Machine Learning 2007: Lecture 7

Instructor: Tim van Erven (Tim.van.Erven@cwi.nl)
Website: `www.cwi.nl/˜erven/teaching/0708/ml/`

October 18, 2007

Overview

- **Organisational Matters**
- Answers Exercises 2
- Linear Functions as Inner Products
- Vector Valued Outputs in Regression and Classification
- Neural Networks and the Perceptron
  - Neural Networks
  - The Perceptron
  - Implementing Boolean Functions with a Perceptron
- Convex Functions
- Gradient Descent (part 1)

Course Organisation

- Room of the intermediate exam changed to: **Q105**.
- Not necessary to enroll on tisvu.

Course Organisation

- Room of the intermediate exam changed to: **Q105**.
- Not necessary to enroll on tisvu.
- Next lecture (in two weeks) will be on Wednesday at 13.30-15.15 in room KC159.

Course Organisation

- Room of the intermediate exam changed to: **Q105**.
- Not necessary to enroll on tisvu.
- Next lecture (in two weeks) will be on Wednesday at 13.30-15.15 in room KC159.
- Do not submit Office 2007 (.docx) files for the homework. Pdf is preferred; older Office (.doc) is acceptable.

Course Organisation

- Room of the intermediate exam changed to: **Q105**.
- Not necessary to enroll on tisvu.
- Next lecture (in two weeks) will be on Wednesday at 13.30-15.15 in room KC159.
- Do not submit Office 2007 (.docx) files for the homework. Pdf is preferred; older Office (.doc) is acceptable.

Mitchell:

- Read: Chapter 4, sections 4.1–4.4.

This Lecture:

- Explanation of linear functions as inner products is needed to understand Mitchell.
- Neural networks are in Mitchell. I have some extra pictures.
- Convex functions are not discussed in Mitchell.
- I will give more background on gradient descent.

Overview

- Organisational Matters
- **Answers Exercises 2**
- Linear Functions as Inner Products
- Vector Valued Outputs in Regression and Classification
- Neural Networks and the Perceptron
  - Neural Networks
  - The Perceptron
  - Implementing Boolean Functions with a Perceptron
- Convex Functions
- Gradient Descent (part 1)

Overview

- Organisational Matters
- Answers Exercises 2
- **Linear Functions as Inner Products**
- Vector Valued Outputs in Regression and Classification
- Neural Networks and the Perceptron
  - Neural Networks
  - The Perceptron
  - Implementing Boolean Functions with a Perceptron
- Convex Functions
- Gradient Descent (part 1)

Linear Functions as Inner Products

Linear Function:

$$h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + \ldots + w_d x_d$$

- $\mathbf{x} = (x_1, \ldots, x_d)^\top$ is a d-dimensional feature vector.
- $\mathbf{w} = (w_0, w_1, \ldots, w_d)^\top$ is a $d+1$-dimensional weight vector.

Linear Functions as Inner Products

Linear Function:

$$h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + \ldots + w_d x_d$$

- $\mathbf{x} = (x_1, \ldots, x_d)^\top$ is a d-dimensional feature vector.
- $\mathbf{w} = (w_0, w_1, \ldots, w_d)^\top$ is a $d+1$-dimensional weight vector.

As Inner Products (a standard trick):

We may change $\mathbf{x}$ into a $d+1$-dimensional vector $\mathbf{x}'$ by adding an imaginary extra feature x_0, which always has value 1:

$$\mathbf{x} = (x_1, \ldots, x_d)^\top \quad \Rightarrow \quad \mathbf{x}' = (1, x_1, \ldots, x_d)^\top$$

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{i=0}^{d} w_i x_i' = \langle \mathbf{w}, \mathbf{x}' \rangle$$

- Mitchell writes $\mathbf{w} \cdot \mathbf{x}'$ for $\langle \mathbf{w}, \mathbf{x}' \rangle$.

Overview

- Organisational Matters
- Answers Exercises 2
- Linear Functions as Inner Products
- **Vector Valued Outputs in Regression and Classification**
- Neural Networks and the Perceptron
  - Neural Networks
  - The Perceptron
  - Implementing Boolean Functions with a Perceptron
- Convex Functions
- Gradient Descent (part 1)

Vector Valued Outputs

Reminder:

- Regression: Predict the label y for any feature vector $\mathbf{x}$. Typically y can take infinitely many values.
- Classification: Predict the class label y for any new feature vector $\mathbf{x}$. Only finitely many categories for y.

Vector Valued Outputs

Reminder:

- Regression: Predict the label y for any feature vector $\mathbf{x}$. Typically y can take infinitely many values.
- Classification: Predict the class label y for any new feature vector $\mathbf{x}$. Only finitely many categories for y.

Vector Valued Outputs:

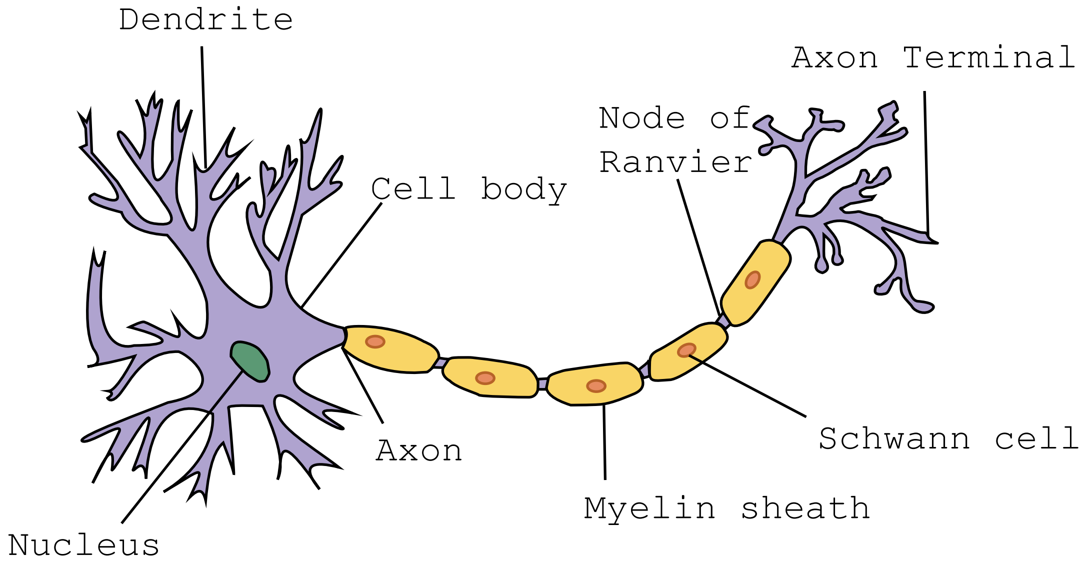
- In our definition the label y is a single value.
- This can be generalised to a label **vector** $\mathbf{y}$.
- Neural networks typically output label vectors.

Overview

- Organisational Matters
- Answers Exercises 2
- Linear Functions as Inner Products
- Vector Valued Outputs in Regression and Classification
- Neural Networks and the Perceptron
  - **Neural Networks**
  - The Perceptron
  - Implementing Boolean Functions with a Perceptron
- Convex Functions
- Gradient Descent (part 1)

Biology

A Neuron [Wikimedia Commons]:



The Brain:

- The brain is a complex network of approximately $10^{11} = 100\,000\,000\,000$ neurons.
- On average each neuron is connected to approximately $10^4 = 10\,000$ other neurons.
- Each neuron has many input channels (dendrites) and one output channel (axon).

Artificial Neurons

An Artificial Neuron:

An (artificial) **neuron** is some function h that gets a feature vector $\mathbf{x}$ as input and outputs a (single) label y.

Artificial Neurons

An Artificial Neuron:

An (artificial) **neuron** is some function h that gets a feature vector $\mathbf{x}$ as input and outputs a (single) label y.

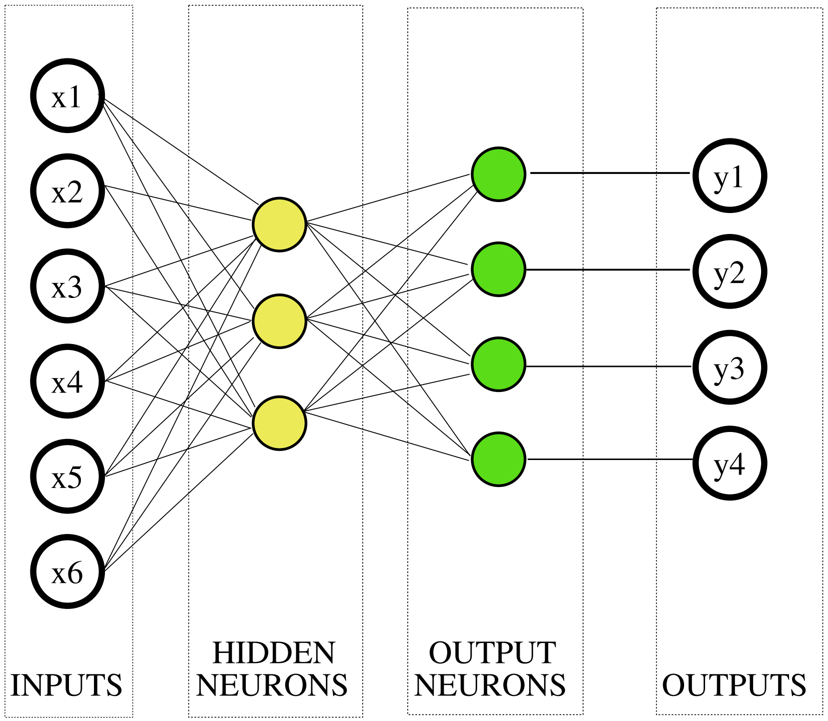
The Perceptron:

The most famous type of (artificial) neuron is the perceptron:

$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \ldots w_d x_d > 0, \\ -1 & \text{otherwise.} \end{cases}$$

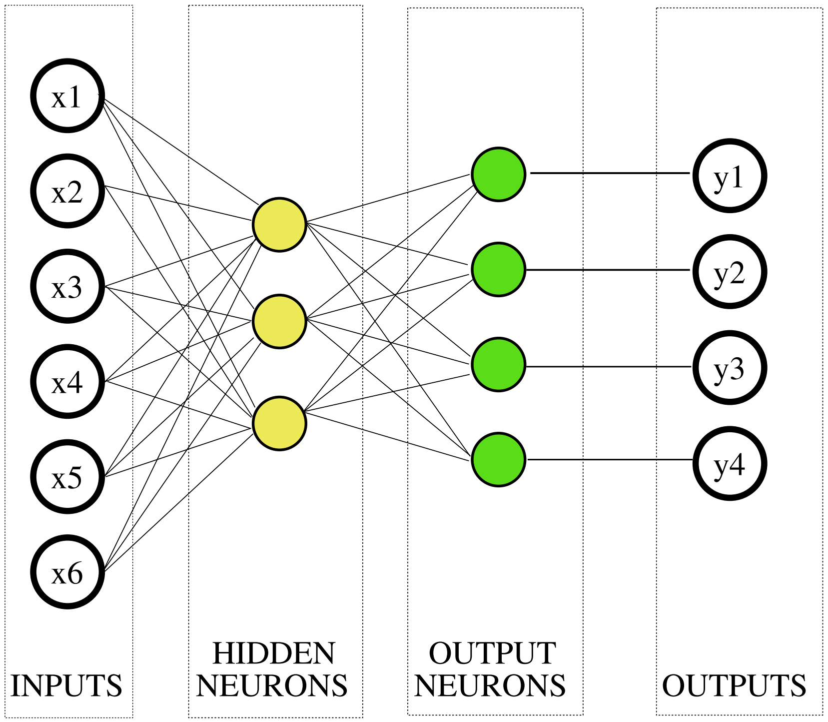
- Applies a threshold to a linear function of $\mathbf{x}$.
- Has parameters $\mathbf{w}$.

Artificial Neural Networks



- We can create an (artificial) **neural network** (NN) by connecting neurons together.
- We hook up our feature vector $\mathbf{x}$ to the input neurons in the network. We get a label vector $\mathbf{y}$ from the output neurons.

Artificial Neural Networks



- We can create an (artificial) **neural network** (NN) by connecting neurons together.
- We hook up our feature vector $\mathbf{x}$ to the input neurons in the network. We get a label vector $\mathbf{y}$ from the output neurons.
- The parameters of the neural network $\mathbf{w}$ consist of all the parameters of the neurons in the network taken together in one vector.

Why Study Neural Networks?

Modelling Biology:

- Some researchers want to study biological learning processes.
- They may try to model them using artificial neural networks.

Why Study Neural Networks?

Modelling Biology:

- Some researchers want to study biological learning processes.
- They may try to model them using artificial neural networks.
- This is not us!
- In machine learning we often use artificial neural networks that are poor models of biological neural networks.

Why Study Neural Networks?

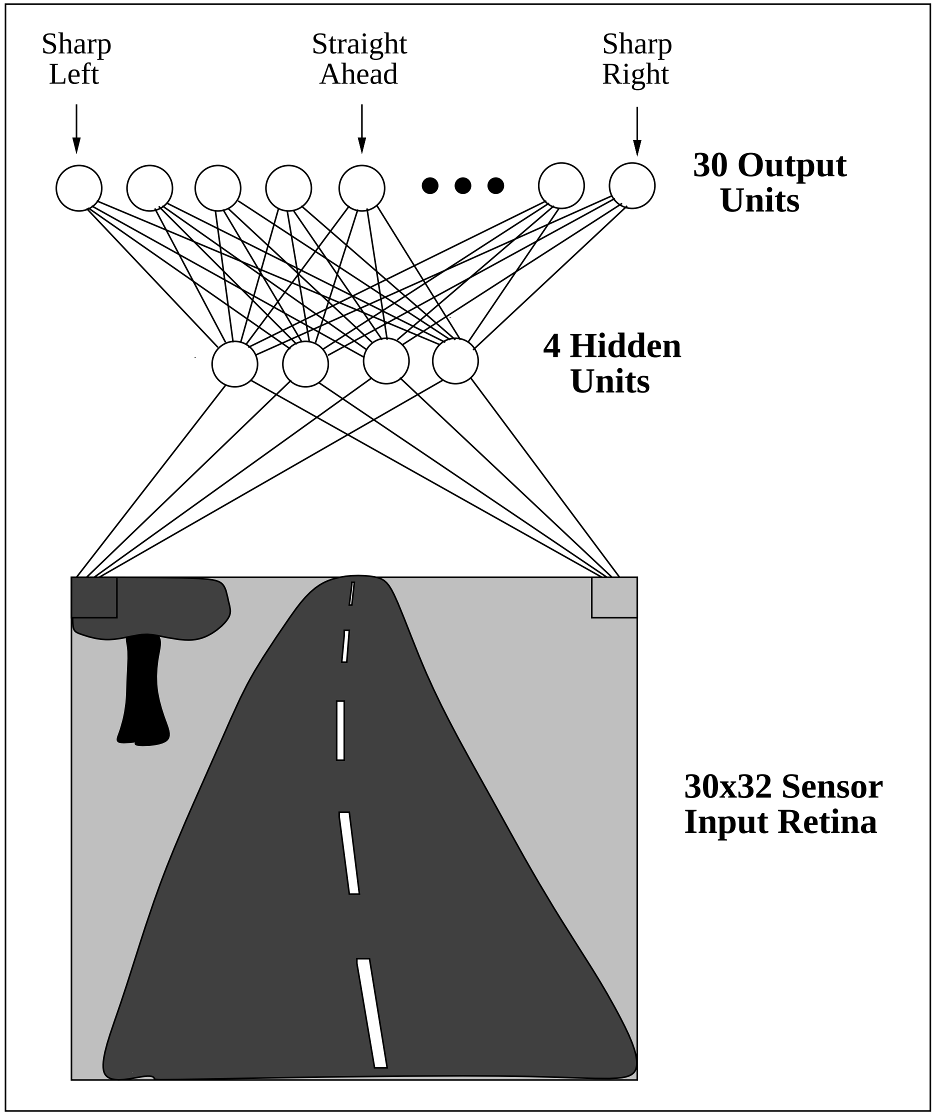
Modelling Biology:

- Some researchers want to study biological learning processes.
- They may try to model them using artificial neural networks.
- This is not us!
- In machine learning we often use artificial neural networks that are poor models of biological neural networks.

Obtaining Effective ML Algorithms:

- We want effective machine learning algorithms.
- An (artificial) neural network is a **hypothesis space** $\mathcal{H}$.
- Each setting of the parameters $\mathbf{w}$ corresponds to a different hypothesis $h_{\mathbf{w}} \in \mathcal{H}$.
- This hypothesis space may be used for **regression** or **classification**.

NN Example: ALVINN

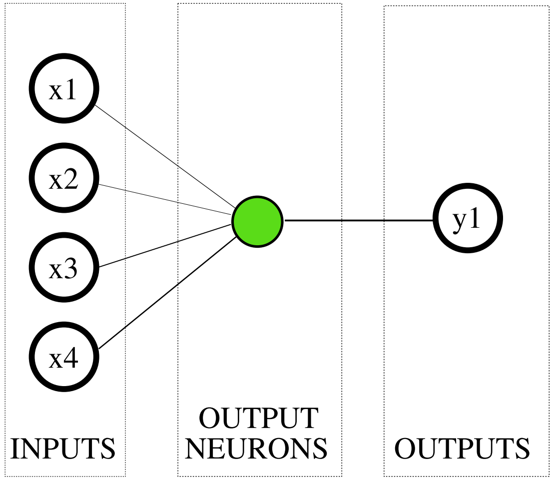


Overview

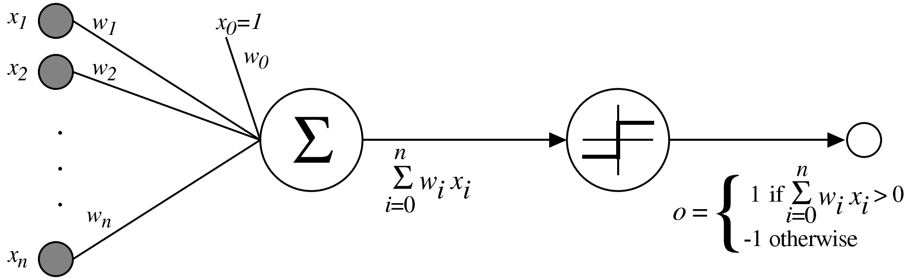
- Organisational Matters
- Answers Exercises 2
- Linear Functions as Inner Products
- Vector Valued Outputs in Regression and Classification
- Neural Networks and the Perceptron
  - Neural Networks
  - **The Perceptron**
  - Implementing Boolean Functions with a Perceptron
- Convex Functions
- Gradient Descent (part 1)

Different Views of The Perceptron

Simple Neural Network:



Mitchell's Drawing:

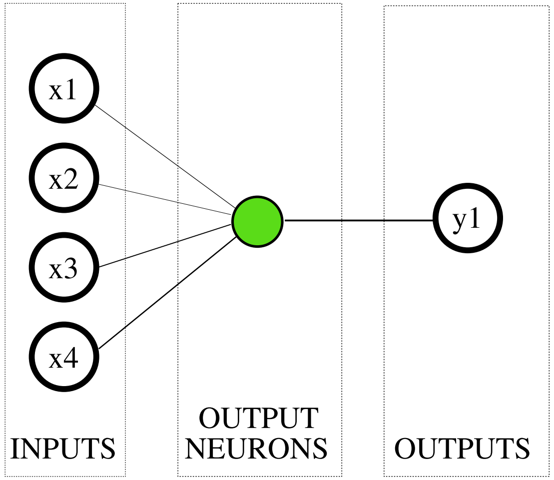


Equation:

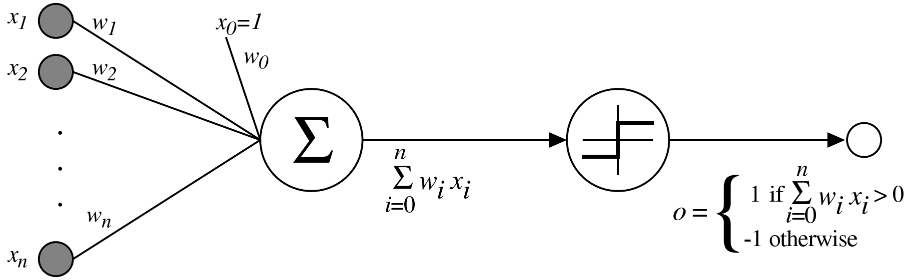
$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \ldots w_d x_d > 0, \\ -1 & \text{otherwise.} \end{cases}$$

Different Views of The Perceptron

Simple Neural Network:



Mitchell's Drawing:



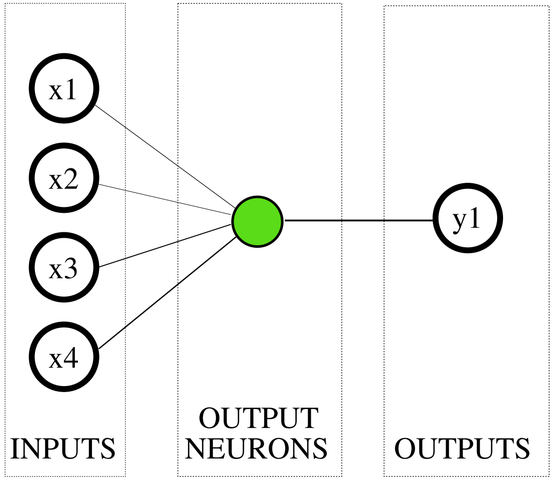
Equation:

$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \ldots w_d x_d > 0, \\ -1 & \text{otherwise.} \end{cases}$$

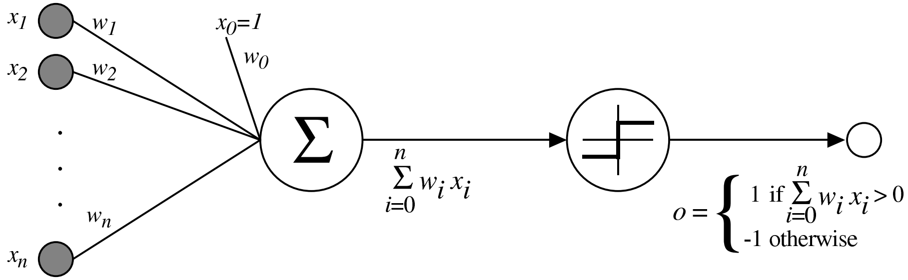
- One of the most simple neural networks consists of just one perceptron neuron.
- A perceptron does **classification**.

Different Views of The Perceptron

Simple Neural Network:



Mitchell's Drawing:



Equation:

$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \dots w_d x_d > 0, \\ -1 & \text{otherwise.} \end{cases}$$

- One of the most simple neural networks consists of just one perceptron neuron.
- A perceptron does **classification**.
- The network has no hidden units, and just one output.
- It may have any number of inputs.

Overview

- Organisational Matters
- Answers Exercises 2
- Linear Functions as Inner Products
- Vector Valued Outputs in Regression and Classification
- Neural Networks and the Perceptron
  - Neural Networks
  - The Perceptron
  - **Implementing Boolean Functions with a Perceptron**
- Convex Functions
- Gradient Descent (part 1)

Decision Boundary of the Perceptron

Decision boundary: $w_0 + w_1 x_1 + \ldots + w_d x_d = 0$

- This is where the perceptron changes its output y from -1 (-) to $+1$ (+) if we change $\mathbf{x}$ a little bit.
- Always a line.

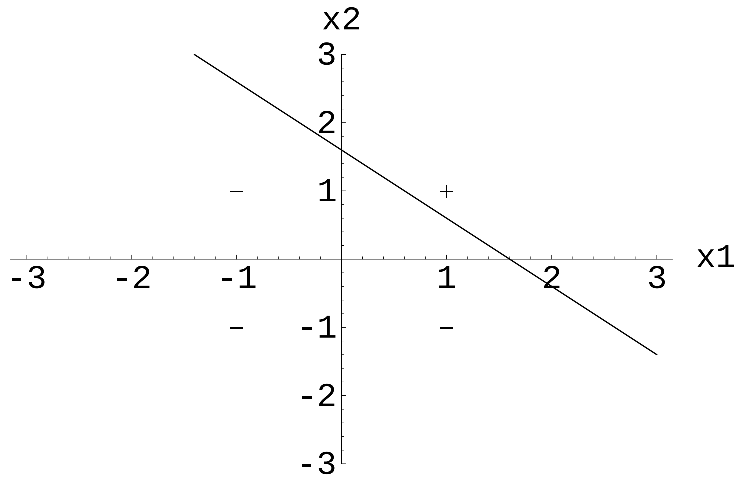
Decision Boundary of the Perceptron

Decision boundary: $w_0 + w_1 x_1 + \ldots + w_d x_d = 0$

- This is where the perceptron changes its output y from -1 (-) to $+1$ (+) if we change $\mathbf{x}$ a little bit.
- Always a line.

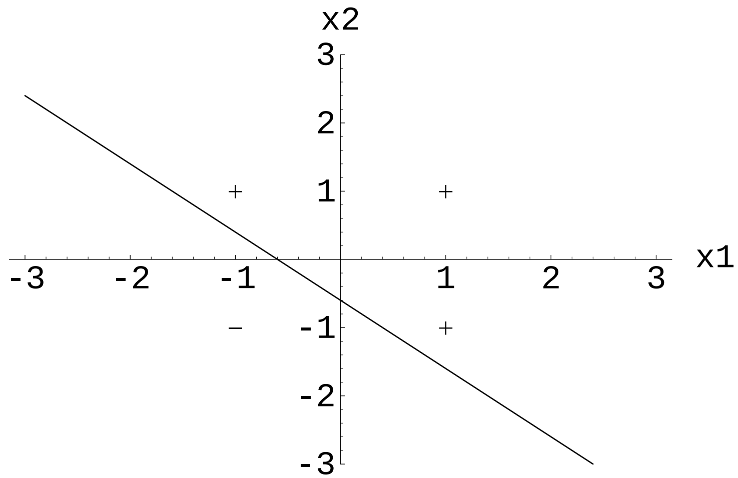
Examples of different Weights (with Boolean inputs: -1 = false, 1 = true):

AND



$w_0 = -0.8, w_1 = 0.5, w_2 = 0.5$

OR

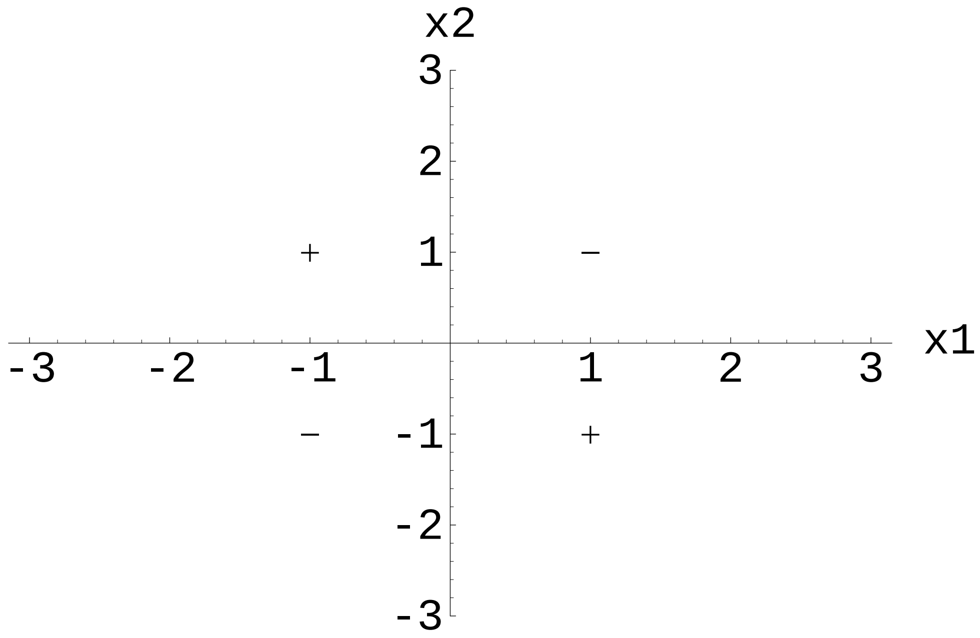


$w_0 = 0.3, w_1 = 0.5, w_2 = 0.5$

Wrong in Mitchell!

Perceptron Cannot Represent Exclusive Or

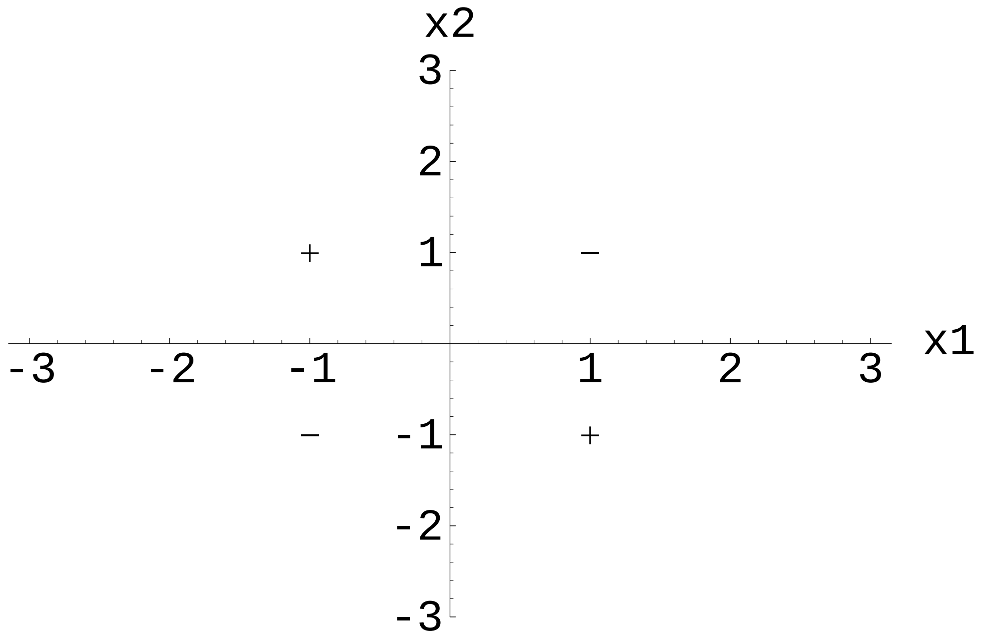
Exclusive Or:



- There exists no line that separates the inputs with label '-' from the inputs with label '+'. They are not **linearly separable**.

Perceptron Cannot Represent Exclusive Or

Exclusive Or:



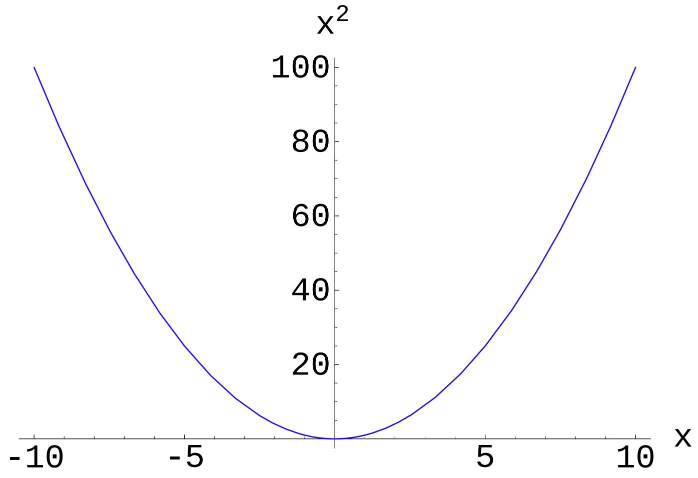
- There exists no line that separates the inputs with label '-' from the inputs with label '+'. They are not **linearly separable**.
- The decision boundary for the perceptron is always a line.
- Hence a perceptron can never implement the 'exclusive or' function, whichever weights we choose.

Overview

- Organisational Matters
- Answers Exercises 2
- Linear Functions as Inner Products
- Vector Valued Outputs in Regression and Classification
- Neural Networks and the Perceptron
  - Neural Networks
  - The Perceptron
  - Implementing Boolean Functions with a Perceptron
- **Convex Functions**
- Gradient Descent (part 1)

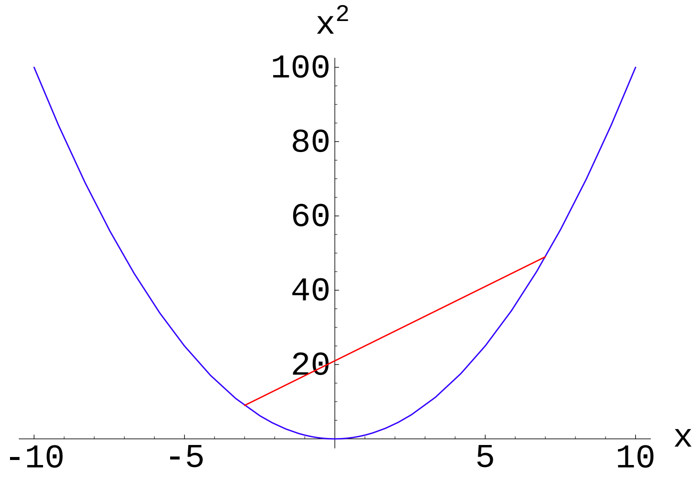
Convex Functions

Intuition:



Convex Functions

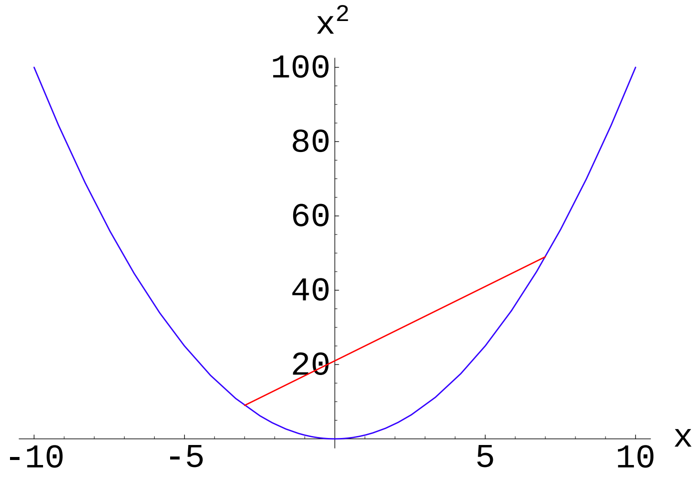
Intuition:



- A function is convex if it lies below the line between any two of its points. For example, $f(-3)$ and $f(7)$.

Convex Functions

Intuition:



- A function is convex if it lies below the line between any two of its points. For example, $f(-3)$ and $f(7)$.

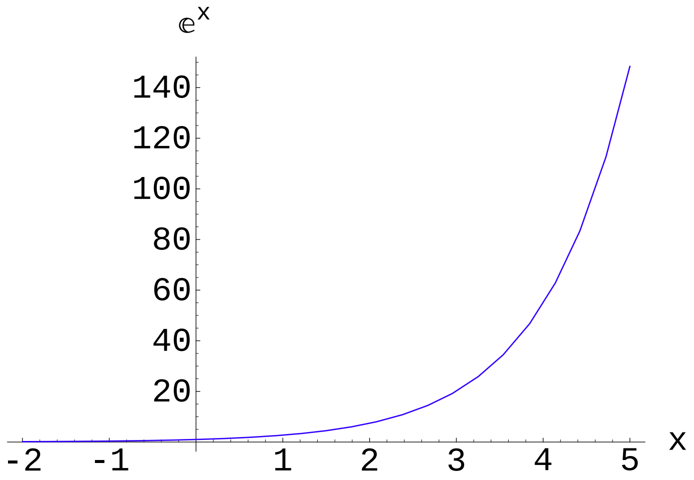
Definition: A function $f(x)$ is **convex** if

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2)$$

for any two inputs x_1, x_2 and any $0 \leq \alpha \leq 1$.

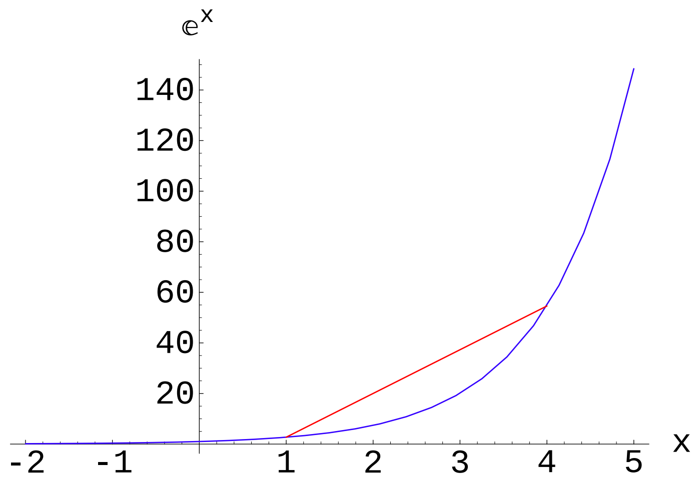
Examples

Convex:



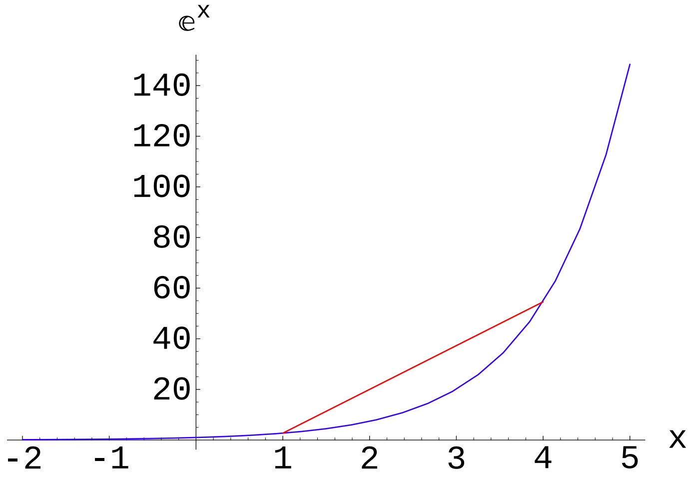
Examples

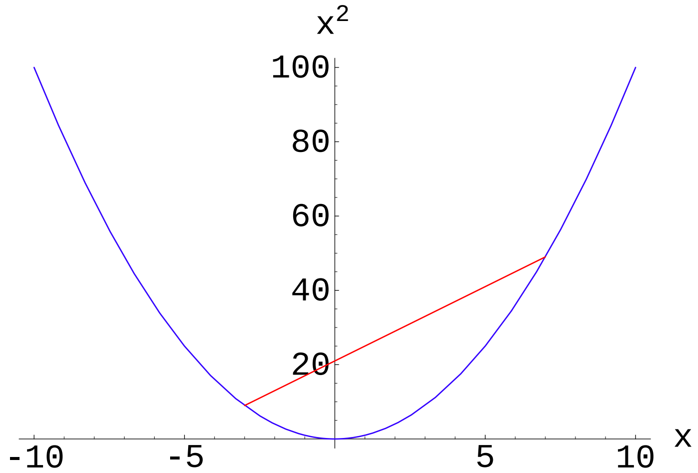
Convex:



Examples

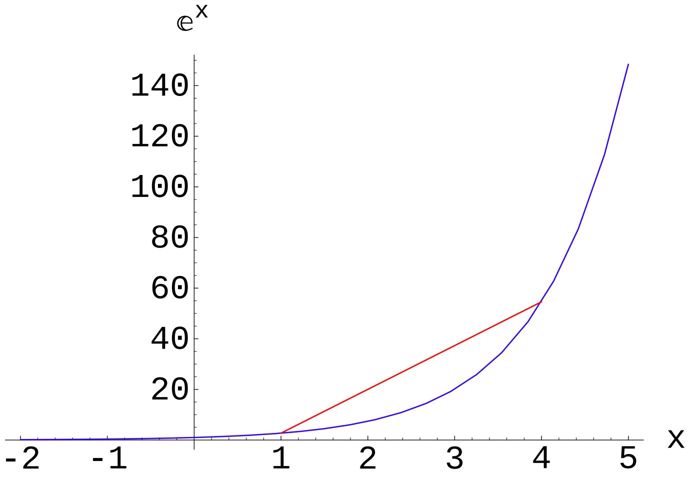
Convex:

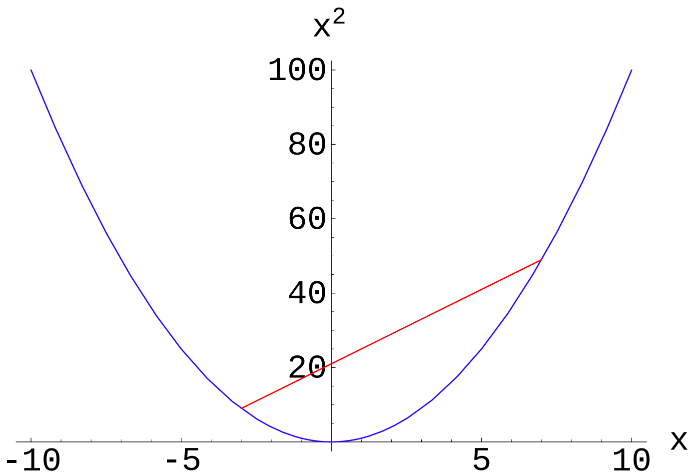




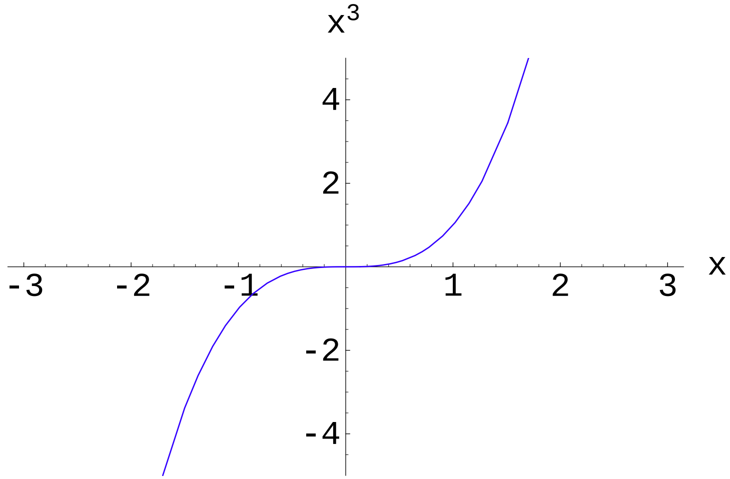
Examples

Convex:



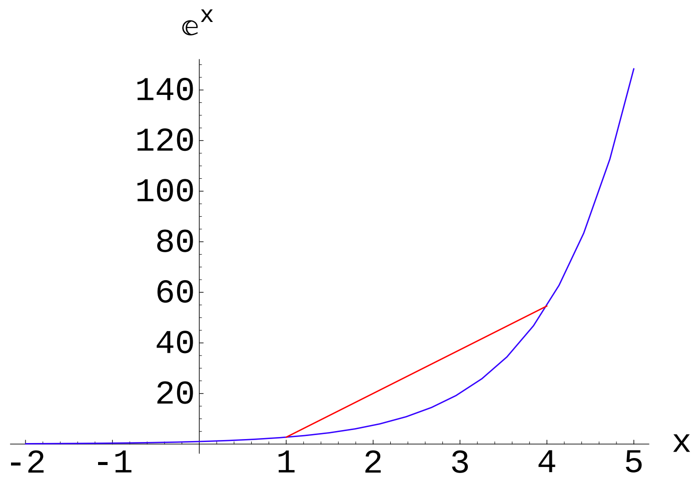


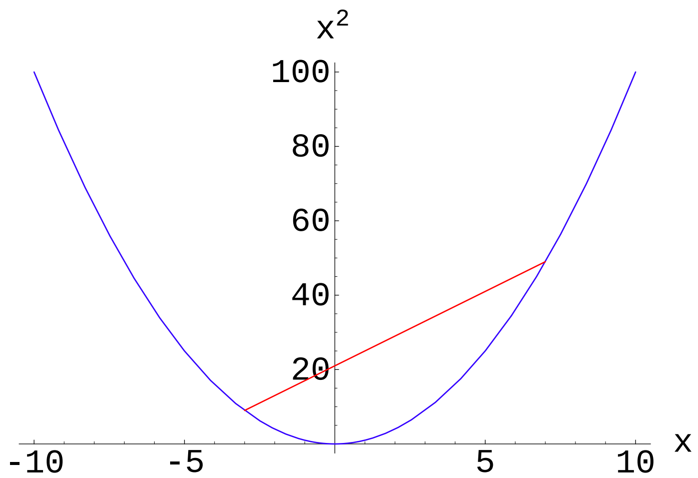
Not Convex:



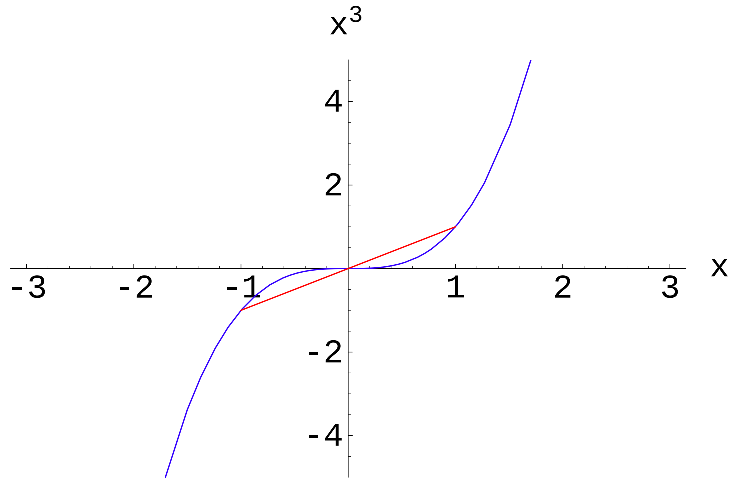
Examples

Convex:



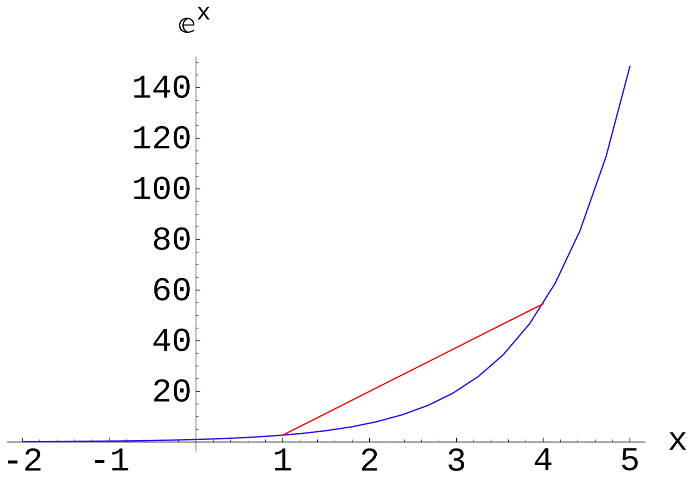


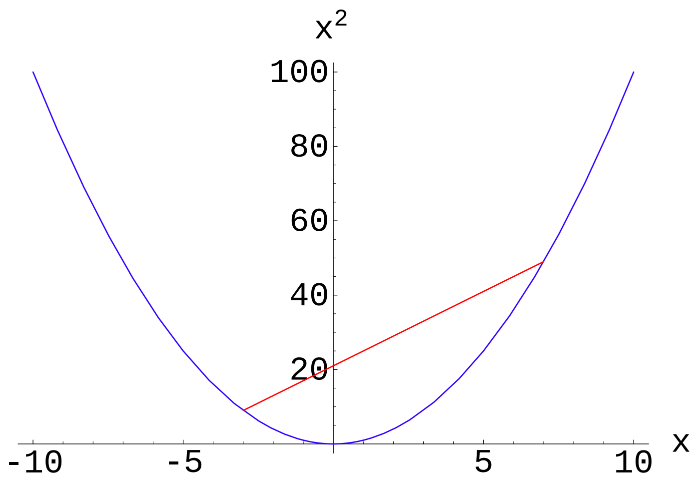
Not Convex:



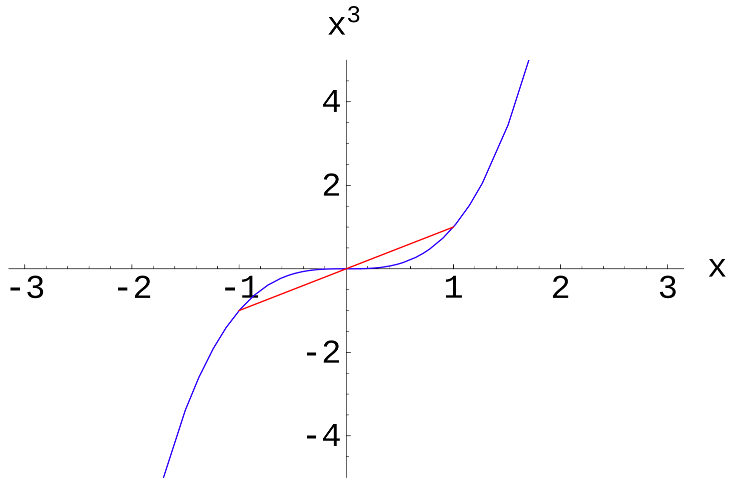
Examples

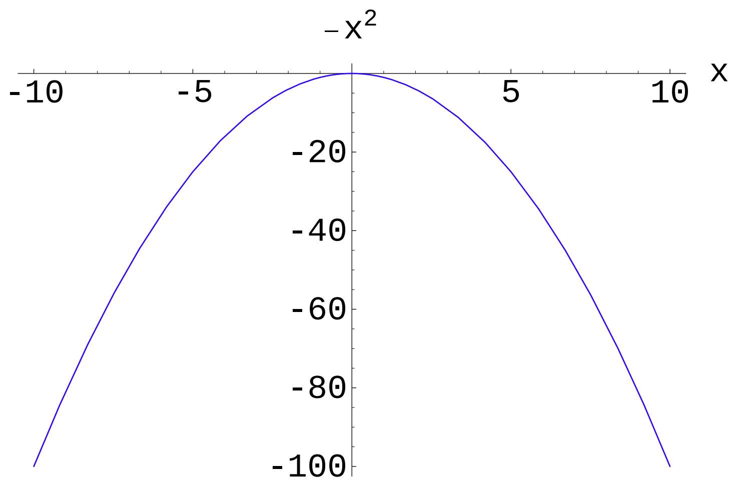
Convex:





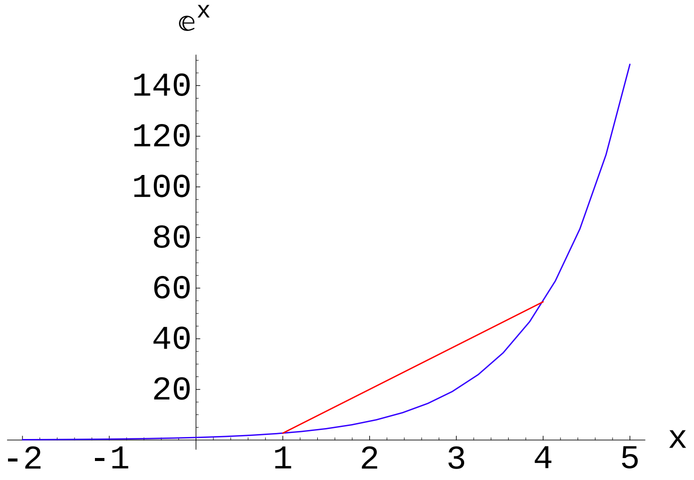
Not Convex:

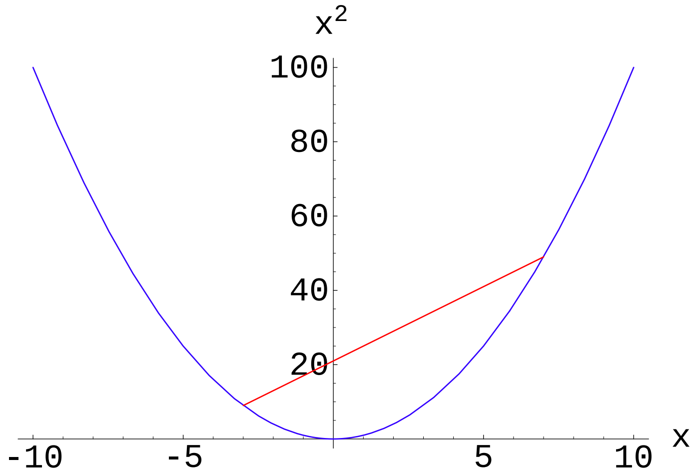




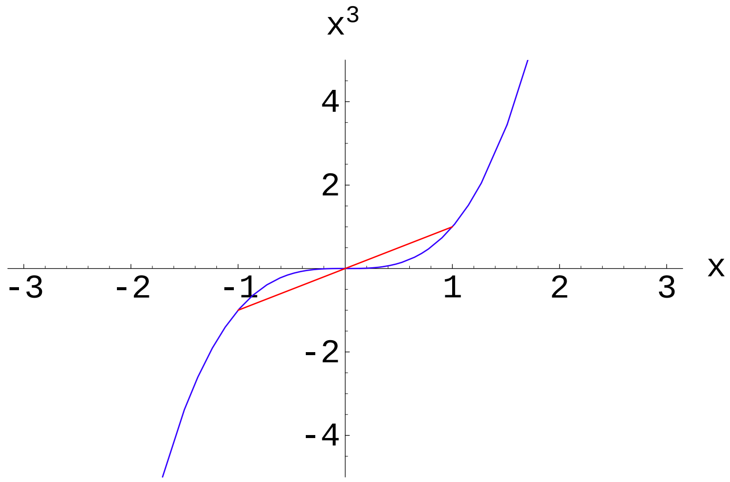
Examples

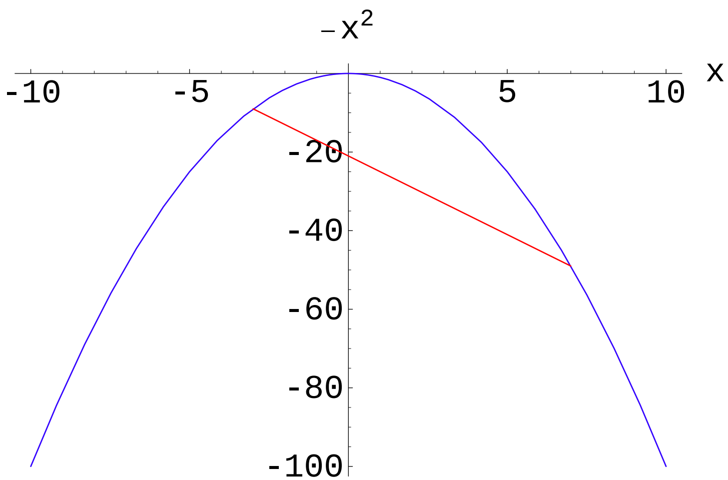
Convex:





Not Convex:





Overview

- Organisational Matters
- Answers Exercises 2
- Linear Functions as Inner Products
- Vector Valued Outputs in Regression and Classification
- Neural Networks and the Perceptron
  - Neural Networks
  - The Perceptron
  - Implementing Boolean Functions with a Perceptron
- Convex Functions
- **Gradient Descent (part 1)**

Gradient Descent

- Gradient descent is a method to find the minimum $\min_x f(x)$ of a function.
- It works for convex functions.
- But not for some other functions.

Gradient Descent

- Gradient descent is a method to find the minimum $\min_x f(x)$ of a function.
- It works for convex functions.
- But not for some other functions.

General Idea:

1. Pick a random starting point x_1.
2. Do a little step in the direction of the derivative: $f'(x_1)$.
3. Now we are at x_2.
4. Do a little step in the direction of the derivative: $f'(x_2)$.
5. Keep doing little steps until $f'(x_m) \approx 0$: we have reached the minimum.

Gradient Descent

- Gradient descent is a method to find the minimum $\min_x f(x)$ of a function.
- It works for convex functions.
- But not for some other functions.

General Idea:

1. Pick a random starting point x_1.
2. Do a little step in the direction of the derivative: $f'(x_1)$.
3. Now we are at x_2.
4. Do a little step in the direction of the derivative: $f'(x_2)$.
5. Keep doing little steps until $f'(x_m) \approx 0$: we have reached the minimum.

To be continued next lecture. . .

Overview

- Organisational Matters
- Answers Exercises 2
- Linear Functions as Inner Products
- Vector Valued Outputs in Regression and Classification
- Neural Networks and the Perceptron
  - Neural Networks
  - The Perceptron
  - Implementing Boolean Functions with a Perceptron
- Convex Functions
- Gradient Descent (part 1)

References

- Picture of a neuron taken from Wikimedia Commons, http://commons.wikimedia.org/wiki/Image:Neuron.svg: Originally Neuron.jpg taken from the US Federal (public domain) (Nerve Tissue, retrieved March 2007), redrawn by User:Dhp1080 in Illustrator. Source: ”Anatomy and Physiology” by the US National Cancer Institute’s Surveillance, Epidemiology and End Results (SEER) Program.
- S. Boyd and L. Vandenberghe. Convex Optimization. Cambridge University Press, 2004
- T.M. Mitchell, “Machine Learning”, McGraw-Hill, 1997