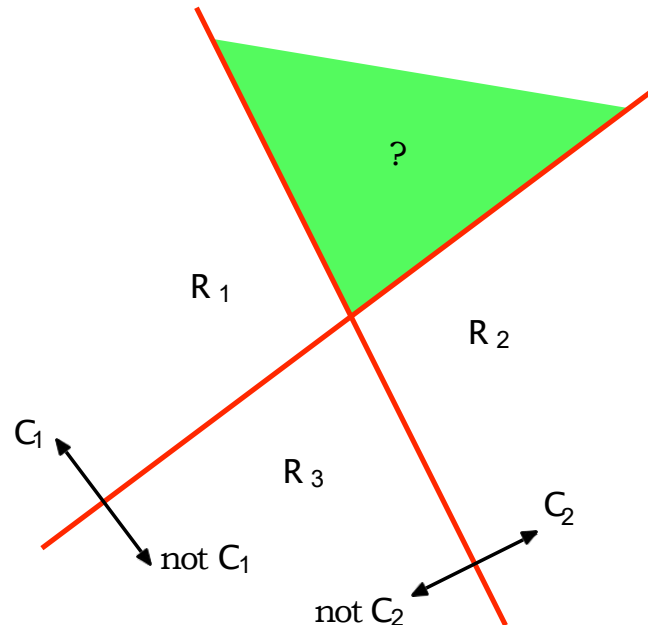
Generalization to Multiclass Problems

- How can we use perceptron, or linear classifiers in general, to classify inputs when there are $K > 2$ classes?



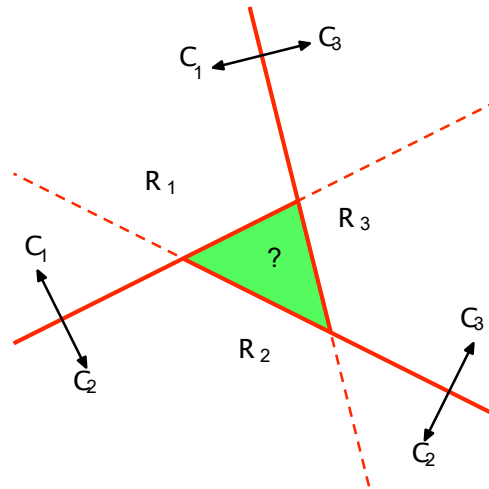
$K > 2$ Classes

- Idea #1: Just use $K-1$ discriminant functions, each of which separates one class C_k from the rest. (One-versus-the-rest classifier.)
- Problem: Ambiguous regions



$K > 2$ Classes

- Idea #2: Use $K(K-1)/2$ discriminant functions, each of which separates two classes C_i, C_k from each other. (One-versus-one classifier.)
- Each point classified by majority vote.
- Problem: Ambiguous regions



$K > 2$ Classes

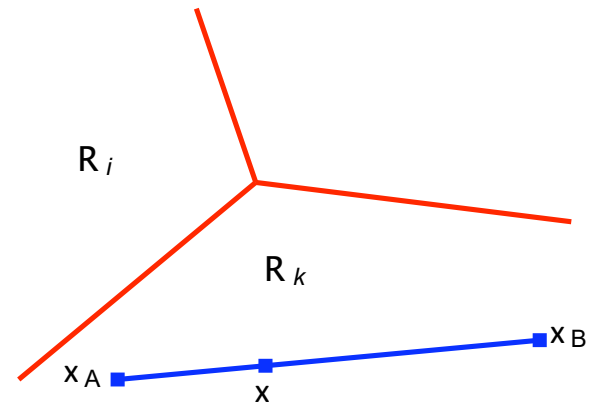
- Idea #3: Use K discriminant functions $y_k(\mathbf{x})$
- Use the magnitude of $y_k(\mathbf{x})$, not just the sign.

$$y_k(\mathbf{x}) = \mathbf{w}_k^t \mathbf{x}$$

\mathbf{x} assigned to C_k if $y_k(\mathbf{x}) > y_j(\mathbf{x}) \forall j \neq k$

Decision boundary $y_k(\mathbf{x}) = y_j(\mathbf{x}) \rightarrow (\mathbf{w}_k - \mathbf{w}_j)^t \mathbf{x} + (\mathbf{w}_{k0} - \mathbf{w}_{j0}) = 0$

- Results in decision regions that are simply-connected and convex.




Example: Kesler's Construction

- The perceptron algorithm can be generalized to K- class classification problems.
- Example:
 - Kesler's Construction:
 - Allows use of the perceptron algorithm to simultaneously learn K separate weight vectors \mathbf{w}_i .
 - Inputs are then classified in Class i if and only if
$$\mathbf{w}_i^t \mathbf{x} > \mathbf{w}_j^t \mathbf{x} \quad \forall j \neq i$$
 - The algorithm will converge to an optimal solution if a solution exists, i.e., if all training vectors can be correctly classified according to this rule.

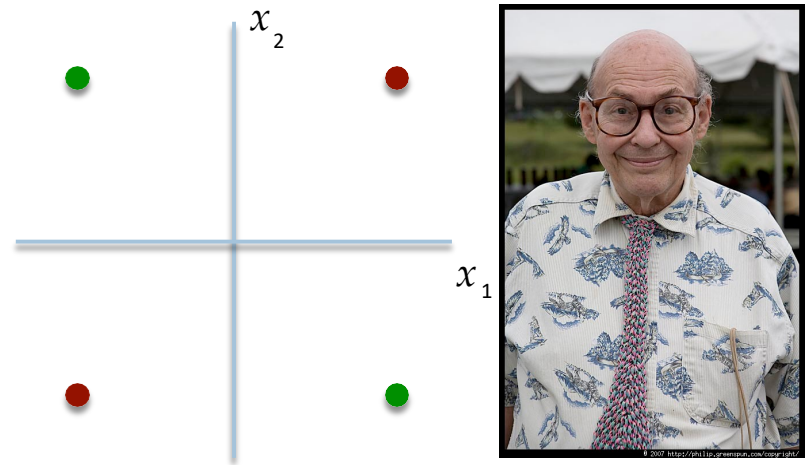
1-of-K Coding Scheme

- When there are $K > 2$ classes, target variables can be coded using the 1-of-K coding scheme:
- Input from Class $C_i \iff t = [0 \ 0 \ \dots 1 \ \dots 0 \ 0]^t$


Element i

Computational Limitations of Perceptrons

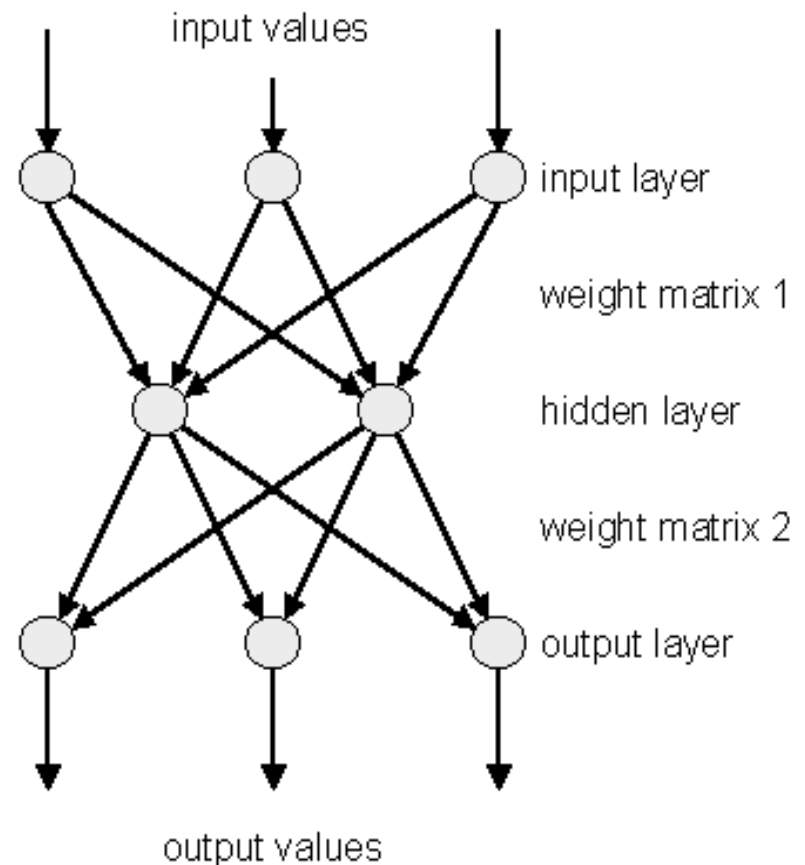
- Initially, the perceptron was thought to be a potentially powerful learning machine that could model human neural processing.
- However, Minsky & Papert (1969) showed that the single-layer perceptron could not learn a simple XOR function.
- This is just one example of a non-linearly separable pattern that cannot be learned by a single-layer perceptron.



Marvin Minsky (1927 -)

Multi-Layer Perceptrons

- Minsky & Papert's book was widely misinterpreted as showing that artificial neural networks were inherently limited.
- This contributed to a decline in the reputation of neural network research through the 70s and 80s.
- However, their findings apply only to single-layer perceptrons. Multi-layer perceptrons are capable of learning highly nonlinear functions, and are used in many practical applications.
- **Deep learning now**



Outline

- Basic Introduction
- The Perceptron Algorithm
- **Logistic Classifiers**
- Support Vector Machines
- Evaluation for Classification/Imbalanced Issues

Logistic regression

- Logistic regression is actually a classification method
- LR introduces an extra non-linearity over a linear classifier, $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$, by using a logistic (or sigmoid) function, $\sigma()$.
- The LR classifier is defined as

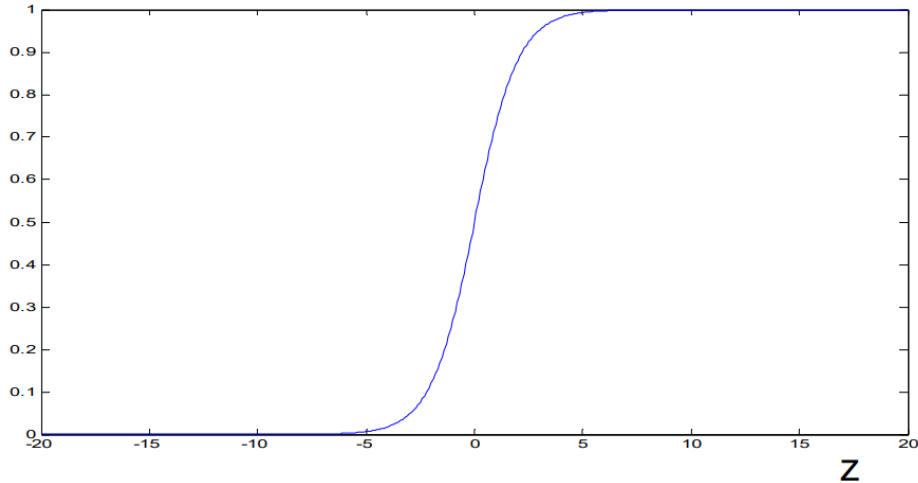
$$\sigma(f(\mathbf{x}_i)) \begin{cases} \geq 0.5 & y_i = +1 \\ < 0.5 & y_i = -1 \end{cases}$$

where $\sigma(f(\mathbf{x})) = \frac{1}{1 + e^{-f(\mathbf{x})}}$

Logistic regression

- As z goes from $-\infty$ to ∞ , $\sigma(z)$ goes from 0 to 1, a “squashing function”.
- It has a “sigmoid” shape (i.e. S-like shape)

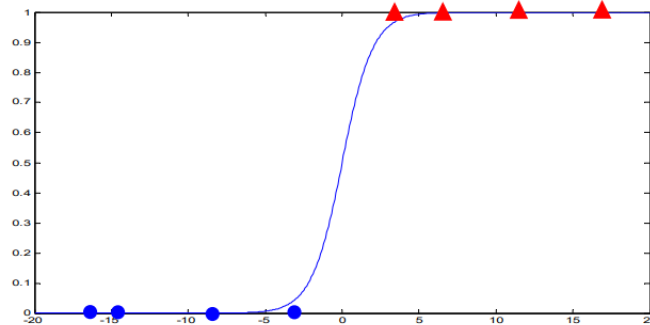
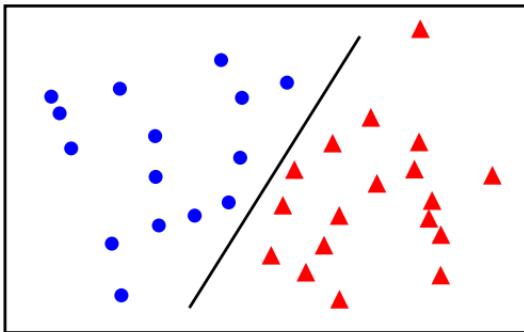
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Learning

- In logistic regression fit a sigmoid function to the data $\{x_i, y_i\}$ by minimizing the classification errors

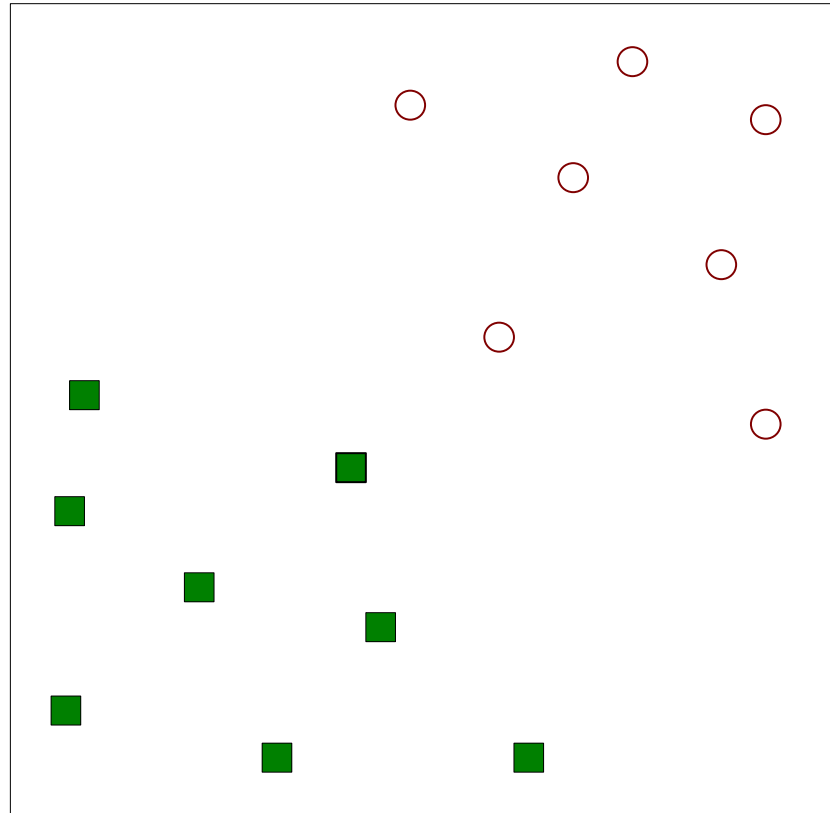
$$y_i - \sigma(\mathbf{w}^\top \mathbf{x}_i)$$



Outline

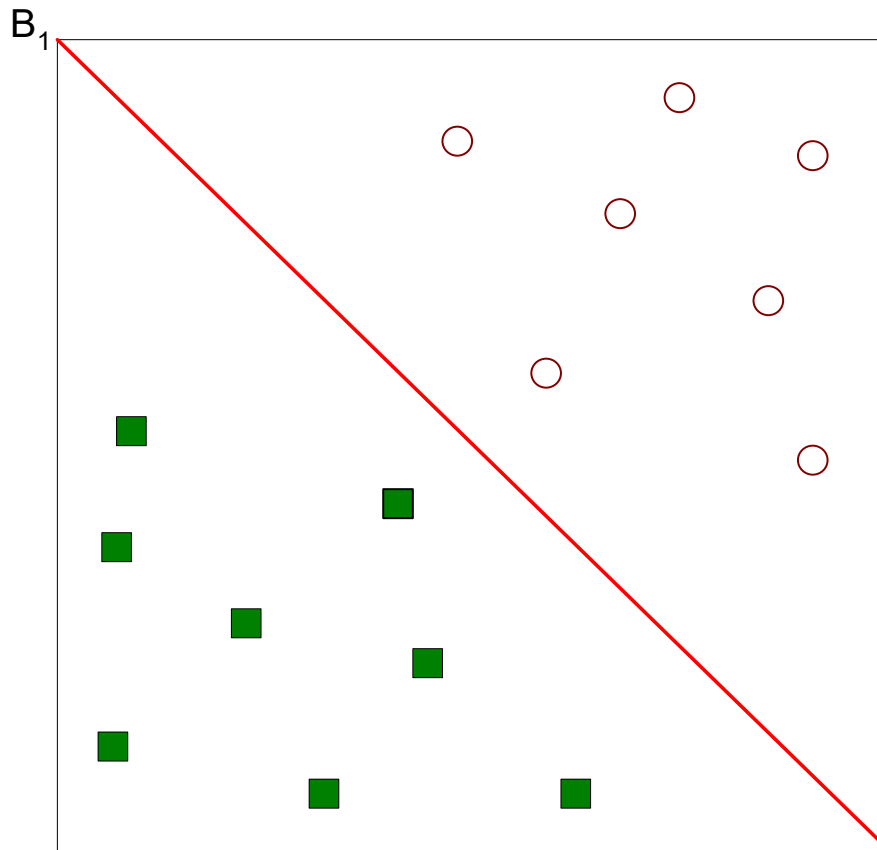
- Basic Introduction
- The Perceptron Algorithm
- Logistic Classifiers
- **Support Vector Machines**
- Evaluation for Classification/Imbalanced Issues

Support Vector Machines



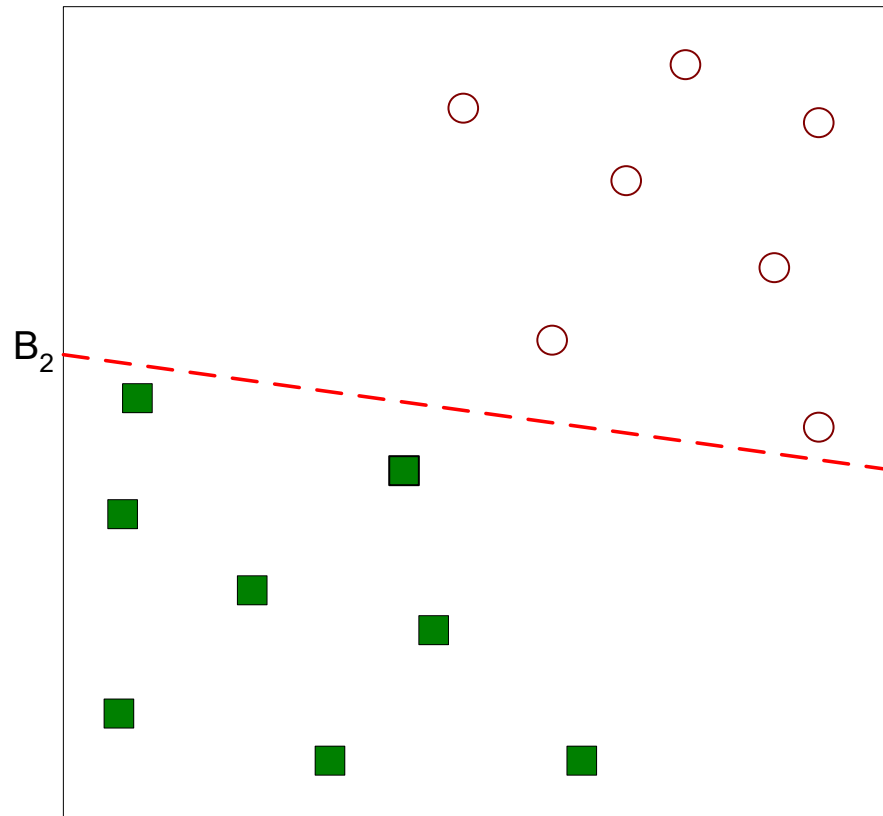
- Classification: find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



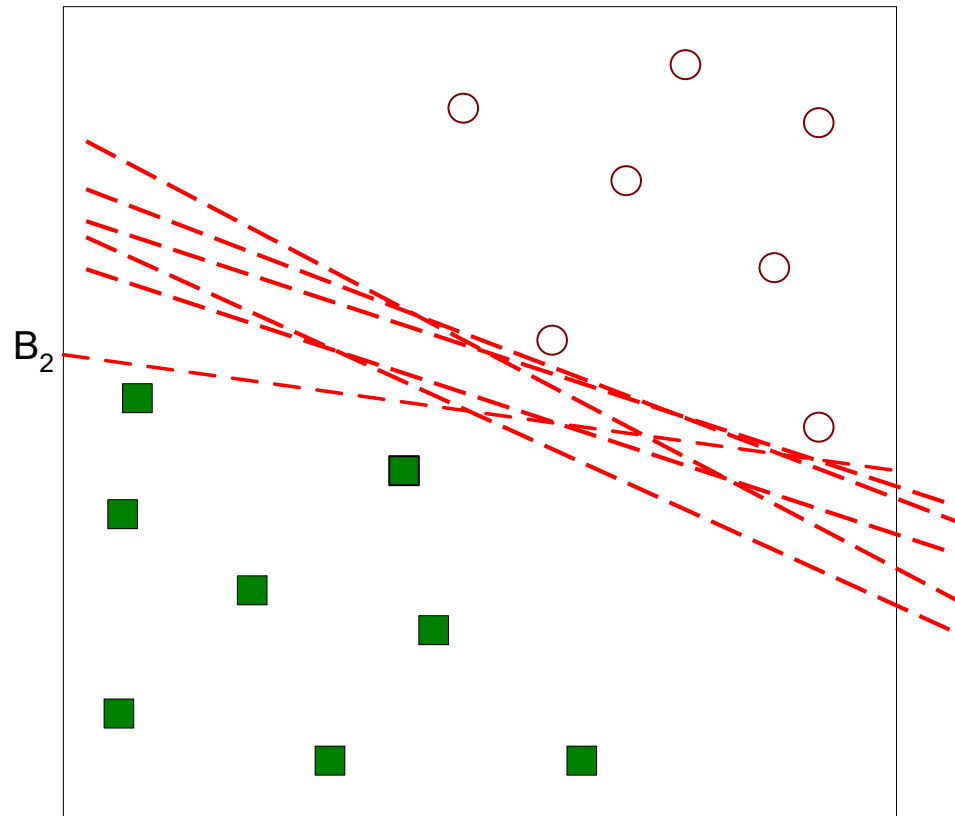
- One Possible Solution

Support Vector Machines



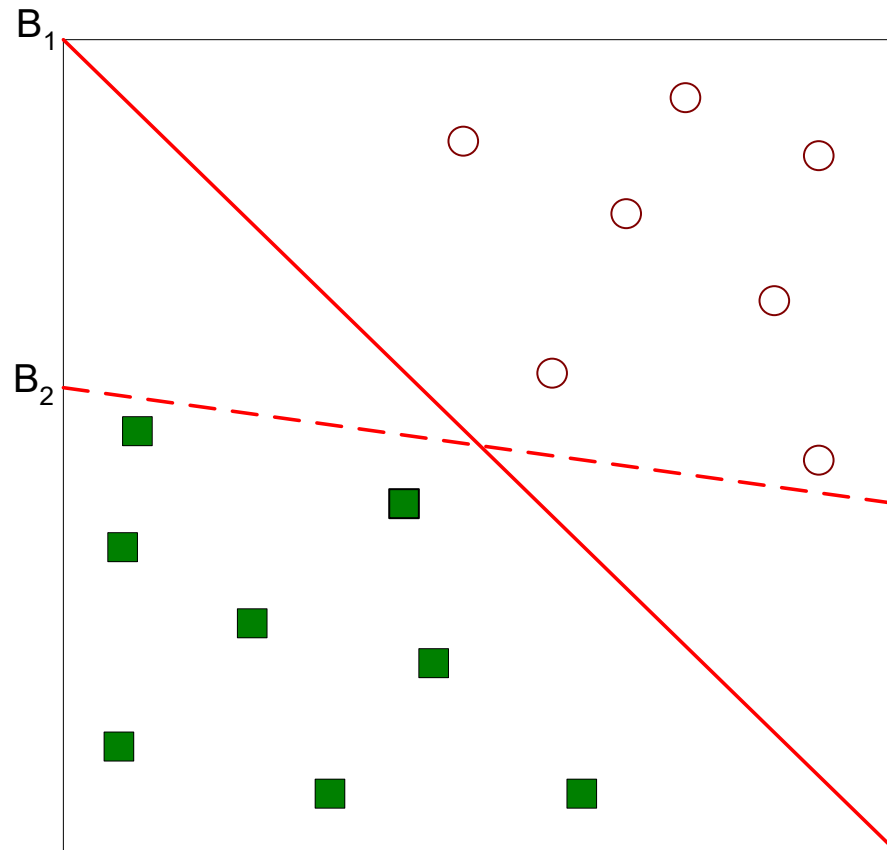
- Another possible solution

Support Vector Machines



- Other possible solutions

Support Vector Machines



- Which one is better? B_1 or B_2 ?
- How do you define better?

Motivation

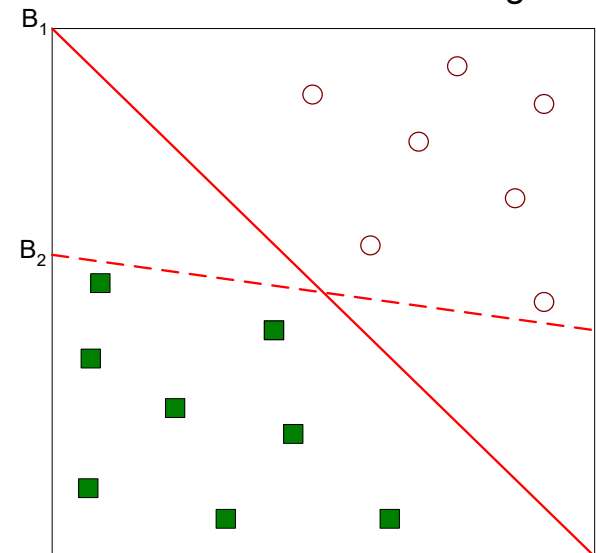
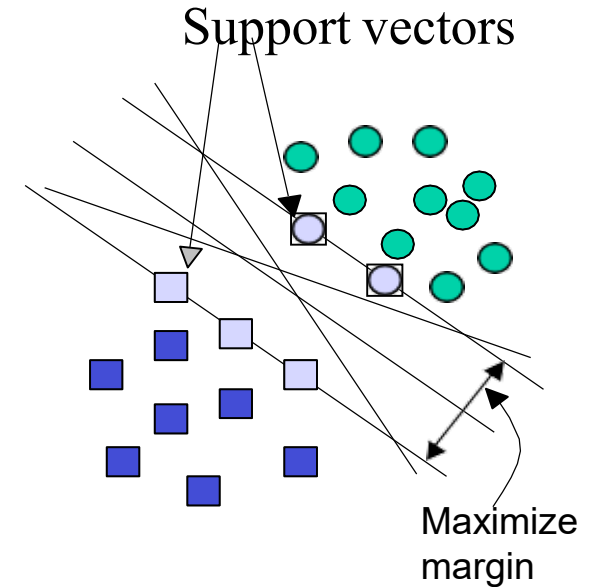
- The perceptron algorithm is guaranteed to provide a linear decision surface that separates the training data, if one exists.
- However, if the data are linearly separable, there are in general an infinite number of solutions, and the solution returned by the **perceptron algorithm** depends in a **complex** way on the initial conditions, the learning rate and the order in which training data are processed.
- While all solutions achieve a perfect score on the training data, they won't all necessarily generalize as well to new inputs.

The Large Margin Classifier

- Unlike the Perceptron Algorithm, Support Vector Machines solve a problem that has a **unique solution**: they return the linear classifier with the **maximum margin**, that is, the hyperplane that separates the data and is farthest from any of the training vectors.

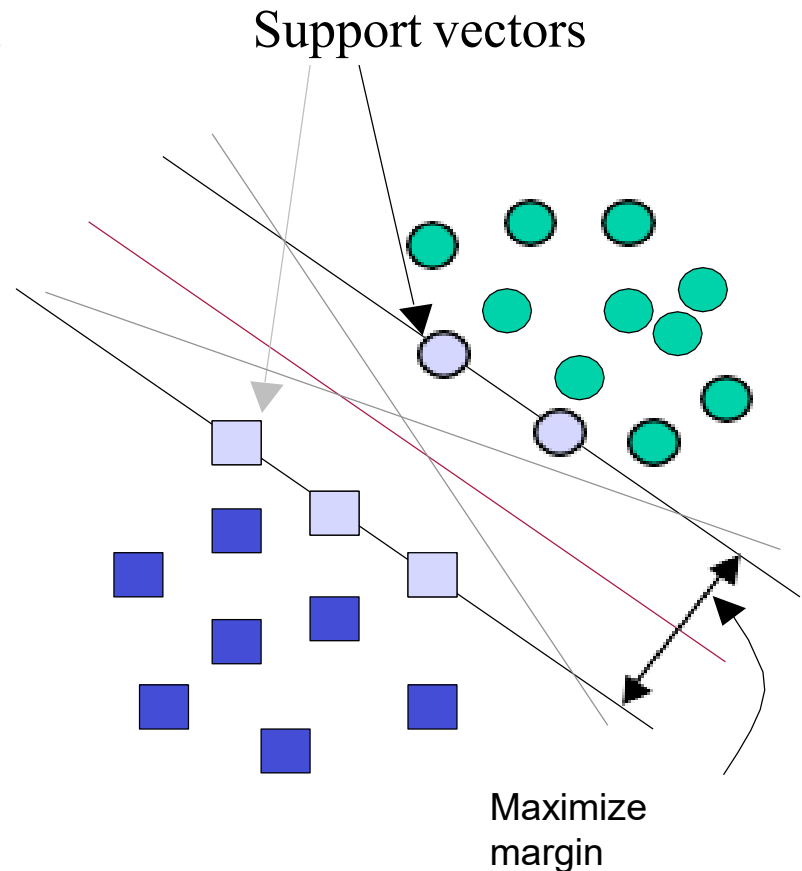
Support vectors

- Support vectors are the data points that lie closest to the decision surface (or hyperplane)
- They are the data points most difficult to classify
- They decide the optimum location of the decision surface

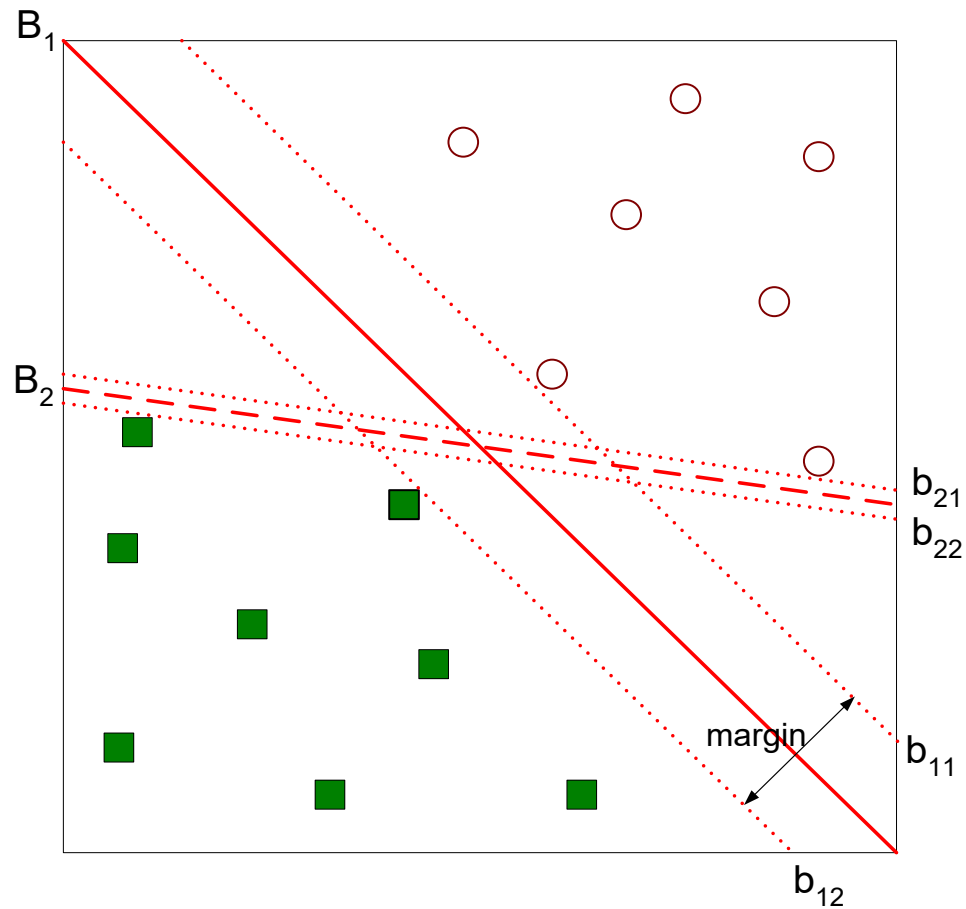


Support Vector Machines

- SVMs maximize the margin (Winston terminology: the 'street') around the separating hyperplane.
- The decision function is fully specified by a (usually very small) subset of training samples, the support vectors.
- This becomes a Quadratic programming problem that is easy to solve by standard methods



Support Vector Machines



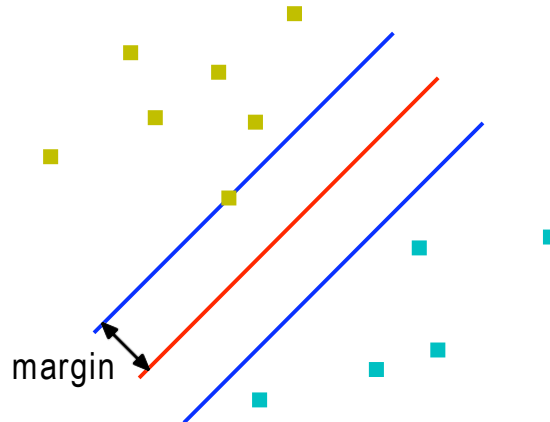
- Find hyperplane **maximizes** the margin => B1 is better than B2

General input/output for SVMs

- Input: set of (input, output) training pair samples; call the input sample features $x_1, x_2 \dots x_n$, and the output result y .
- Typically, there can be lots of input features x_i .
- Output: set of weights w (or w_i), one for each feature, whose linear combination predicts the value of y . (So far, just like previous mentioned methods)
- **Important difference**: we use the optimization of maximizing the margin ('street width') to reduce the number of weights that are nonzero to just a few that correspond to the important features that 'matter' in deciding the separating line (hyperplane)... these nonzero weights correspond to the support vectors (because they 'support' the separating hyperplane)

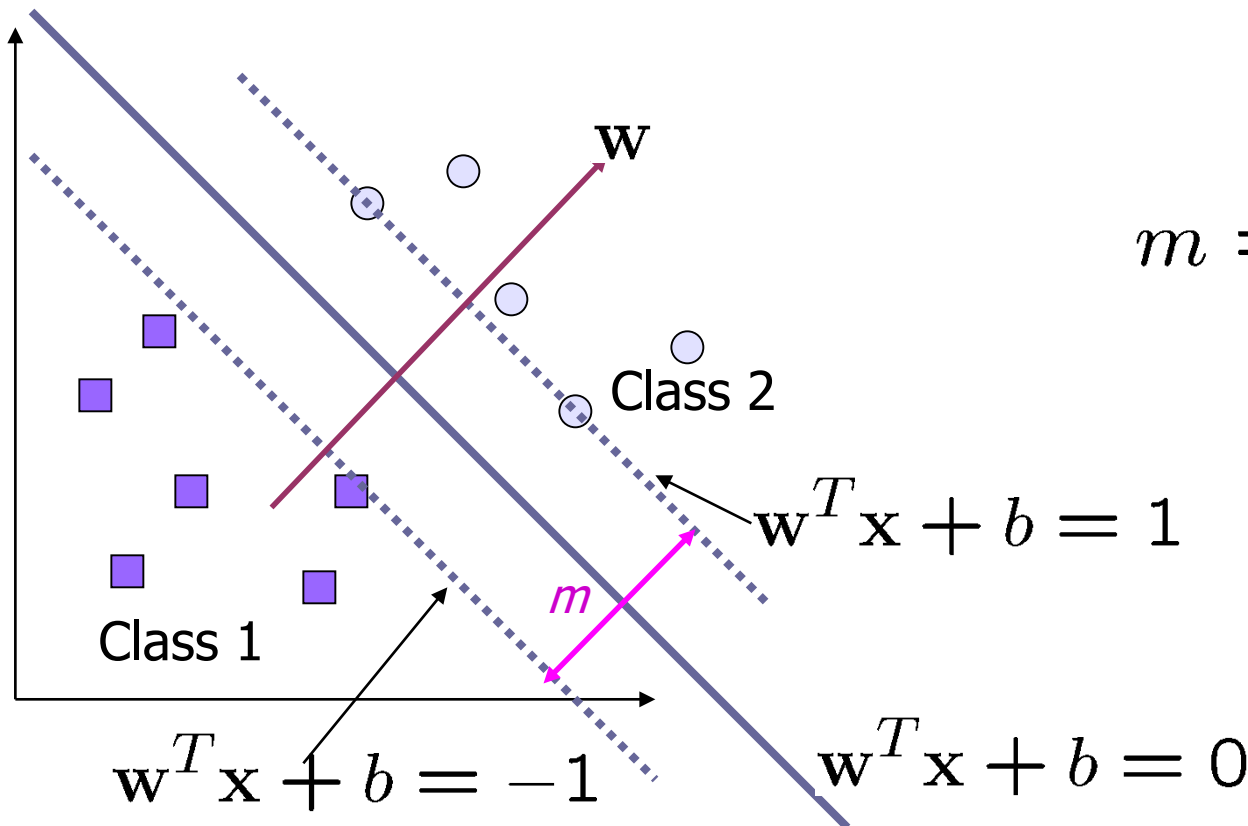
Maximum Margin Classifiers

- When the training data are linearly separable, there are generally an infinite number of solutions for (w, b) that separate the classes exactly.
- The margin of such a classifier is defined as the orthogonal distance in feature space between the decision boundary and the closest training vector.
- SVMs are an example of a maximum margin classifier, which finds the linear classifier that maximizes the margin.



Large-margin Decision Boundary

- The decision boundary should be as far away from the data of both classes as possible
 - We should maximize the margin, m
 - Distance between the origin and the line $\mathbf{w}^T \mathbf{x} = -b$ is $b / ||\mathbf{w}||$



$$m = \frac{2}{||\mathbf{w}||}$$

Finding the Decision Boundary

- Let $\{x_1, \dots, x_n\}$ be our data set and let $y_i \in \{1, -1\}$ be the class label of x_i
- The decision boundary should classify all points correctly
 $\Rightarrow y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad \forall i$
- To see this: when $y=-1$, we wish $(\mathbf{w}\mathbf{x}+b)<1$, when $y=1$, we wish $(\mathbf{w}\mathbf{x}+b)>1$. For support vectors, we wish $y(\mathbf{w}\mathbf{x}+b)=1$.
- The decision boundary can be found by solving the following constrained optimization problem

$$\begin{aligned} & \text{Minimize } \frac{1}{2} \|\mathbf{w}\|^2 \\ & \text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \quad \forall i \end{aligned}$$

Learning Linear SVM

- Decision boundary depends only on support vectors
 - If you have data set with **same support vectors**, decision boundary will **not change**
 - How to classify using SVM once w and b are found? Given a test record, x_i

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 \end{cases}$$

Outline

- Basic Introduction
- The Perceptron Algorithm
- Logistic Classifiers
- Support Vector Machines
- Evaluation for Classification/Imbalanced Issues

Class Imbalance Problem

- Lots of classification problems where the classes are skewed (more records from one class than another)
 - Credit card fraud
 - Intrusion detection
 - Defective products in manufacturing assembly line
 - COVID-19 test results on a random sample

Challenges

- Evaluation measures such as accuracy are not well-suited for imbalanced class
- Detecting the rare class is like finding a needle in a haystack

Confusion Matrix

- Confusion Matrix:

ACTUAL CLASS	PREDICTED CLASS	
	Class=Yes	Class=No
Class=Yes	a	b
	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Accuracy

ACTUAL CLASS	PREDICTED CLASS	
	Class=Yes	Class=No
Class=Yes	a (TP)	b (FN)
	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}$$

Problem with Accuracy

- Consider a 2-class problem
 - Number of Class NO examples = 990
 - Number of Class YES examples = 10

Problem with Accuracy

- Consider a 2-class problem
 - Number of Class NO examples = 990
 - Number of Class YES examples = 10

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	0	10
	Class=No	0	990

Problem with Accuracy

- Consider a 2-class problem
 - Number of Class NO examples = 990
 - Number of Class YES examples = 10
- If a model predicts everything to be class NO, accuracy is $990/1000 = 99\%$
 - This is misleading because the model does not detect any class YES example
 - Detecting the rare class is usually more important (e.g., frauds, intrusions, defects, etc)

Which model is better?

A

	PREDICTED		
		Class=Yes	Class=No
	ACTUAL	Class=Yes	Class=No
		Class=No	Class=No
		0	10
		0	990

B

	PREDICTED		
		Class=Yes	Class=No
	ACTUAL	Class=Yes	Class=No
		Class=No	Class=No
		10	0
		90	900

Which model is better?

A

	PREDICTED		
		Class=Yes	Class=No
	ACTUAL	Class=Yes	5
		Class=No	990

B

	PREDICTED		
		Class=Yes	Class=No
	ACTUAL	Class=Yes	0
		Class=No	900

Alternative Measures

	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

$$\text{Precision (p)} = \frac{a}{a + c}$$

$$\text{Recall (r)} = \frac{a}{a + b}$$

$$\text{F - measure (F)} = \frac{2rp}{r + p} = \frac{2a}{2a + b + c}$$

Alternative Measures

ACTUAL CLASS	PREDICTED CLASS		
		Class=Yes	Class=No
	Class=Yes	10	0
	Class=No	10	980

$$\text{Precision (p)} = \frac{10}{10+10} = 0.5$$

$$\text{Recall (r)} = \frac{10}{10+0} = 1$$

$$\text{F - measure (F)} = \frac{2 * 1 * 0.5}{1 + 0.5} = 0.62$$

$$\text{Accuracy} = \frac{990}{1000} = 0.99$$

ACTUAL CLASS	PREDICTED CLASS		
		Class=Yes	Class=No
	Class=Yes	1	9
	Class=No	0	990

$$\text{Precision (p)} = \frac{1}{1+0} = 1$$

$$\text{Recall (r)} = \frac{1}{1+9} = 0.1$$

$$\text{F - measure (F)} = \frac{2 * 0.1 * 1}{1 + 0.1} = 0.18$$

$$\text{Accuracy} = \frac{991}{1000} = 0.991$$

Measures of Classification Performance

	PREDICTED CLASS		
ACTUAL CLASS		Yes	No
	Yes	TP	FN
	No	FP	TN

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

$$ErrorRate = 1 - accuracy$$

$$Precision = \text{Positive Predictive Value} = \frac{TP}{TP + FP}$$

$$Recall = \text{Sensitivity} = TP \text{ Rate} = \frac{TP}{TP + FN}$$

$$Specificity = TN \text{ Rate} = \frac{TN}{TN + FP}$$

$$FP \text{ Rate} = \alpha = \frac{FP}{TN + FP} = 1 - specificity$$

$$FN \text{ Rate} = \beta = \frac{FN}{FN + TP} = 1 - sensitivity$$

$$Power = sensitivity = 1 - \beta$$

Alternative Measures

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	40	10
	Class=No	10	40

Precision (p) = 0.8

TPR = Recall (r) = 0.8

FPR = 0.2

F-measure (F) = 0.8

Accuracy = 0.8

$$\frac{\text{TPR}}{\text{FPR}} = 4$$

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	40	10
	Class=No	1000	4000

Precision (p) = 0.038

TPR = Recall (r) = 0.8

FPR = 0.2

F-measure (F) = 0.07

Accuracy = 0.8

$$\frac{\text{TPR}}{\text{FPR}} = 4$$

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	40	10
	Class=No	40	10

Precision (p) = 0.5

TPR = Recall (r) = 0.8

FPR = 0.8

F – measure = 0.61

	ACTUAL CLASS		
PREDICTED CLASS		Class=Yes	Class=No
	Class=Yes	40	40
	Class=No	10	10

	ACTUAL CLASS		
PREDICTED CLASS		Class=Yes	Class=No
	Class=Yes	35	35
	Class=No	15	15

Alternative Measures

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	10	40
	Class=No	10	40

$$\text{Precision (p)} = 0.5$$

$$\text{TPR} = \text{Recall (r)} = 0.2$$

$$\text{FPR} = 0.2$$

$$\text{F-measure} = 0.28$$

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	25	25
	Class=No	25	25

$$\text{Precision (p)} = 0.5$$

$$\text{TPR} = \text{Recall (r)} = 0.5$$

$$\text{FPR} = 0.5$$

$$\text{F-measure} = 0.5$$

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	40	10
	Class=No	40	10

$$\text{Precision (p)} = 0.5$$

$$\text{TPR} = \text{Recall (r)} = 0.8$$

$$\text{FPR} = 0.8$$

$$\text{F-measure} = 0.61$$

EE Career Talk

Grab the opportunity to:

- Get comprehensive information on job requirements, career development, recruitment procedure, etc. from each company; and
- Get valuable advice and tips on application and interview.

Date	29 October 2021 (Friday)
Time	3:00 –5:00pm
Venue	Over Zoom
Company	GP Electronics(HK) Limited http://www.gp-industries.com/manuf_elect.htm Lee Kum Kee International Holdings Ltd. https://corporate.lkk.com/en Crypto.com https://crypto.com/ Origo Ltd www.theorigo.com Centre for Intelligent Multidimensional Data Analysis Limited https://www.innocimda.com
Zoom Details	Link: https://cityu.zoom.us/j/94596447356?pwd=aXU4MlRtSEVDY0lsYVZjQXRmWWZVUT09 Meeting ID: 945 9644 7356 Password: 931024

Event details can also be found at:

- EE website – Upcoming Event https://www.ee.cityu.edu.hk/events/upcoming_events
- Email – Subject “EE Career Talk (29 Oct) - Updates with Zoom Details”