Attendance Code:
26444542

Artificial Intelligence and Machine Learning (AIML)

2023–24

Attendance Code:
26444542

- **Last lecture**: classification in ML
- **This lecture**: classification using the perceptron algorithm

AIML

Attendance Code:
26444542

Example: health insurance company

- Data on whether customers bought the plan

Client	Age (yrs)	Income (k £)	Bought?
1	25	30	No
2	45	60	Yes
3	30	50	Yes
4	22	25	No
5	35	45	Yes
6	55	70	Yes
7	40	55	No
8	60	80	Yes
9	50	40	No
10	28	35	No

- Task: predict whether a new customer is likely to buy or not the plan, given their age and income.
  - Goal: split the data into 2 **classes** (bought/didn't buy) that best match **class-labeled training data**.

AIML

Attendance Code: **26444542**

Example: health insurance company

- Data on whether customers bought the plan

Client	Age (yrs)	Income (k £)	Bought?
1	25	30	No
2	45	60	Yes
3	30	50	Yes
4	22	25	No
5	35	45	Yes
6	55	70	Yes
7	40	55	No
8	60	80	Yes
9	50	40	No
10	28	35	No

- Task: predict whether a new customer is likely to buy or not the plan, given their age and income.
  - Goal: split the data into 2 **classes** (bought/didn't buy) that best match **class-labeled training data**.
  - Classification model: linear classifier

$$f(w, x) = \text{sign}(w^T x)$$

$$f: \mathbb{R}^D \to \{-1, +1\}$$

AIML

Attendance Code:
26444542

Example: health insurance company

- Data on whether customers bought the plan

Client	Age (yrs)	Income (k £)	Bought?
1	25	30	No
2	45	60	Yes
3	30	50	Yes
4	22	25	No
5	35	45	Yes
6	55	70	Yes
7	40	55	No
8	60	80	Yes
9	50	40	No
10	28	35	No

- Task: predict whether a new customer is likely to buy or not the plan, given their age and income.
  - Goal: split the data into 2 **classes** (bought/didn't buy) that best match **class-labeled training data**.
  - Classification model: linear classifier

$$f(w, x) = \text{sign}(w^T x)$$

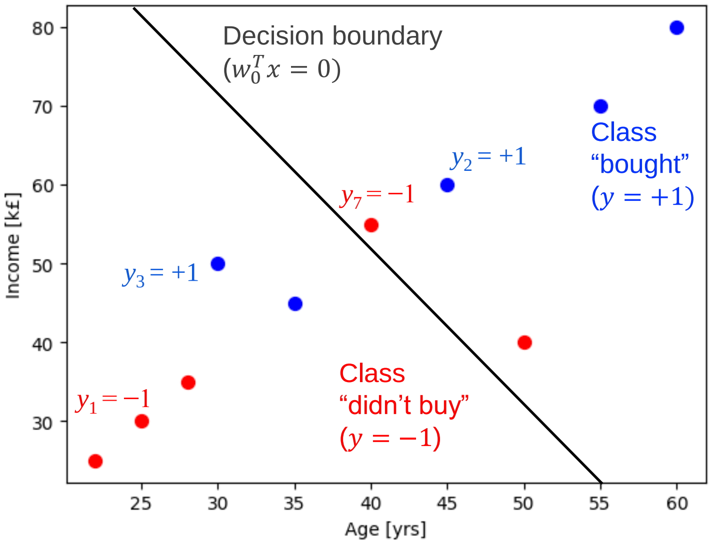
$$f: \mathbb{R}^D \to \{-1, +1\}$$

  - Initial guess: $w_0 = [-130, 2, 1]^T$

Attendance Code:
26444542

Example: health insurance company

- Data on whether customers bought the plan



- Task: predict whether a new customer is likely to buy or not the plan, given their age and income.
  - Goal: split the data into 2 **classes** (bought/didn't buy) that best match **class-labeled training data**.
  - Classification model: linear classifier

$$f(w, x) = \text{sign}(w^T x)$$

$$f\colon \mathbb{R}^D \to \{-1, +1\}$$

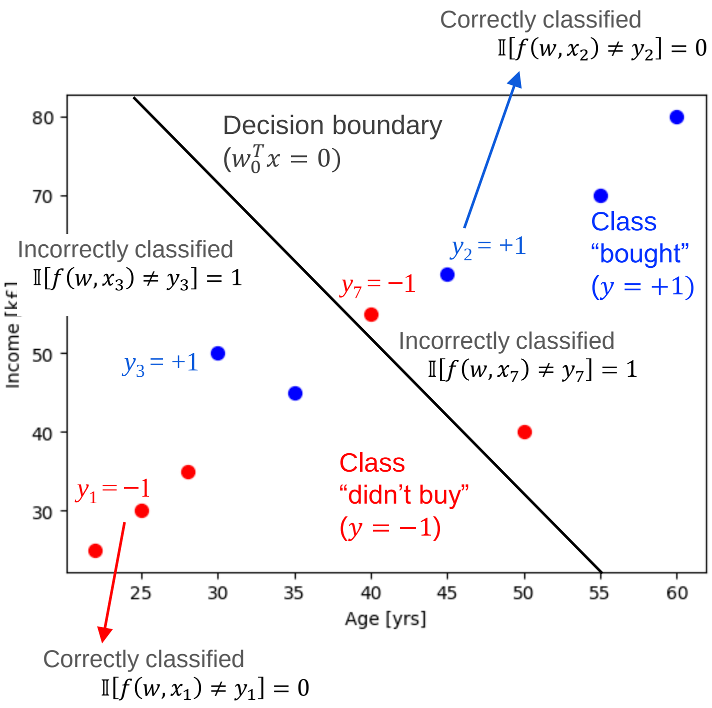
  - Initial guess: $w_0 = [-130, 2, 1]^T$

AIML

Attendance Code: **26444542**

Example: health insurance company

- Data on whether customers bought the plan



  - Misclassification error: number of misclassified data points

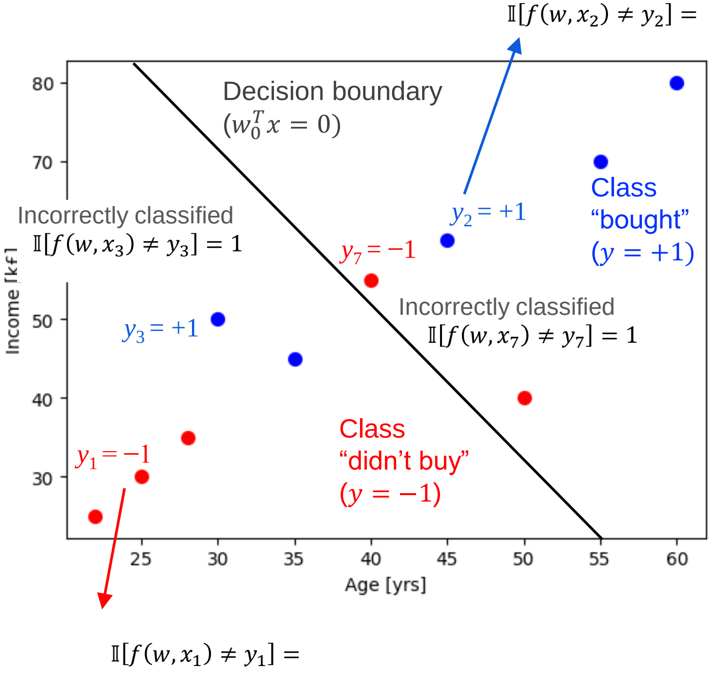
$$F(w) = \sum_{i=1}^{N} \mathbb{I}[f(w, x_i) \neq y_i]$$

    - assigns the same penalty to all incorrect decisions, regardless of how 'bad' they are.

Attendance Code:
26444542

Example: health insurance company

Initial guess: $w_0 = [-130, 2, 1]^T$



- Now let's look at y1

Client	Age (yrs)	Income (k £)	Bought?
1	25	30	No

What should be the loss here?

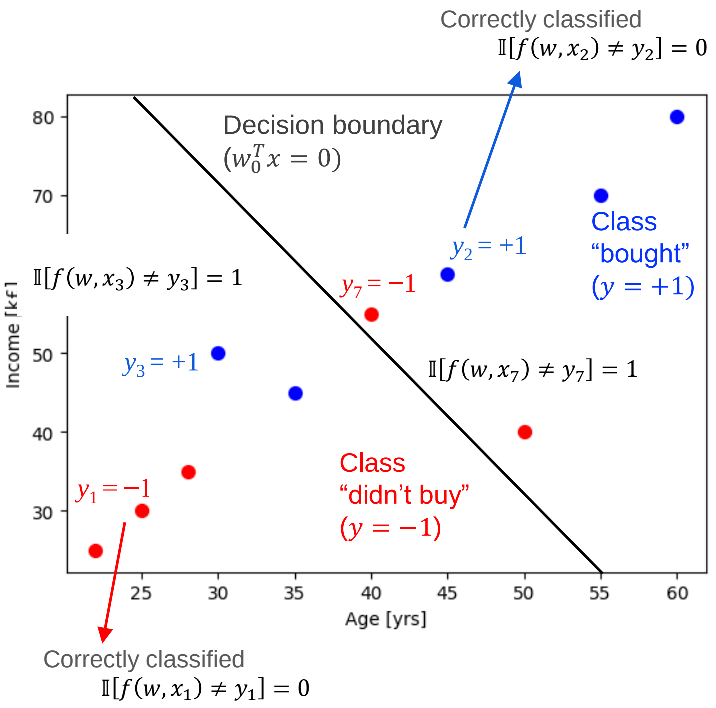
How about y2?

2	45	60	Yes

Attendance Code:
26444542

Example: health insurance company

Initial guess: $w_0 = [-130, 2, 1]^T$



- Now let's look at y3

3	30	50	Yes

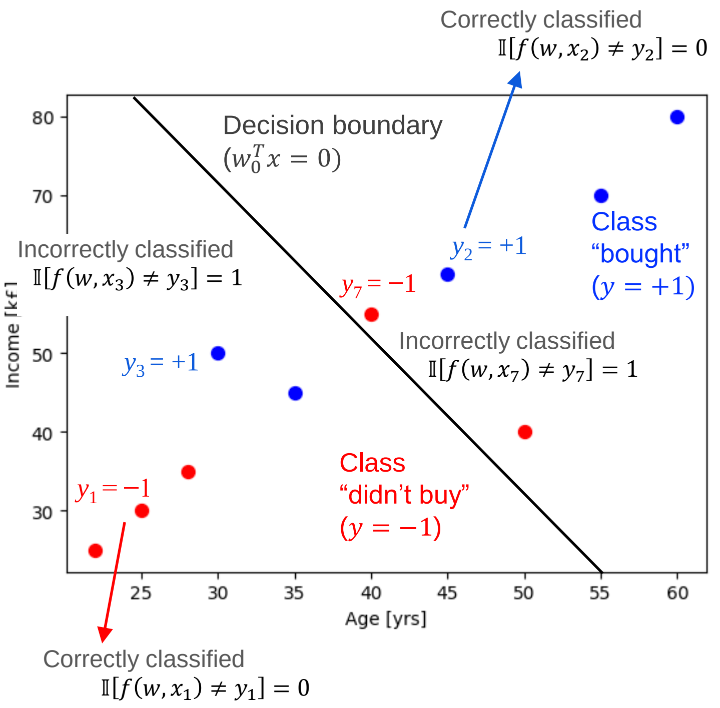
- How about y7?

7	40	55	No

Attendance Code:
26444542

Example: health insurance company

Initial guess: $w_0 = [-130, 2, 1]^T$



- Now let's look at y3

3	30	50	Yes

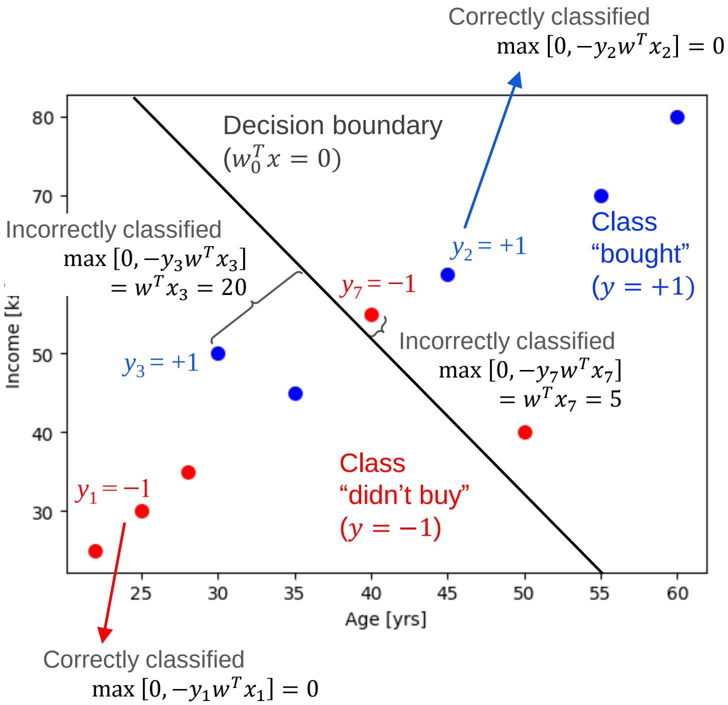
- How about y7?

7	40	55	No

Attendance Code:
26444542

Example: health insurance company

- Data on whether customers bought the plan



- Misclassification error: number of misclassified data points

$$F(w) = \sum_{i=1}^{N} \mathbb{I}[f(w, x_i) \neq y_i]$$

  - assigns the same penalty to all incorrect decisions, regardless of how 'bad' they are.

- Perceptron error: sum of perpendicular distances of every **misclassified** data point to the decision boundary,

$$F(w) = \sum_{i=1}^{N} \max\,(0, -y_i w^T x_i)$$

  - 'Penalizes' incorrect decisions by the distance from the decision boundary $w^T x$ in the direction w (perpendicular distance).

Attendance Code:
26444542

SGD: algorithm (Section 9 Lecture Notes)

- **Step 1**. *Initialization*: Select an initial guess for w_0, a convergence tolerance $\varepsilon > 0$, step size (learning rate) parameter $\alpha > 0$, set iteration number $n=0$
- **Step 2**. *Gradient descent step*: Compute new model parameters,

$$w_{n+1} = w_n - \alpha F_w(w_n)$$

- **Step 3**. *Convergence test*: Compute new loss function value $F(w_{n+1})$, and loss function improvement, $\Delta F = |F(w_{n+1}) - F(w_n)|$ and if $\Delta F < \varepsilon$, exit with solution $w^*=w_{n+1}$
- **Step 4**. *Iteration*: update $n=n+1$ and go to step 2.

Attendance Code:
26444542

Perceptron classification

- Classification model (D-dimensional):

$$f(w, x) = \mathrm{sign}(w_1 x^1 + w_2 x^2 + \cdots + w_D x^D) = \mathrm{sign}(w^T x)$$

Attendance Code:
26444542

Perceptron classification

- Classification model (D-dimensional):

$$f(w, x) = \text{sign}(w_1 x^1 + w_2 x^2 + \cdots + w_D x^D) = \text{sign}(w^T x)$$

- Perceptron error function:

$$F(w) = \sum_{i=1}^{N} \max(0, -y_i w^T x_i)$$

Attendance Code:
26444542

Perceptron classification

- Classification model (D-dimensional):

$$f(w, x) = \text{sign}(w_1 x^1 + w_2 x^2 + \cdots + w_D x^D) = \text{sign}(w^T x)$$

- Perceptron error function:

$$F(w) = \sum_{i=1}^{N} \max\,(0, -y_i w^T x_i)$$

- Gradient with respect to w:

$$F_w(w) = -\sum_{i=1}^{N} y_i\, x_i\, \mathbb{I}[-y_i w^T x_i \geq 0]$$

- Intuitively, gradient is just sum of $-y_i x_i$ over incorrectly classified points

Attendance Code:
26444542

Perceptron training: algorithm

- **Step 1**. *Initialization*: Select a starting candidate classification model w_0, set iteration number $n = 0$, choose maximum number of iterations R and learning rate $\alpha > 0$
- **Step 2**. *Gradient descent step*: Compute new model parameters: taking each $i = 1, 2, \ldots, N$ in turn, if $\mathrm{sign}(w_n^T x_i) \neq y_i$,then
$$\mathrm{w}_{n+1} = w_n + \alpha y_i x_i$$
- **Step 3**. *Iteration*: If $n < R$, update $n = n + 1$, go to step 2, otherwise exit with solution $w^* = w_n$.

Perceptron training: algorithm

- **Step 1**. *Initialization*: Select a starting candidate classification model w_0, set iteration number $n = 0$, choose maximum number of iterations R and learning rate $\alpha > 0$
- **Step 2**. *Gradient descent step*: Compute new model parameters: taking each $i = 1, 2, \ldots, N$ in turn, if $\text{sign}(w_n^T x_i) \neq y_i$, then

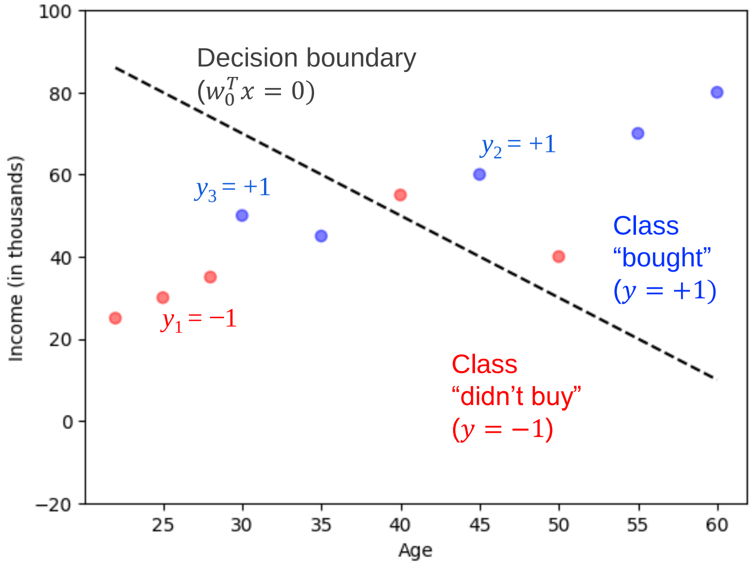
$$w_{n+1} = w_n + \alpha y_i x_i$$

- **Step 3**. *Iteration*: If $n < R$, update $n = n + 1$, go to step 2, otherwise exit with solution $w^* = w_n$.

Attendance Code:
26444542

Example: health insurance company

- Perceptron algorithm in action

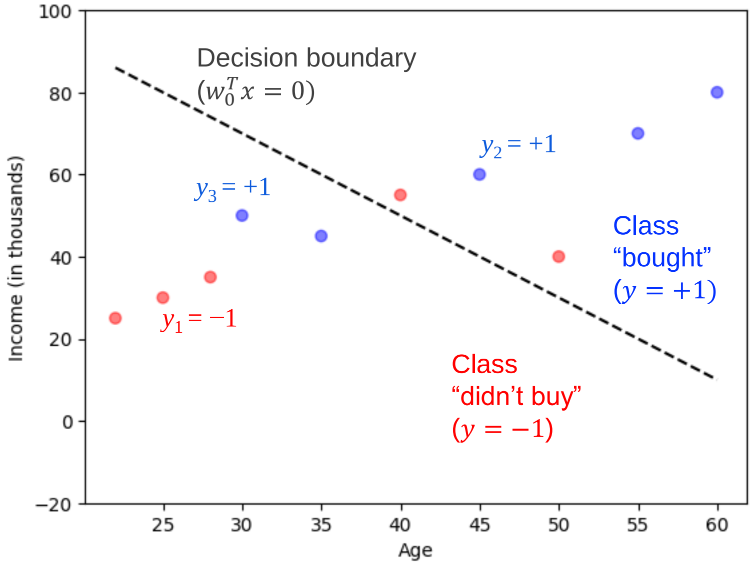


Client	Age (yrs)	Income (k £)	Bought?
1	25	30	No
2	45	60	Yes
3	30	50	Yes
4	22	25	No
5	35	45	Yes
6	55	70	Yes
7	40	55	No
8	60	80	Yes
9	50	40	No
10	28	35	No

Attendance Code:
26444542

Example: health insurance company

- Perceptron algorithm in action



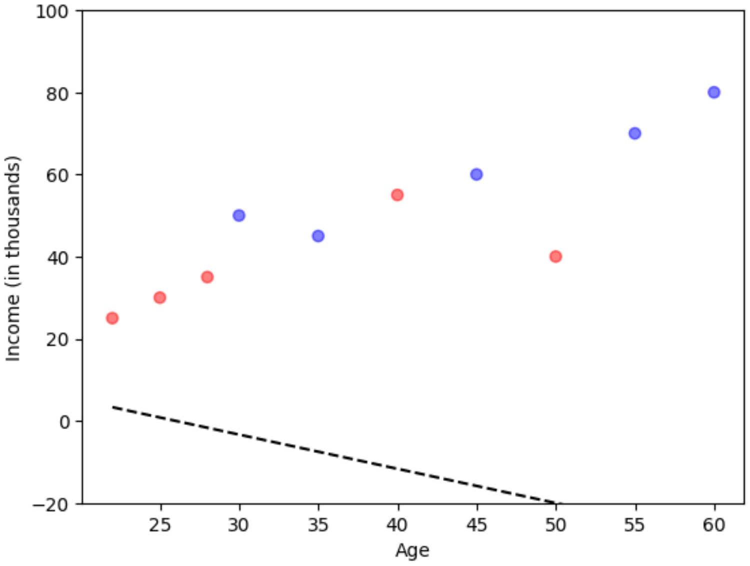
  - $n = 0,\ w_0 = [-130, 2, 1]^T,\ R = 10, \alpha = 0.1$:
    - $i = 3, w_1 = w_0 + 0.1 \times (+1) \times \begin{bmatrix} 1 \\ 30 \\ 50 \end{bmatrix} = [-129.9, 5, 6]^T$

AIML

Attendance Code: **26444542**

Example: health insurance company

- Perceptron algorithm in action



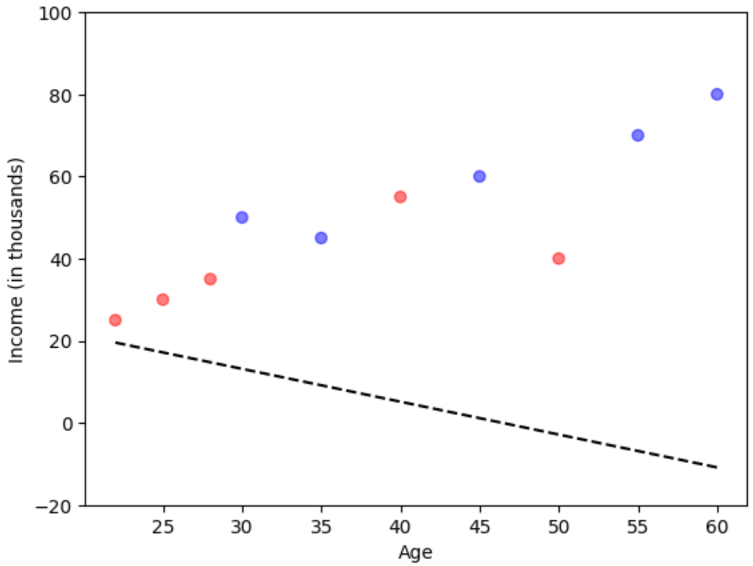
  - $n = 0,\ w_0 = [-130, 2, 1]^T,\ R = 10, \alpha = 0.1$:
    - $i = 3, w_1 = w_0 + 0.1 \times (+1) \times \begin{bmatrix} 1 \\ 30 \\ 50 \end{bmatrix} = [-129.9, 5, 6]^T$
    - $i = 4, w_1 = [-129.9, 5, 6]^T + 0.1 \times (-1) \times \begin{bmatrix} 1 \\ 22 \\ 25 \end{bmatrix} = [-130, 2.8, 3.5]^T$

AIML

Attendance Code:
26444542

Example: health insurance company

- Perceptron algorithm in action



  - $n = 0,\ w_0 = [-130, 2, 1]^T,\ R = 1000,\ \alpha = 0.1:$
    - $i = 3, w_1 = w_0 + 0.1 \times (+1) \times \begin{bmatrix} 1 \\ 30 \\ 50 \end{bmatrix} = [-129.9, 5, 6]^T$
    - $i = 4, w_1 = [-129.9, 5, 6]^T + 0.1 \times (-1) \times \begin{bmatrix} 1 \\ 22 \\ 25 \end{bmatrix} = [-130, 2.8, 3.5]^T$
    - $i = 7, w_1 = [-130, 2.8, 3.5]^T + 0.1 \times (+1) \times \begin{bmatrix} 1 \\ 40 \\ 55 \end{bmatrix} = [-130.1, -1.2, -2]^T$

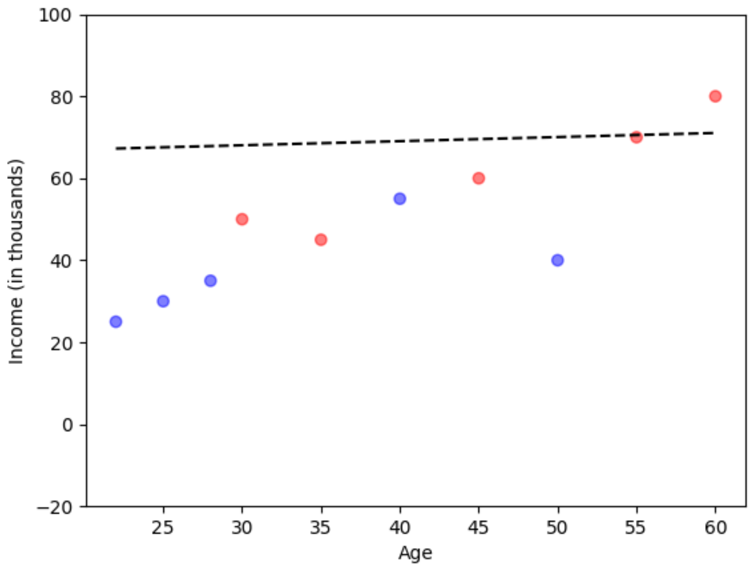
    ....

AIML

Attendance Code: **26444542**

Example: health insurance company

- Perceptron algorithm in action



- $n = 0,\ w_0 = [-130, 2, 1]^T,\ R = 1000,\ \alpha = 0.1$:
  - $i = 3, w_1 = w_0 + 0.1 \times (+1) \times \begin{bmatrix} 1 \\ 30 \\ 50 \end{bmatrix} = [-129.9, 5, 6]^T$
  - $i = 4, w_1 = [-129.9, 5, 6]^T + 0.1 \times (-1) \times \begin{bmatrix} 1 \\ 22 \\ 25 \end{bmatrix} = [-130, 2.8, 3.5]^T$
  - $i = 7, w_1 = [-130, 2.8, 3.5]^T + 0.1 \times (+1) \times \begin{bmatrix} 1 \\ 40 \\ 55 \end{bmatrix} = [-130.1, -1.2, -2]^T$

  ....
- $n = 1,\ w_1 = [-130, -0.2, 2]^T$:

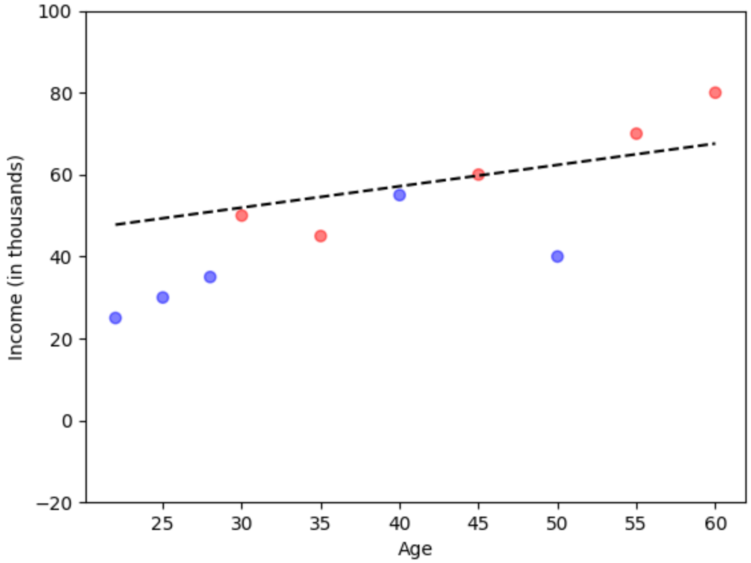
  Update weights, and so on... until n = R

AIML

Attendance Code:
26444542

Example: health insurance company

- Perceptron algorithm in action



- $n = 0,\ w_0 = [-130, 2, 1]^T,\ R = 1000,\ \alpha = 0.1$:
  - $i = 3, w_1 = w_0 + 0.1 \times (+1) \times \begin{bmatrix} 1 \\ 30 \\ 50 \end{bmatrix} = [-129.9, 5, 6]^T$
  - $i = 4, w_1 = [-129.9, 5, 6]^T + 0.1 \times (-1) \times \begin{bmatrix} 1 \\ 22 \\ 25 \end{bmatrix} = [-130, 2.8, 3.5]^T$
  - $i = 7, w_1 = [-130, 2.8, 3.5]^T + 0.1 \times (+1) \times \begin{bmatrix} 1 \\ 40 \\ 55 \end{bmatrix} = [-130.1, -1.2, -2]^T$

  ....
- $n = 1,\ w_1 = [-130, -0.2, 2]^T$:

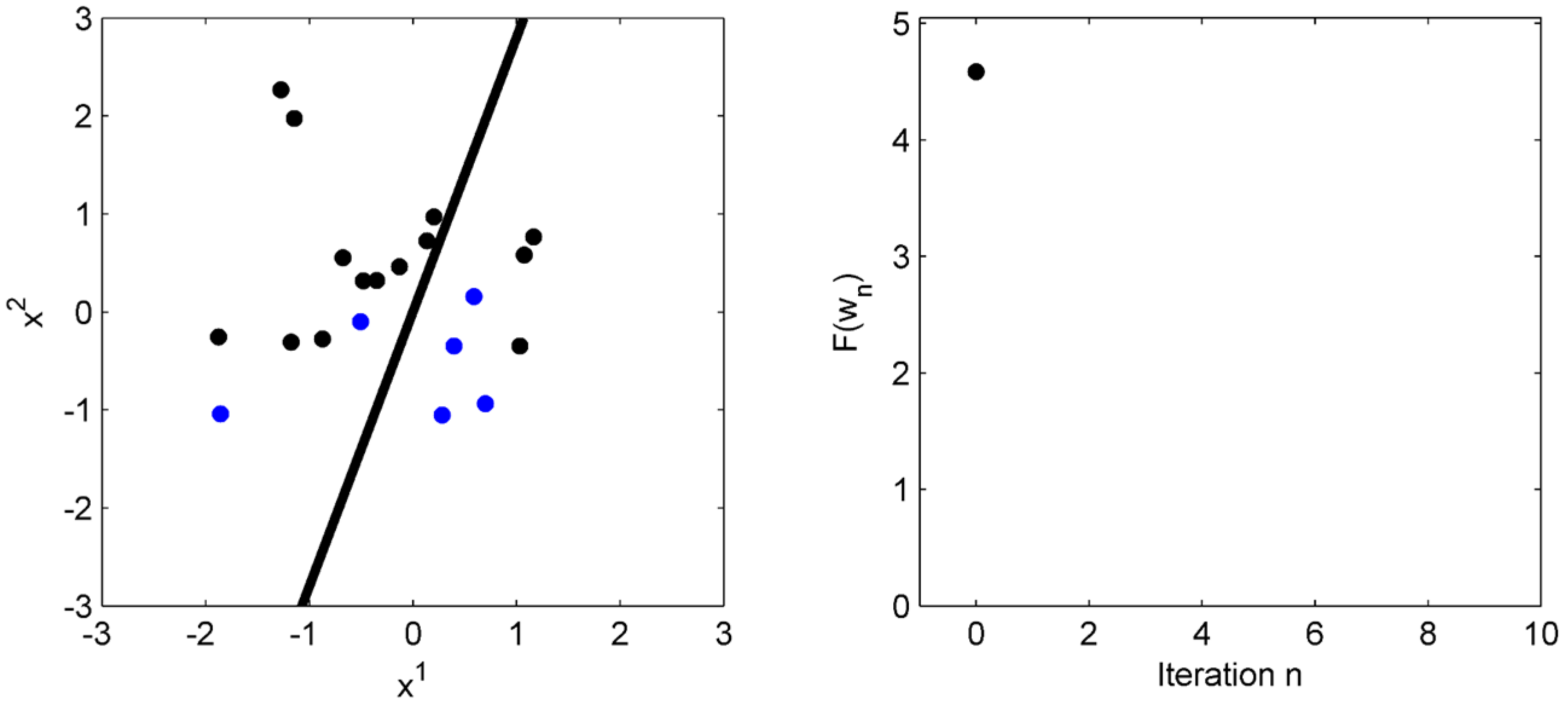
  Update weights, and so on… until n = R
- $n = 999,\ w^\star = [-128.98, -1.85, 3.55]^T$:

Perceptron algorithm: analysis

- If the data is linearly separable, perceptron algorithm always converges on a decision boundary with zero error but no guarantee on the number of iterations required to reach fixed point
- If data is not linearly separable, no convergence guarantee – can cycle between local optima of the perceptron error function, so we need to stop after some number of iterations R

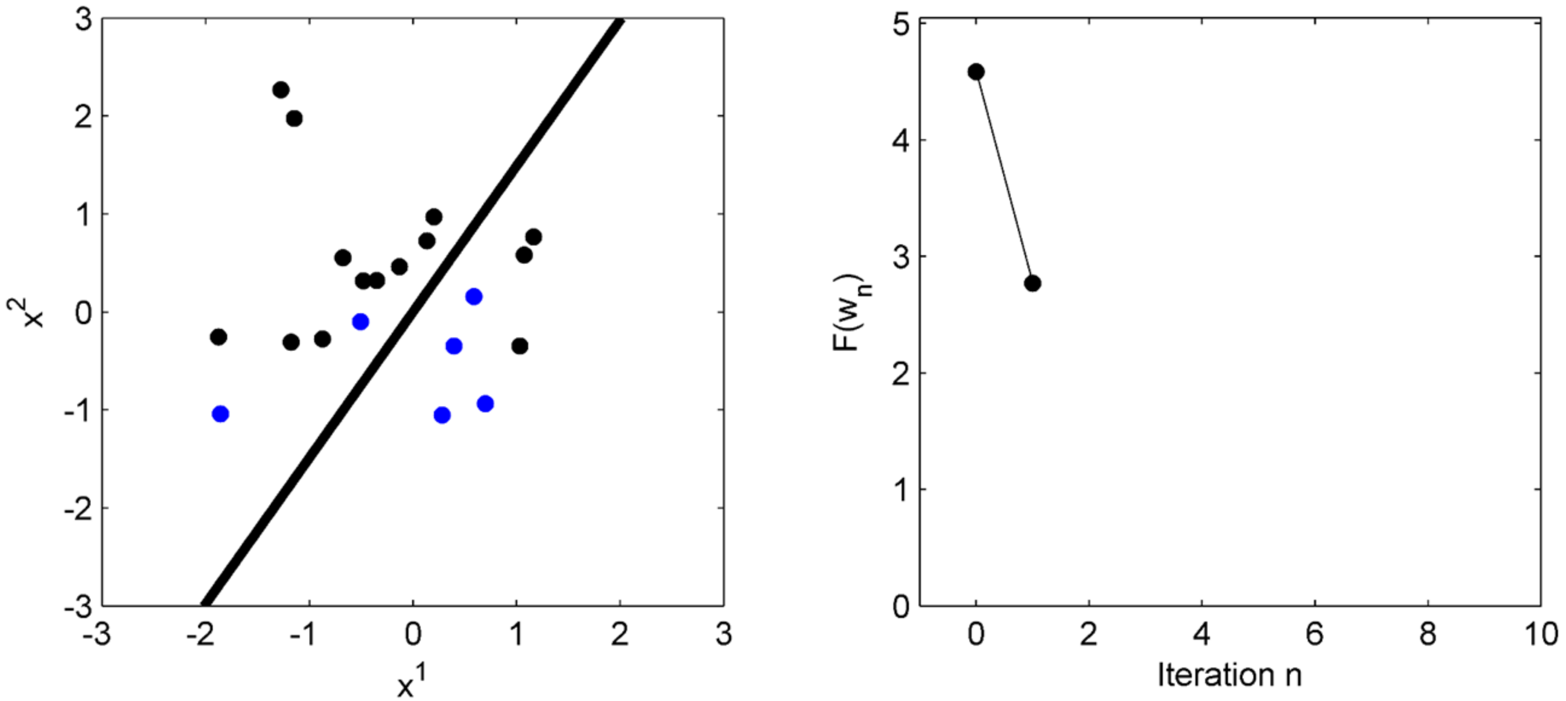
Attendance Code:
26444542

Perceptron training in action



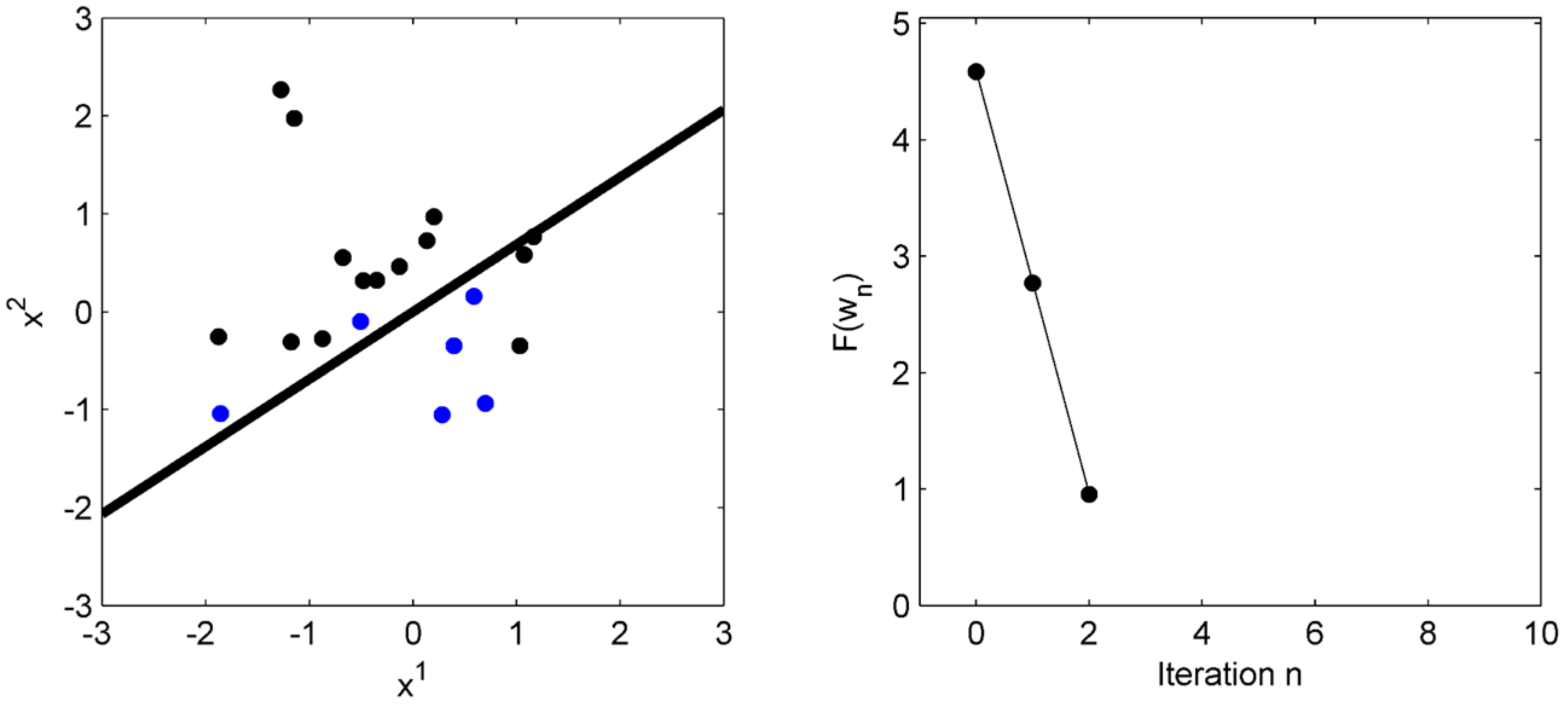
Attendance Code:
26444542

Perceptron training in action



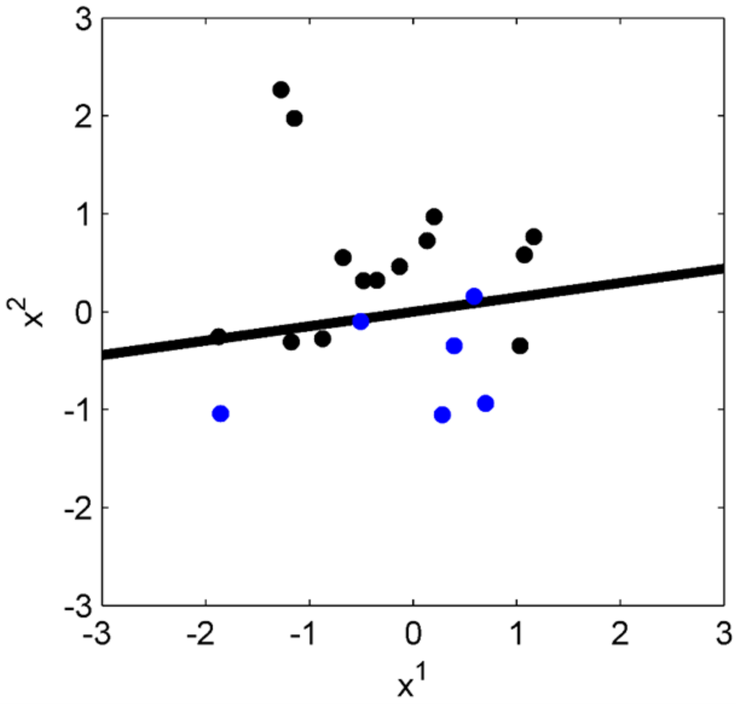
Attendance Code:
26444542

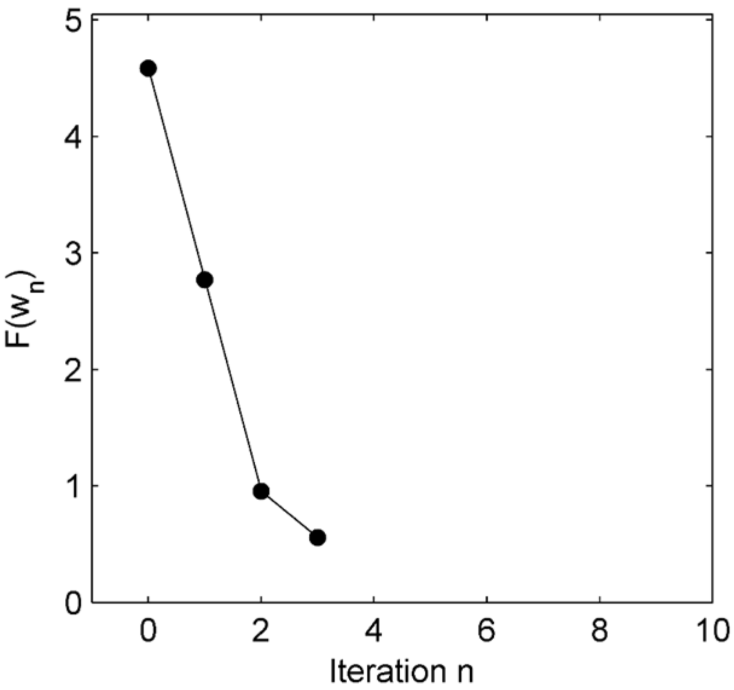
Perceptron training in action



Attendance Code:
26444542

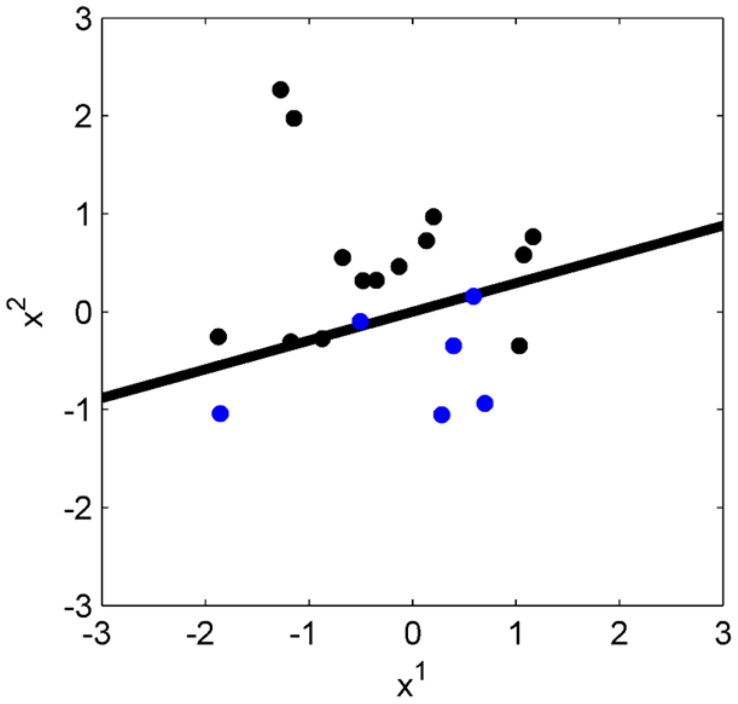
Perceptron training in action

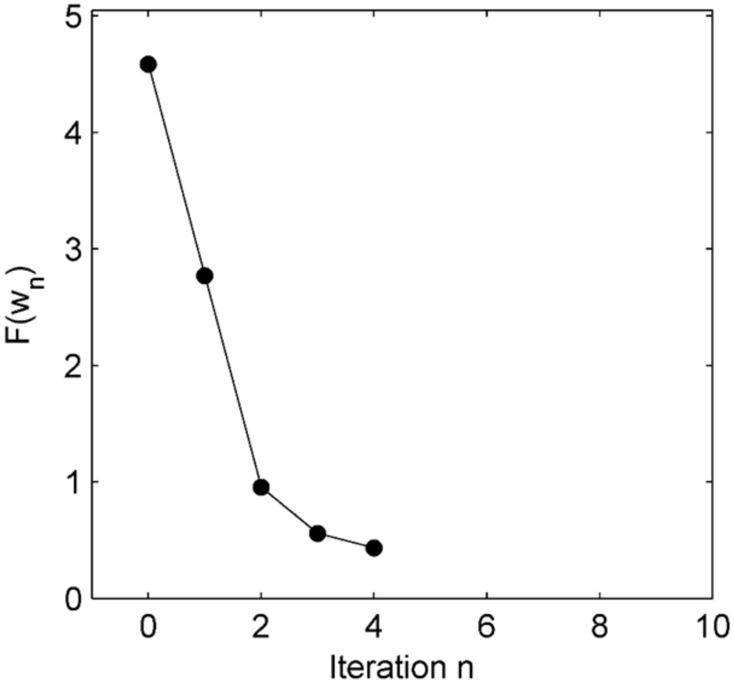




Attendance Code:
26444542

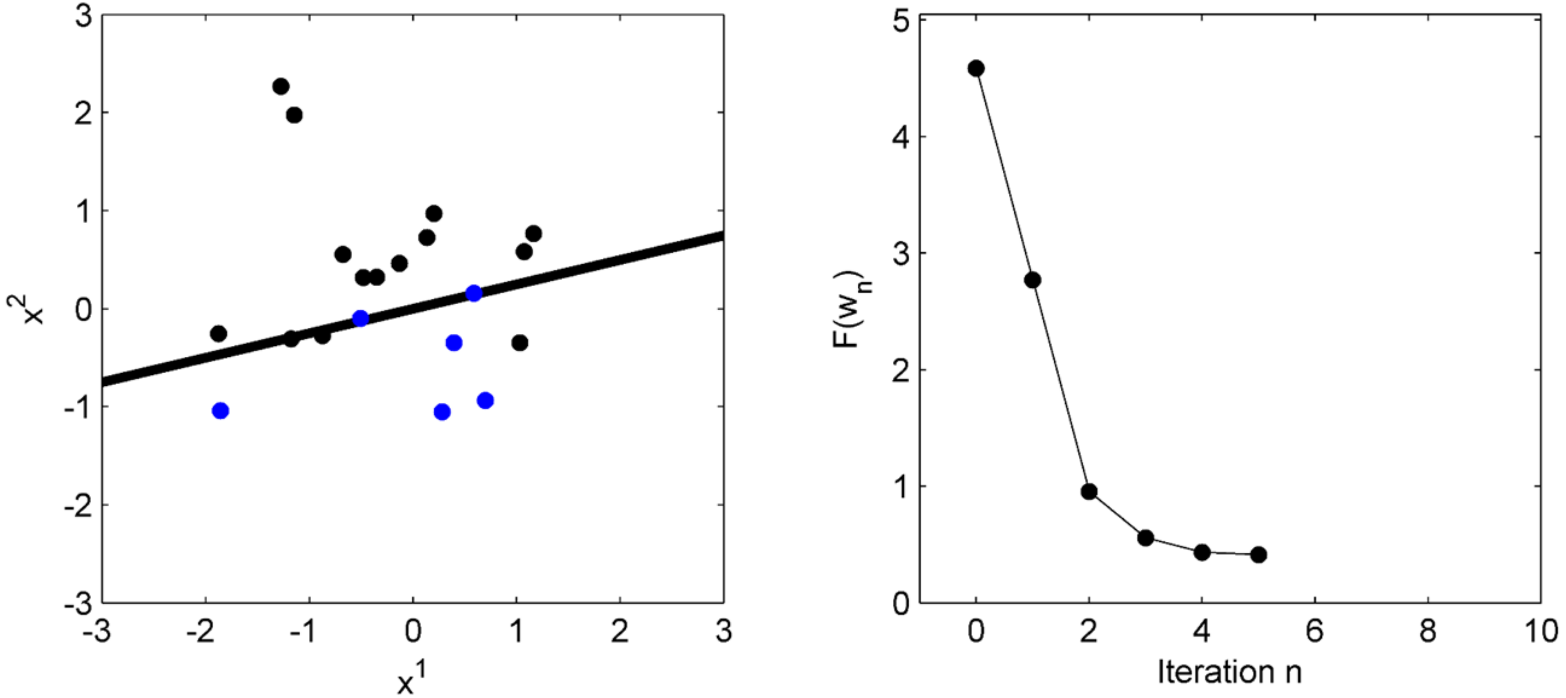
Perceptron training in action





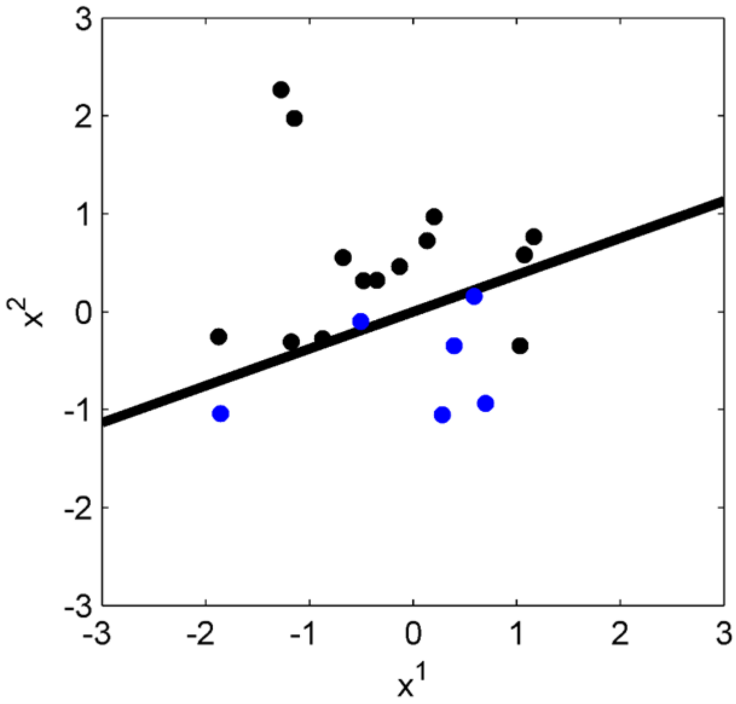
Attendance Code:
26444542

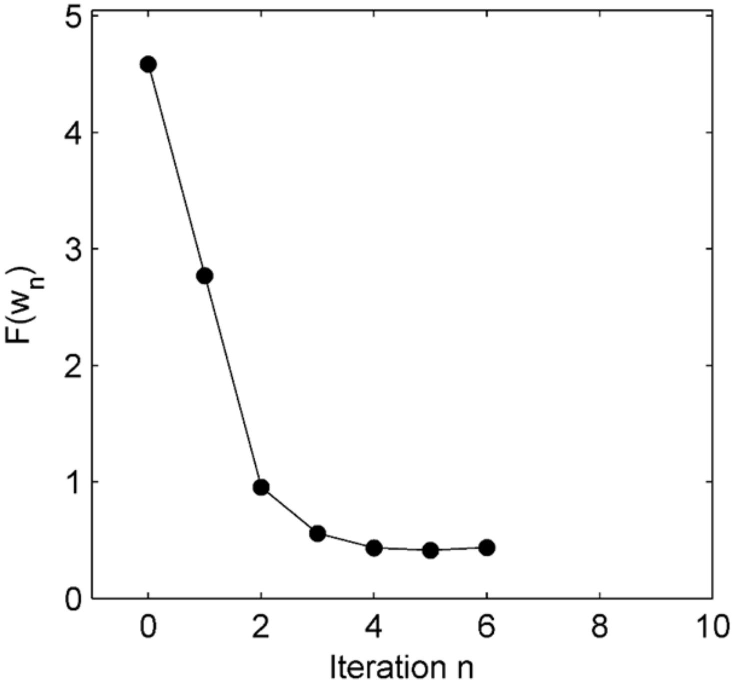
Perceptron training in action



Attendance Code:
26444542

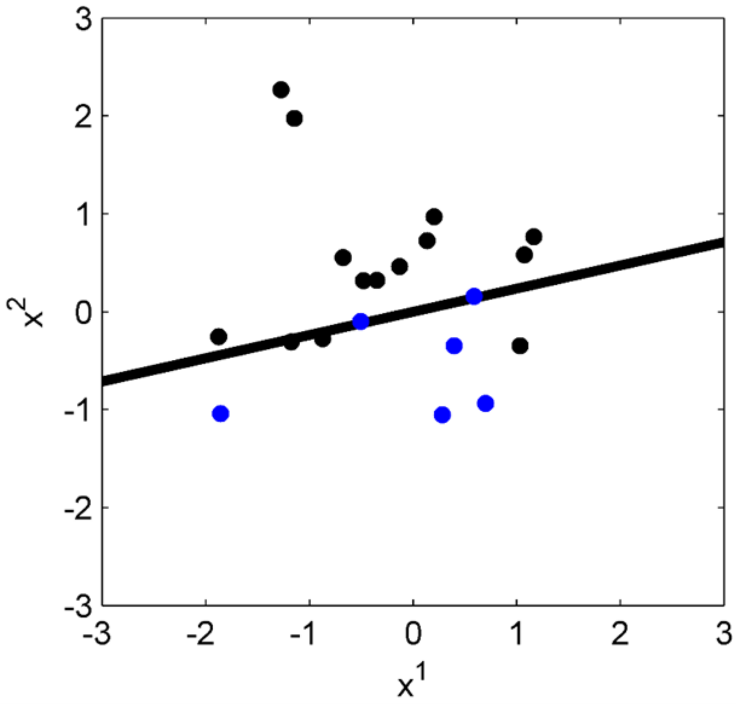
Perceptron training in action

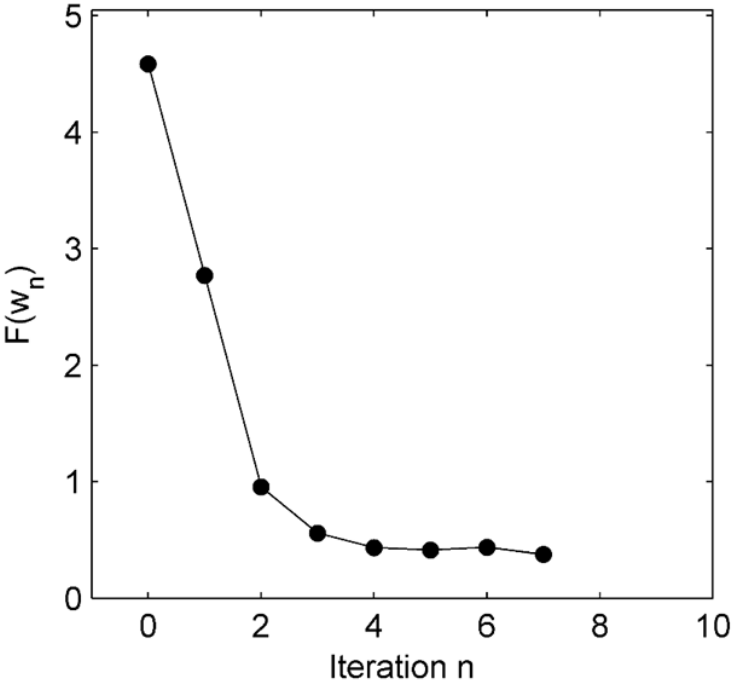




Attendance Code:
26444542

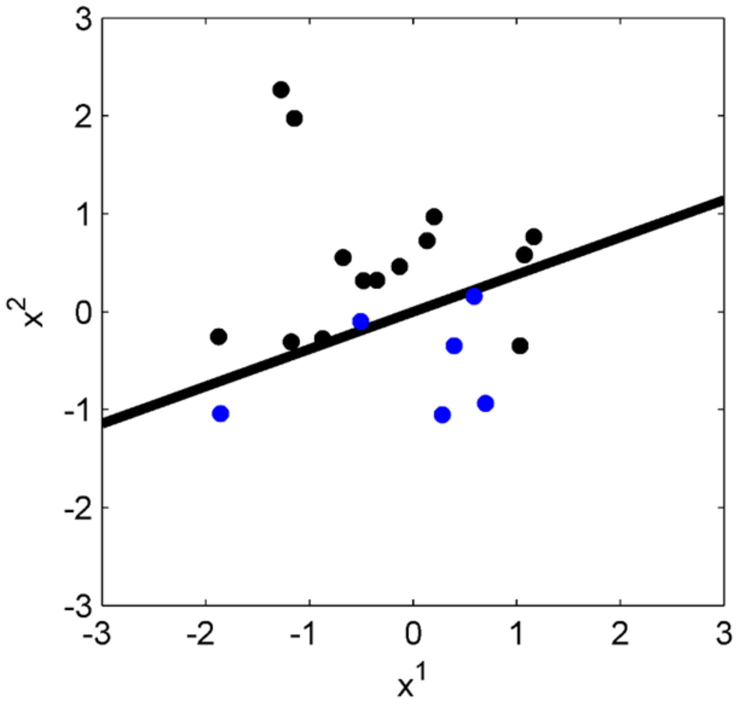
Perceptron training in action

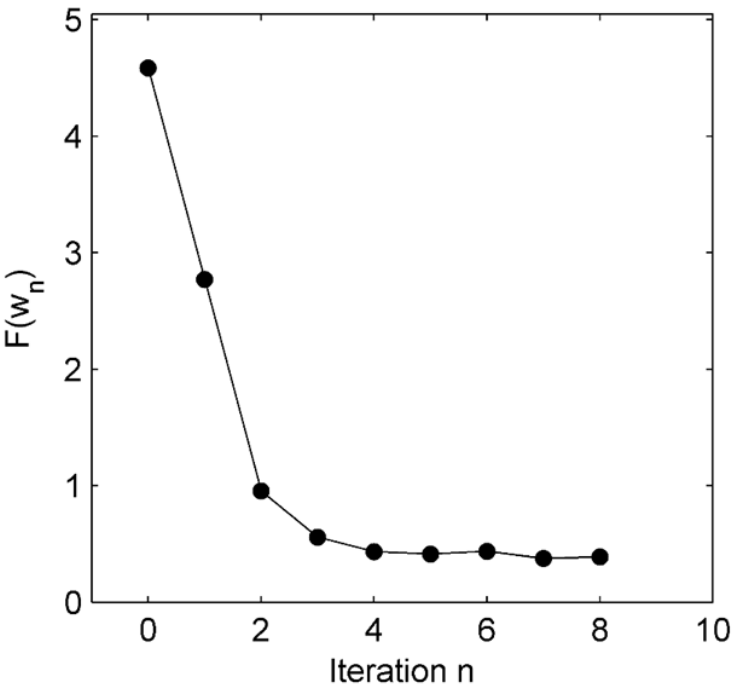




Attendance Code:
26444542

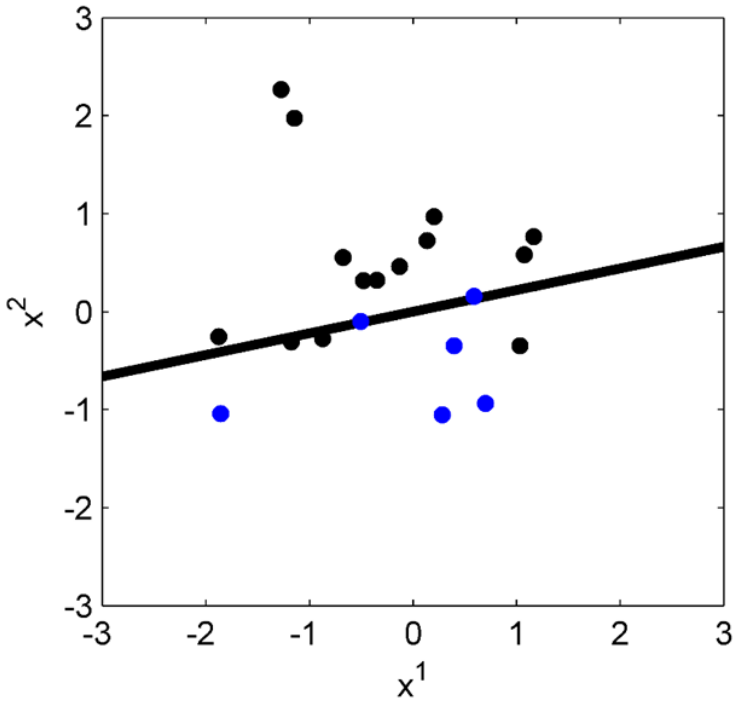
Perceptron training in action

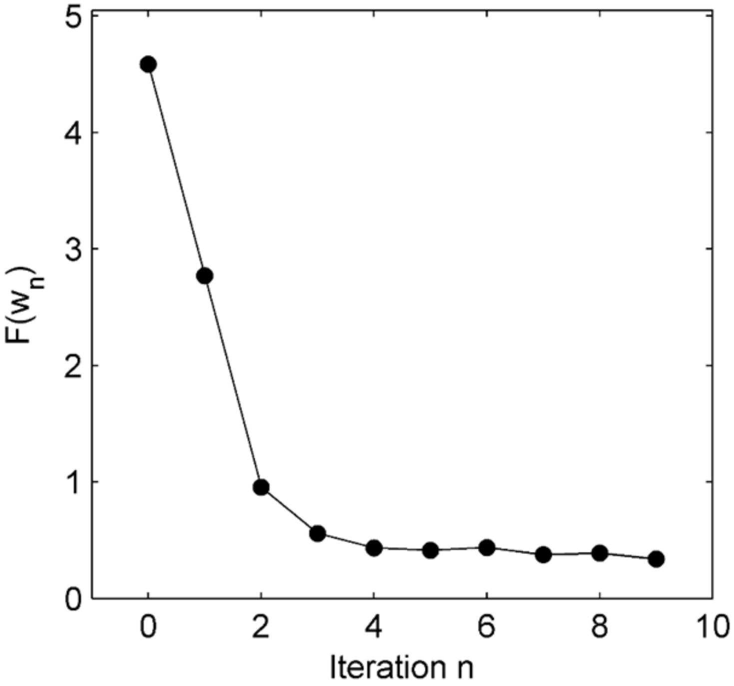




Attendance Code:
26444542

Perceptron training in action





Attendance Code:
26444542

Some remarks about the perceptron

- Simple linear classifier based on the **perceptron error function** rather than the misclassification error function
- Very important classic algorithm in the history of ML, direct precursor to modern deep learning algorithms
- Extremely simple; there are mathematically better "linear single-layer" algorithms (e.g. support vector machines) so the perceptron is rarely used in practice today
- Understanding the perceptron critical to understanding most of the main principles of modern ML classification

AIML

Attendance Code:
26444542

To recap

- We learned the **perceptron algorithm** for classifying data.
  - It only converges if training data is linearly separable (and solution may not be unique)
- Question: how would you generalize the algorithm to K > 2 classes?
- **Next**: Neural networks (we will answer the question above)

Further Reading

- **PRML**, Section 4.1.7
- **R&N**, Section 18.6.3
- **H&T**, Section 4.5.1