Introduction to Machine Learning

Tu Bui
Central for Vision Speech and Signal Processing (CVSSP)
University of Surrey
Guildford, Surrey, United Kingdom

Contents

- What is machine learning (ML)
- Linear regression
- Optimisation
- Classification
- ML in practice

Machine Learning (ML)

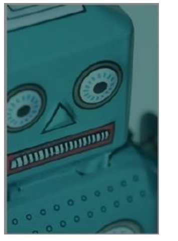
- ML appears frequently in our daily life

Web search

Auto tagging

Product recommendation

Machine Learning (ML)



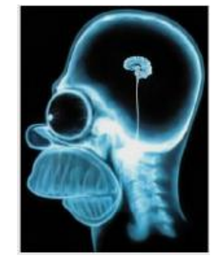
- Definition

"Field of study that gives computers the ability to learn from and make predictions on **data** without explicitly programmed"

Arthur Samuel, 1959

- ➢ Learn from data
- ➢ NOT following static program instructions
- ➢ Humans are learning machines

Why ML?



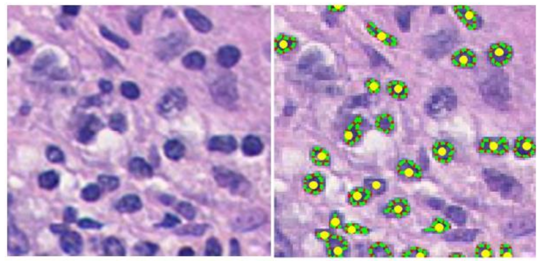
- We cannot program everything

- Some tasks are difficult to define algorithmically
  - Especially in computer vision, e.g. recognise objects

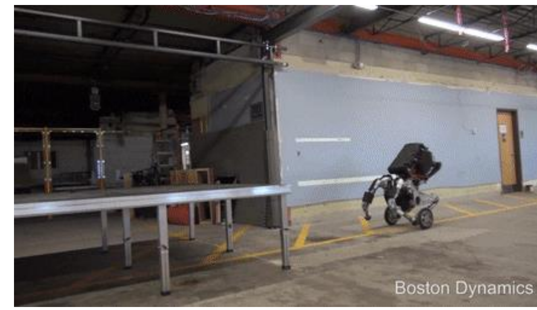
- Cost effective

ML applications

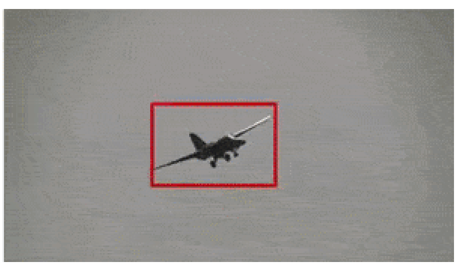
Market analysis



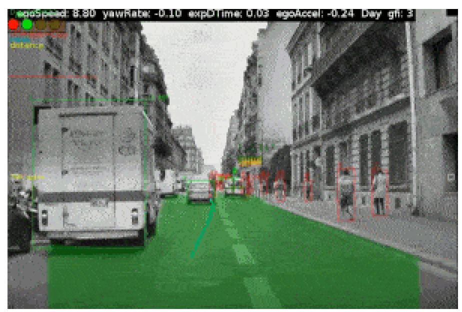
Medical imaging
http://www.robots.ox.ac.uk



Robot



Tracking/surveillance

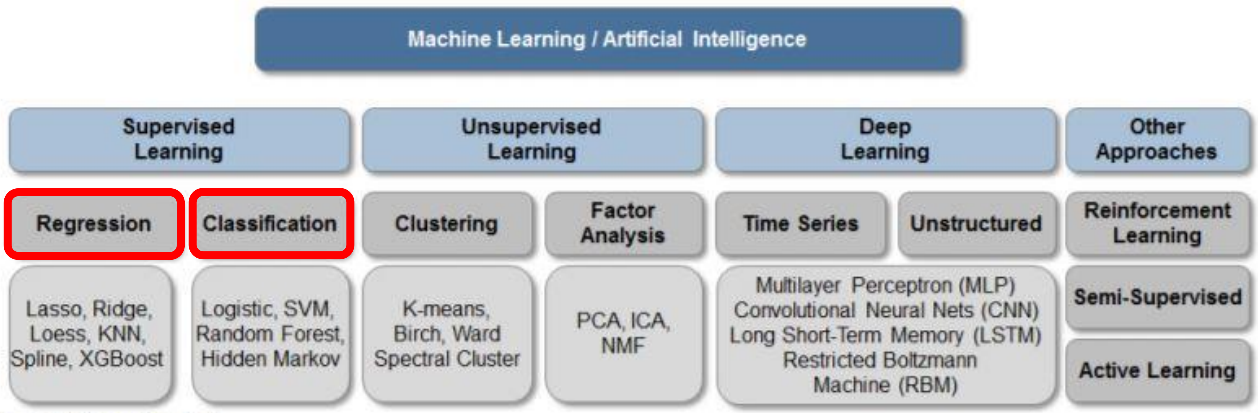


Self-driving car

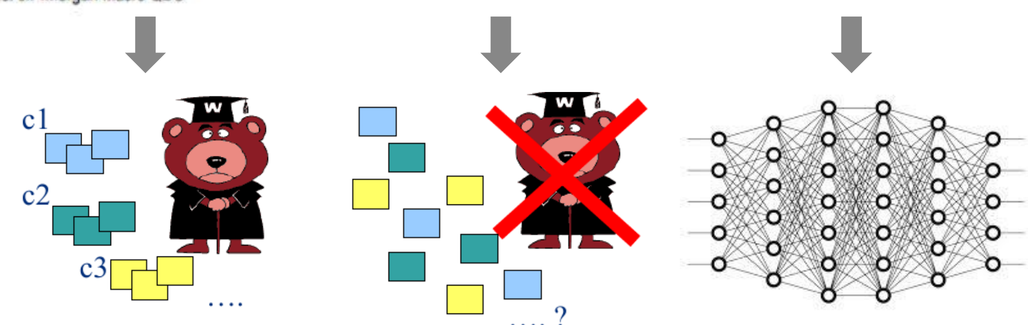
3D reconstruction/analysis

ML types

Figure 39: Classification of Machine Learning techniques



Source: J.P.Morgan Macro QDS



Supervised ML: regression vs. classification

- Regression
  - **“Continuous”** valued output
- Example: house price prediction

- Classification
  - **Discrete** set of output labels/classes
- Example: gender prediction

♂ ? ♀

Quiz: regression or classification?

1. You are helping a bank to write two programs that help decide whether the bank should provide credit card to new customers.
   a) Your 1st program will output credit score for each customer based on their credit history.
   b) Your 2nd program will rely on the output of the 1st program plus personal ratings of the bank manager(s) to decide if a customer will get a credit card or not.
2. You write a program to filter spam emails in your inbox. If an email is marked as spam by your program, it will be moved to the Junk folder.
3. You are running a company that sells umbrella. You want to predict how many umbrellas will be sold in the next 3 months based on last 10 year sale data.

1a. regression

1b. classification

2. classification

3. regression

Supervised learning: problem formulation

- Mathematically, the machine is realising a hypothesis function to approximate the output:

$$\boldsymbol{\mathcal{F}}\colon\ \boldsymbol{x} \ \mapsto\ \boldsymbol{y}$$

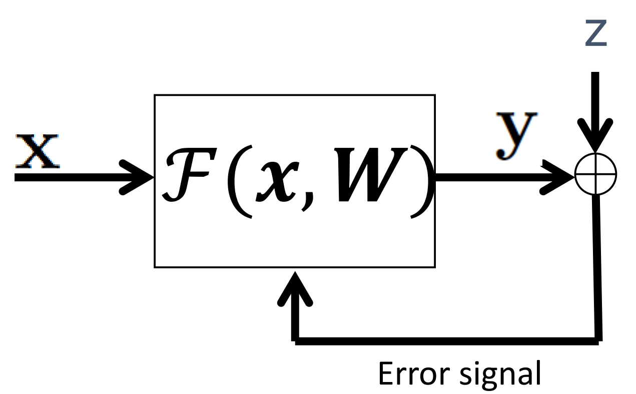
$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, \boldsymbol{W})$$

  - x ….. D dimensional input
  - y ….. p dimensional output
  - W ….. parameters

$$\mathbf{x} \longrightarrow \boxed{\mathcal{F}(\boldsymbol{x}, \boldsymbol{W})} \longrightarrow \mathbf{y}$$

Machine training

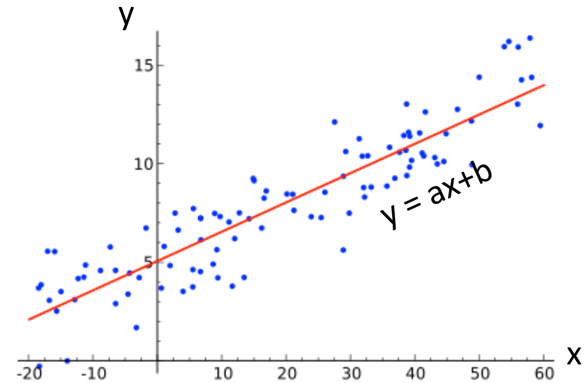
- Pre-requisites
  - Training set $\boldsymbol{X} = \left\{\boldsymbol{x}^{(\mathbf{1})}, \boldsymbol{x}^{(\mathbf{2})}, \dots, \boldsymbol{x}^{(\boldsymbol{N})}\right\}$
  - Groundtruth target values $\boldsymbol{Z} = \left\{\boldsymbol{z}^{(\mathbf{1})}, \boldsymbol{z}^{(\mathbf{2})}, \dots, \boldsymbol{z}^{(\boldsymbol{N})}\right\}$
  - Form of hypothesis function $\boldsymbol{\mathcal{F}}$
  - Cost function (objective, loss or error measure)
  - Optimisation (procedure to update W)



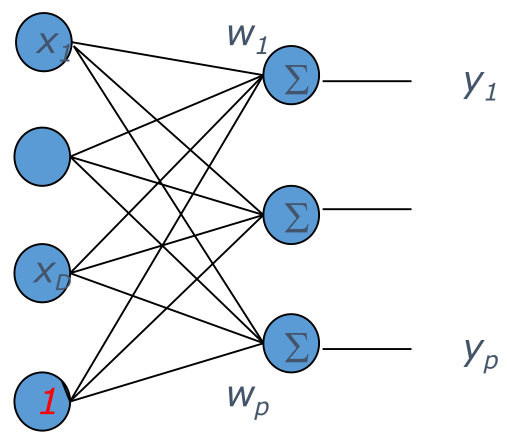
Basic linear machine

- The simplest form of $\mathcal{F}$ is a linear function

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, \boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x}$$

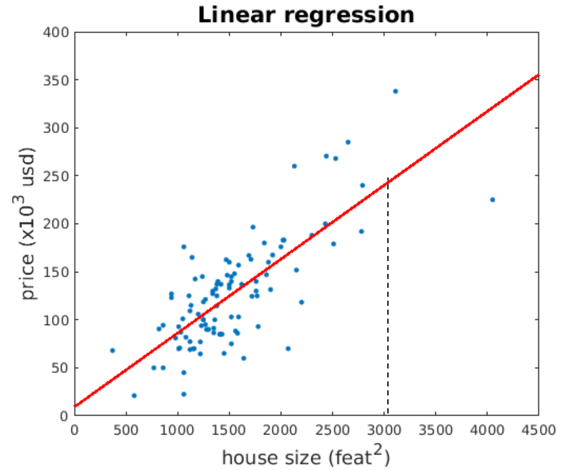


$$\begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_p \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \dots w_{1D} & {\color{red} b_1} \\ w_{21} & w_{22} \dots w_{2D} & {\color{red} b_2} \\ & \dots & \\ & \dots & \\ & \dots & \\ w_{p1} & w_{p2} \dots w_{pD} & {\color{red} b_p} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_D \\ {\color{red} 1} \end{bmatrix} := \begin{bmatrix} w_1^T \\ w_2^T \\ . \\ . \\ . \\ w_p^T \end{bmatrix} \boldsymbol{x}$$



Linear regression: example

- Pre-requisites
  - Training set $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(N)}\}$
  - Groundtruth target values $\boldsymbol{Z} = \{\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \dots, \boldsymbol{z}^{(N)}\}$
  - Form of hypothesis function $\boldsymbol{\mathcal{F}}$
- Cost function (loss, objective or error measure)
- Optimisation (procedure to update W)



$$\boldsymbol{y} = \boldsymbol{\mathcal{F}}(\boldsymbol{x}, \boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x}$$

House size $\rightarrow \boxed{\boldsymbol{\mathcal{F}}(\boldsymbol{x}, \boldsymbol{W})} \rightarrow$ Estimated price

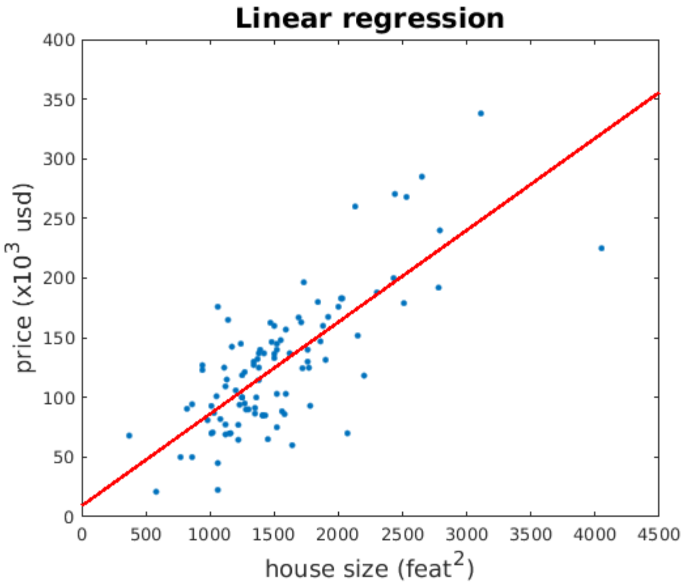
X	House size (feet²)	Price ($ x10³)	Z
$x^{(1)}$	1240	145	$z^{(1)}$
$x^{(2)}$	370	68	$z^{(2)}$
$x^{(3)}$	1130	115	$z^{(3)}$
$x^{(4)}$	1120	69	$z^{(4)}$
$x^{(5)}$	1710	163	$z^{(5)}$
…	…	…	…
$x^{(N)}$	860	50	$z^{(N)}$

Cost/Loss function

$$\mathcal{L}(W) = \frac{1}{2N}\sum_{i=1}^{N}\left(\boldsymbol{y}^{(i)} - \boldsymbol{z}^{(i)}\right)^2$$

$$= \frac{1}{2N}\sum_{i=1}^{N}\left(\mathcal{F}(\boldsymbol{x}^{(i)}, W) - \boldsymbol{z}^{(i)}\right)^2$$

$$\underset{W}{\mathrm{Min}}\ \mathcal{L}(W)$$

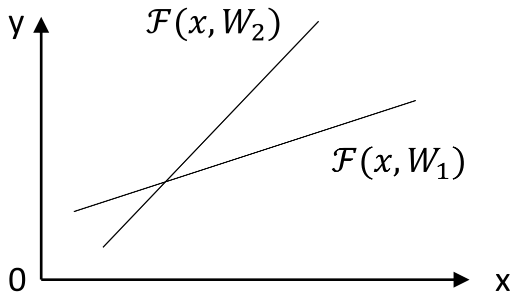


Hypothesis function vs. cost function

Hypothesis function

$$\mathcal{F}(\boldsymbol{x}, W) = W\boldsymbol{x} = [a\ b]\begin{bmatrix} x \\ 1 \end{bmatrix}$$
$$= ax + b$$

- Is a function of **x**, given parameter W

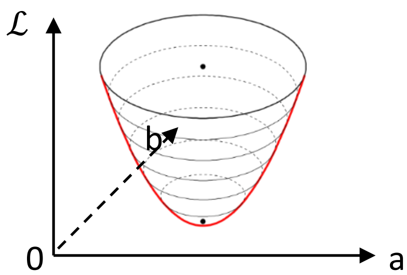


Cost function

$$\mathcal{L}(W) = \frac{1}{2N}\sum_{i=1}^{N}\left(\mathcal{F}\left(\boldsymbol{x}^{(i)}, W\right) - \boldsymbol{z}^{(i)}\right)^2$$
$$= \frac{1}{2N}\sum_{i=1}^{N}\left(W\boldsymbol{x}^{(i)} - \boldsymbol{z}^{(i)}\right)^2$$

- Is a function of W

Todo: 2D map

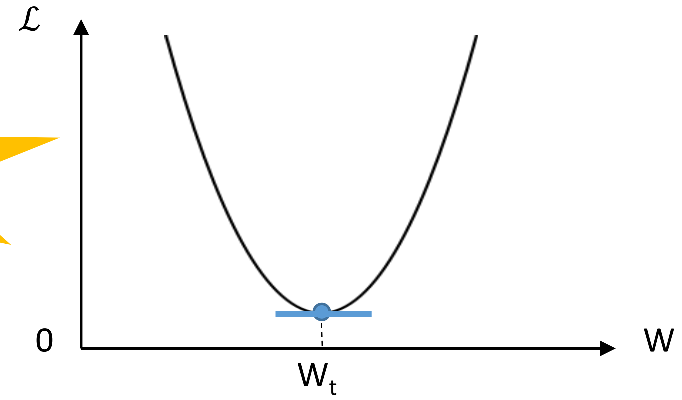


Optimisation method #1

$$\mathcal{F}(\boldsymbol{x}, \boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x}$$

$$\mathcal{L}(W) = \frac{1}{2N}\sum_{i=1}^{N}\left(W\boldsymbol{x}^{(i)} - \boldsymbol{z}^{(i)}\right)^2$$

$$\min_{W} \mathcal{L}(W)$$



Solve for W $\boxed{\frac{\partial \mathcal{L}}{\partial W} = 0}$

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{1}{N}\sum_{i=1}^{N}\left(W\boldsymbol{x}^{(i)} - \boldsymbol{z}^{(i)}\right)\boldsymbol{x}^{(i)} = \frac{1}{N}(W\boldsymbol{X} - \boldsymbol{Z})\boldsymbol{X}^T := 0$$

$$\boxed{W = \boldsymbol{Z}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T)^{-1}}$$

Optimisation method #1

Pros

- Straight forward, no loop.
- No additional parameters.

Cons

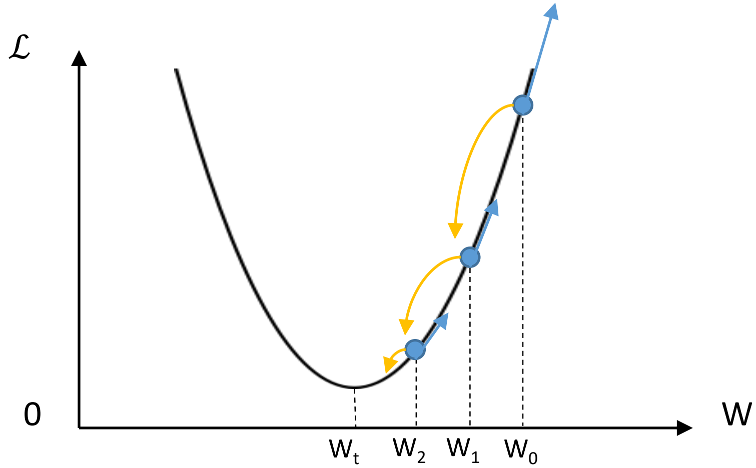
- Need to compute:
  $$(\boldsymbol{X}\boldsymbol{X}^T)^{-1}$$
  - slow when feature dimension is large
  - $\boldsymbol{X}\boldsymbol{X}^T$ may not invertible

$\Longrightarrow$ Gradient descent

Optimisation method #2: Gradient descent

Training procedure:

1. Initialisation: assign a random value to W
$$W := W_0$$
2. Compute gradient of the loss function
$$\frac{\partial \mathcal{L}}{\partial W}$$
3. Update W
$$W := W - k\frac{\partial \mathcal{L}}{\partial W}$$
where k is the learning rate
4. Repeat step 2 & 3.



Stopping criteria

When should we terminate the updating iterations?

- After fixed number of iterations: n = 1k, 1M?
- When improvement in the loss is small enough?

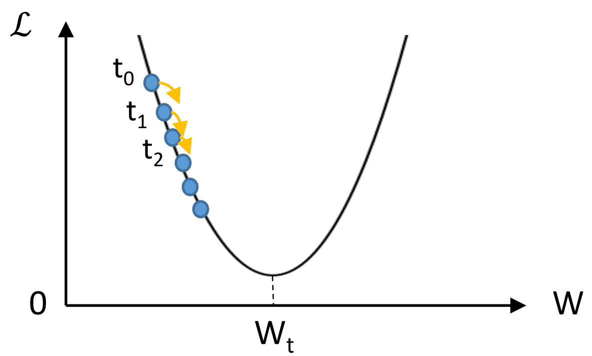
$$\Delta\mathcal{L}_t = |\mathcal{L}(W_t) - \mathcal{L}(W_{t-1})| < \epsilon$$

- Both?

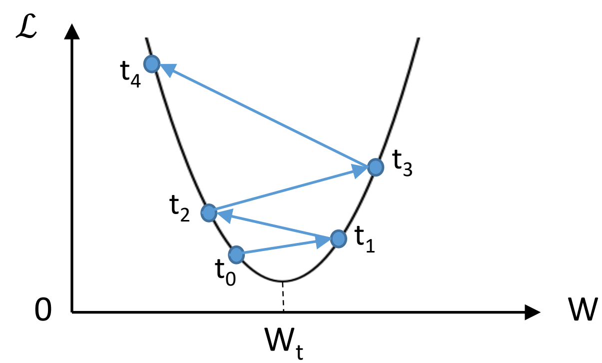
Effect of learning rate

$$W := W - k\frac{\partial \mathcal{L}}{\partial W}$$

K too small: gradient descent can be slow



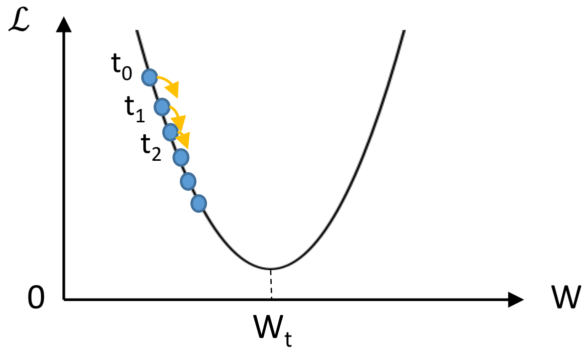
K too large: the training may not converge

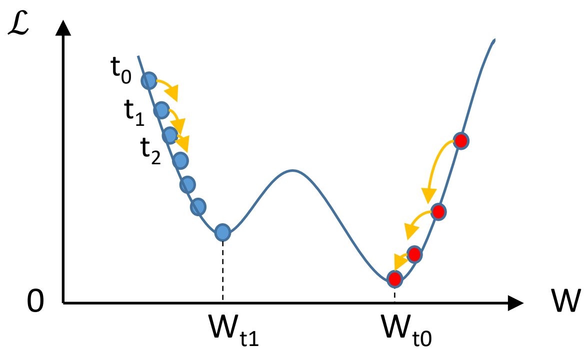


Quiz: should we vary learning rate during the training of our model?

Local optimal in gradient descent

- Linear regression: $\mathcal{L}(W)$ is always convex
  - $\mathcal{L}(W) = \frac{1}{2N}\sum_{i=1}^{N}\left(W\boldsymbol{x}^{(i)} - \boldsymbol{z}^{(i)}\right)^2$
- What if $\mathcal{L}(W)$ non-convex?
  - Gradient descent may converge to local optimal.
- Possible solution:
  - Different parameter (W) initialisations
  - Stochastic gradient descent





Gradient descent types

- **Batch gradient descent**: each update iteration is derived from the whole training set

$$\mathcal{L}(W) = \frac{1}{2N}\sum\left(\mathcal{F}(\boldsymbol{x}^{(1:N)}, W) - \boldsymbol{z}^{(1:N)}\right)^2 \qquad W := W - k\frac{\partial \mathcal{L}}{\partial W}$$

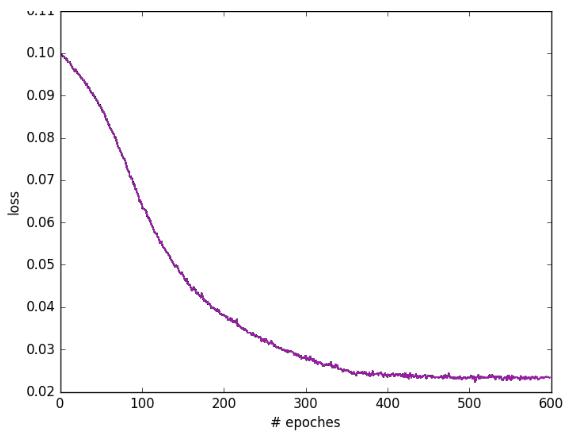
- **Stochastic gradient descent**: each update is derived from one sample in the training set

$$\mathcal{L}(W) = \frac{1}{2}\left(\mathcal{F}\left(\boldsymbol{x}^{(i)}, W\right) - \boldsymbol{z}^{(i)}\right)^2 \qquad W := W - k\frac{\partial \mathcal{L}}{\partial W}$$

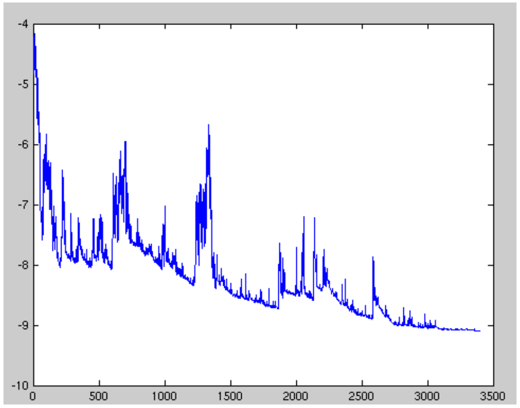
- **Mini-batch gradient descent**: perform update for every mini-batch of p training samples at a time

$$\mathcal{L}(W) = \frac{1}{2p}\sum\left(\mathcal{F}\left(x^{(i:i+p)}, W\right) - z^{(i:i+p)}\right)^2 \qquad W := W - k\frac{\partial \mathcal{L}}{\partial W}$$

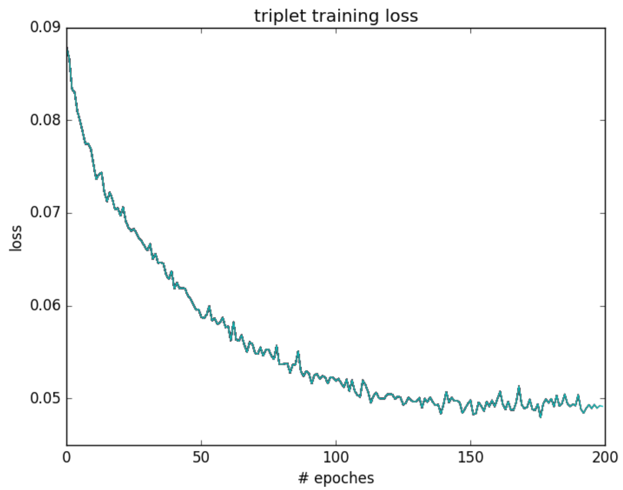
Gradient descent types



Batch gradient descent



Stochastic gradient descent*



Mini-batch gradient descent

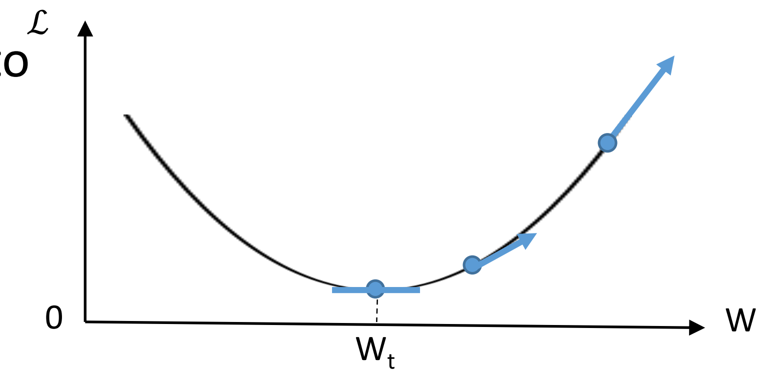
*Source: wikipedia

Gradient descent with momentum

- Problem: update is getting slow when close to the optima (gradient becomes small).

$$W := W - k\frac{\partial \mathcal{L}}{\partial W}$$

especially when W is multi-dimensional.



- Solution: give "velocity" and "momentum" to the update

$$v_t = m v_{t-1} + k\frac{\partial \mathcal{L}}{\partial W}$$
$$W := W - v_t$$

where m is momentum, k is learning rate.
In practice, m = 0.9 (usually).

Other optimisation methods

- Nesterov accelerated gradient
$$v_t = mv_{t-1} + k\frac{\partial}{\partial W}\mathcal{F}(W - mv_{t-1})$$
$$W := W - v_t$$
compute gradient at “future” W.
- Adagrad
  - Assign different learning rates to each component of W.
  - Perform larger updates to infrequent and smaller updates to frequent components.
- Adadelta, AdaMax, Adam, Nadam, RMSprop … all using adaptive learning rate methods.

Multivariate regression

House size (feet^2)	#bedrooms	House age (years)	Distance from city centre (km)
1240	4	25	1.5
370	1	40	20.1
1130	3	5	13.0
1120	2	60	100.5
1710	4	13	30.7
…	…	…	…
860	2	8	46.4

Price ($ x10^3)
145
68
115
69
163
…
50

$$y = ax + b$$

$$W = [a \;\; b] \quad \boldsymbol{x} = \begin{bmatrix} x \\ 1 \end{bmatrix}$$

$$\mathrm{y} = W\boldsymbol{x}$$

Univariate regression

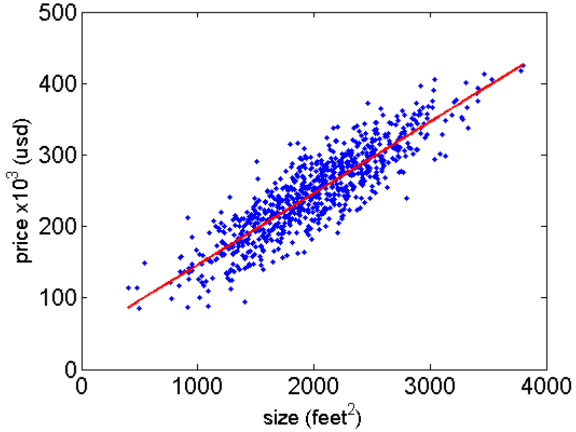
$$y = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_b$$

$$W = [w_1 \; w_2 \; w_3 \; w_4 \; w_b] \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ 1 \end{bmatrix}$$

$$\mathrm{y} = W\boldsymbol{x}$$

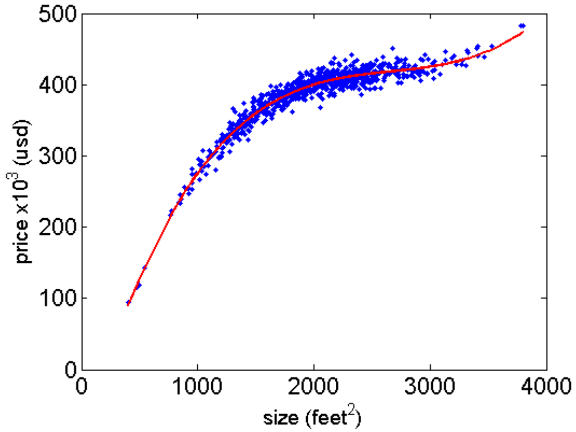
Multivariate regression

Polynomial regression



$$y = ax + b \qquad W = [a\ b]\ , \boldsymbol{x} = \begin{bmatrix} x \\ 1 \end{bmatrix}$$

$$\boxed{\boldsymbol{y = Wx}}$$



$$y = ax^3 + bx^2 + cx + d \qquad W = [a\ b\ c\ d]\ , \boldsymbol{x} = \begin{bmatrix} x^3 \\ x^2 \\ x \\ 1 \end{bmatrix}$$

$$y = ax + b\sqrt{x} + \mathrm{c} \qquad W = [a\ b\ c]\ , \boldsymbol{x} = \begin{bmatrix} x \\ \sqrt{x} \\ 1 \end{bmatrix}$$

Regression summary

Training set

- $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(N)}\}$
- $\boldsymbol{Z} = \{\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \dots, \boldsymbol{z}^{(N)}\}$

Hypothesis function

- $y = \mathcal{F}(x, W) = W\boldsymbol{x}$

Loss function

- $\mathcal{L}(W) = \frac{1}{2N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{z}^{(i)}\right)^2$

Optimisation

- Gradient descent

Classification

- Output: set of **discrete** values (label/class/category).
- Basic example: movie rating

Spyderman homecoming

John: ★ ★ ★ ★

Mary: ★ ★

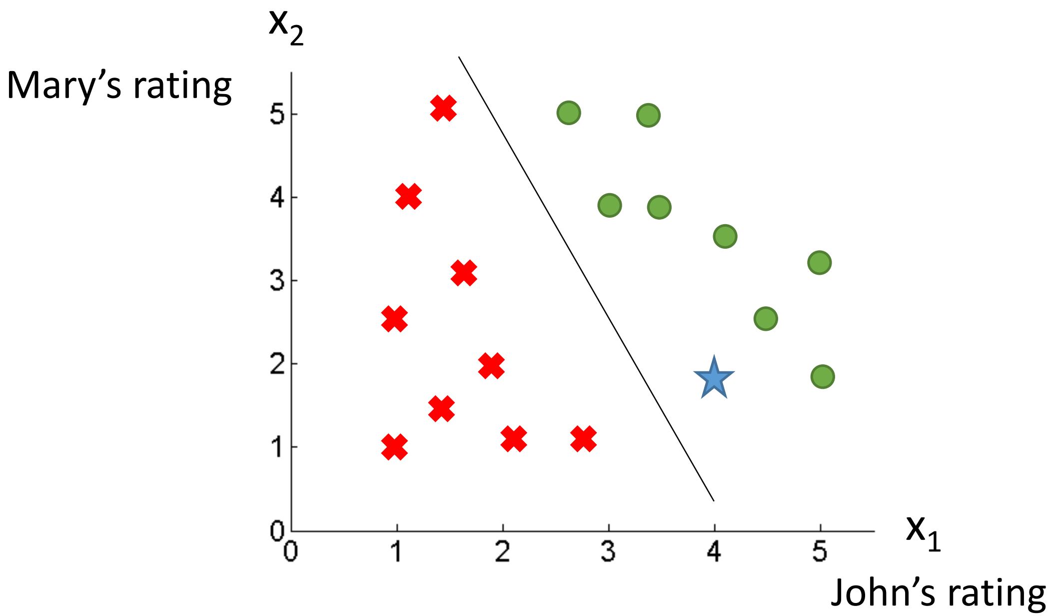
Should I watch it or not?

Binary Classification*

	Rates (1-5★)		Do I like it?
	John	Mary	
Captain America: civil war	5	3	Yes
The hobbit III	4	3.5	Yes
Lalaland	1.5	5	No
…..	…	…	…

* or logistic regression

Binary classification



Problem formulation

- Training set

$$\boldsymbol{x} = \begin{bmatrix} \text{John's rating} \\ \text{Mary's rating} \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, \ \boldsymbol{X} = \left\{ \boldsymbol{x}^{(\mathbf{1})}, \boldsymbol{x}^{(\mathbf{2})}, \dots, \boldsymbol{x}^{(\boldsymbol{N})} \right\}$$

$$\boldsymbol{z} = \begin{cases} 1 & \text{if I like} \\ 0 & \text{if I don't like} \end{cases}, \boldsymbol{Z} = \left\{ \boldsymbol{z}^{(\mathbf{1})}, \boldsymbol{z}^{(\mathbf{2})}, \dots, \boldsymbol{z}^{(\boldsymbol{N})} \right\}$$

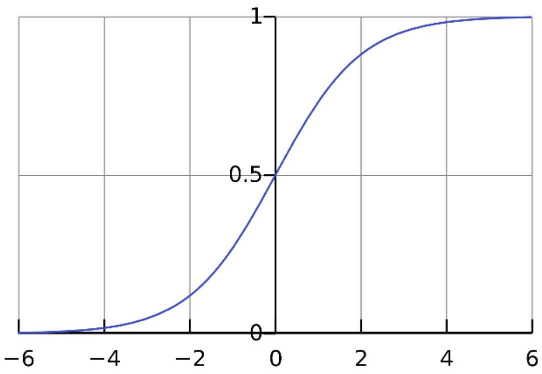
Hypothesis function

- $t = w_1 x_1 + w_2 x_2 + w_b$
- $W = [w_1 \; w_2 \; w_b] \;\; \rightarrow t = W\boldsymbol{x}$
- Hypothesis function

$$\mathcal{F}(x, W) = t = W\boldsymbol{x} \qquad ✗$$

$$\mathcal{F}(x, W) = g(t) \qquad ✓$$

Sigmoid* → $g(t) = \dfrac{1}{1 + e^{-t}}$



* Or logistic function

Decision making

- $\mathcal{F}(x, W) = g(t) > 0.5$

  $t = W\boldsymbol{x} > 0$

  I like the movie

- $\mathcal{F}(x, W) < 0.5$

  $t = W\boldsymbol{x} < 0$

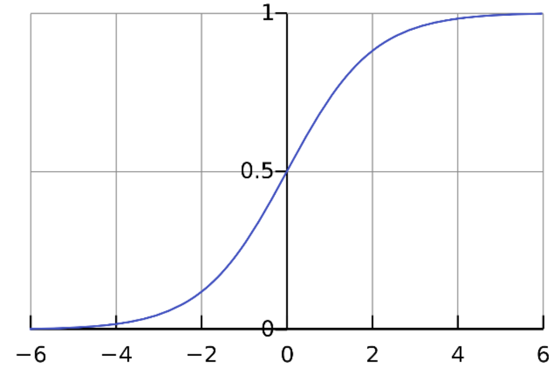
  I DON'T like the movie

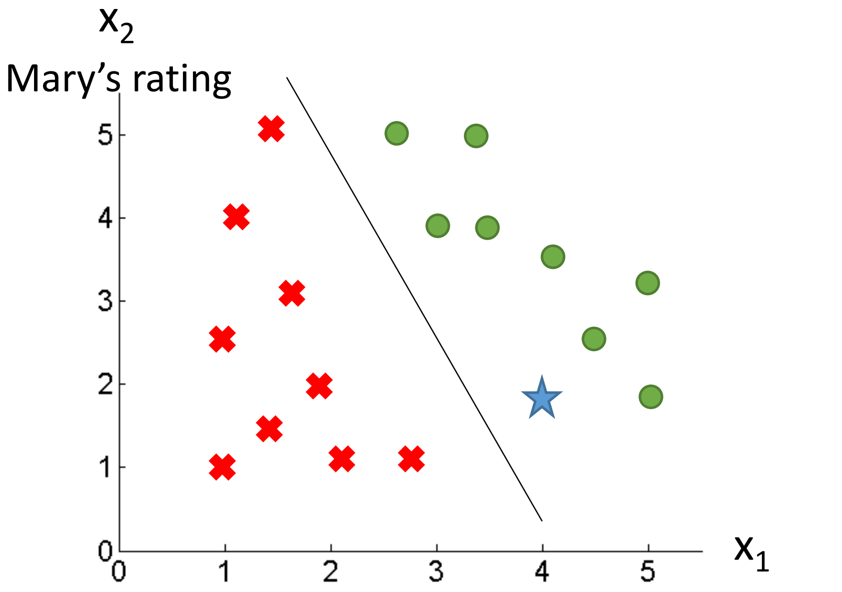
- $\mathcal{F}(x, W) = 0.5$

  $W\boldsymbol{x} = 0$

  $w_1 x_1 + w_2 x_2 + w_b = 0$

  50-50 chance liking the movie





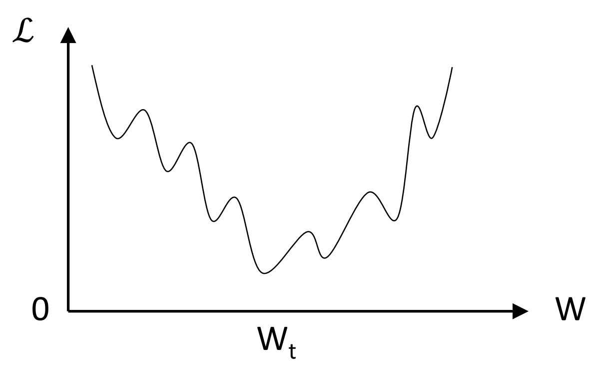
Loss function

- If regression loss

$$\mathcal{L}(W) = \frac{1}{2N}\sum_{i=1}^{N}\left(\mathcal{F}\left(\boldsymbol{x}^{(i)}, W\right) - \boldsymbol{z}^{(i)}\right)^2$$

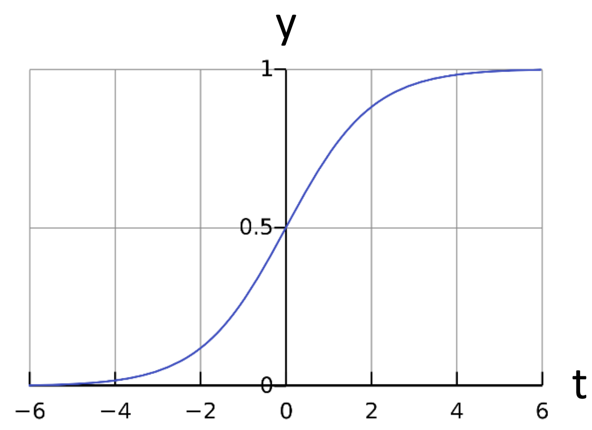
$$= \frac{1}{2N}\sum_{i=1}^{N}\left(\left(1 + e^{-Wx}\right)^{-1} - \boldsymbol{z}^{(i)}\right)^2$$

→ non-convex



Loss function

- $\mathcal{F}(\boldsymbol{x}, W) = g(t) = \frac{1}{1+e^{-Wx}}$ represent probability at y = 1



- $\mathcal{F}(\boldsymbol{x}, W) = P(y = 1|\boldsymbol{x}, W)$
- Probability y = 0 is $1 - \mathcal{F}(\boldsymbol{x}, W)$
- Goal: if groundtruth z=1, maximise $\mathcal{F}(\boldsymbol{x}, W)$; if z=0, maximise $1 - \mathcal{F}(\boldsymbol{x}, W)$
- $loss = \begin{cases} -\mathcal{F}(x, W) & \text{if z} = 1 \\ -(1 - \mathcal{F}(x, W)) & \text{if z} = 0 \end{cases}$

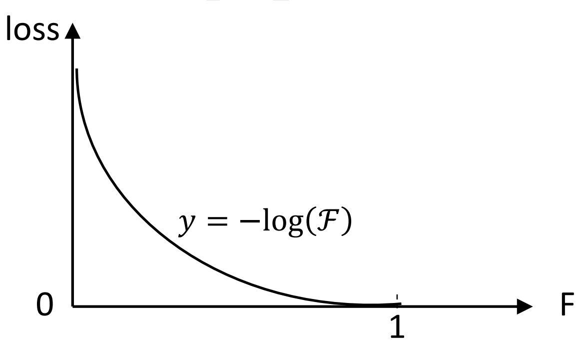
Loss function

- $loss = \begin{cases} -\mathcal{F}(x, W) & \text{if z} = 1 \\ -\left(1 - \mathcal{F}(x, W)\right) & \text{if z} = 0 \end{cases}$ → non-convex

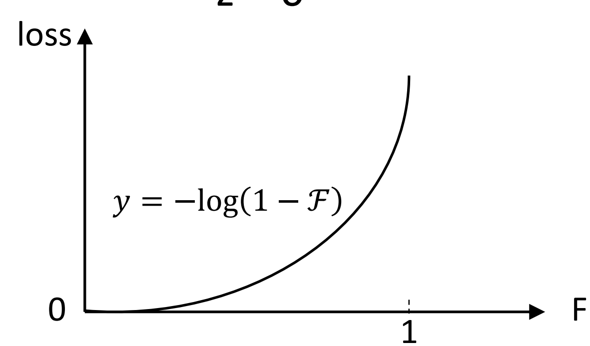
- $loss = \begin{cases} -\log(\mathcal{F}(x, W)) & \text{if z} = 1 \\ -\log\left(1 - \mathcal{F}(x, W)\right) & \text{if z} = 0 \end{cases}$ → convex

Negative log loss

z = 1



z = 0



Loss function

$$\mathcal{L}(W) = \begin{cases} -\log(\mathcal{F}(x, W)) & \text{if z} = 1 \\ -\log\big(1 - \mathcal{F}(x, W)\big) & \text{if z} = 0 \end{cases}$$

$$= -\text{z}\log\big(\mathcal{F}(x, W)\big) - (1 - z)\log(1 - \mathcal{F}(x, W))$$

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial \mathcal{F}} \frac{\partial \mathcal{F}}{\partial W}$$

- $\frac{\partial \mathcal{L}}{\partial \mathcal{F}} = \frac{-z}{\mathcal{F}} + \frac{1-z}{1-\mathcal{F}}$
- $\frac{\partial \mathcal{F}}{\partial W} = x\left(\frac{1}{1+e^{-Wx}} - \frac{1}{(1+e^{-Wx})^2}\right) = x\mathcal{F}(1 - \mathcal{F})$

$$\boxed{\frac{\partial \mathcal{L}}{\partial W} = x(\mathcal{F} - z)}$$

Loss function: matrix form

- Per sample
  - $\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, W) = \frac{1}{1+e^{-W\boldsymbol{x}}}$
  - $\mathcal{L} = -\boldsymbol{z}\log(\mathcal{F}) - (1-\boldsymbol{z})\log(1-\mathcal{F})$
  - $\frac{\partial \mathcal{L}}{\partial W} = \boldsymbol{x}(\mathcal{F} - \boldsymbol{z})$
- Over the whole training set
  - $Y = \mathcal{F}(\boldsymbol{X}, W) = \frac{1}{1+e^{-W\boldsymbol{X}}}$
  - $\mathcal{L} = \frac{-1}{N}[\log(\mathcal{F})\boldsymbol{Z}^T + \log(1-\mathcal{F})(1-\boldsymbol{Z})^T]$
  - $\frac{\partial \mathcal{L}}{\partial W} = \frac{1}{N}(\mathcal{F} - \boldsymbol{Z})\boldsymbol{X}^T$ → convex

Optimisation

- $\frac{\partial \mathcal{L}}{\partial W} = \frac{1}{N}(\mathcal{F} - \boldsymbol{Z})\boldsymbol{X}^T$

- Gradient descent

$$W \coloneqq W - k\frac{\partial \mathcal{L}}{\partial W}$$

- Gradient descent with momentum

$$v_t = mv_{t-1} + k\frac{\partial \mathcal{L}}{\partial W}$$

$$W \coloneqq W - v_t$$

Binary classification summary

Training set

- $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(N)}\}$
- $\boldsymbol{Z} = \{\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \dots, \boldsymbol{z}^{(N)}\}$

Hypothesis function

- $y = \mathcal{F}(x, W) = \frac{1}{1+e^{-Wx}}$

Loss function

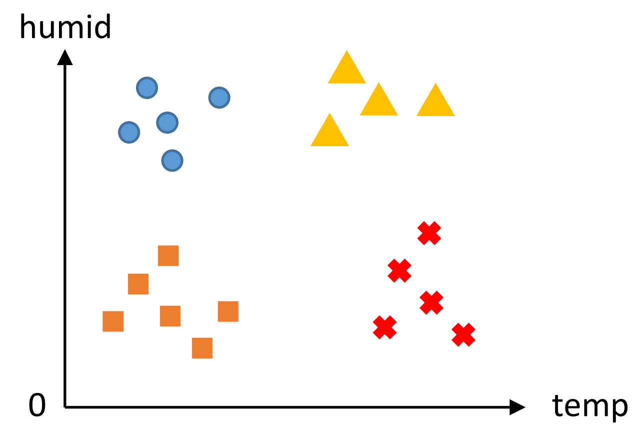
- $\mathcal{L} = \frac{-1}{N}[\log(\mathcal{F})\, Z^T +$

Optimisation

- Gradient descent

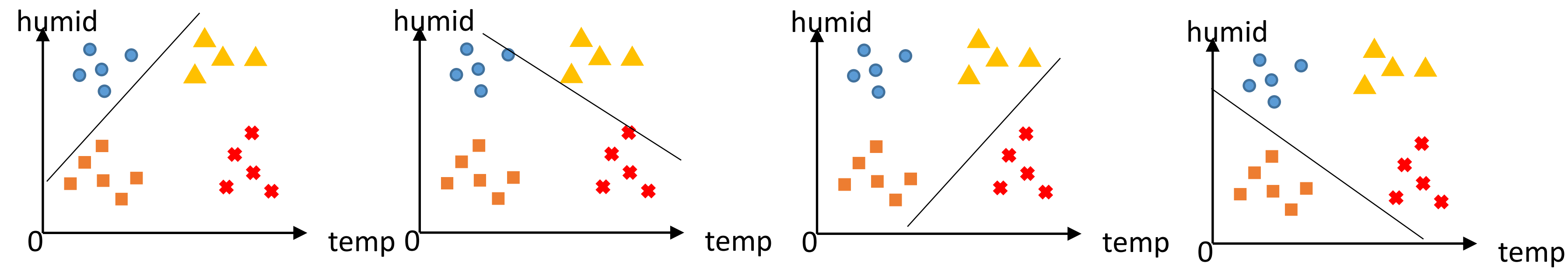
Multi-class classification

- The weather classification example:
  - Input: $x = \begin{bmatrix} \text{temperature} \\ \text{humidity} \\ 1 \end{bmatrix}$
  - Output: sunny (0), cloudy (1), rain (2) or snow (3).
  - $y \in \{0, 1, 2, 3\}$



Method #1: one-vs-all

- For n-class classification, train n binary classifiers $\mathcal{F}_i$, each distinguish one class from the rest.



$\mathcal{F}_0(\boldsymbol{x}) = P(y = 0|\boldsymbol{x}, W_0)$ $\quad$ $\mathcal{F}_0(\boldsymbol{x}) = P(y = 1|\boldsymbol{x}, W_1)$ $\quad$ $\mathcal{F}_0(\boldsymbol{x}) = P(y = 2|\boldsymbol{x}, W_2)$ $\quad$ $\mathcal{F}_0(\boldsymbol{x}) = P(y = 3|\boldsymbol{x}, W_3)$

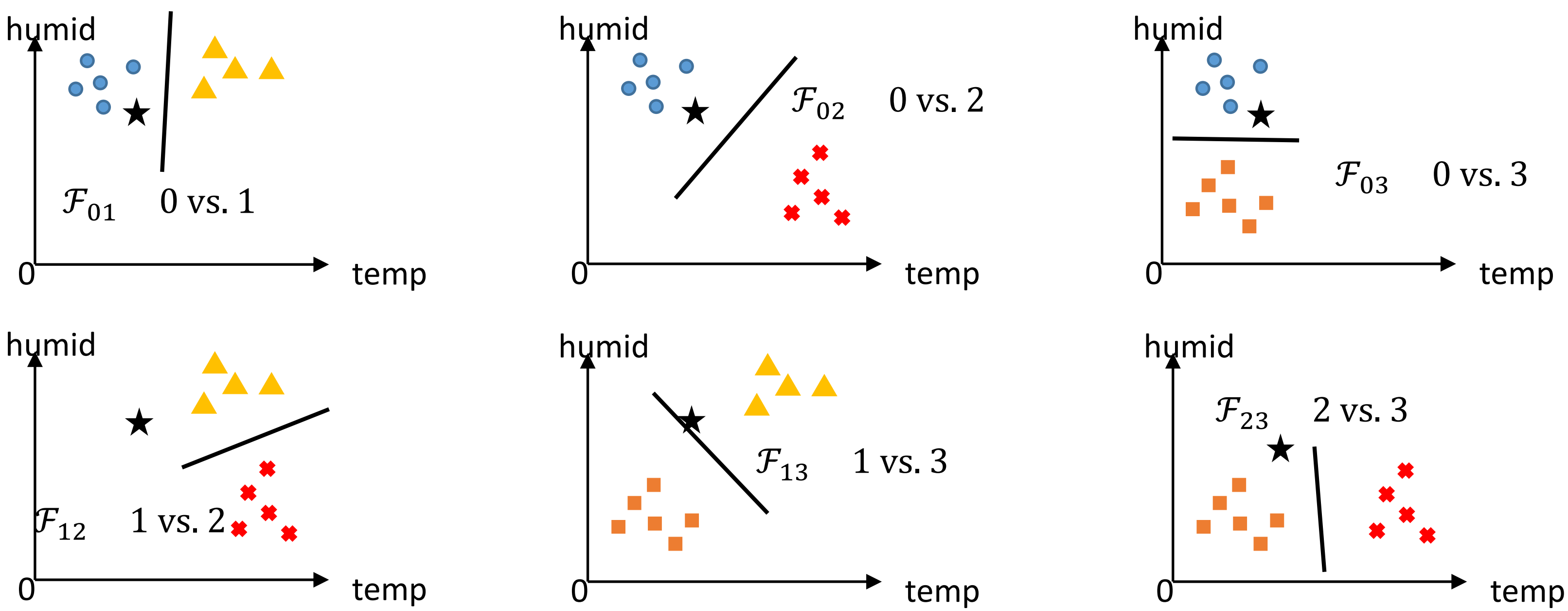
Class 0: $\quad$ Class 1: $\quad$ Class 2: $\quad$ Class 3:

- Unknown sample **x** is classified into class i :

$$\max_i \mathcal{F}_i(\boldsymbol{x}, W_i)$$

Method #2: one-vs-one

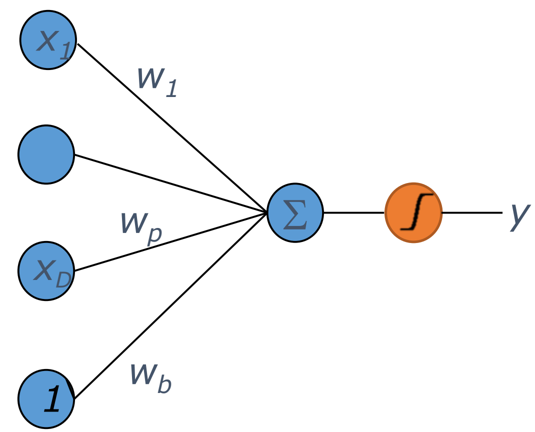
- For n-class classification, train $n(n-1)/2$ binary classifiers $\mathcal{F}_{ij}$, each distinguishes class i and class j.



- Unknown sample **x** belongs to class i if i gets the highest #predictions in $n(n-1)/2$ classifiers.

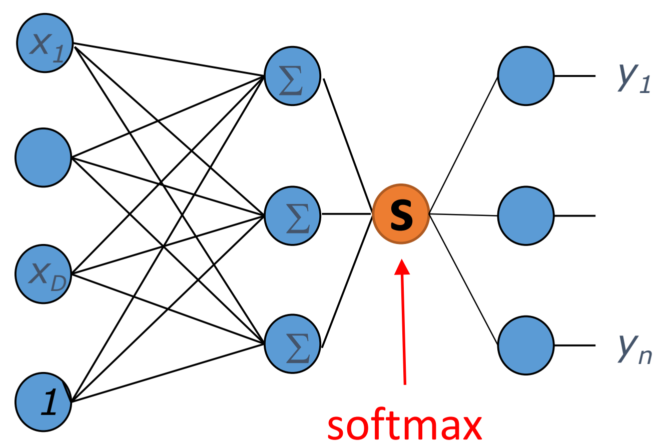
Method #3: softmax

- Binary classification
  - $x \in \mathbb{R}^{D+1}; W \in \mathbb{R}^{D+1};\ y, z \in \mathbb{R}$
  - $y = \mathcal{F}(x, W) = g(W\boldsymbol{x})$



- Multi-class classification
  - $x \in \mathbb{R}^{D+1}; W \in \mathbb{R}^{n \times (D+1)};\ y, z \in \mathbb{R}^n$
  - n number of classes
  - $y = \mathcal{F}(x, W) = h(W\boldsymbol{x})$

$$z = \begin{bmatrix} \text{sunny} \\ \text{cloudy} \\ \text{rain} \\ \text{snow} \end{bmatrix} \qquad \text{e.g. rain } \boldsymbol{z} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$



Classification with softmax

- Softmax function

$$t = \begin{bmatrix} t_1 \\ t_2 \\ \dots \\ t_n \end{bmatrix} \longrightarrow \boxed{\boldsymbol{h}(\boldsymbol{t})} \longrightarrow y = \begin{bmatrix} \frac{e^{t_1}}{\sum_i e^{t_i}} \\ \frac{e^{t_2}}{\sum_i e^{t_i}} \\ \dots \\ \frac{e^{t_n}}{\sum_i e^{t_i}} \end{bmatrix} = \begin{bmatrix} P(y=1|x,W) \\ P(y=2|x,W) \\ \dots \\ P(y=n|x,W) \end{bmatrix}$$

- Loss function: negative log loss for class c

$$\mathcal{L}_c = -\log(P(y=c|x,W))$$

Machine learning in practice

- Preprocessing data: feature normalisation

House size (feet²)	#bedrooms	House age (years)	Distance from city centre (km)
1240	4	25	1.5
370	1	40	20.1
1130	3	5	13.0
1120	2	60	100.5
1710	4	13	30.7
…	…	…	…
860	2	8	46.4

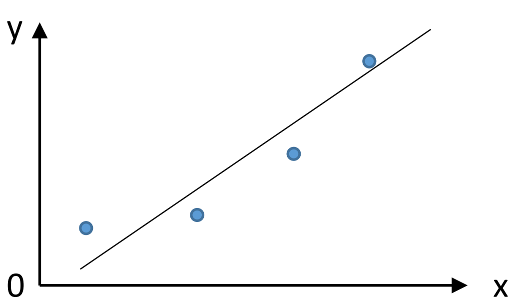
$$\boldsymbol{x} = \begin{bmatrix} \text{house size} \\ \text{bedrooms} \\ \text{age} \\ \text{distance} \\ 1 \end{bmatrix}$$

$x_1 \in$ 200-5000 $x_2 \in$ 1-5 $x_3 \in$ 1-100 $x_4 \in$ 1-200

Normalise each feature $$x \coloneqq \frac{x - \bar{x}}{\max x - \min x} \in [-0.5, 0.5]$$

Machine learning in practice

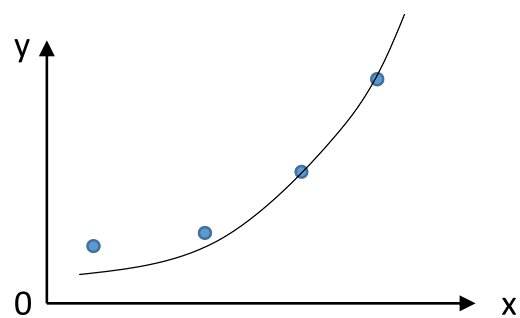
- The overfitting problem
  - Work well on training data, poor performance on test data.
  - Incorrect selection of the hypothesis function can be a source of the problem
  - Often occurs when there are many features, but not enough training samples -> more parameters to learn.



$$y = ax + b$$

W $= [a \;\; b]$

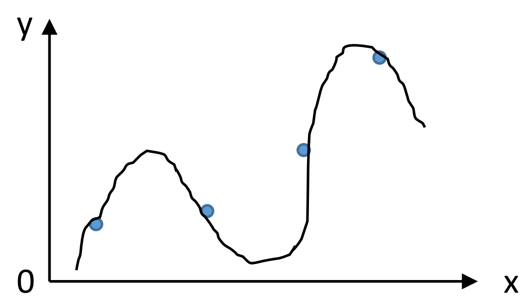
Under-fitting



$$y = ax^2 + bx + c$$

W $= [a \;\; b \; c]$

good



$$y = a_5x^5 + a_4x^4 + \cdots + a_0$$

W $= [a_5 \; a_4 \;\; \ldots a_0]$

over-fitting

Machine learning in practice

- Combat over-fitting using regularisation
  - Keep all features but reduce the magnitude of W.
  - Implemented by adding |W| into the loss function

regression

$$\mathcal{L}(W) = \frac{1}{2N}\sum_{i=1}^{N}\left(\mathcal{F}\left(\boldsymbol{x}^{(i)}, W\right) - \boldsymbol{z}^{(i)}\right)^2 + \frac{1}{2}|W|^2$$
$$= \frac{1}{2N}|WX - Z|^2 + \frac{1}{2}|W|^2$$

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{1}{N}\sum \boldsymbol{x}(\mathcal{F} - \boldsymbol{z}) + W$$
$$= \frac{1}{N}(W\boldsymbol{X} - \boldsymbol{Z})\boldsymbol{X}^T + \mathrm{W}$$

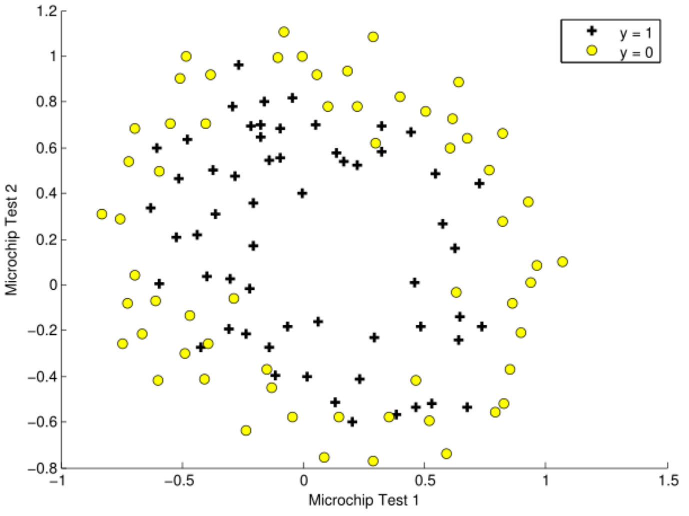
classification

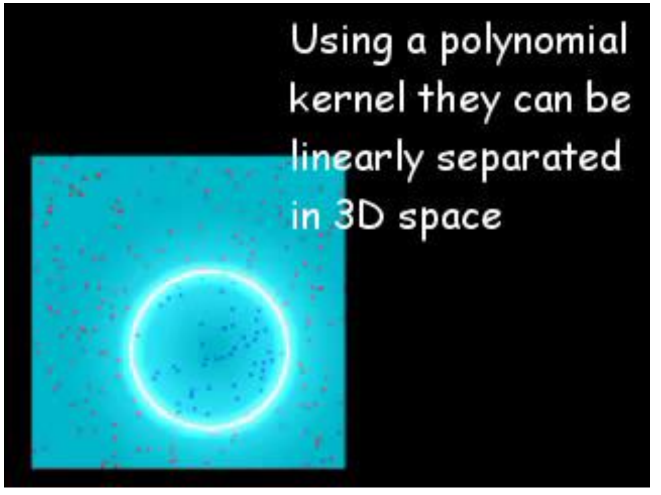
$$\mathcal{L}(W) = \frac{-1}{N}\sum[\boldsymbol{z}\log(\mathcal{F}) + (1 - \boldsymbol{z})\log(1 - \mathcal{F})] + \frac{1}{2}|W|^2$$
$$= \frac{-1}{N}[\log(\mathcal{F})\boldsymbol{Z}^T + \log(1 - \mathcal{F})(1 - \boldsymbol{Z})^T] + \frac{1}{2}|W|^2$$

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{1}{N}\sum \boldsymbol{x}(\mathcal{F} - \boldsymbol{z}) + W$$
$$= \frac{1}{N}(W\boldsymbol{X} - \boldsymbol{Z})\boldsymbol{X}^T + \mathrm{W}$$

Advanced ML techniques

- Support vector machine (SVM)

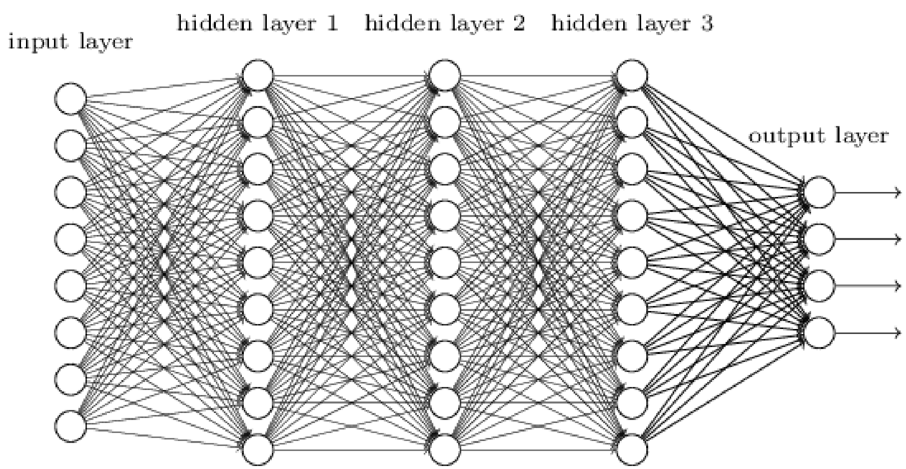




Courtesy: youtube

Advanced ML techniques

- Deep neural networks (DNN)
  - Multi-layer neurons, each consists of a linear operation (convolution) followed by a non-linear one (activation).
  - Huge number of parameters (millions-D).
  - Compute $\partial\mathcal{L}/\partial W$ based on chain rule and backpropagation
  $$\frac{\partial}{\partial W} f\left(g\left(h(\ldots k(W))\right)\right) = \frac{\partial f}{\partial g}\frac{\partial g}{\partial h}\cdots\frac{\partial k}{\partial W}$$
  - Currently state-of-the-art in many applications



Conclusion: supervised learning procedure

Training set

- $\boldsymbol{X} = \left\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}\right\}$
- $\boldsymbol{Z} = \left\{\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \ldots, \boldsymbol{z}^{(N)}\right\}$

Hypothesis function

- $y = \mathcal{F}(x, W)$

Loss function

- $\mathcal{L}(W) = \frac{1}{2N} \sum_{i=1}^{N} g\left(y^{(i)}, z^{(i)}\right)$

Optimisation

- $\min_W \mathcal{L}(W)$

Acknowledgement

This presentations use materials from the following sources:

- Machine Learning Intro – UDRC’17 by Josef Kittler – University of Surrey
- Deep Learning Tutorials – UDRC’17 by Muhammad Awais – University of Surrey
- Machine Learning – coursera.org by Andrew Ng – Standford University
- Neural Network reviews – MLSS’14 by Quoc Le – Google Brain

Thank you.