

# CENG 506 Deep Learning

## Lecture 1 Introduction & Linear Regression

# Objectives

Gain an understanding of how neural networks operate

What type of neural networks are implemented for different learning tasks (especially in computer vision)

Learn to use python libraries to train/build your own neural networks

# Course Outline

Introduction

Linear Classification

Backpropagation

Neural Network Training

Convolutional Neural Networks

ConvNets for Object Detection and Semantic Segmentation

Understanding and Visualizing Convolutional Neural Networks

Adversarial Networks (GANs)

Recurrent Neural Networks

Transformers

# Resources

Our course content is a mixture of some Stanford's courses

CS230 Deep Learning: <https://cs230.stanford.edu/syllabus/>

CS231n Deep Learning for Computer Vision:

<https://cs231n.stanford.edu/>

[link](#) to 2017 videos of CS231n

Among many other resources,

<http://neuralnetworksanddeeplearning.com/>

by Michael Nielsen

<http://www.deeplearningbook.org/>

by Ian Goodfellow and Yoshua Bengio and Aaron Courville

(in Turkish <http://www.buzdagiyayinevi.com/derin-ogrenme/>)

# Prerequisites

## Familiarity with Python programming

Programming examples and assignments will be in Python. If a term project assigned, again python will be the preferred language.

## Calculus and some Linear Algebra

You should be comfortable taking derivatives, understanding matrix vector operations and using related notation.

## Basic Probability and Statistics

You should know basics of probabilities, mean, standard deviation, etc.

## Machine Learning (Equivalence of CENG463 or Stanford's CS229)

Familiarity with the ideas of supervised learning, logistic regression, gradient descent. We will be formulating cost functions, taking derivatives and performing optimization with gradient descent.

# Course Material

Lecture slides, assignments and grades will be posted on MS-Teams.

## Grading

Midterm: 30-35%

Final: 40-45%

Homeworks/project: 20-30%

- Group work is not allowed in homeworks!

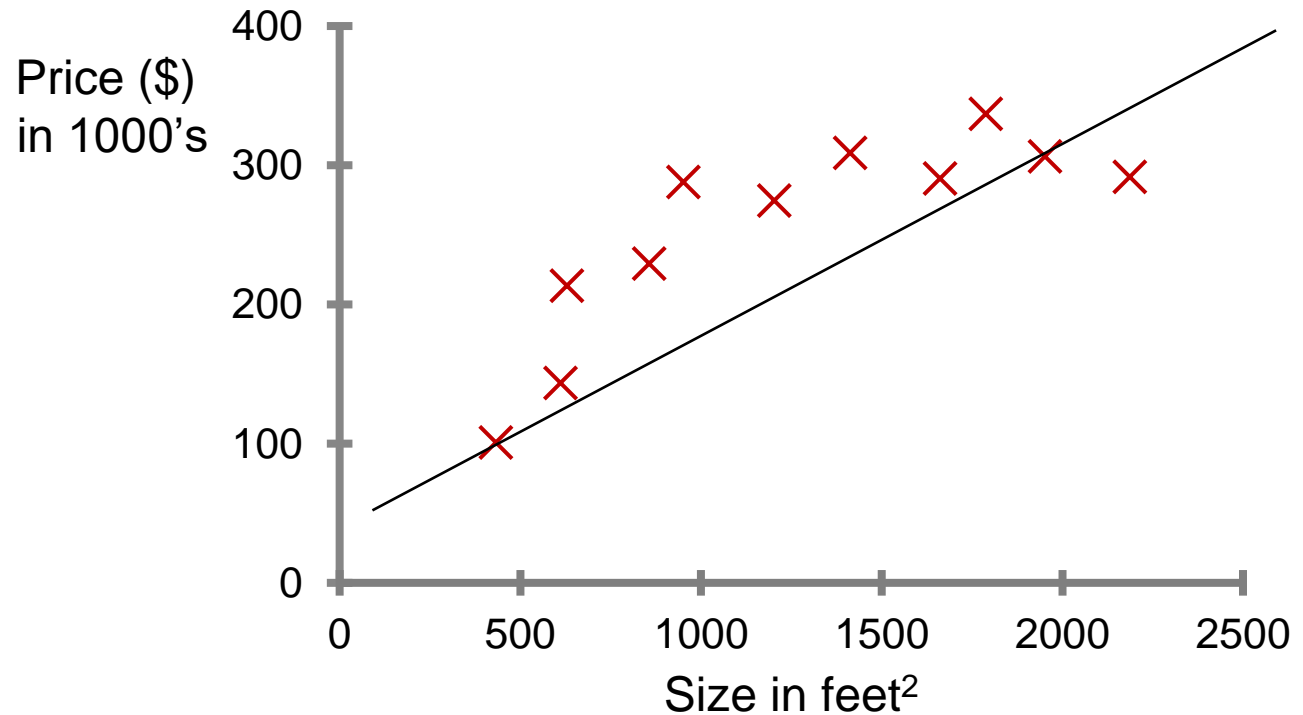
Homeworks will be graded by our T.A.: Ersin Çine

# What is Linear Regression?

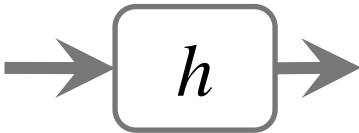
It is a way to fit a mathematical model to a given input, so that we can make estimations for new data.

Example:

House prices according to house size



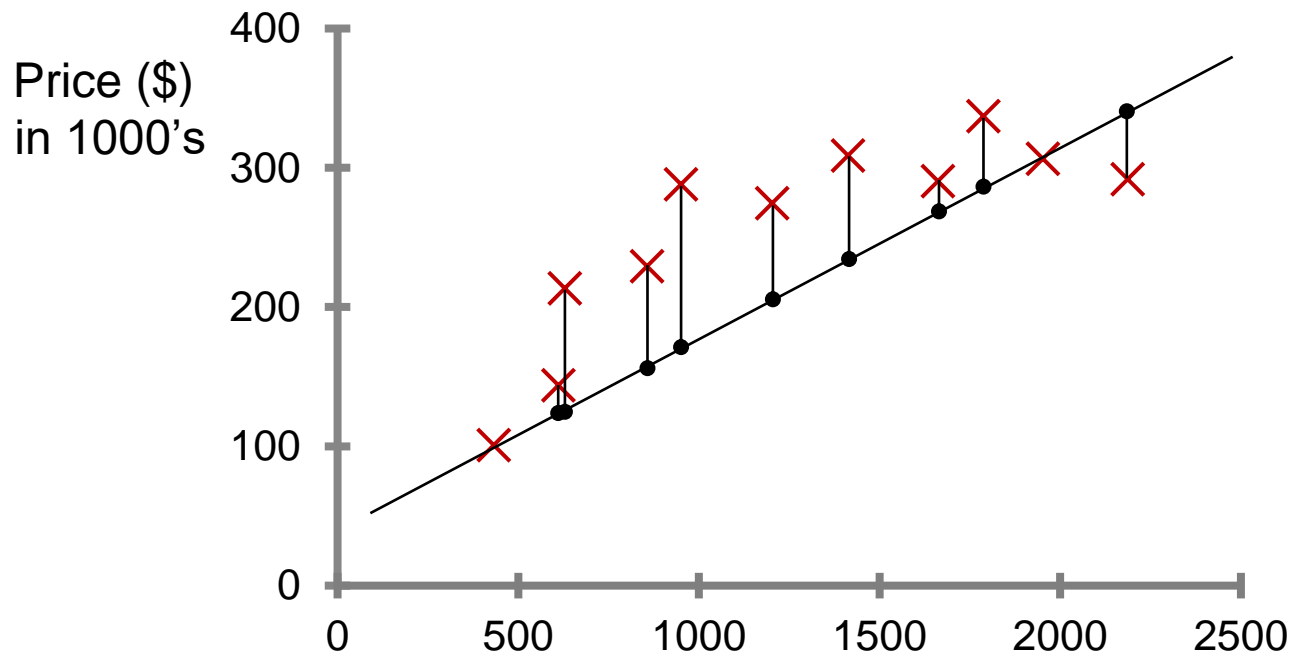
Size of house



Estimated price

$$h_w(x) = w_0 + w_1x$$

# What is a Cost Function?



$y$ : real prices  
(red crosses)

Our hypothesis:  
 $h(x) = w_0 + w_1 x$

First sample cost:  
 $(y^{(1)} - h(x^{(1)}))^2$

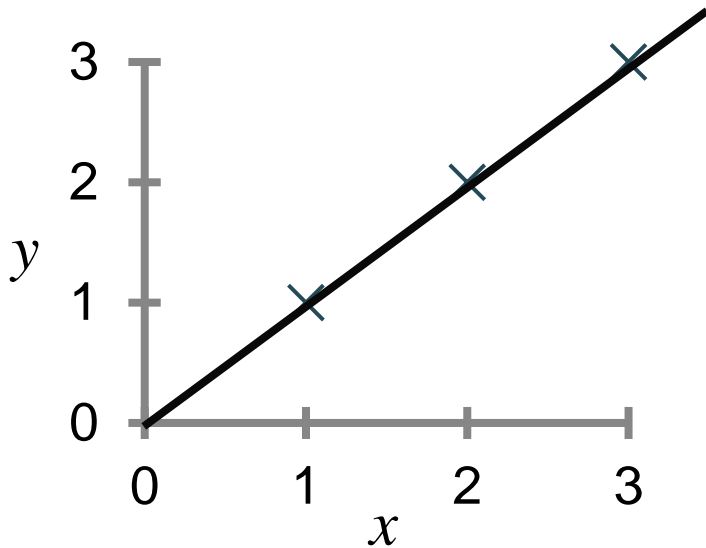
Total cost:  
 $\sum (y^{(i)} - h(x^{(i)}))^2$



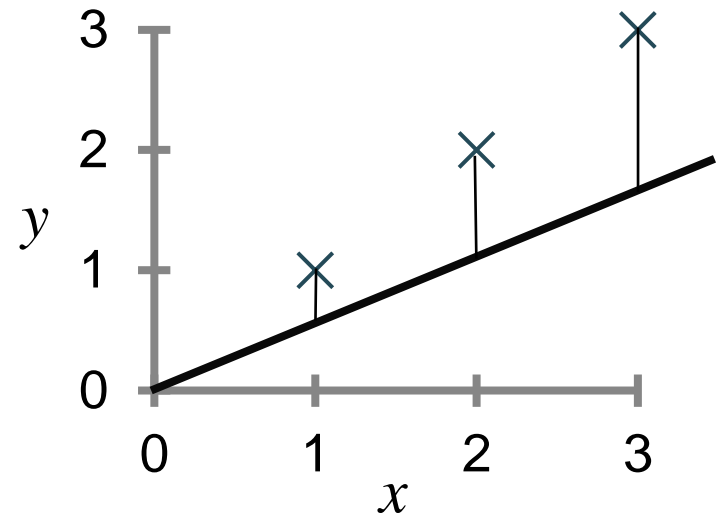
# Cost Function: Intuition

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^m (h_w(x^{(i)}) - y^{(i)})^2$$

What is  $J(0,1)$ ?



What is  $J(0,0.5)$ ?



# Gradient Descent

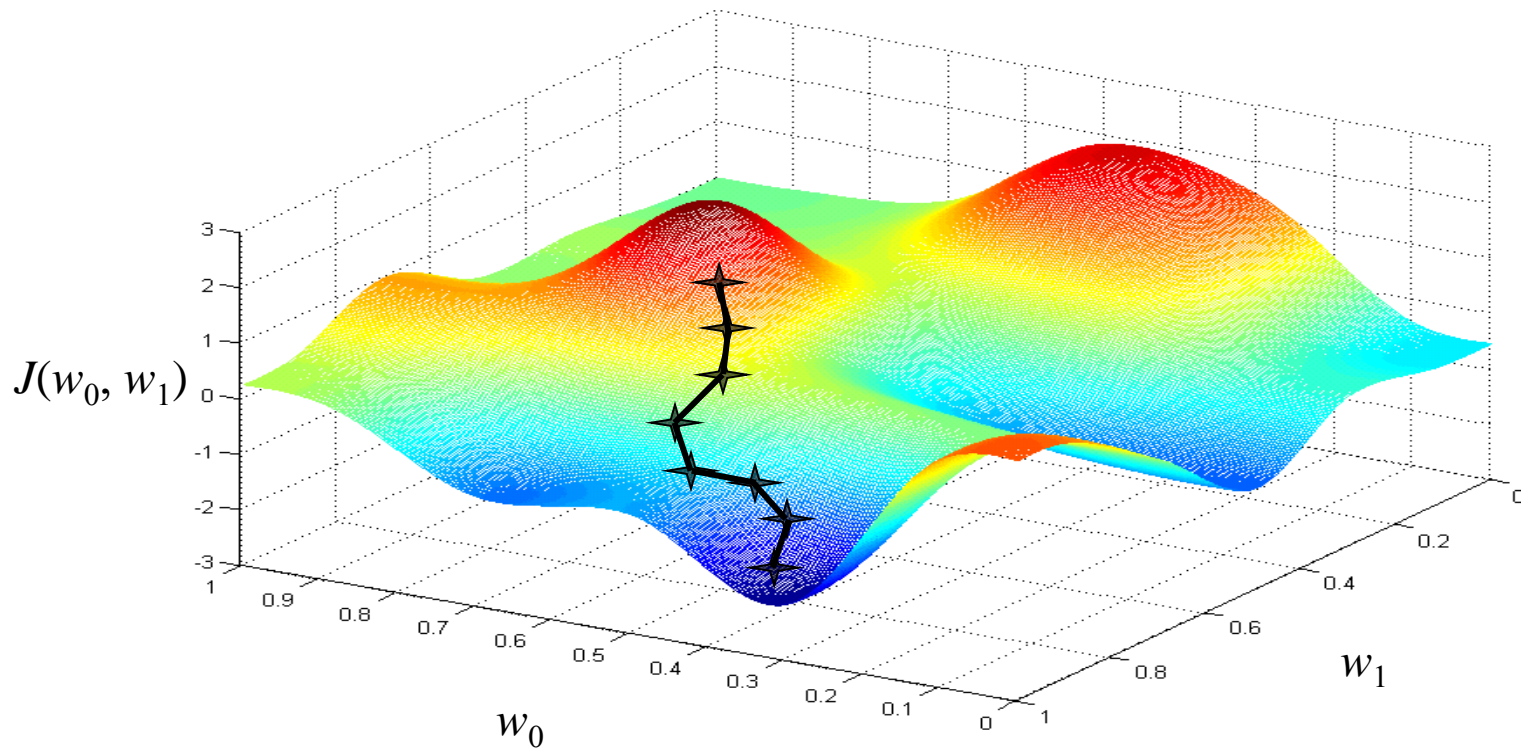
We have cost function  $J(w_0, w_1)$   
that we want to minimize by changing  $w_0, w_1$

## Outline:

- Start with some  $w_0, w_1$
- Keep changing  $w_0, w_1$  to reduce  $J(w_0, w_1)$   
until we hopefully end up at a minimum

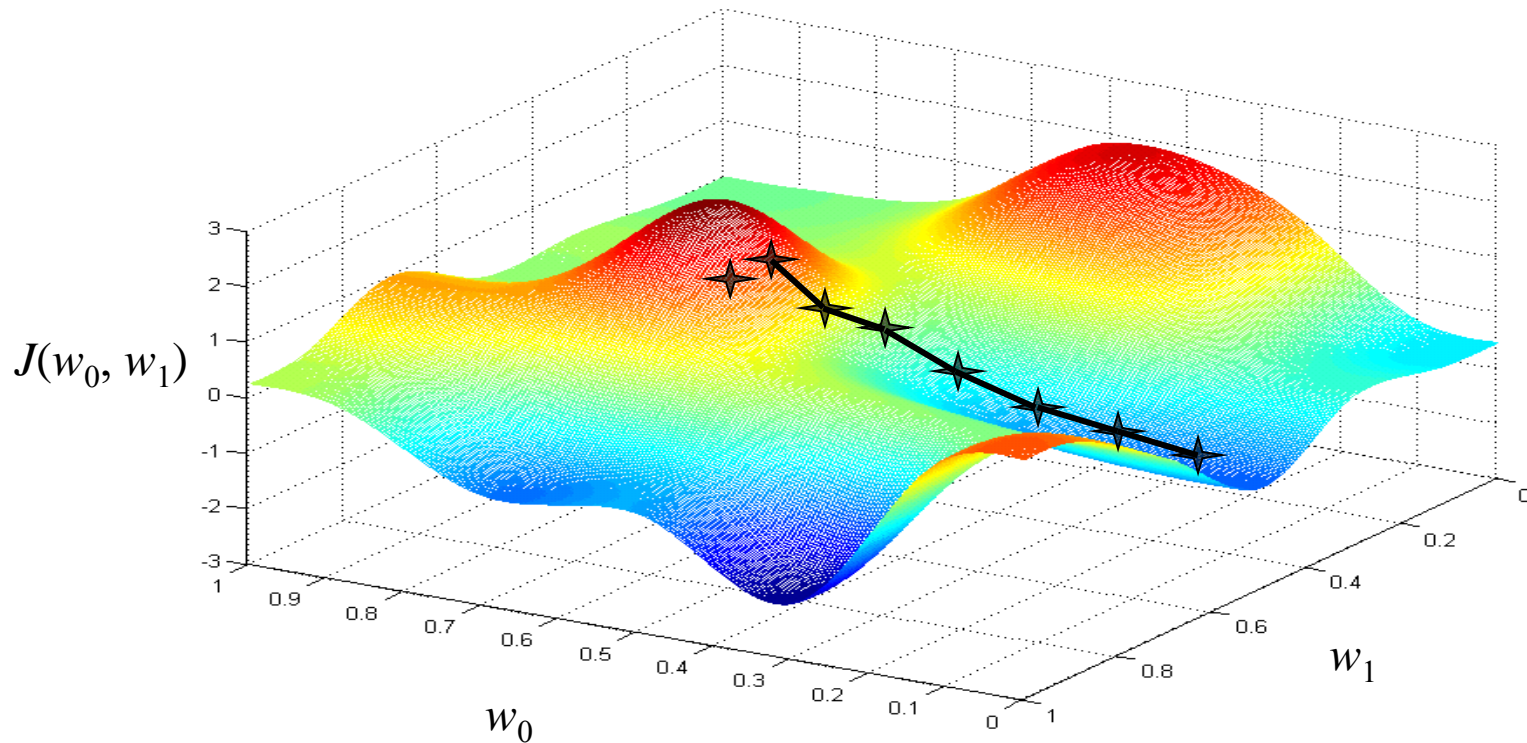
# Gradient Descent

Gradient descent takes steps in the negative of the gradient direction.



# Gradient Descent

Gradient descent does not guarantee to reach the global minimum.



# Gradient Descent Algorithm

Gradient descent algorithm for two-parameter case:

Repeat until convergence {

$$w_j := w_j - \alpha \frac{\partial J(w_0, w_1)}{\partial w_j} \quad (\text{for } j = 0 \text{ and } j = 1)$$

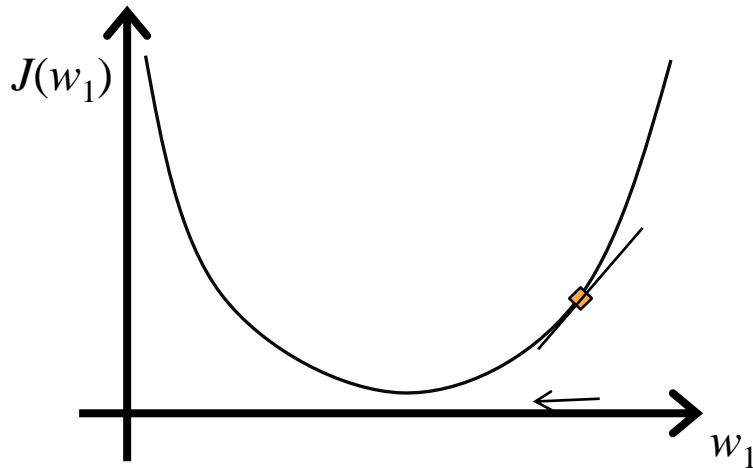
}

This is the derivative term, indicates the direction of step.

This number is 'learning rate', controls the size of the step we take.

# Gradient Descent Intuition

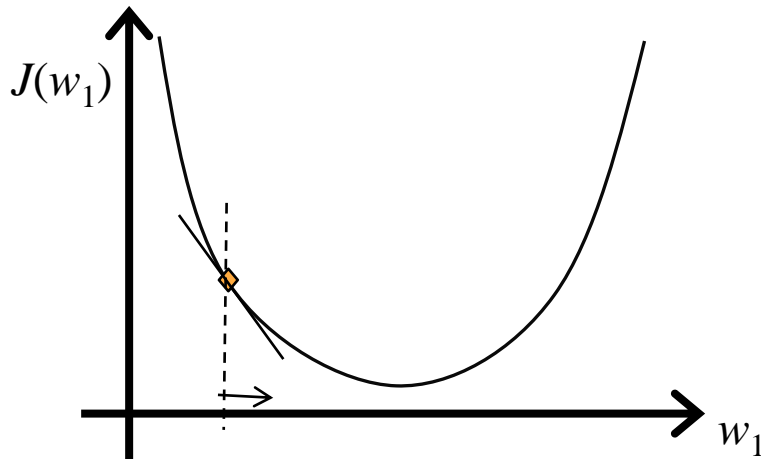
Let's assume a cost function with one variable ( $w_1$ ):



$$w_1 := w_1 - \alpha \frac{\partial J(w_1)}{\partial w_1}$$

An arrow points from the text below to the derivative term in the equation.

The derivative gives the slope at that point. Since we subtract it, positive slope decreases the value of  $w_1$ .



Negative slope increases  $w_1$ .

# Gradient Descent for Lin. Reg.

Let's merge two things we have learned:

Gradient descent algorithm

Repeat until convergence {

$$w_j := w_j - \alpha \frac{\partial J(w_0, w_1)}{\partial w_j}$$

}

Linear Regression Cost Func.

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^m (h_w(x^{(i)}) - y^{(i)})^2$$

# Gradient Descent for Lin. Reg.

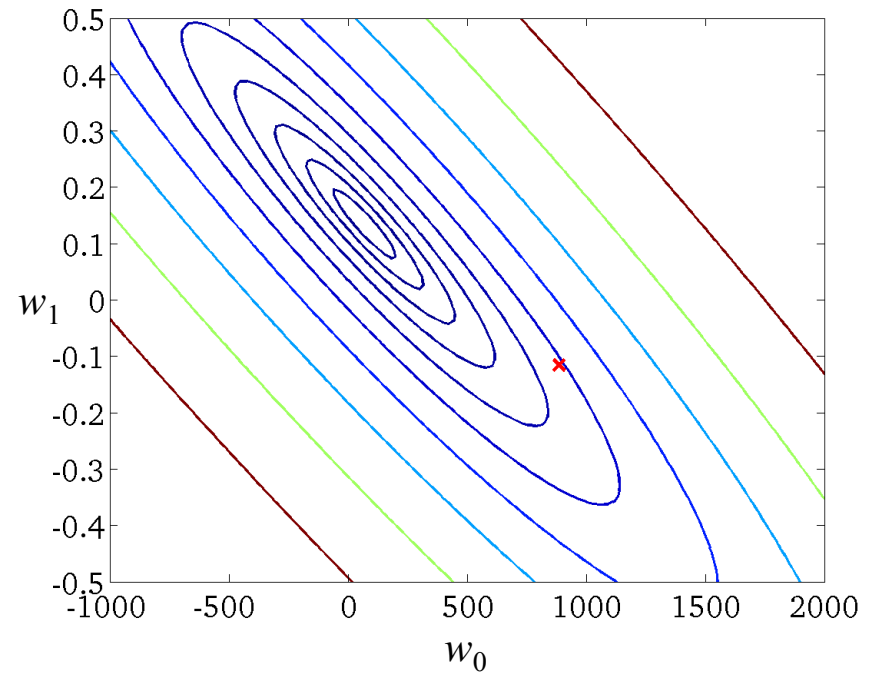
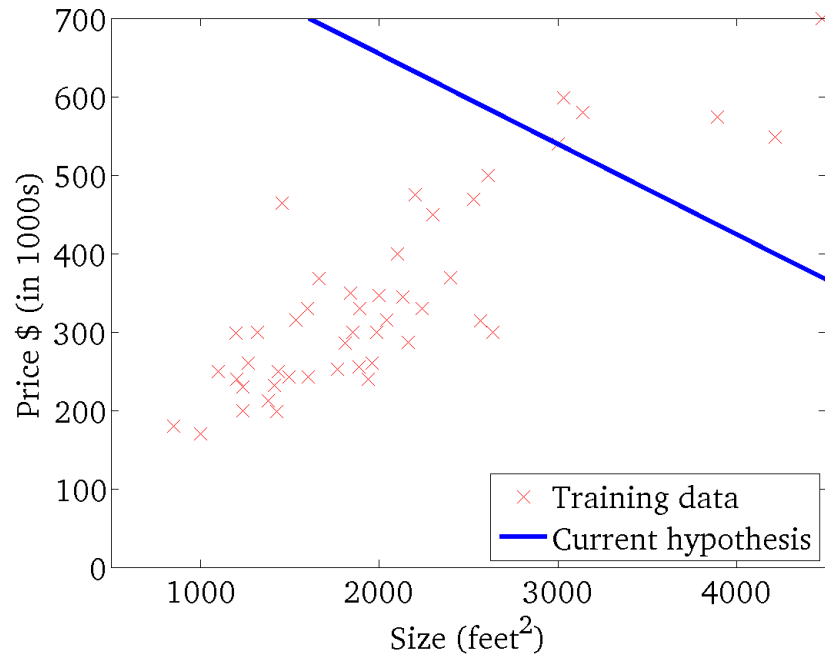
$$\begin{aligned}\frac{\partial J(w_0, w_1)}{\partial w_j} &= \partial \frac{\frac{1}{2m} \sum_{i=1}^m (h_w(x^{(i)}) - y^{(i)})^2}{\partial w_j} \\ &= \partial \frac{\frac{1}{2m} \sum_{i=1}^m (w_0 + w_1 x^{(i)} - y^{(i)})^2}{\partial w_j}\end{aligned}$$

$$\frac{\partial J(w_0, w_1)}{\partial w_0} = \frac{1}{m} \sum_{i=1}^m (h_w(x^{(i)}) - y^{(i)})$$

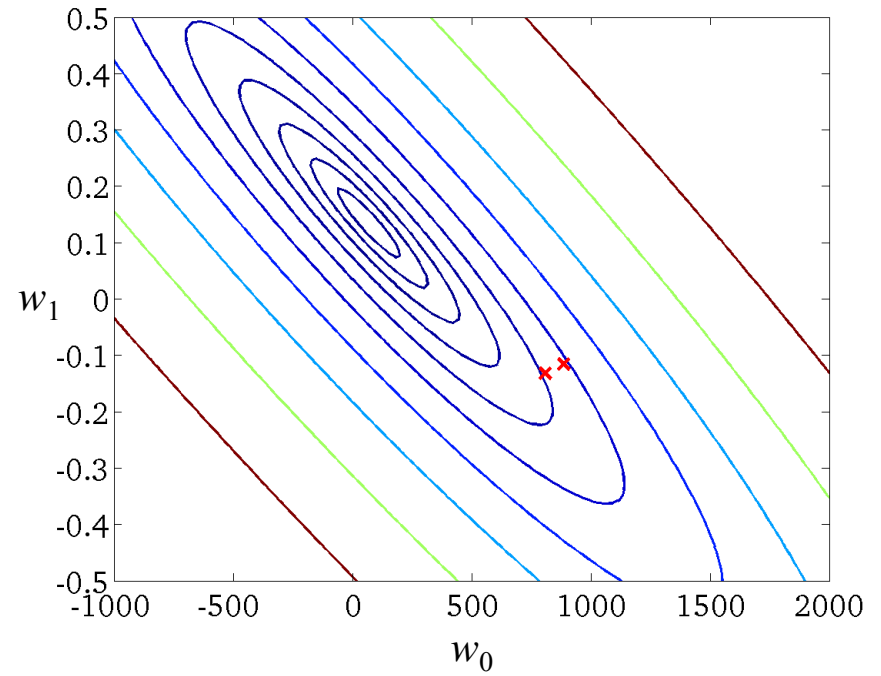
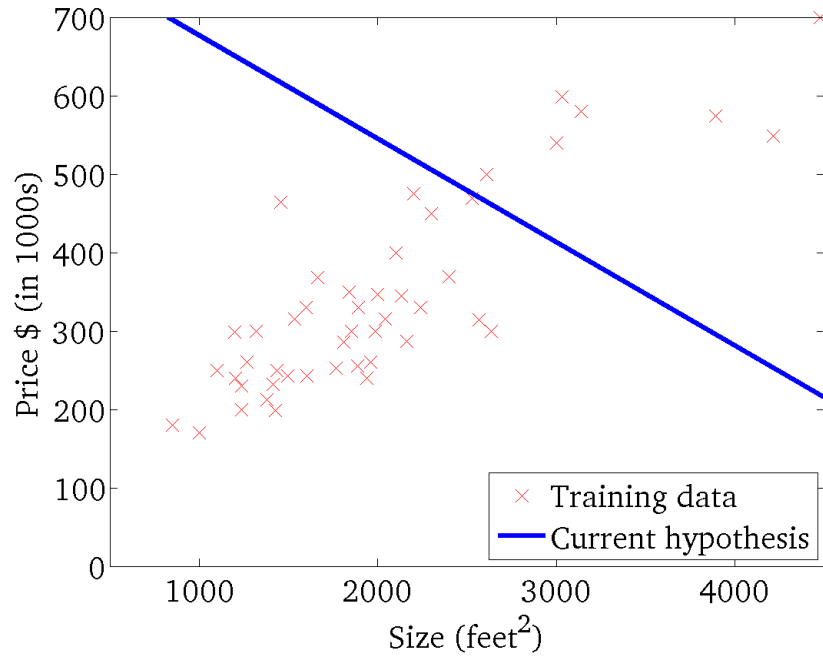
$$\frac{\partial J(w_0, w_1)}{\partial w_1} = \frac{1}{m} \sum_{i=1}^m (h_w(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$



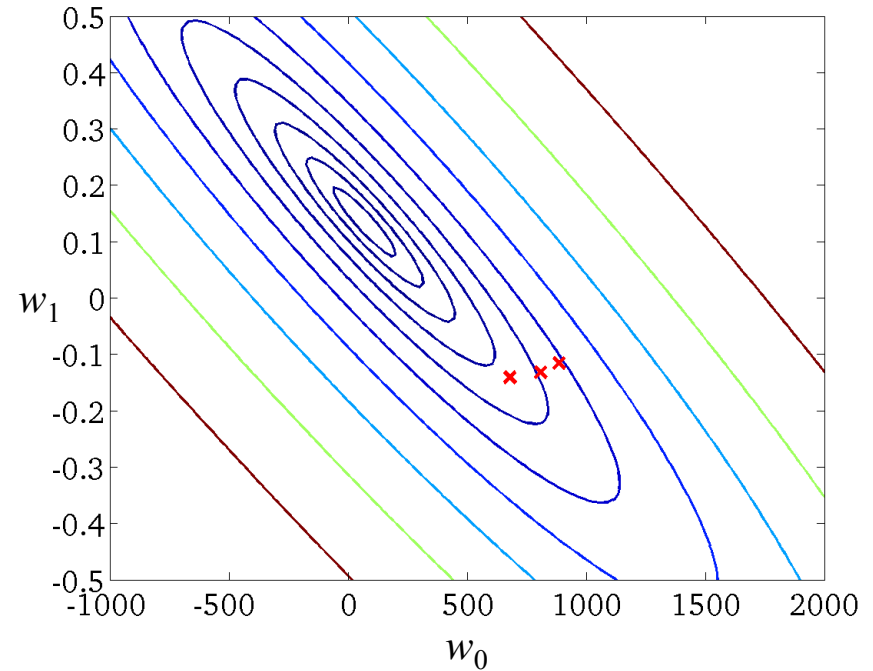
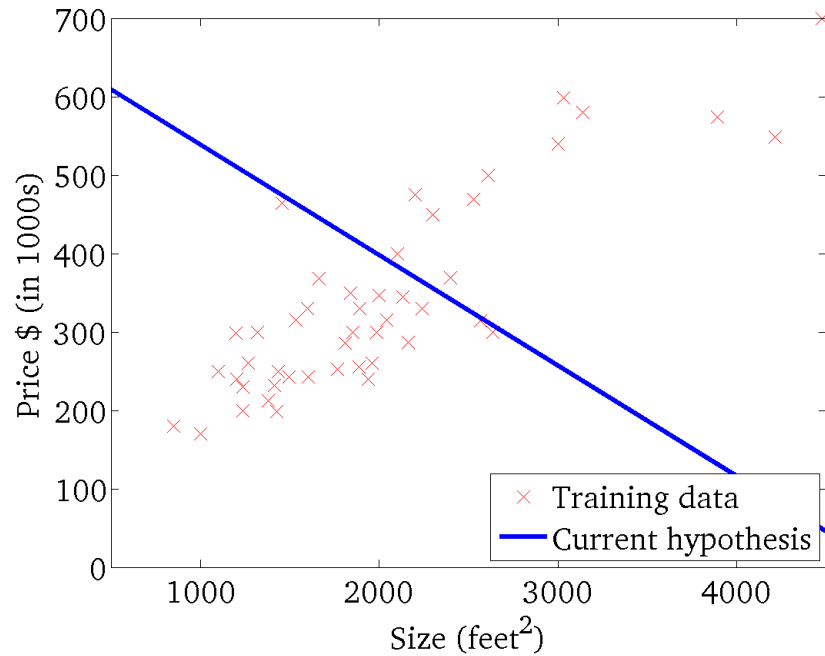
# Gradient Descent in Action



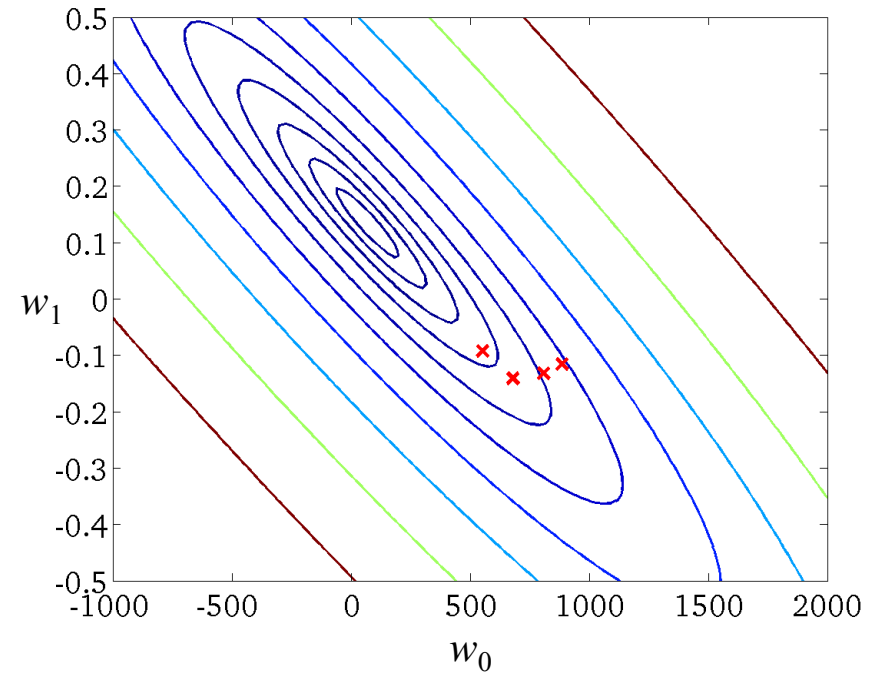
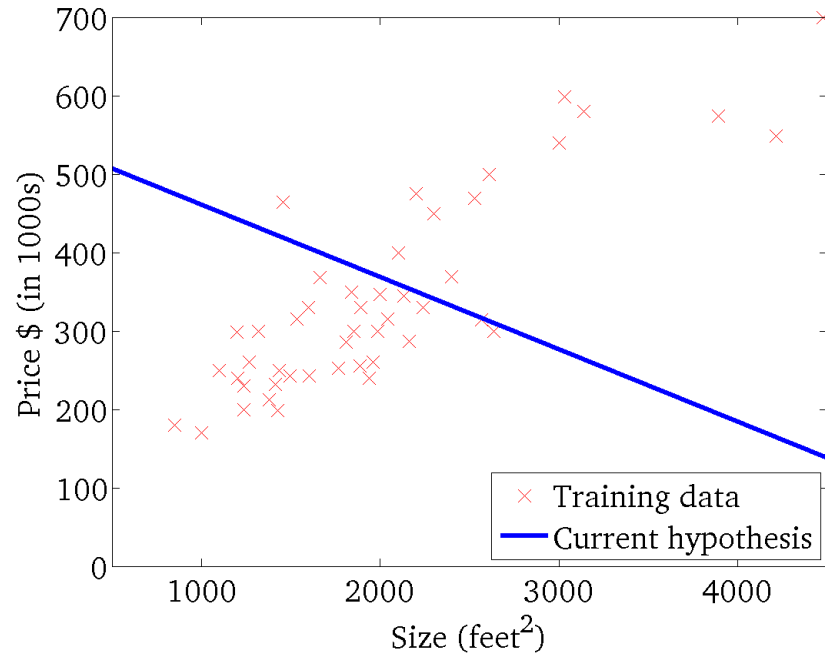
# Gradient Descent in Action



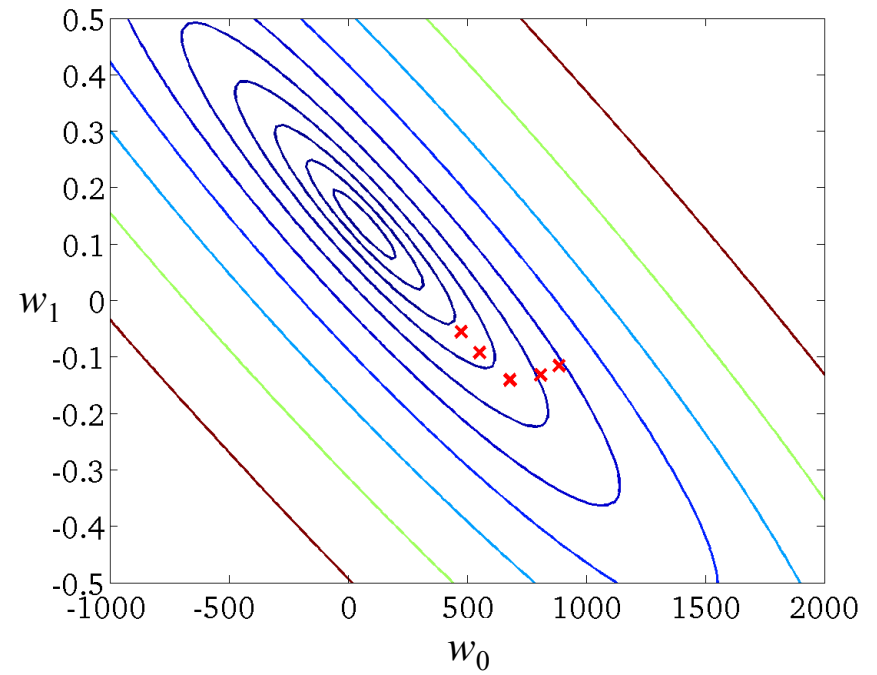
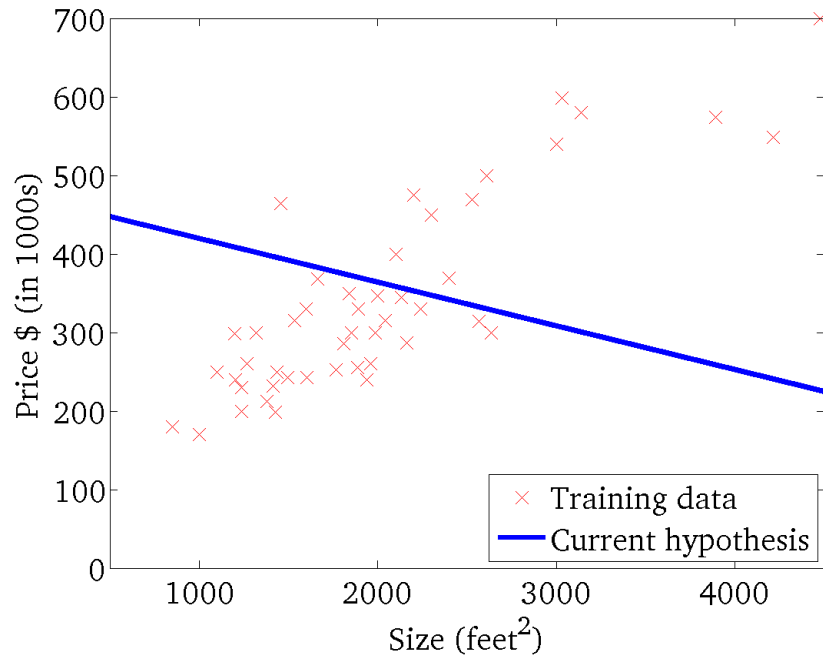
# Gradient Descent in Action



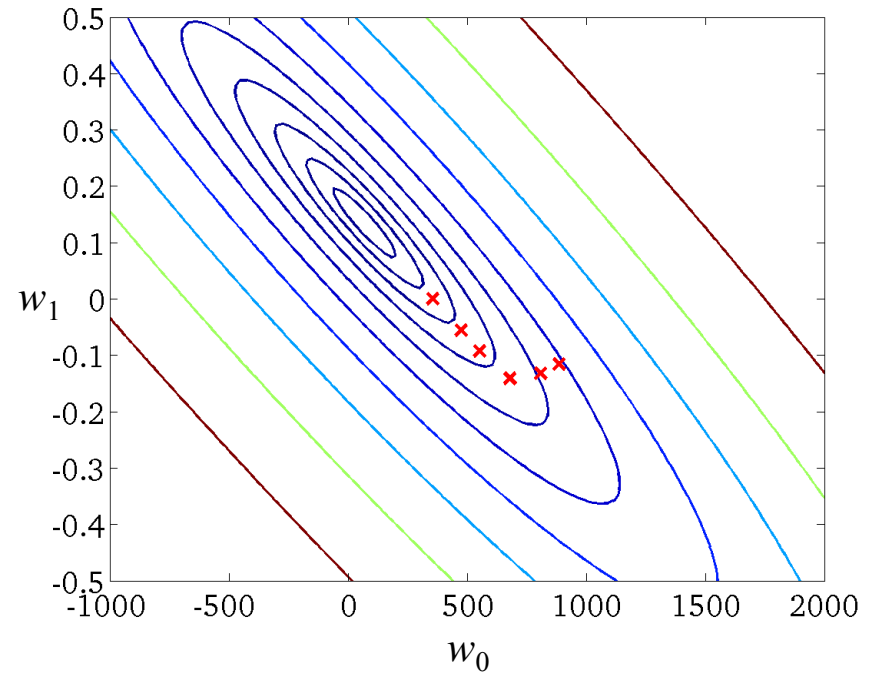
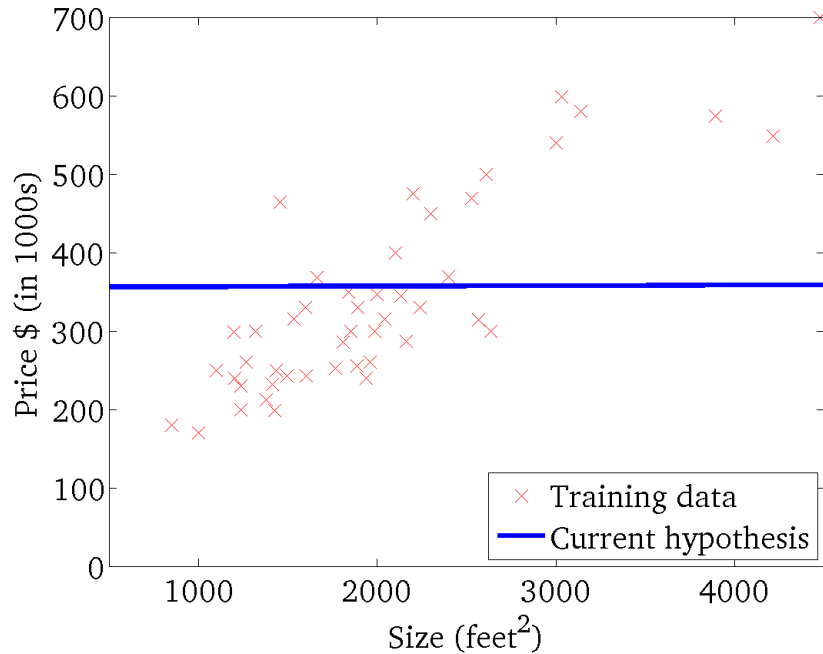
# Gradient Descent in Action



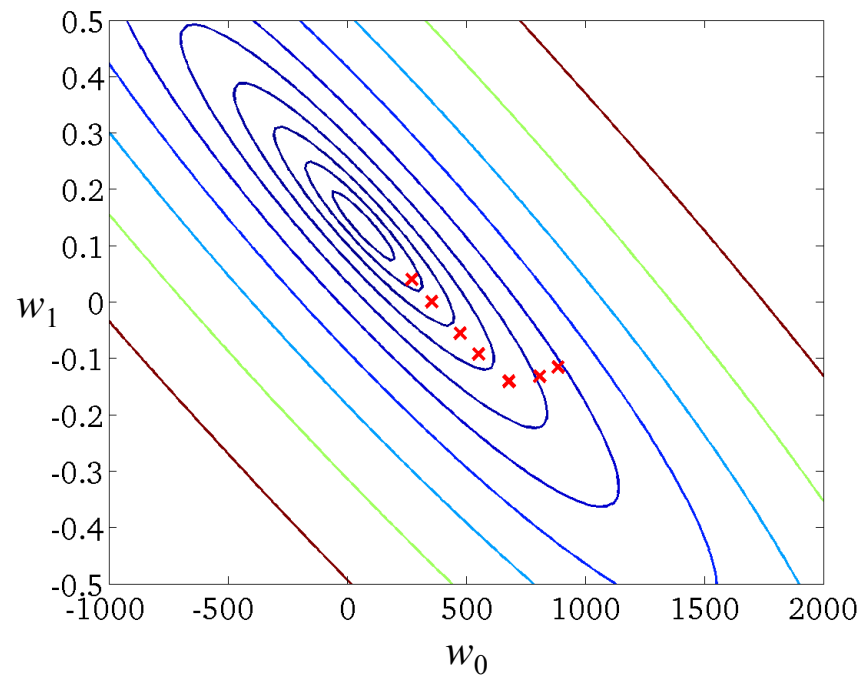
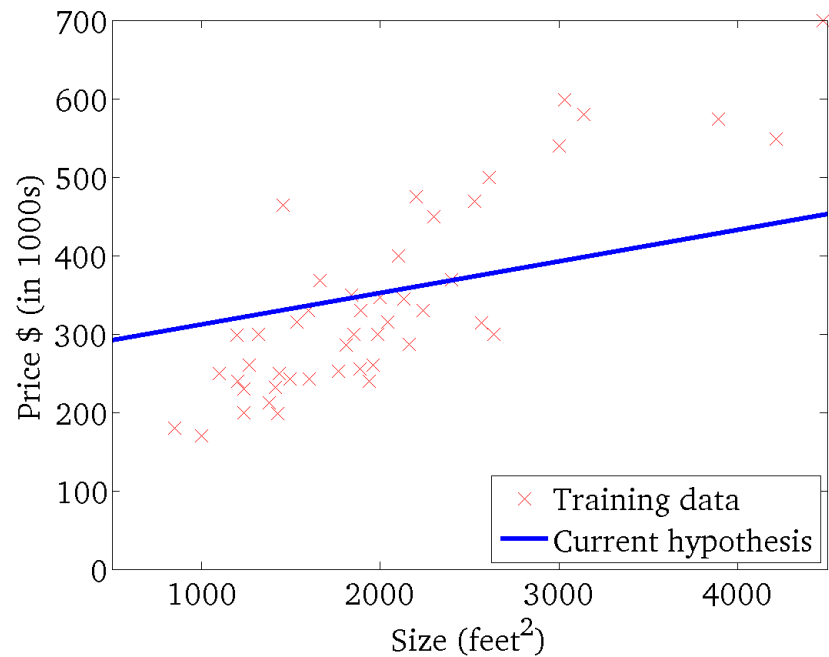
# Gradient Descent in Action



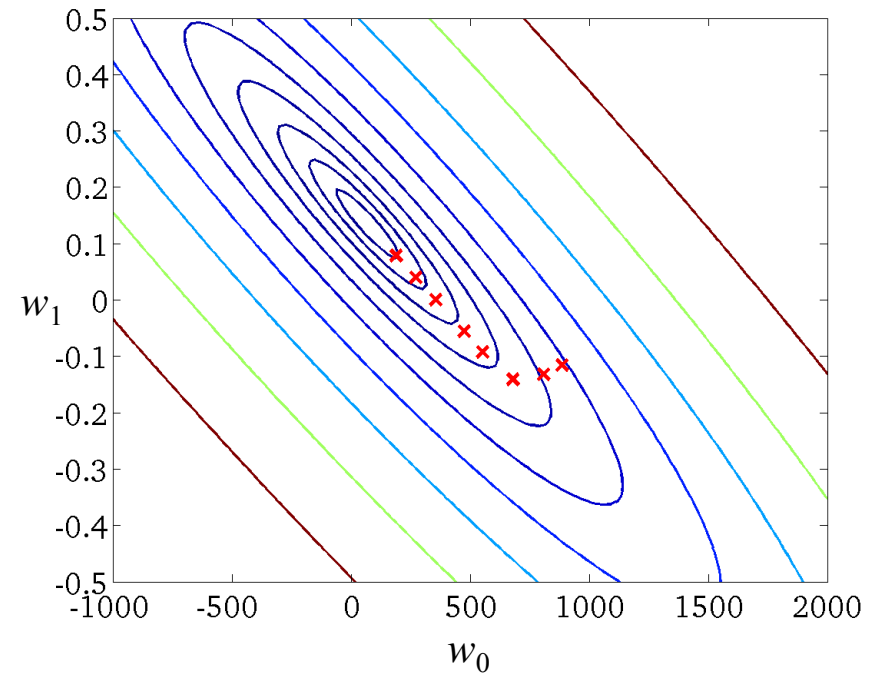
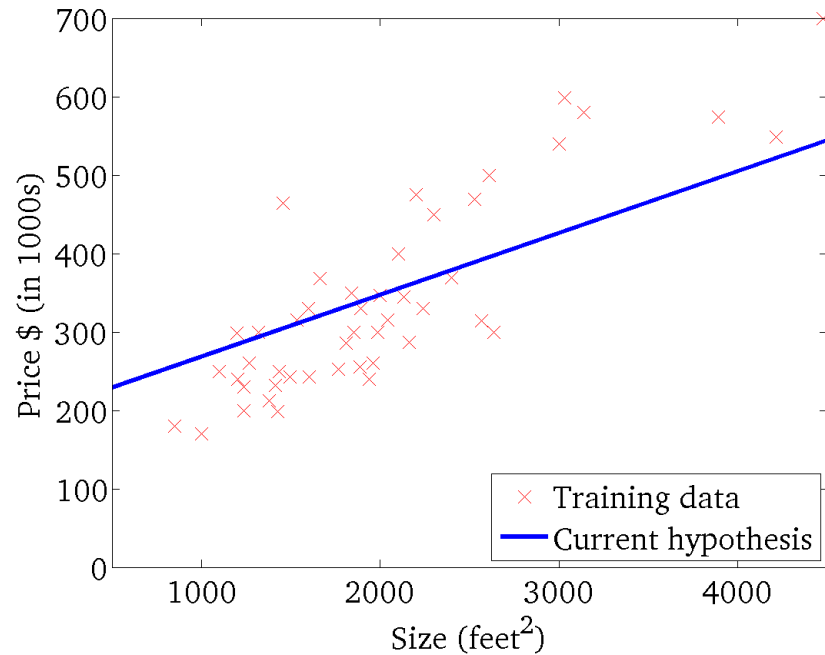
# Gradient Descent in Action



# Gradient Descent in Action

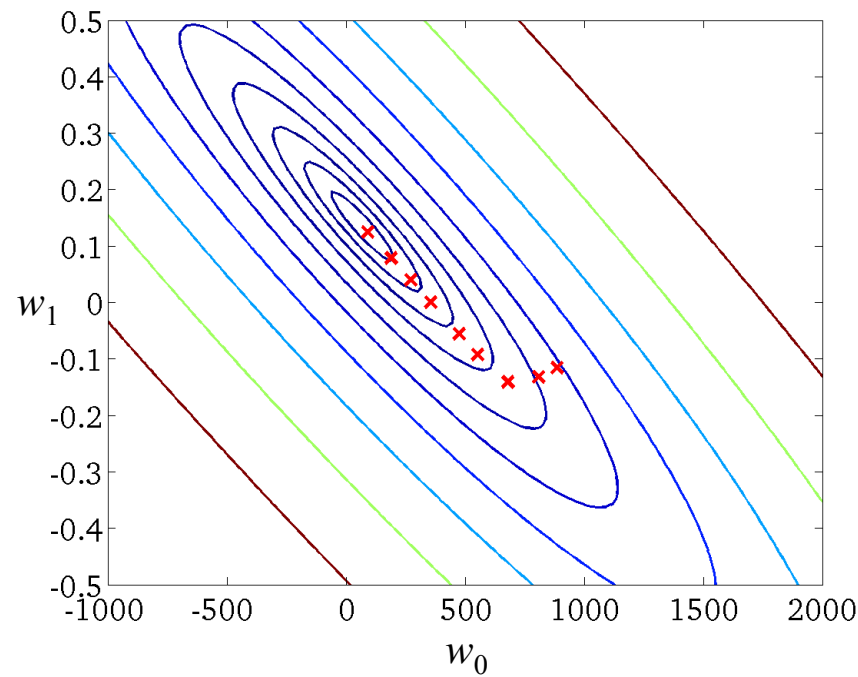
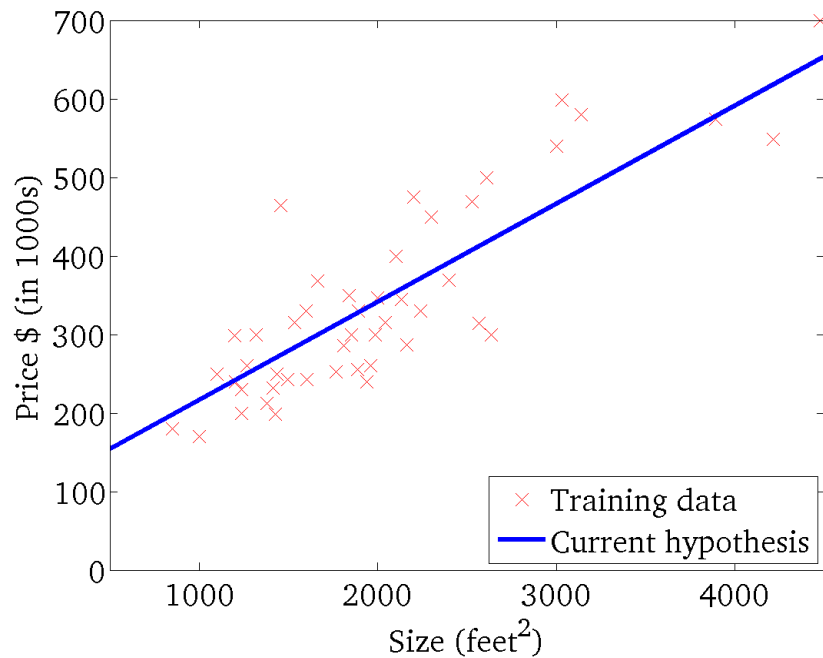


# Gradient Descent in Action





# Gradient Descent in Action



# Multiple Variables

We generally have multiple variables in our data.

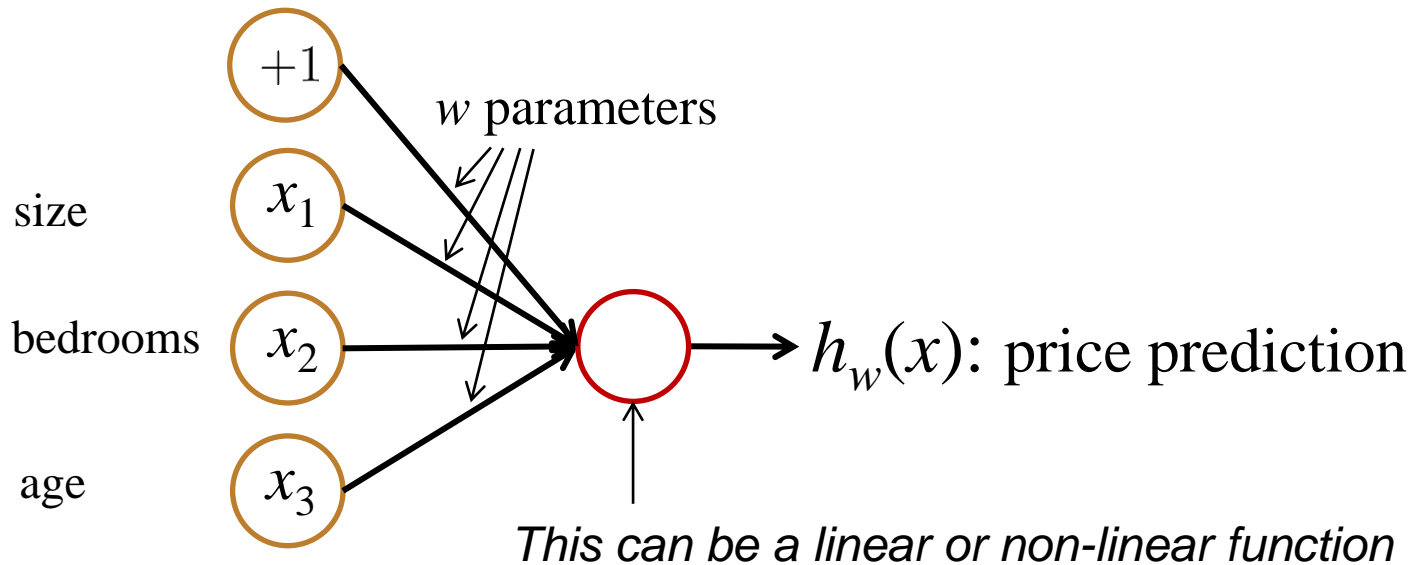
We can not visualize the line since it has >2 features, but cost function and gradient descent works the same.

Size (feet <sup>2</sup> )	Number of bedrooms	Age of home (years)	Price (\$1000)
$x_1$	$x_2$	$x_3$	$y$
2104	5	46	460
1416	3	40	232
1534	3	30	315
852	2	36	178
...	...	...	...

With new features:  $h_w(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_3$

# What is a Neural Network?

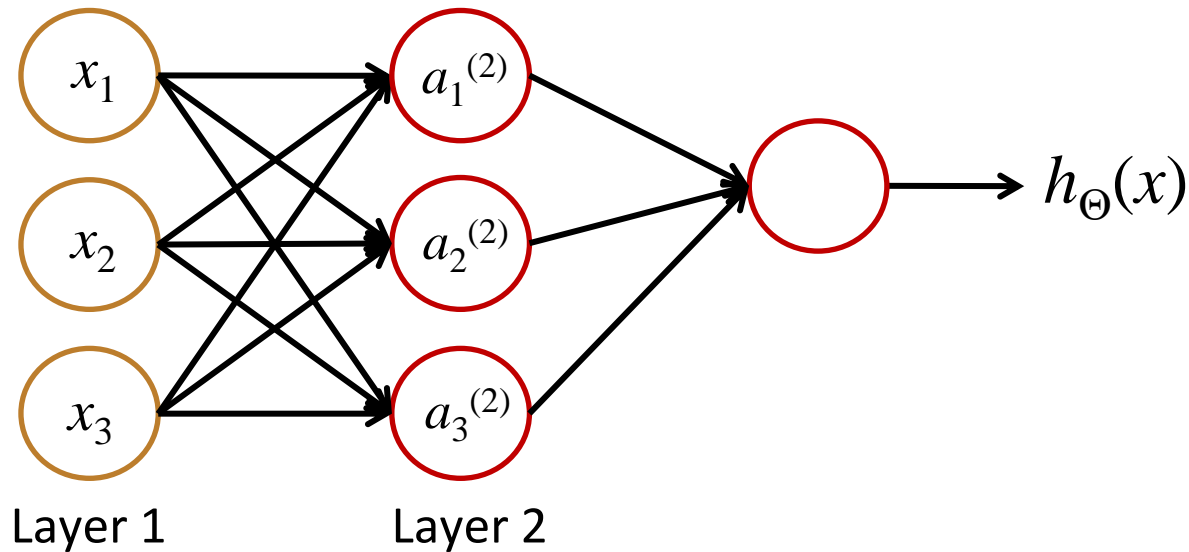
It is a mathematical model to make an estimation with given input. Think about the same linear regression problem:



Linear function:  $h_w(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_3$

Non-linear function:  $h_w(x) = f(w_0 + w_1x_1 + w_2x_2 + w_3x_3)$

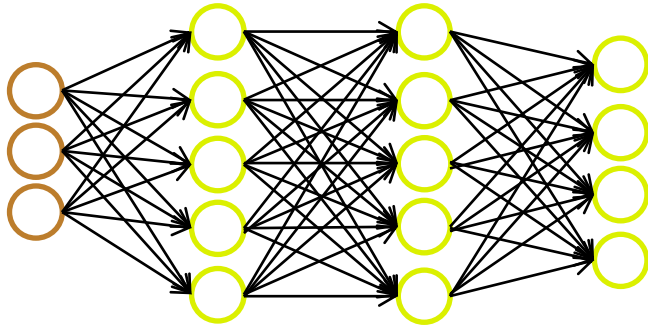
# What is a Neural Network?



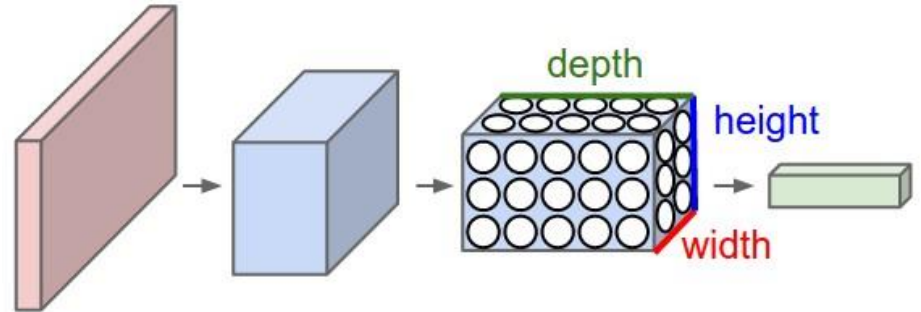
What neural network does in this example is similar to what linear regression does, but it creates new features (activations  $a_1^{(2)}$ ,  $a_2^{(2)}$ ,  $a_3^{(2)}$ ) and learns them.

Also, a non-linear function is used for activations.

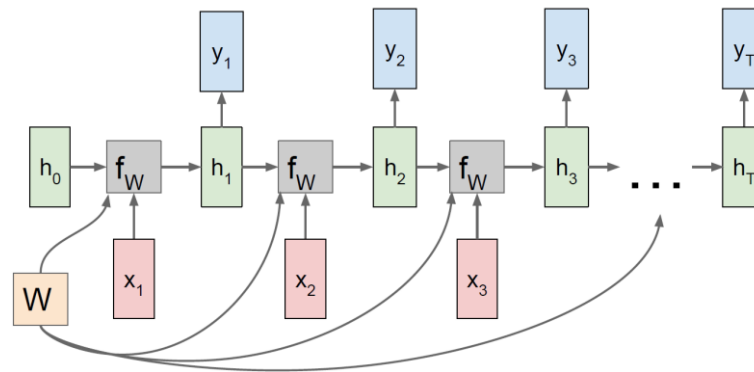
# Neural Network Examples



Standard NN



Convolutional NN



Recurrent NN

# Supervised Learning: Classification

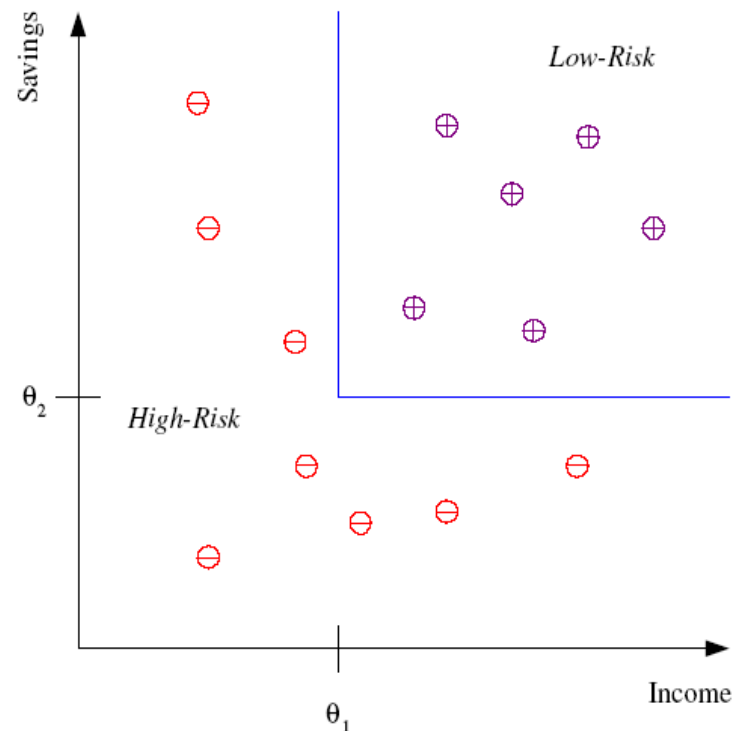
Estimating the house prices is a regression problem, we estimate a real-valued output in a continuous range.

However most problems in ML are classification:

E.g.: Bank credit scoring

Differentiating between **low-risk** and **high-risk** customers from their *income* and *savings*.

There are two **classes** to be distinguished from each other. This is **classification**.



# Neural Networks are Especially Used in Supervised Learning

Input	Output	Application	NN type
Home features	Price	Real Estate	Standard NN
Customer features	Customer type	Bank credit scoring	Standard NN
Image	Object (1000s)	Photo tagging	CNN
Audio	Text transcript	Speech recognition	RNN
Text (English)	Text (French)	Machine translation	RNN

# Why is Deep Learning Taking Off? The Case of 'Image Classification'

Multi-class image classification:



Pedestrian



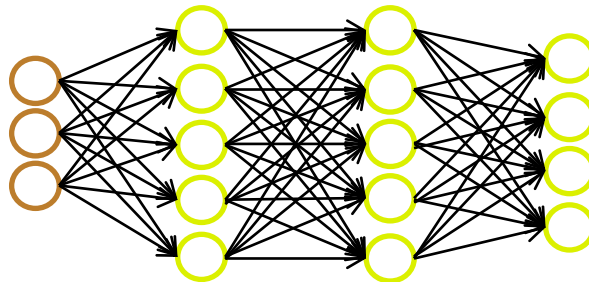
Car



Motorcycle



Truck



$$h_w(x) \in R^4$$


E.g. For pedestrians, we want  $h_w(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$



# An Image Classification Dataset

IMAGENET is dataset of 14M images from 22K categories.

Using a part of this dataset (1.4 M images from 1000 classes) a competition is organized: ImageNet Large-Scale Visual Recognition Challenge (ILSVRC).



www.image-net.org

**22K** categories and **14M** images

- Animals
  - Bird
  - Fish
  - Mammal
  - Invertebrate
- Plants
  - Tree
  - Flower
  - Food
  - Materials
- Structures
  - Artifact
  - Tools
  - Appliances
  - Structures
- Person
  - Scenes
    - Indoor
    - Geological Formations
  - Sport Activities

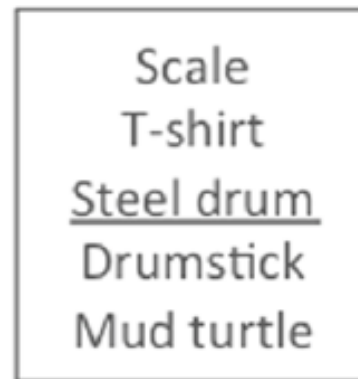
# Image Classification

In ILSVRC, algorithms produce 5 top guesses for each image. If actual image category is one of these 5 labels than guess is 'correct'.

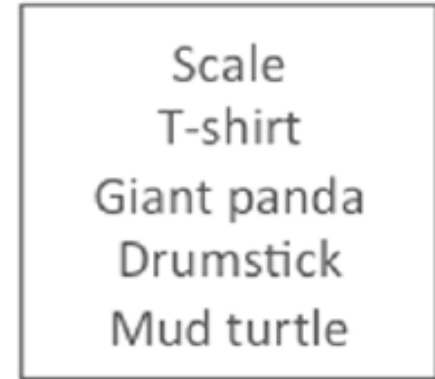
Total no of incorrect answers in this sense is called top-5 error.



A test image



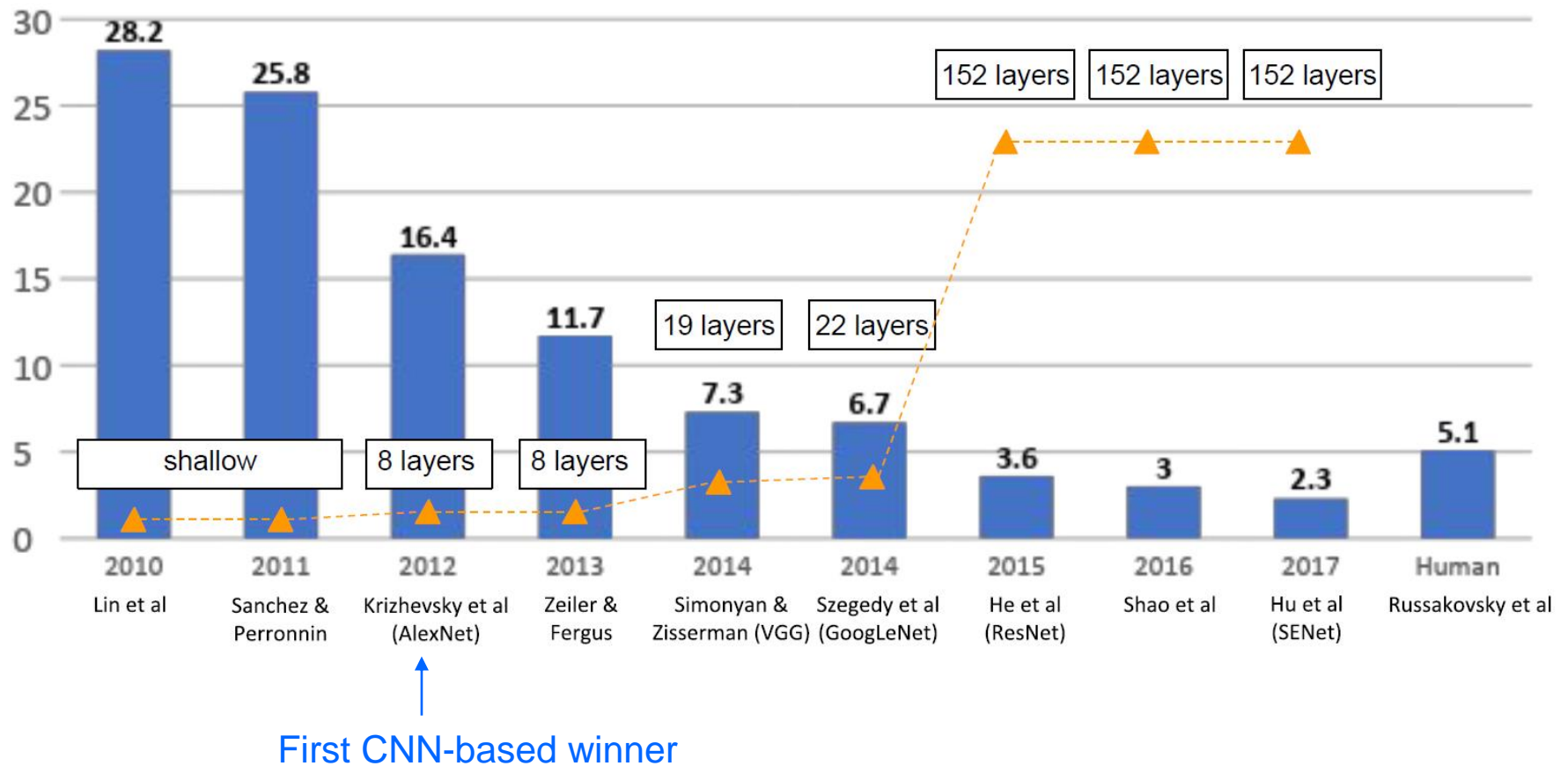
Accuracy: 1



Accuracy: 0

# Image Classification

ILSVRC image classification task top-5 error chart



# Other Example Vision Tasks

CNNs became an important tool for tasks like image captioning and segmentation.

Figure from He et al, Mask R-CNN

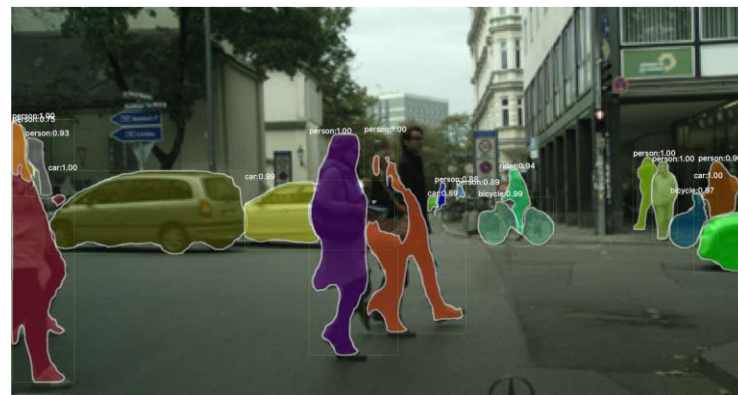


Figure by Andrej Karpathy, Fei-Fei Li



"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."



"two young girls are playing with  
lego toy."

# Other Example Vision Tasks

Style transfer: A given ‘content’ is regenerated using a style image.

## Artistic style transfer for videos

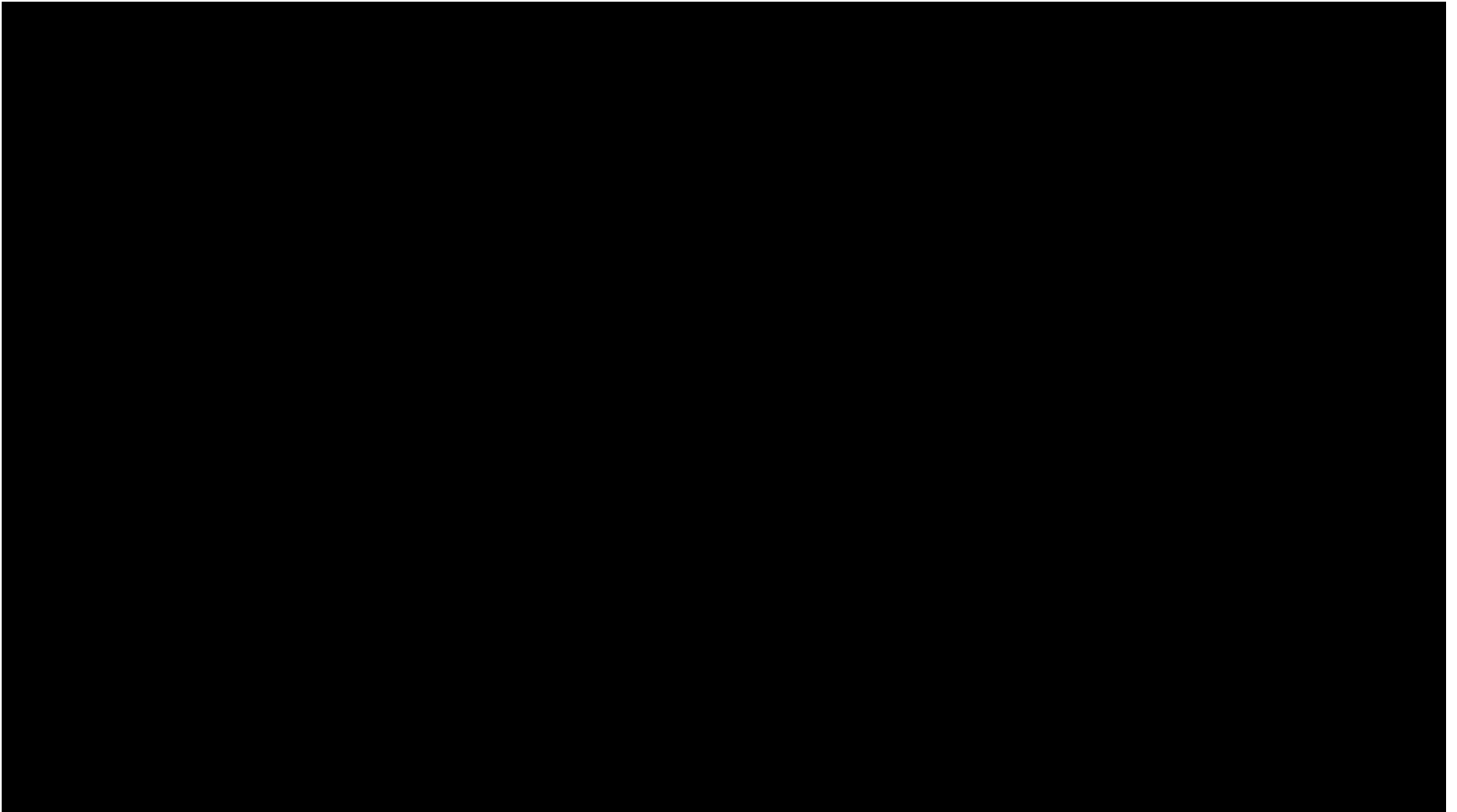
Manuel Ruder  
Alexey Dosovitskiy  
Thomas Brox

University of Freiburg  
Chair of Pattern Recognition and Image Processing

Ruder, Dosovitsky, Brox, “Artistic style transfer for videos”, Arxiv 2016  
Video available at <https://www.youtube.com/watch?v=Khuj4ASldmU>

# Other Example Vision Tasks

Text-to-image and Inpainting



<https://openai.com/dall-e-2/>

<https://openai.com/dall-e-3/>



# Text-to-video (Sora by OpenAI)

Prompt: A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She walks confidently and casually. The street is damp and reflective, creating a mirror effect of the colorful lights. Many pedestrians walk about.



<https://openai.com/sora/>

# A non-vision example: ChatGPT

User     this code is not working like i expect — how do i fix it?

```
resultWorkerErr := make(chan error)
defer close(resultWorkerErr)
go func() {
    defer cancel()
    resultWorkerErr <- b.resultWorker(ctx)
}()

err := b.worker(ctx)
cancel()
if err == nil {
    return <-resultWorkerErr
}
return multierror.Append(err, <-resultWorkerErr)
```



# But..

DL becomes proficient in a single task (no general intelligence).

Even 'very impressive' ChatGPT is not as smart as we think.

Main problems with ChatGPT and generative AI:

- 1) Plagiarism: The results are a cut-and-paste synthesis drawn from various sources. In some cases, the borrowing is so obvious.
- 2) Information is not knowledge: A well-trained machine can find the right answer in a very large database. Unpredictability, meanwhile, is what drives creative innovation. AI can not create a concept that was not thought by humans before (because it is not in the training data).
- 3) Undetected bias: If the training data set is inaccurate or biased, the results will reflect it. Prejudicial opinions and partisanship have been shown to find their way into AI models.

Source: David Wilson <https://thetechnologyexpress.com/10-reasons-to-worry-about-generative-ai/>