TEXAS A&M UNIVERSITY

CSCE 633: Machine Learning

Lecture 2

Outline

- Perceptron
- Example & Representation
- Perceptron Learning Algorithm (PLA)

(* Part of the following slides are taken from Dr. Malik Magdon-Ismail's Machine Learning class.)

Perceptron: Example

Credit approval or denial

- Task: Approve or deny credit (binary classification task)
- Features: Salary, debt, years in residence, etc.

Perceptron: Example & Representation

- Input vector $\mathbf{x} = [x_1, \ldots, x_d]^T$
- Assign importance to input features and compute a Credit Score

$$CreditScore = \sum_{i=1}^{D} w_i x_i$$

In the above, weights convey importance if features are in the same range

- Approve credit if $\sum_{i=1}^{D} w_i x_i > threshold$
- Deny credit if $\sum_{i=1}^{D} w_i x_i < threshold$
- How to choose the importance of weights?

Perceptron: Example & Representation

- Approve credit if $\sum_{i=1}^{D} w_i x_i > \mathit{threshold}$
- Deny credit if $\sum_{i=1}^{D} w_i x_i < \mathit{threshold}$
- Can be written more formally as

$$h(\mathbf{x}) = \mathit{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) + w_0\right)$$

or

$$h(\mathbf{x}) = \mathit{sign}(\mathbf{w}^T \mathbf{x})$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_d]^T$ and $\mathbf{x} = [1, x_1, \ldots, x_d]^T$

Perceptron: Representation

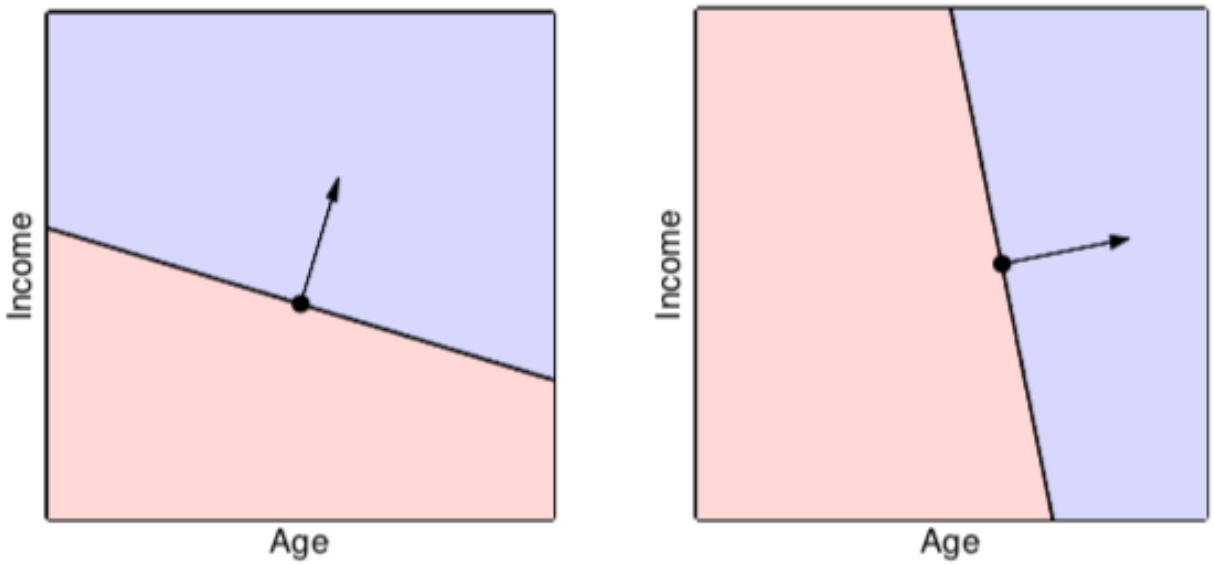
Classify data into two classes

- Input: $\mathbf{x} \in \mathbb{R}^D$ (features, attributes, etc.)
- Output: $y \in \{-1, 1\}$ (labels)
- Model: $h : \mathbf{x} \to y$

$$h(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x}) = \begin{cases} 1\,, & \mathbf{w}^T\mathbf{x} > 0 \\ -1\,, & \mathbf{w}^T\mathbf{x} < 0 \end{cases}$$

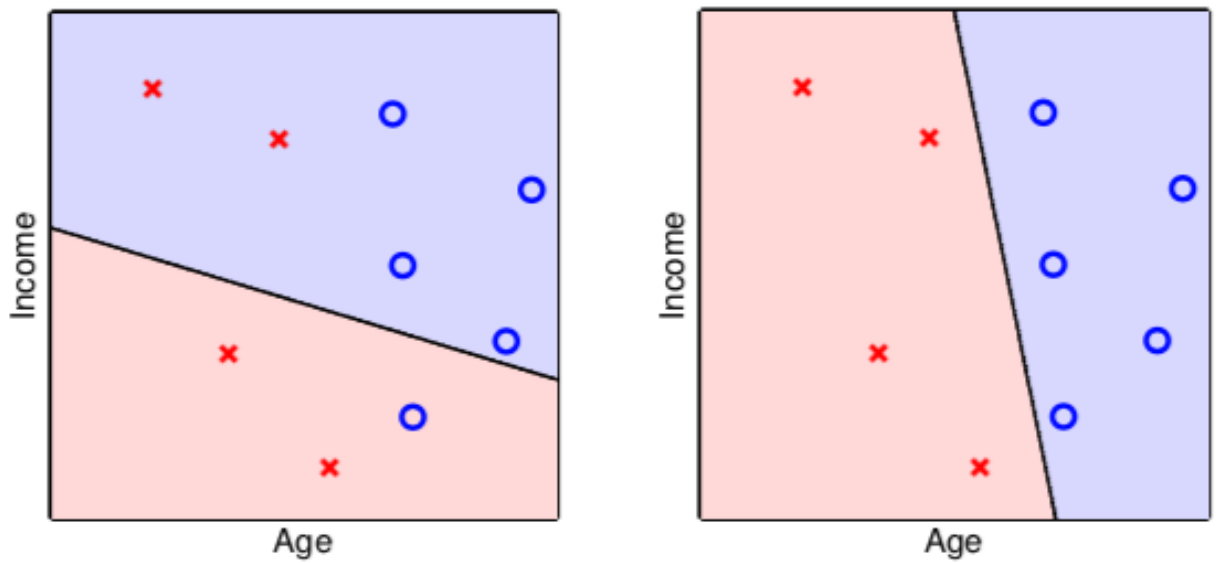
Perceptron: Representation

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^{\text{T}}\mathbf{x})$$ (Problem 1.2 in LFD)



How to find the perpendicular vector to a line?
See Handout on Piazza

Perceptron: Representation



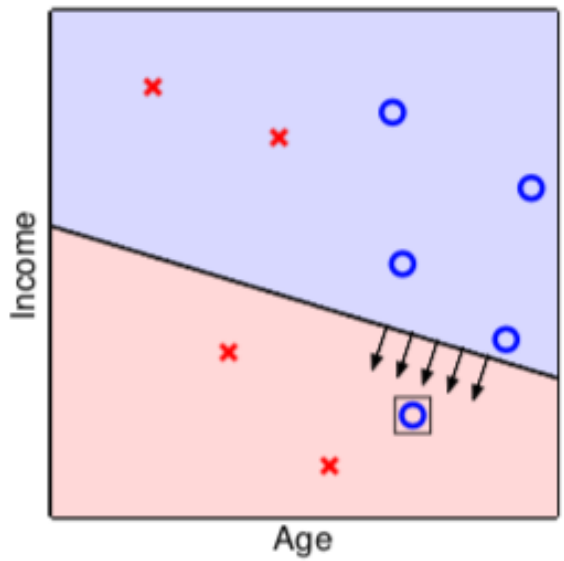
A perceptron fits the data by using a line to separate the $+1$ from -1 data.

Fitting the data: How to find a hyperplane that *separates* the data?

("It's obvious - just look at the data and draw the line," is not a valid solution.)

Perceptron Learning Algorithm

Idea! Start with some weight vector and try to improve it.



Perceptron Learning Algorithm: Example

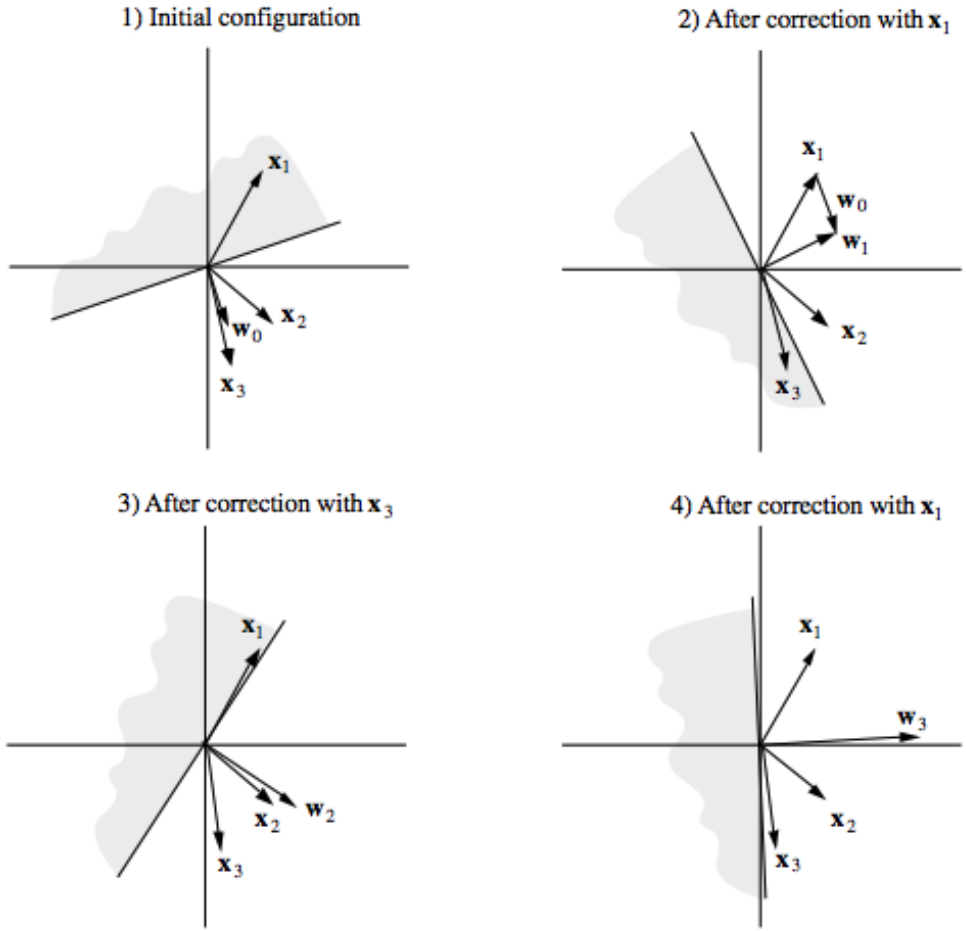


Fig. 4.10. Convergence behavior of the learning algorithm

Assuming that $\{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}\}$ belong to the same class, after the initial updates, successive corrections become smaller and the algorithm "fine tunes" the position of the weight vector.

Perceptron Learning Algorithm

A simple iterative method

Incremental learning on single example at a time

1. Initialization $\mathbf{w(0)} = 0$ (or any other vector)
2. for $t = 1, 2, 3, \ldots$
   a. From $\{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_N}, y_N)\}$ pick a misclassified sample
   b. Call the misclassified sample $(\mathbf{x_s}, y_s)$: $sign(\mathbf{w(t)}^T \mathbf{x_s}) \neq y_s$
   ($\mathbf{w(t)}^T \mathbf{x_s} = -1$ if $y_s = 1$; $\mathbf{w(t)}^T \mathbf{x_s} = 1$ if $y_s = -1$)
   c. Update the weight:
   $\mathbf{w(t+1)} = \mathbf{w(t)} + y_s \mathbf{x_s}$
   d. $t \leftarrow t + 1$

Perceptron Learning Algorithm

Algorithm Learning Rate

- Vectors $\mathbf{w}$ are not necessarily normalized
- If $\|\mathbf{w}(t)\| \gg \|\mathbf{x_s}\|$ the new weight vector $\mathbf{w}(t) + \mathbf{x_s}$ is almost equal to $\mathbf{w}(t)$
- We can add a learning rate $\alpha > 0$ in the update
  - $\mathbf{w}(t+1) = \mathbf{w}(t) \pm \alpha \mathbf{x_s}$
  - if $\alpha \gg 0$, $\mathbf{x_s}$ will have a large impact on the update
  - if $\alpha \equiv 0$, $\mathbf{x_s}$ will not influence much the update
  - Learning rate a typical hyperparameter of many learning algorithms and depends on the data of each problem

Perceptron Learning Algorithm

Algorithm Convergence

- The rule update considers a training sample at a time and may "destroy" the classification of other samples
- If the data can be fit by a linear separator (linearly separable), then after some finite number of steps, PLA is guaranteed to arrive to a correct solution.
- What if the data cannot be fit by a perceptron?
  - We can find infinitely many combinations of perceptrons that fit the data $\rightarrow$ neural networks

Perceptron Learning Algorithm

Algorithm Cost

- PLA is a local, greedy algorithm
- This can lead to an exponential number of updates of the weight
- In the following figure:
  - Two almost antiparallel vectors are to be classified in the same half-space
  - PLA rotates the separation line in one of the two directions and will require more and more time when the angle between the two vectors approaches 180 degrees

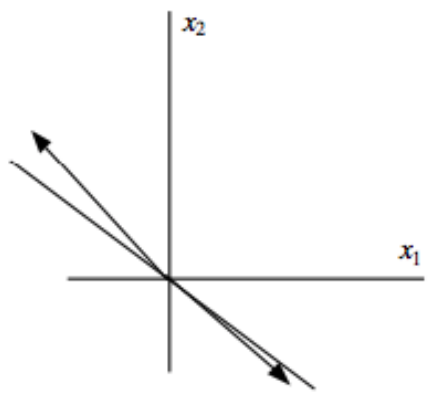


Fig. 4.13. Worst case for perceptron learning (input space)

Parametric v.s. non-parametric models

- Many possible ways to categorize learning models
- One way is to ask this question
  - Does the model have a fixed number of parameters or does it grow with the amount of training data?
- Non-parametric models (or instance/memory-based)
  - more flexible
  - computationally intractable for large datasets
  - e.g. K-NN
- Parametric models
  - faster to use
  - make strong assumptions about data
  - e.g. linear perceptron, linear regression

Perceptron Learning Algorithm

Question

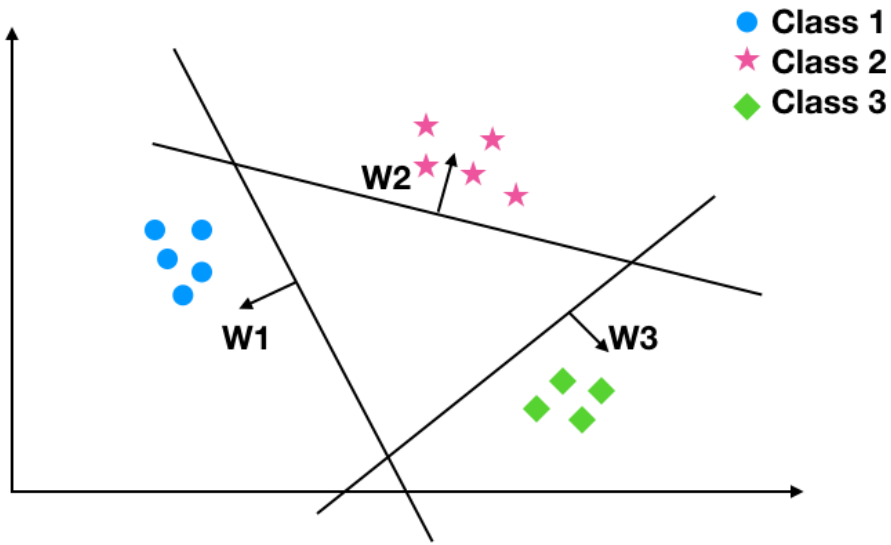
Assume the following 2-dimensional data space with three classes C_1, C_2, C_3, and three linear classification boundaries $\mathbf{w_1}$, $\mathbf{w_2}$, and $\mathbf{w_3}$. If $\mathbf{x_1} \in C_1$, $\mathbf{x_2} \in C_2$, and $\mathbf{x_3} \in C_3$, please select the equations that hold:

(A) $\mathbf{w}_1^T\mathbf{x}_1 > 0$, $\mathbf{w}_2^T\mathbf{x}_2 > 0$, $\mathbf{w}_3^T\mathbf{x}_3 > 0$

(B) $\mathbf{w}_1^T\mathbf{x}_1 < 0$, $\mathbf{w}_2^T\mathbf{x}_2 < 0$, $\mathbf{w}_3^T\mathbf{x}_3 < 0$

(C) $\mathbf{w}_1^T\mathbf{x}_1 > 0$, $\mathbf{w}_2^T\mathbf{x}_2 < 0$, $\mathbf{w}_3^T\mathbf{x}_3 < 0$

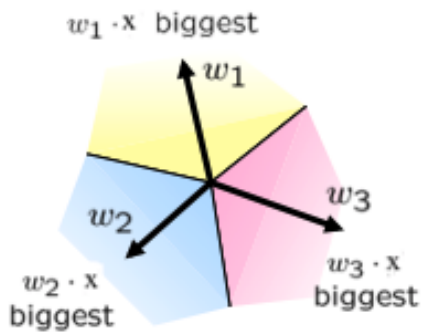
(D) $\mathbf{w}_1^T\mathbf{x}_1 < 0$, $\mathbf{w}_2^T\mathbf{x}_2 > 0$, $\mathbf{w}_3^T\mathbf{x}_3 > 0$



Multiclass perceptron

Perceptron representation for more than two classes

- Input: $\mathbf{x} \in \mathbb{R}^D$
- Output: $y \in \{1, 2, \ldots, K\}$
- Training data: $\mathcal{D}^{train} = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_N}, y_N)\}$
- Model: $y = f(\mathbf{x}) = arg\,\max_{k \in \{1,\ldots,K\}} \mathbf{w_k}^T \mathbf{x}$
- Model parameters: weights $\mathbf{w_1}, \ldots, \mathbf{w_K}$



Learning occurs in the similar manner as the PLA for the 2-class perceptron.

Three Components Of Learning

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search
Hyperplanes	Likelihood	Branch-and-bound
Naive Bayes	Posterior probability	Continuous optimization
Logistic regression	Information gain	Unconstrained
Decision trees	K-L divergence	Gradient descent
Sets of rules	Cost/Utility	Conjugate gradient
Propositional rules	Margin	Quasi-Newton methods
Logic programs		Constrained
Neural networks		Linear programming
Graphical models		Quadratic programming
Bayesian networks		
Conditional random fields		

a learner must be represented in some formal language

an evaluation function assesses the performance of a learner

find the highest-scoring learner

Summary

- Perceptron is a simple learning algorithm
  - decision boundary defined by a hyper-plane
  - hyper-plane is learned from the training data
- Reading materials
  - Abu-Mostafa 1.1.2