

# Daily Inspiration

---



**It's better to be a real  
goofball than a real  
spouse**

**I'm InspiroBot.**

I am an artificial intelligence dedicated to generating unlimited amounts of unique inspirational quotes for endless enrichment of pointless human existence.



# Today

---

- Intro
- Linear regression, gradient descent
- Bias and variance, cross-validation, learning curves
- Linear algebra primer



# Who am I?

---

- Some random guy off the streets
- Studied Biology, then Molecular and Cellular Life Sciences with Bioinformatics profile, doing mainly ML. Homegrown at UU.

# Who am I?

- Some random guy off the streets
- Studied Biology, then Molecular and Cellular Life Sciences with Bioinformatics profile, doing mainly ML. Homegrown at UU.
- Hidden talent:

~~SPECIAL TALENT: Can cook Minute rice in 58 seconds~~

Can sing Tom Lehrer's Elements Song (used to, anyway)

- I can also tie my own shoes →





# Why am I teaching this?

---

- *I don't know man, they told me to so I did it!*

# Why am I teaching this?

---

- ~~I don't know man, they told me to so I did it!~~
- Are you good at ML?
  - Like *good* good? No.
- Do you know more about ML than us?
  - *Maybe?* I sure hope so.
- This doesn't help your credibility, you know
  - Sorry guys.

# Real MVPs

---



Andrew Ng,  
Coursera ML guru



Jeroen de Ridder,  
resident ML  
maestro UMCU



Dieter Stoker,  
Some rando off  
the streets



# Course content: what I hope to teach you

---

- What is ML? Cost functions, gradient descent, generalisation, bias and variance.
- Week 1: low-level understanding: able to implement linear regression, logistic regression, neural networks, clustering and PCA yourself using numpy in Python.
- Week 2: modern ML library (scikit-learn) workflow (one day), + a hands-on project (~2 days). Written exam about lecture and practical concepts at the end.



# Setup per day

- Morning/early afternoon:
  - Lectures of ~45-60 minutes, interspersed with (2) short practical(s).
- Rest of the day:
  - Somewhat longer afternoon practical
- Taken together:
  - Lecture
  - Short practical 1
  - Lecture
  - Short practical 2
  - Lecture
  - Afternoon practical

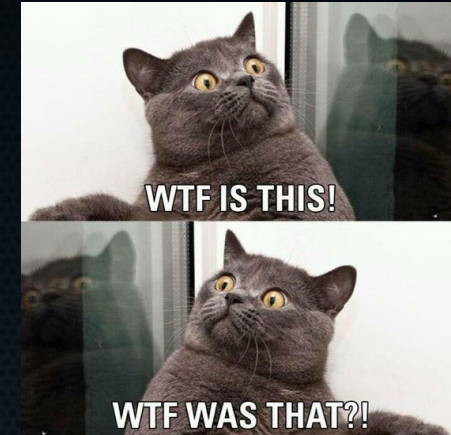


Lunch somewhere here  
(12:15-13:00 is the idea)  
Might shift times a bit.

# I need your help

---

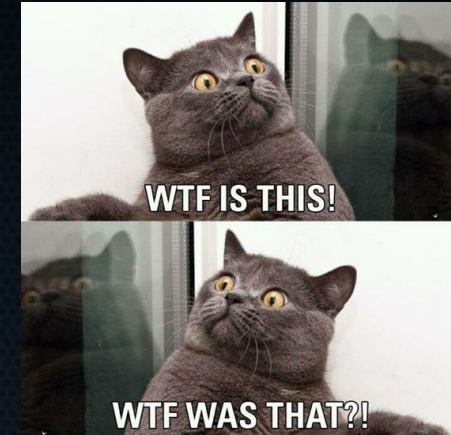
- This is a new course. So probably, you'll encounter difficulty spikes, things that don't make sense, or other things that are lacklustre.
- At the end of each practical I ask you to anonymously rate it and give comments.
- In this way, I can hopefully take things on board quickly and perhaps change practicals or lectures during the course, rather than only after!





# I need your help

- This is a new course. So probably, you'll encounter difficulty spikes, things that don't make sense, or other things that are lacklustre.
- This also means that we are going to discover together how much is reasonable to do: if there's far too much material, say so, and we can scrap some!



# I need your help 2

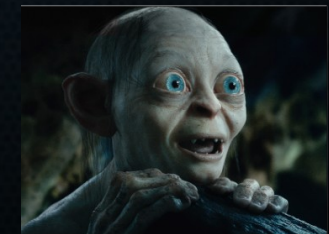
- You might wonder what's up with the coloured baubles I blessed you with.





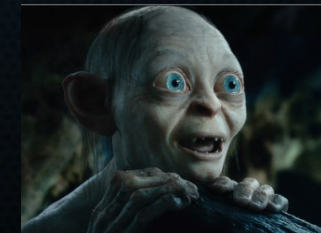
# I need your help 2

- You might wonder what's up with the coloured baubles I blessed you with.
- They're mood indicators for during the lecture:
  - Green: „I am positively brimming with enthusiasm to learn“ and/or „I can follow this material well enough“
  - Yellow/Orange: „This is somewhat difficult“ and/or „I feel my attention is slipping and I can't absorb the information so well anymore“
  - Red: „MAKE IT STOP! PLEASE, PLEASE MAKE IT STOP!!!“



# I need your help 2

- Put the bauble that matches your mood at the front of your table.
- I tend to make lectures a bit too long. In this way, I can *notice* that's happening and stop/address it, without you having to tell me to shut up. Win-win!





# Questions

---

- Besides this, feel free to raise your hand and ask questions when something is unclear.
- If there's no hands raised right now then we'll dive right in!

# This presentation

---

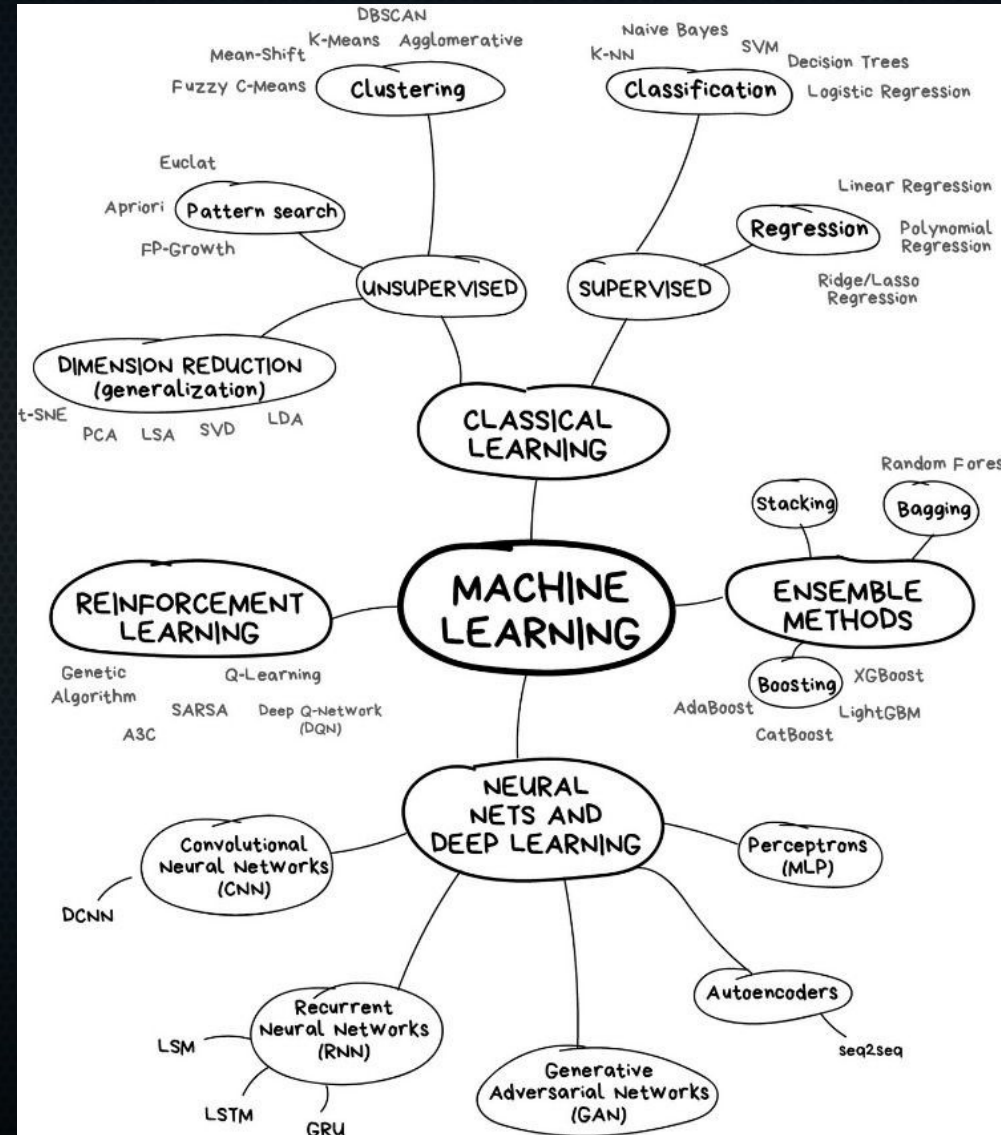
- Two branches of ML: supervised and unsupervised
- Terminology
- Linear regression & cost function
- Gradient descent & partial derivatives



# The ugly truth about ML

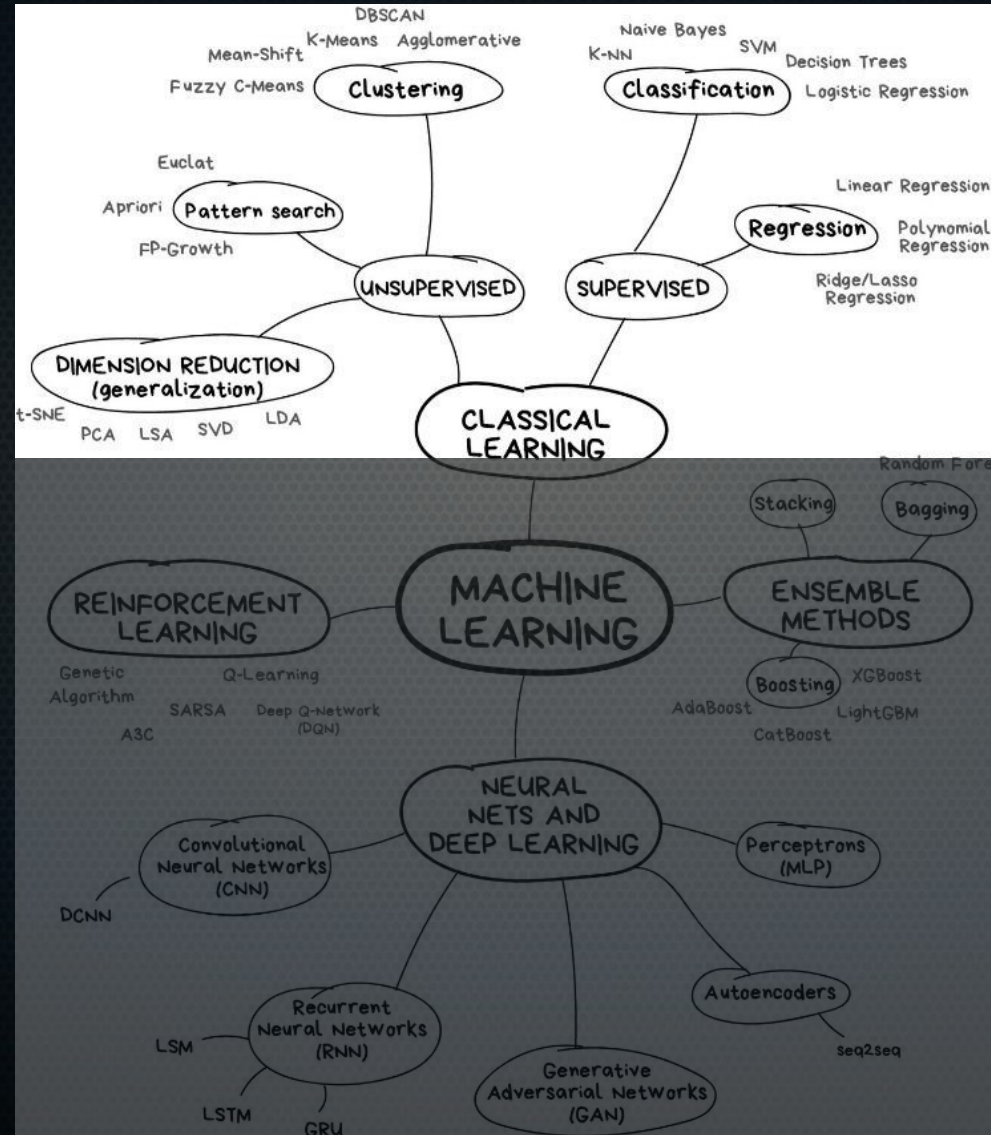


# Map of Machine Learning

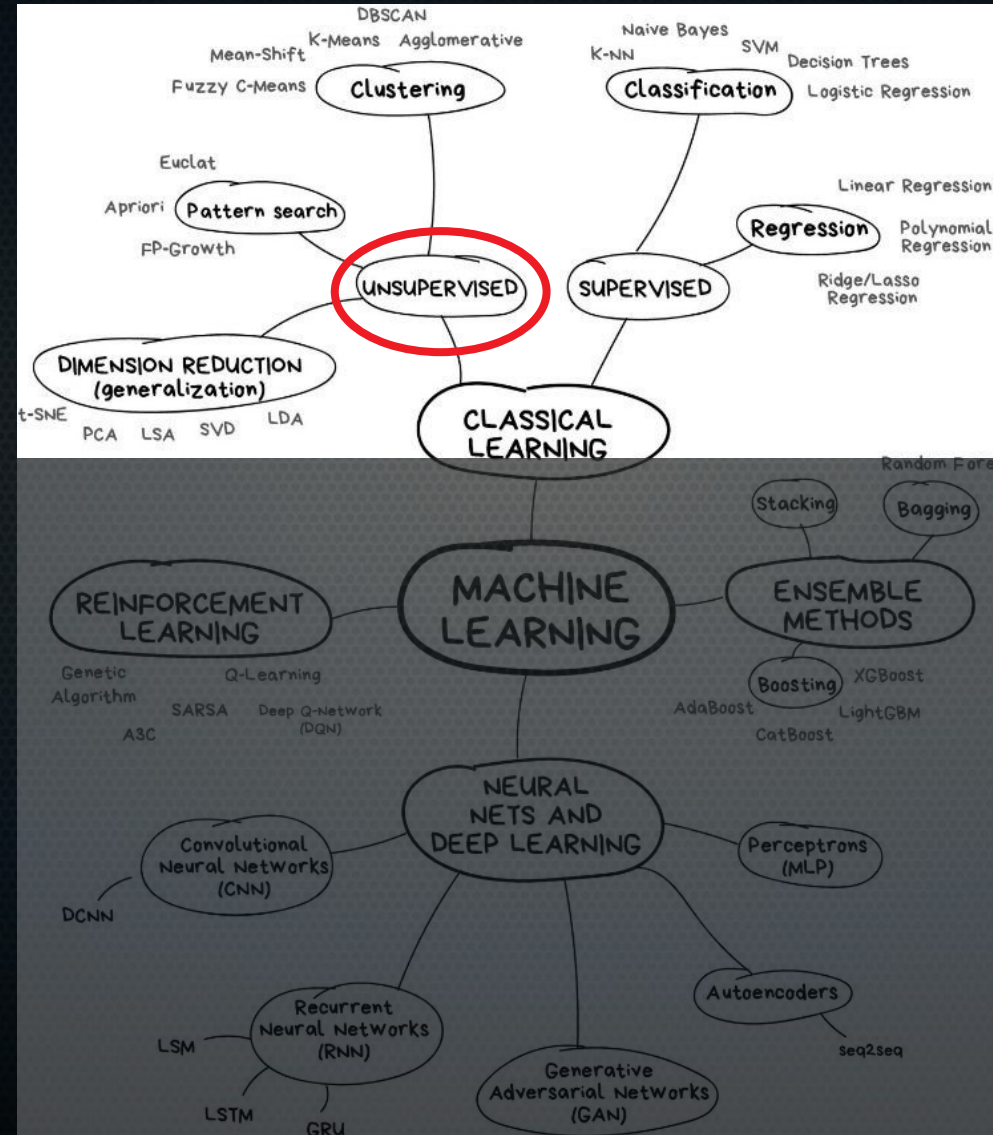




# Map of Machine Learning

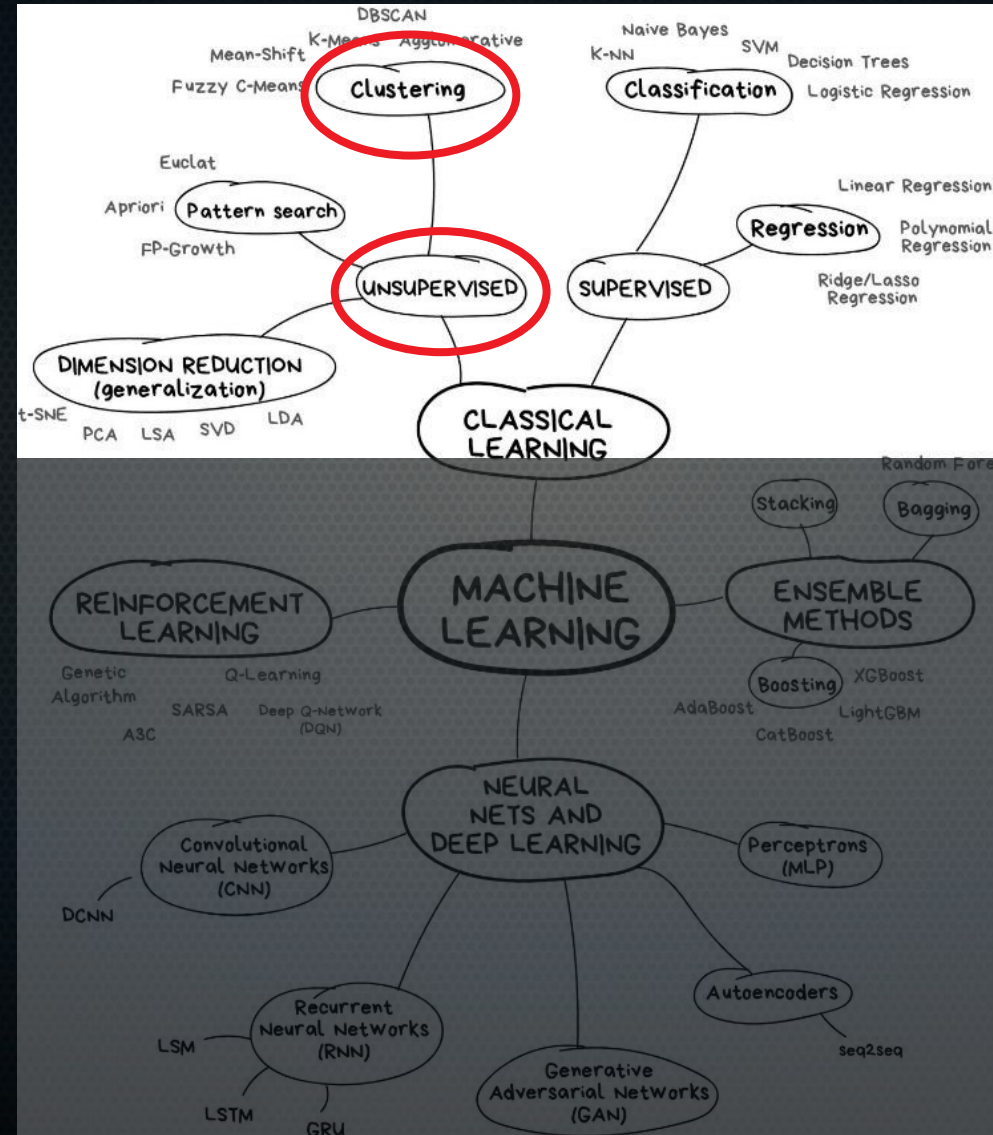


# Map of Machine Learning



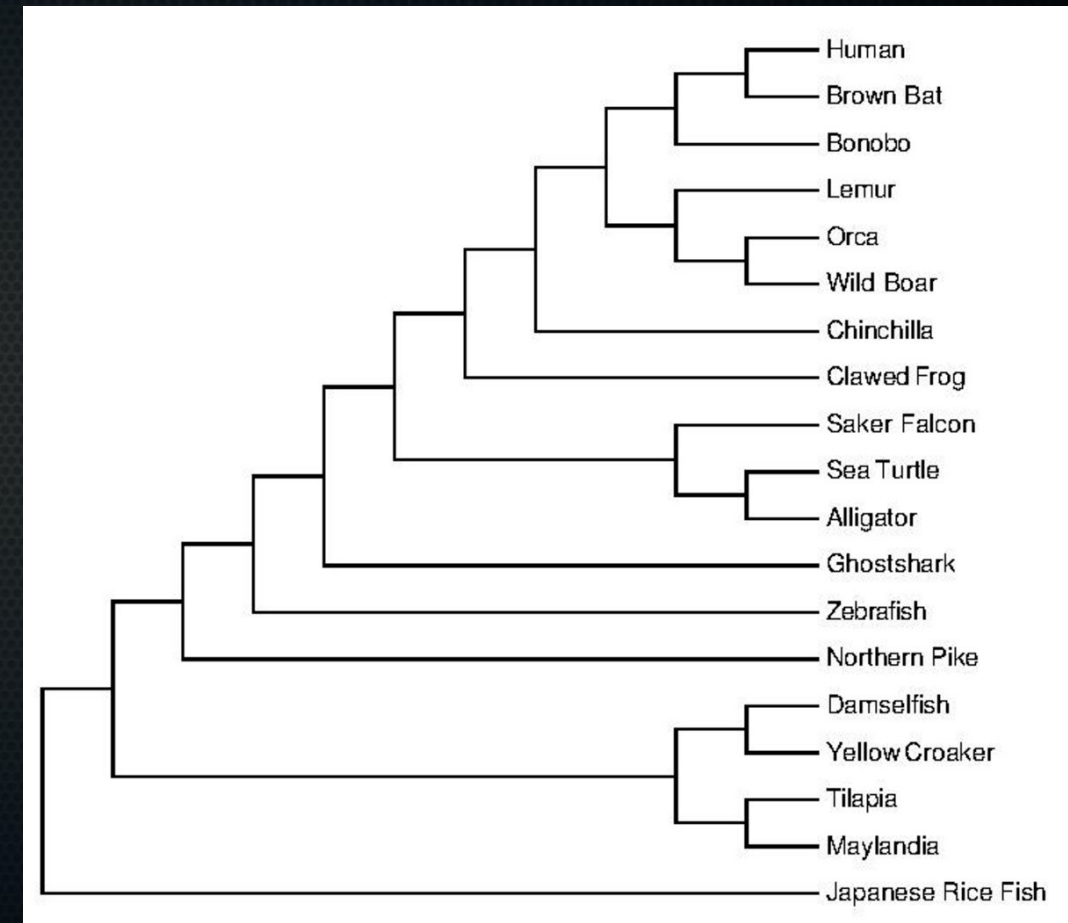


# Map of Machine Learning



# Unsupervised learning: clustering

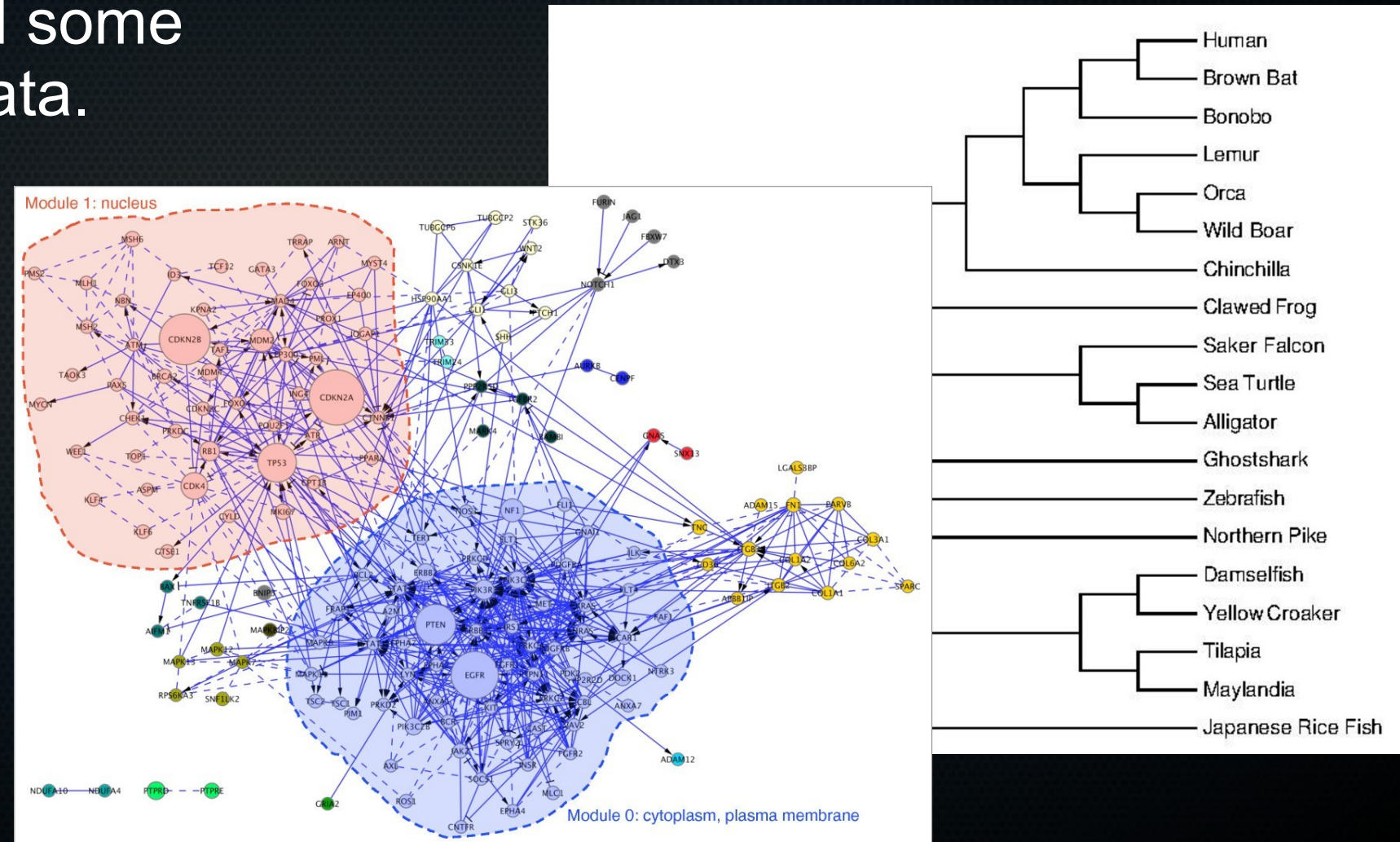
- Automatically find some structure in the data.





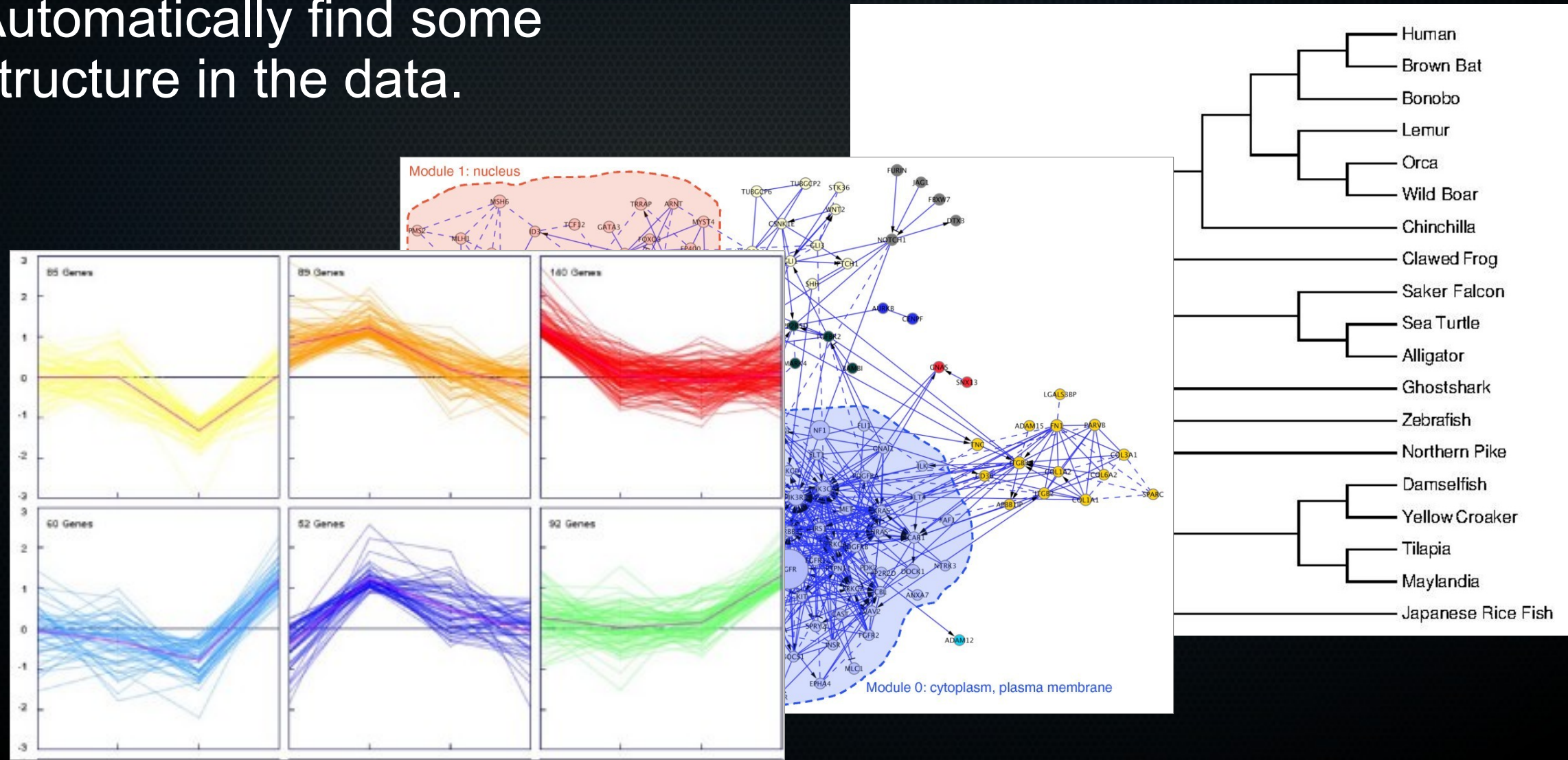
# Unsupervised learning: clustering

- Automatically find some structure in the data.



# Unsupervised learning: clustering

- Automatically find some structure in the data.



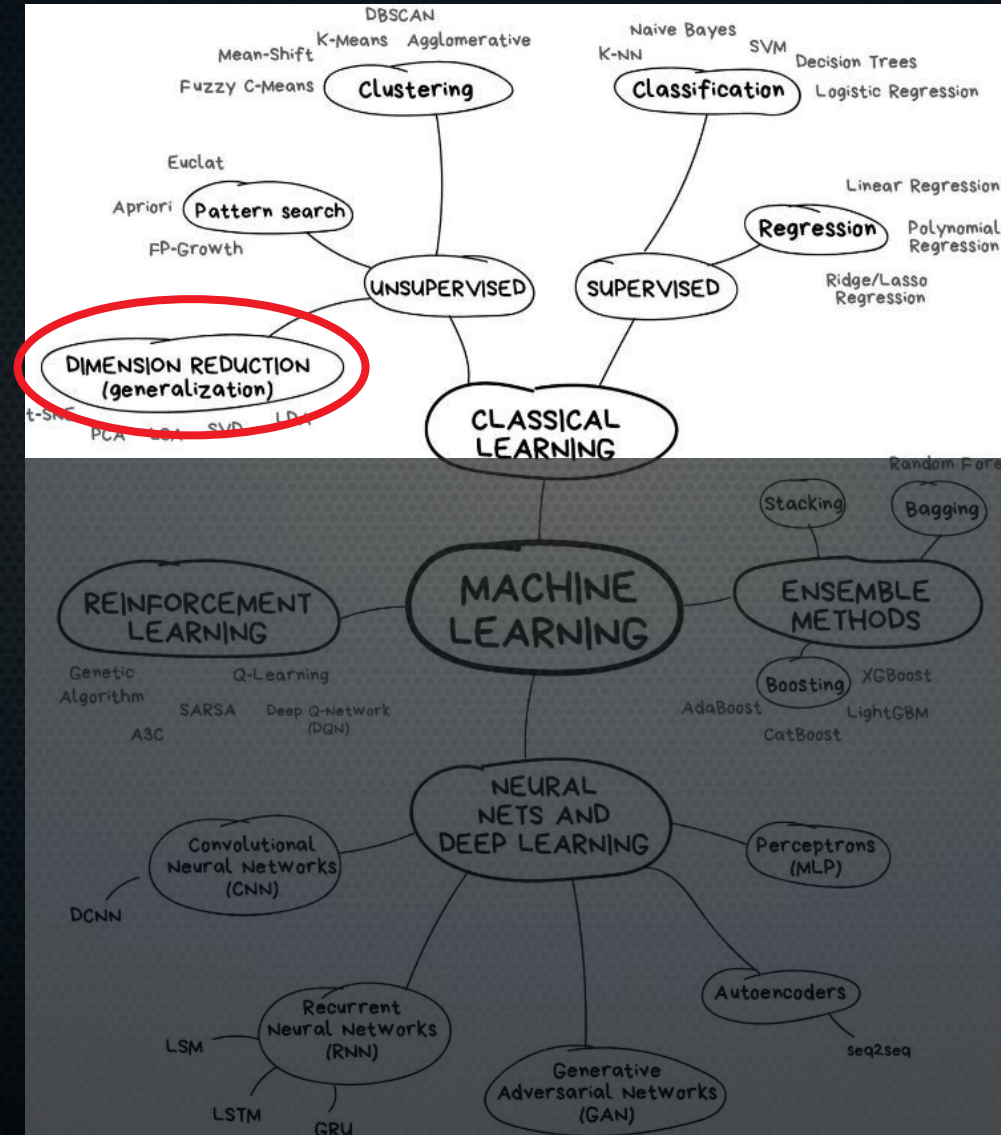


# Unsupervised learning: clustering

---

- No right or wrong:
  - Back-and-forth between different clustering algorithms, your knowledge, and the data.
  - You don't *know* correct clustering.

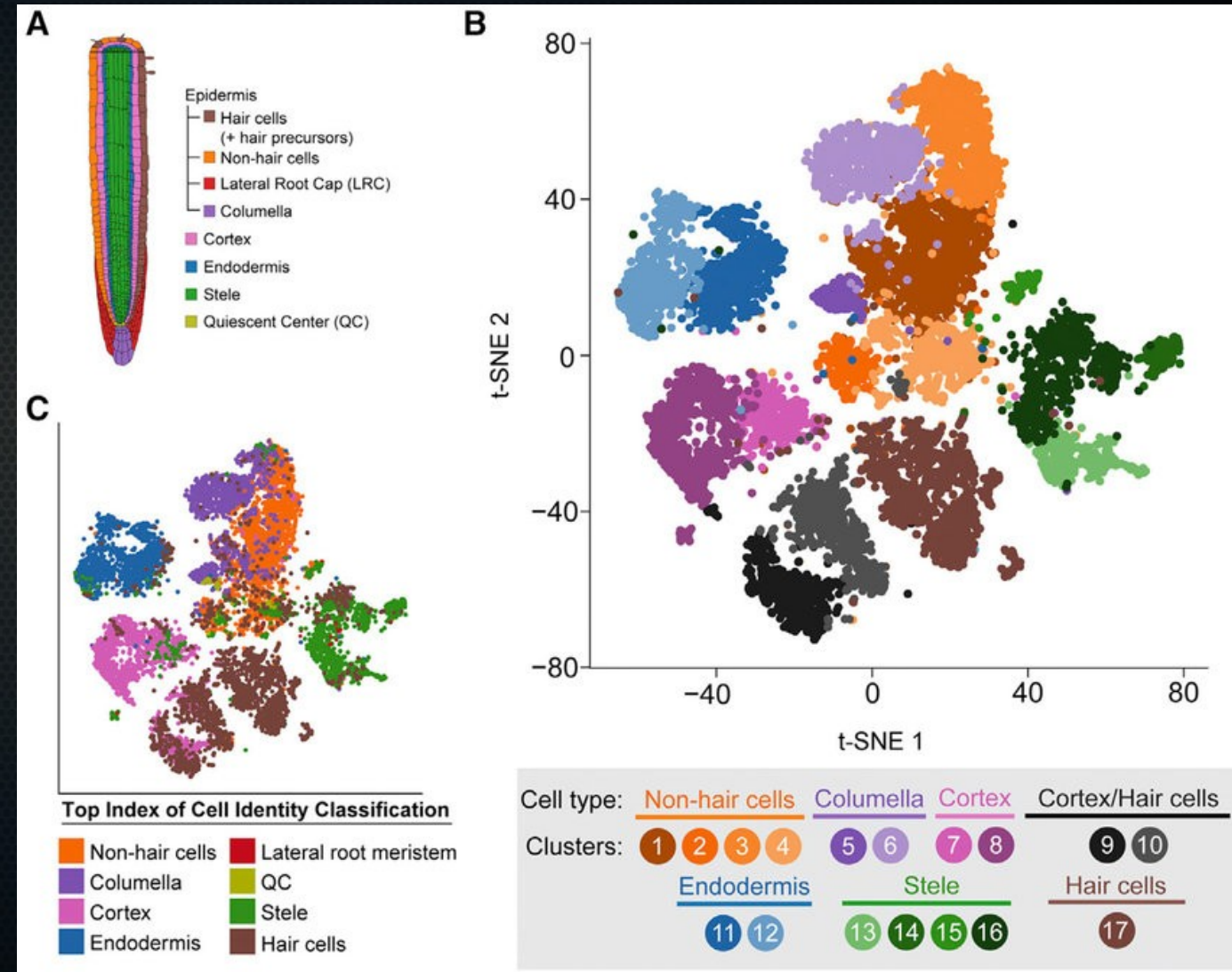
# Map of machine learning





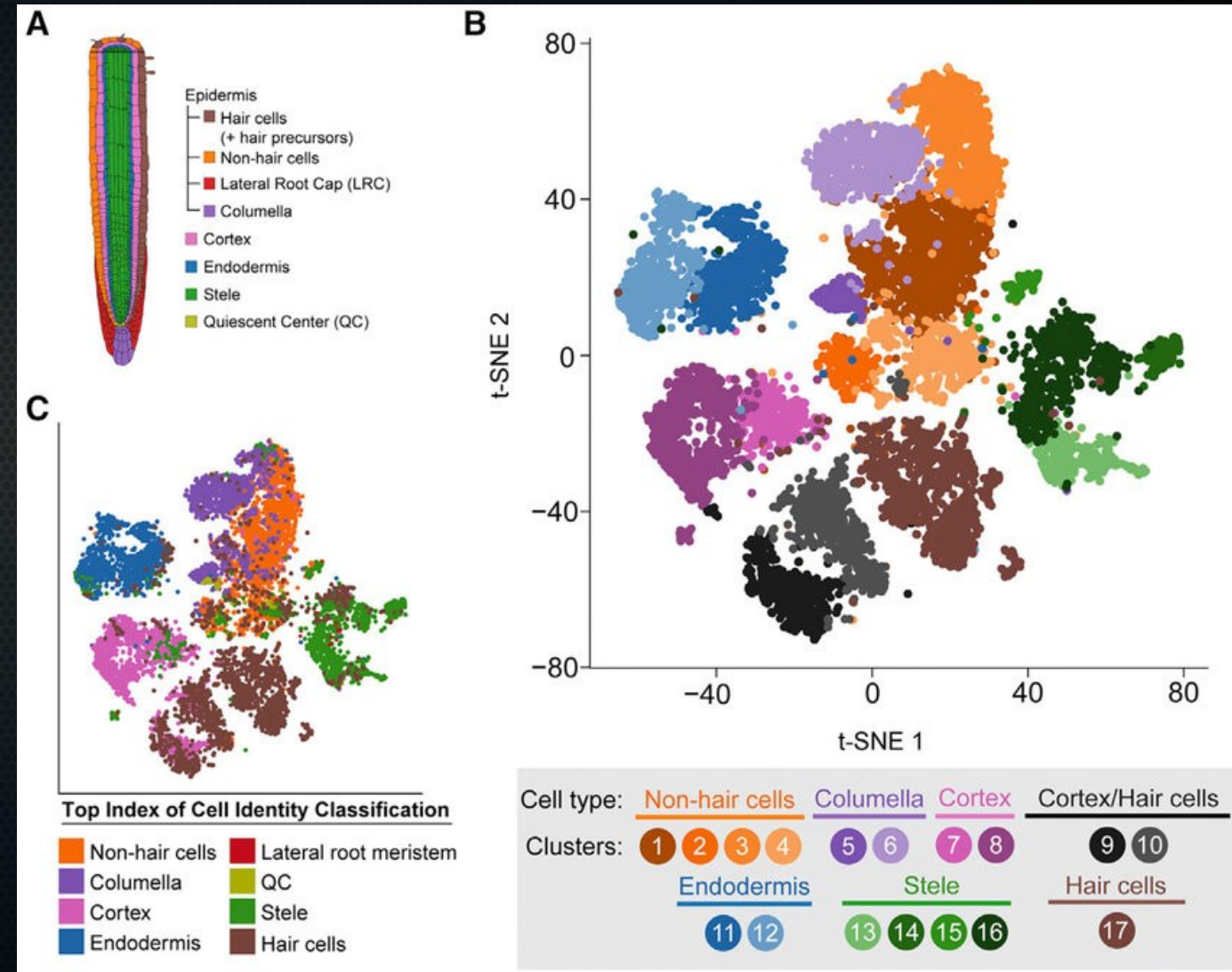
# Unsupervised learning: dimensionality reduction

- Single-cell RNAseq of 12,198 Arabidopsis root cells.
- How do they differ?



# Unsupervised learning: dimensionality reduction

- Single-cell RNAseq of 12,198 Arabidopsis root cells.
- How do they differ?
- Visualise differences in all RNAs between all these cells in 2 dimensions.
- Used in conjunction with clustering (colours)



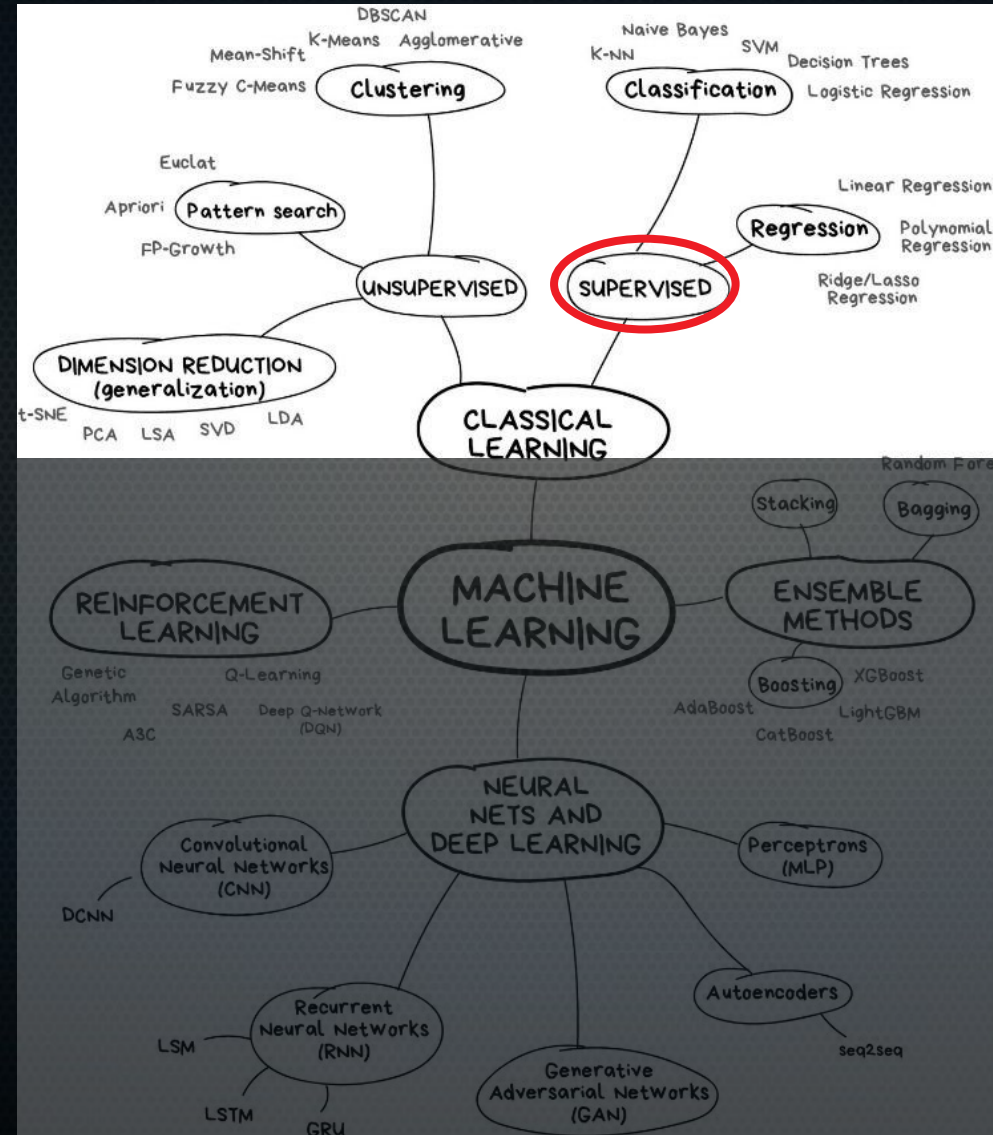


# Unsupervised learning: dimensionality reduction

---

- Used for:
  - Visualisation (our visual systems cannot deal with  $> 3D$ )
  - Compressing data (capture 90% of the variation with much less data, say)
  - Preventing overfitting and other ill effects of high dimensionality

# Map of Machine Learning





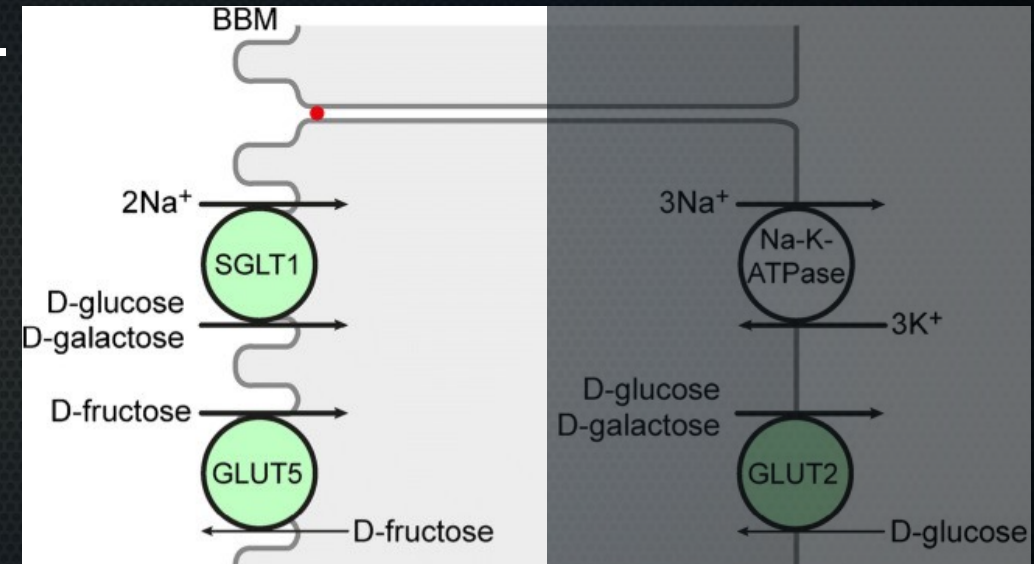
# Supervised learning

---

- Given known examples, automatically find a function to map new examples.

# Supervised learning: regression

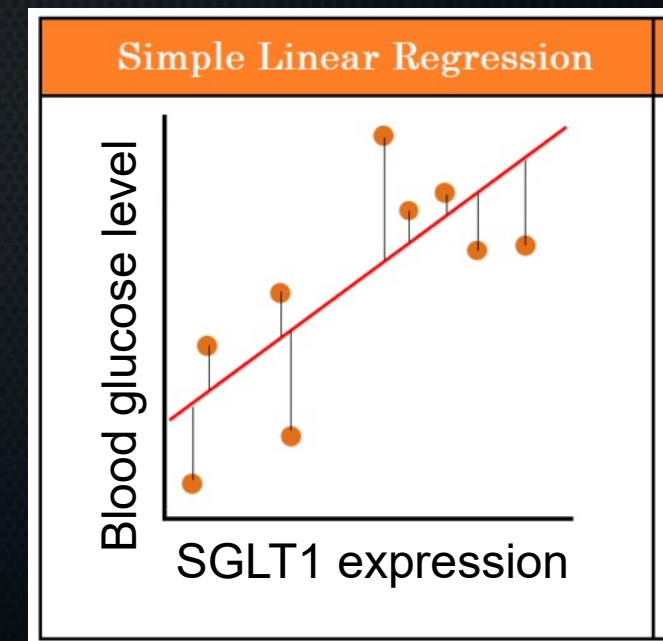
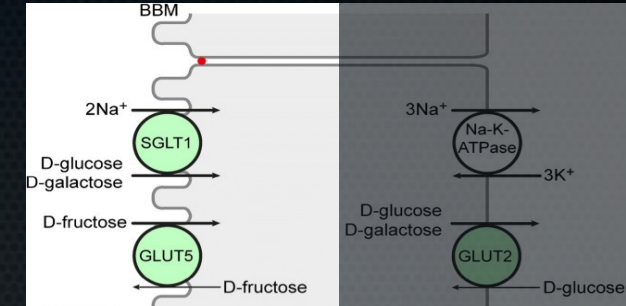
- Given known examples, automatically find a function to map new examples.
- Real-valued outputs.





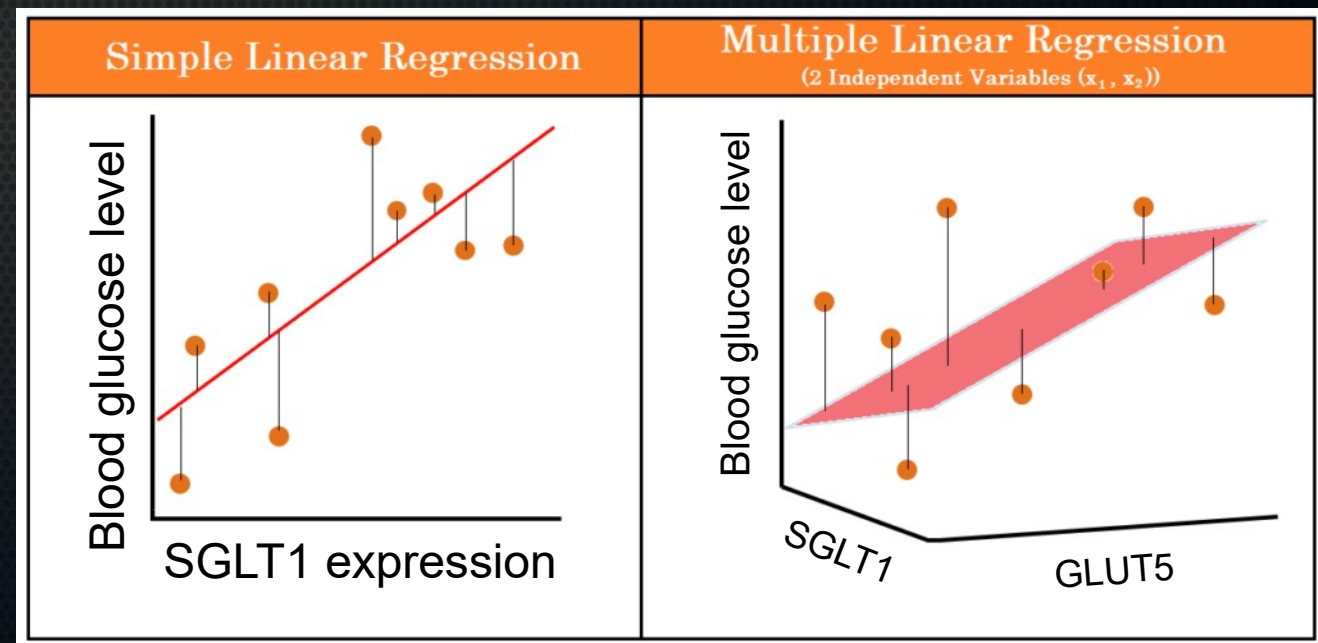
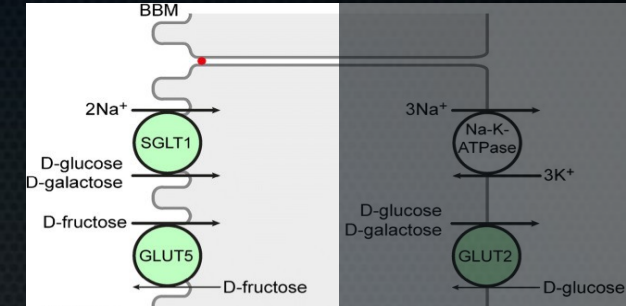
# Supervised learning: regression

- Given known examples, automatically find a function to map new examples.
- Real-valued outputs.



# Supervised learning: regression

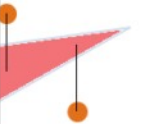
- Given known examples, automatically find a function to map new examples.
- Real-valued outputs.





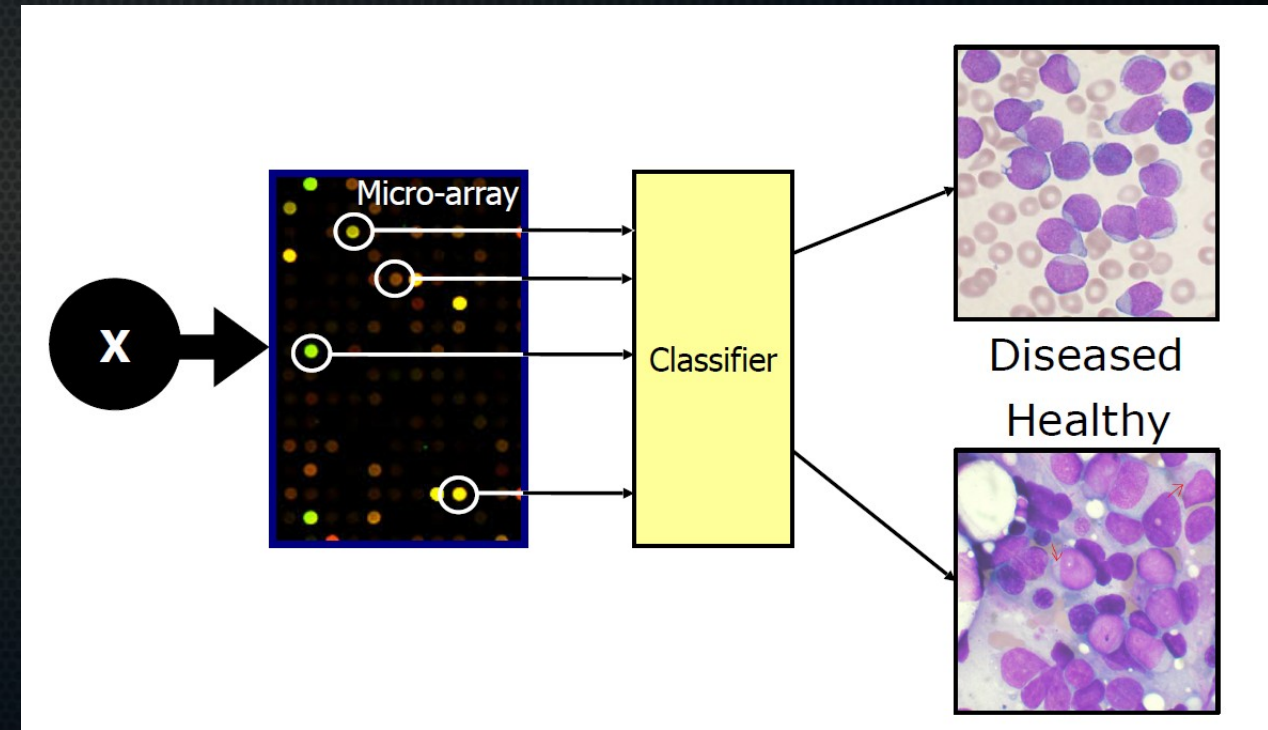
# Supervised learning: regression

- Given known examples, automatically find a function to map new examples.
- Real-valued outputs: regression.

	Simple Linear Regression	Multiple Linear Regression (2 Independent Variables ( $x_1, x_2$ ))
Simple Linear Regression	$y = b_0 + b_1x_1$	
Multiple Linear Regression	$y = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n$	
Polynomial Linear Regression	$y = b_0 + b_1x_1 + b_2x_1^2 + \dots + b_nx_1^n$	

# Supervised learning: classification

- Given known examples, automatically find a function to map new examples.
- Discrete outputs.

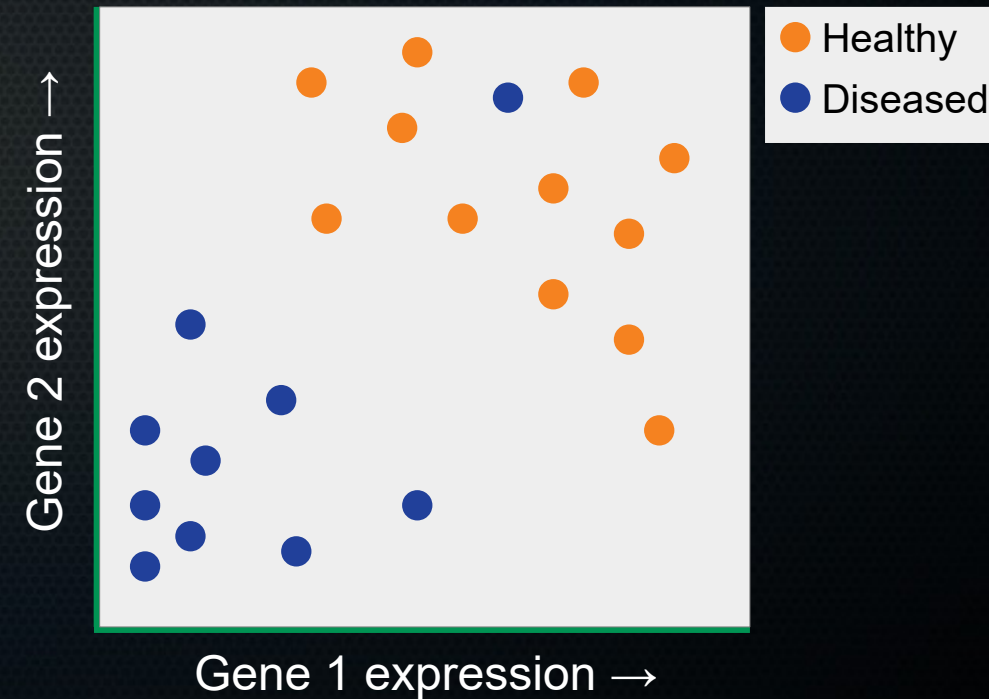


Source: Jeroen de Ridder



# Supervised learning: classification

- Given known examples, automatically find a function to map new examples.
- Discrete outputs.



# Supervised learning: classification

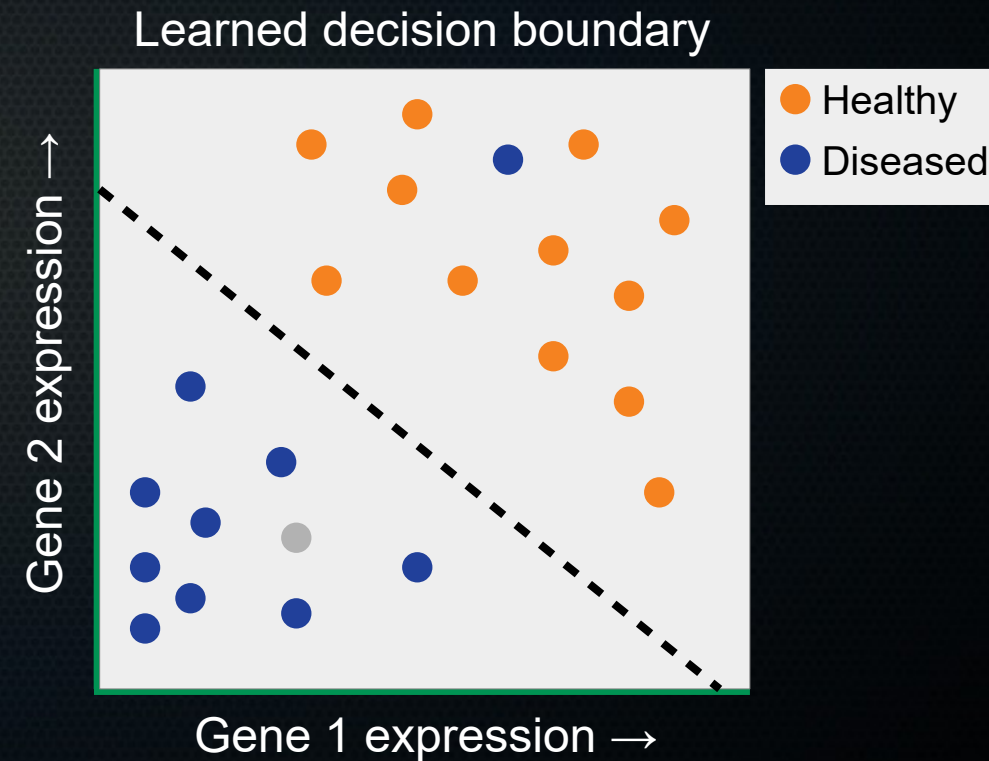
- Given known examples, automatically find a function to map new examples.
- Discrete outputs.





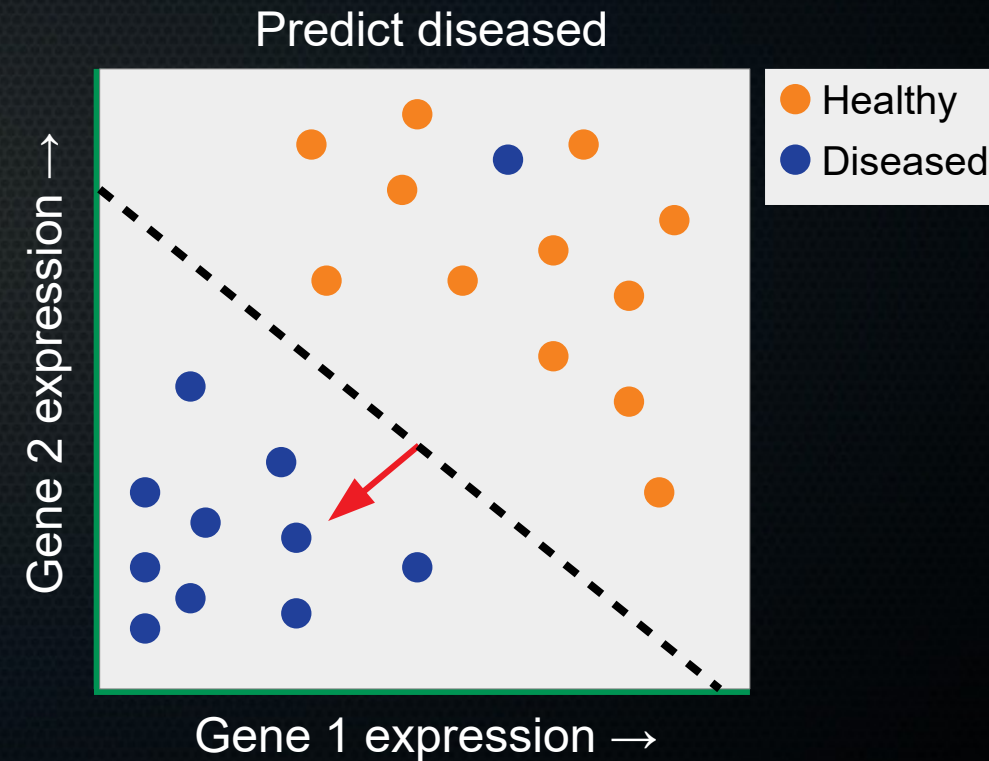
# Supervised learning: classification

- Given known examples, automatically find a function to map new examples.
- Discrete outputs.



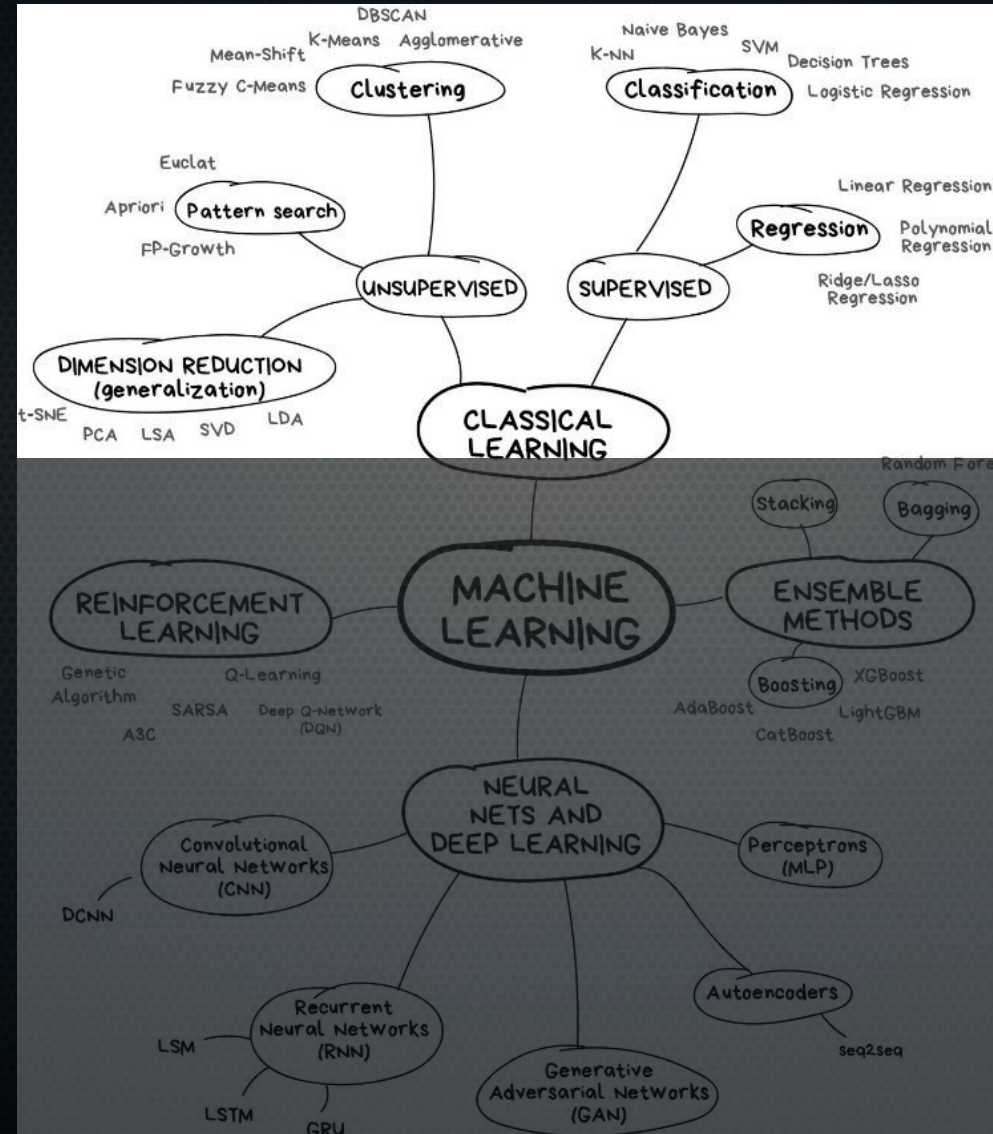
# Supervised learning: classification

- Given known examples, automatically find a function to map new examples.
- Discrete outputs.





# Map of Machine Learning



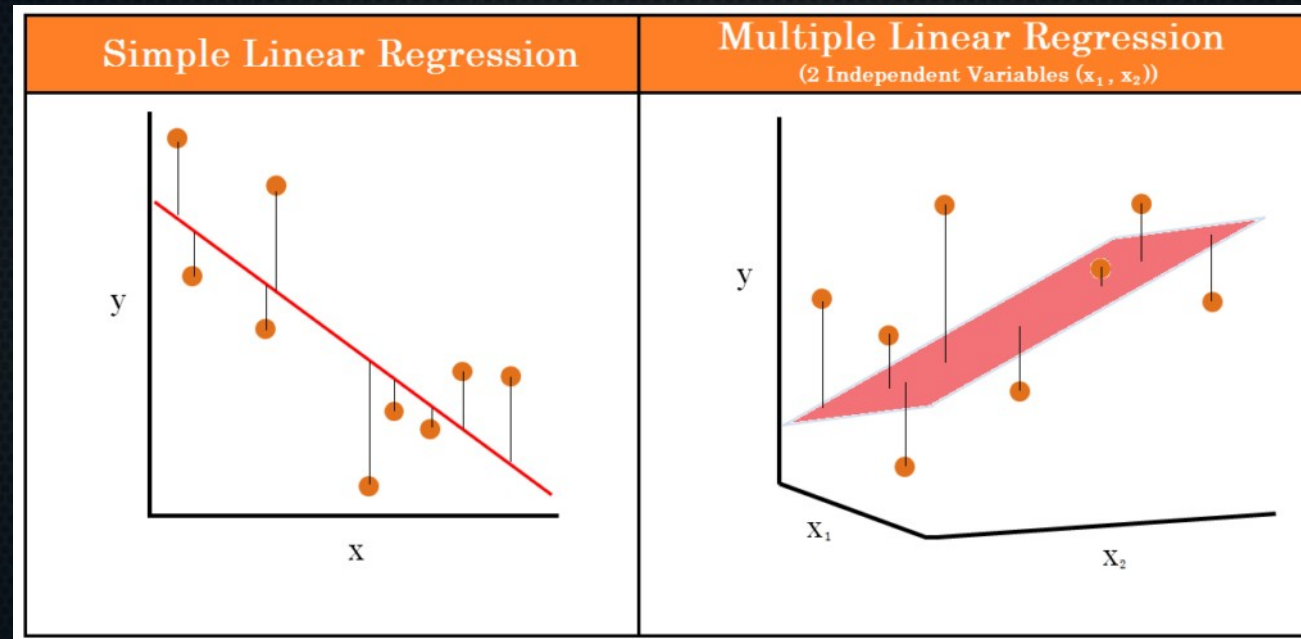
# Summary

---

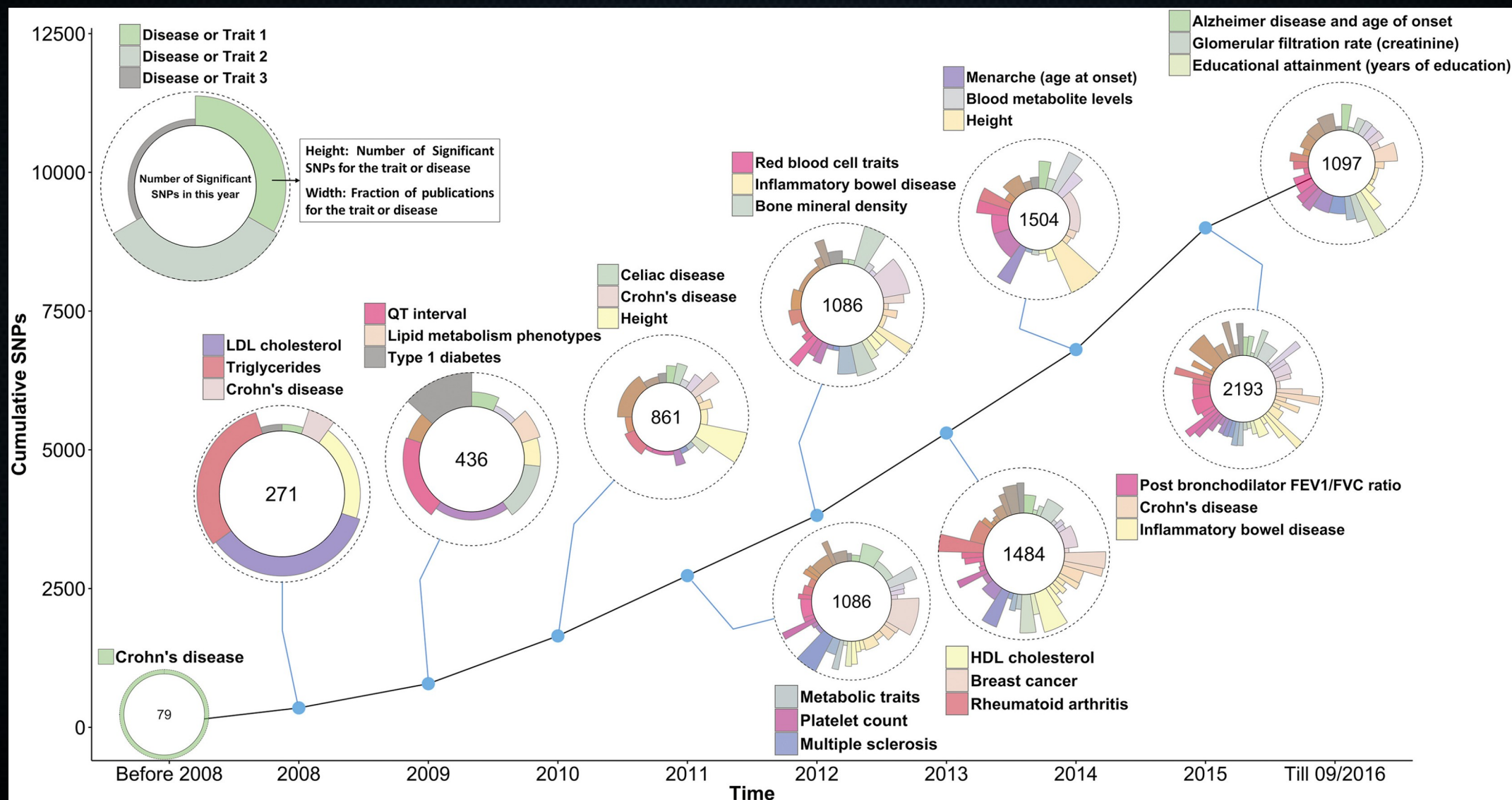
- Unsupervised:
  - Find some structure in your data (clusters, projection into lower-dimensional space that captures much of variance)
  - Don't know the „correct“ structure (no *labels*)
- Supervised:
  - Automatically learn a function that maps *features* (e.g. gene expression) to a real-valued output (regression, e.g. blood glucose level) or discrete *classes/labels* (classification, e.g. healthy/diseased) using training data for which you know these outcomes.



# Linear regression

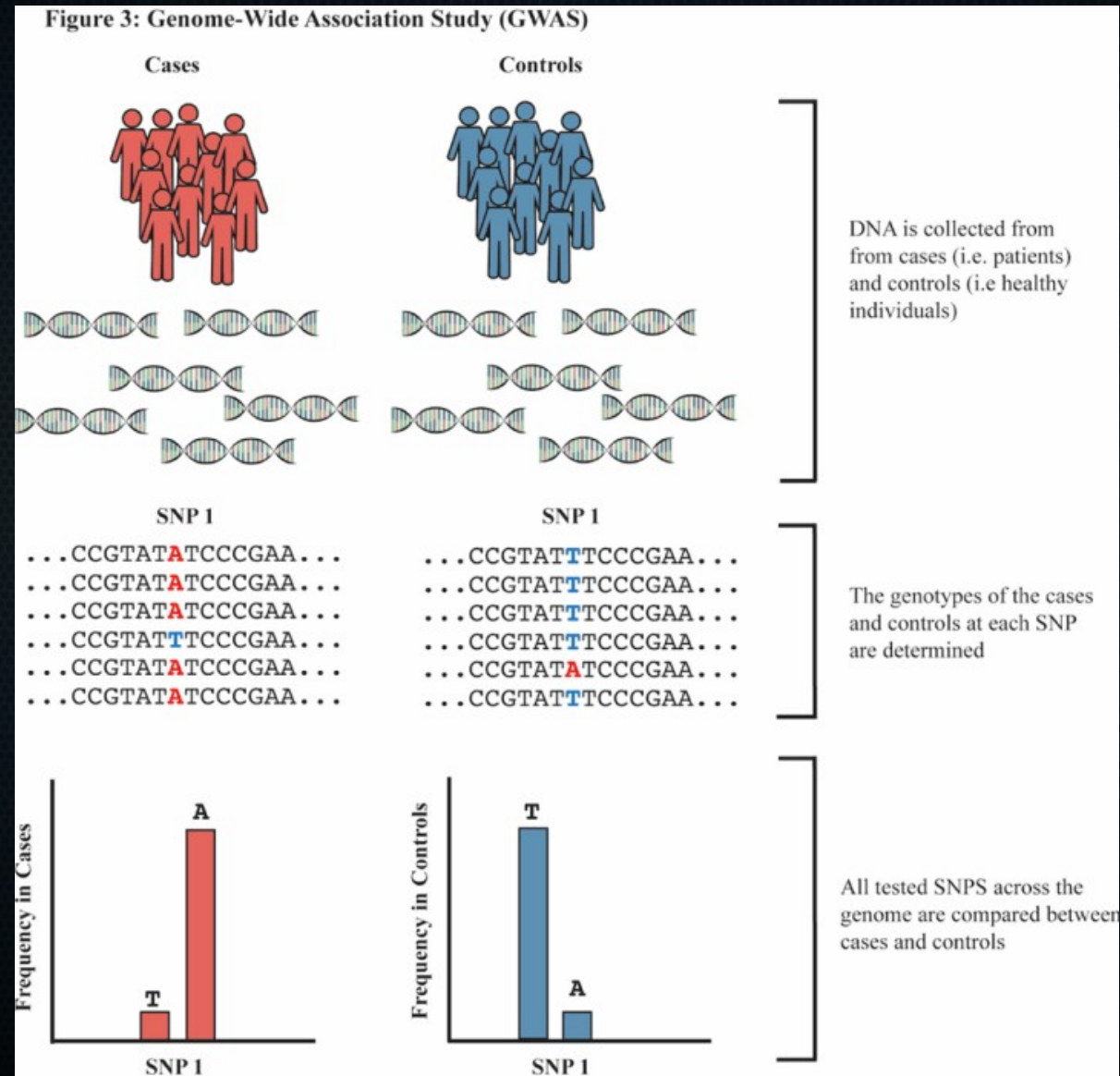


# Linear regression: GWAS

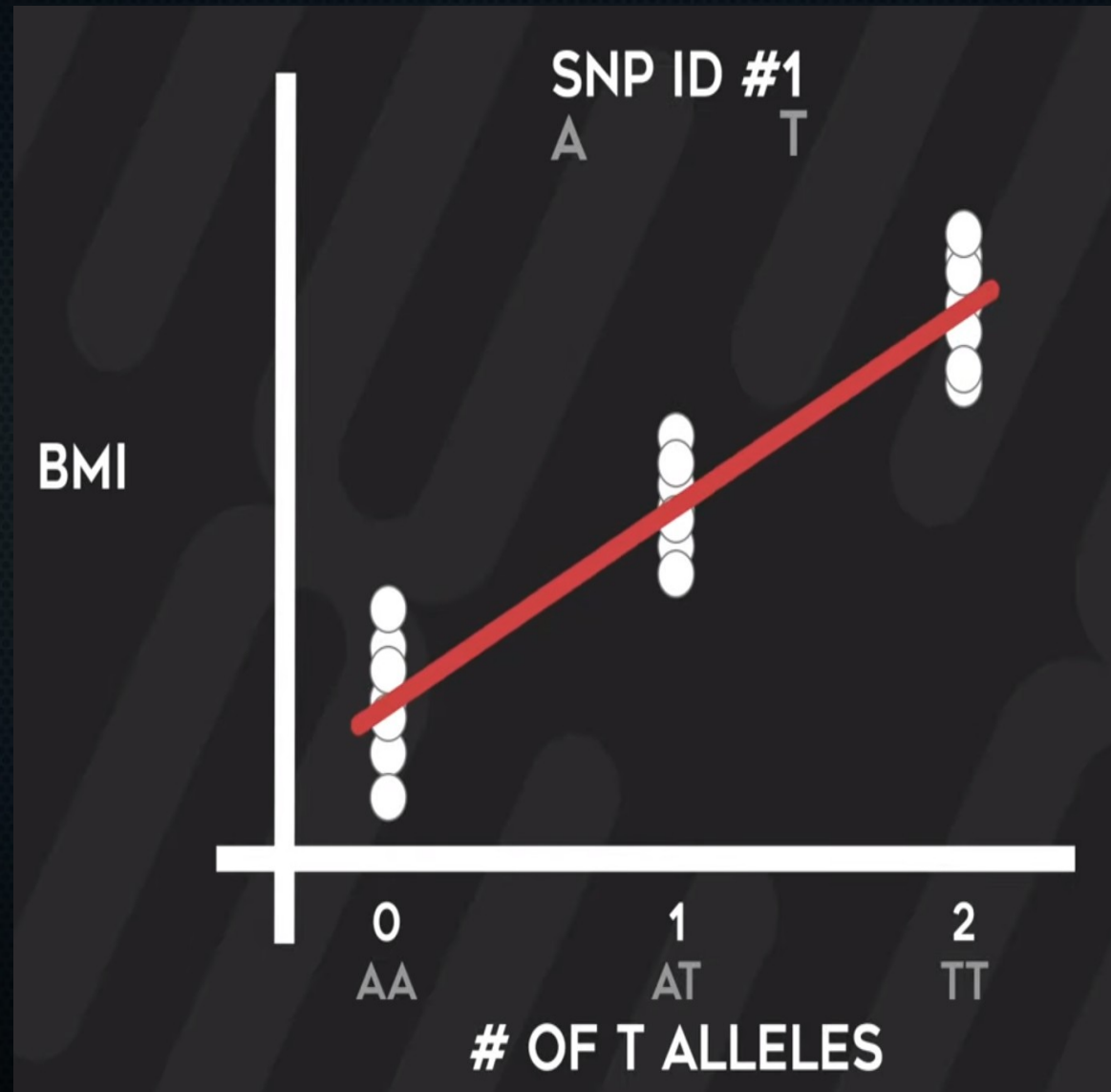




# Linear regression: GWAS

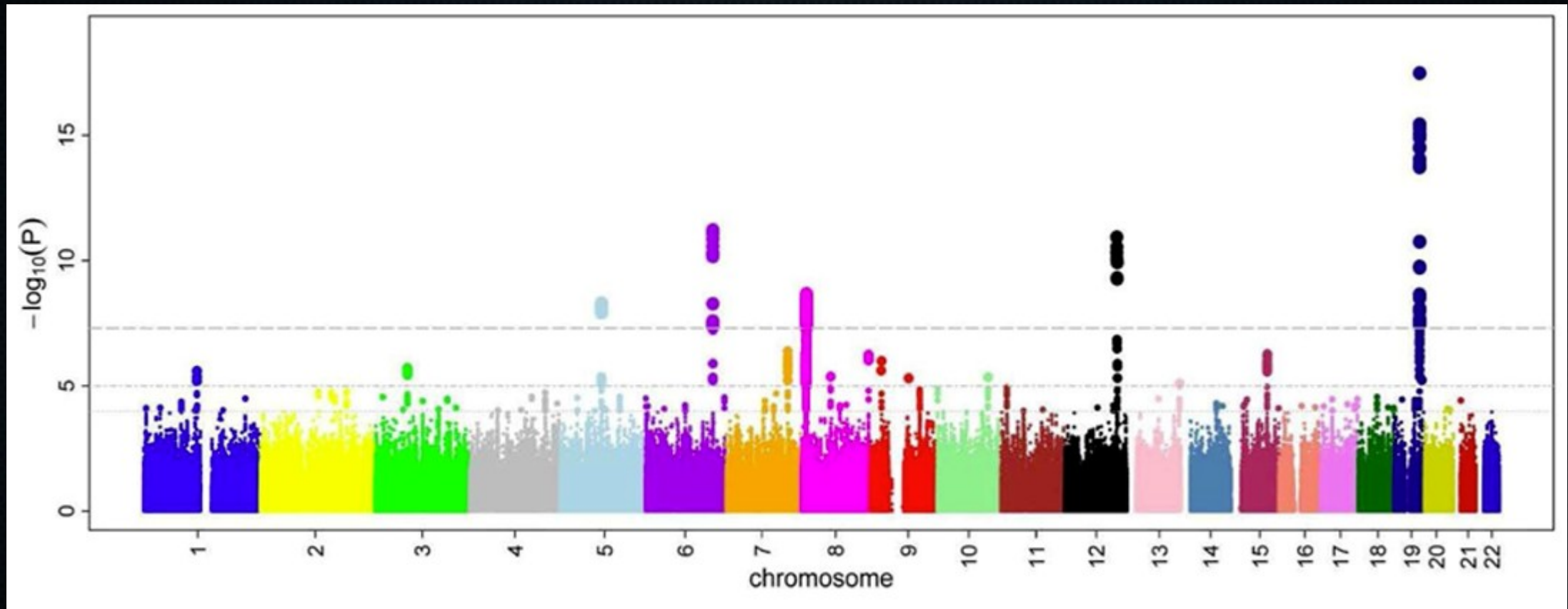


# Linear regression: GWAS





# Linear regression: GWAS



# Linear regression: GWAS

- Connect SNPs to nearby genes (non-trivial!)
- Yielded huge advances in our knowledge on many complex diseases over the past ~15 years.



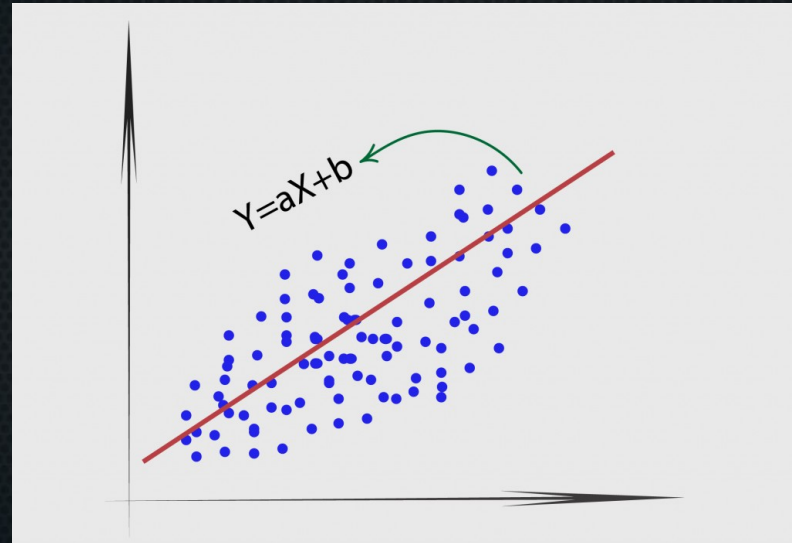


# Univariate linear regression

---

$$y = ax + b$$

$$y = ax + b$$

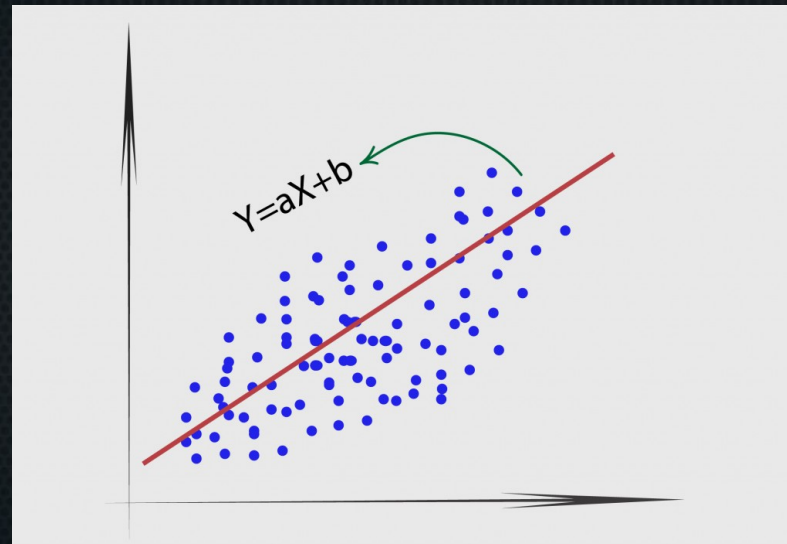


# Univariate linear regression

---

$$y = ax + b$$

$$y = ax + b$$

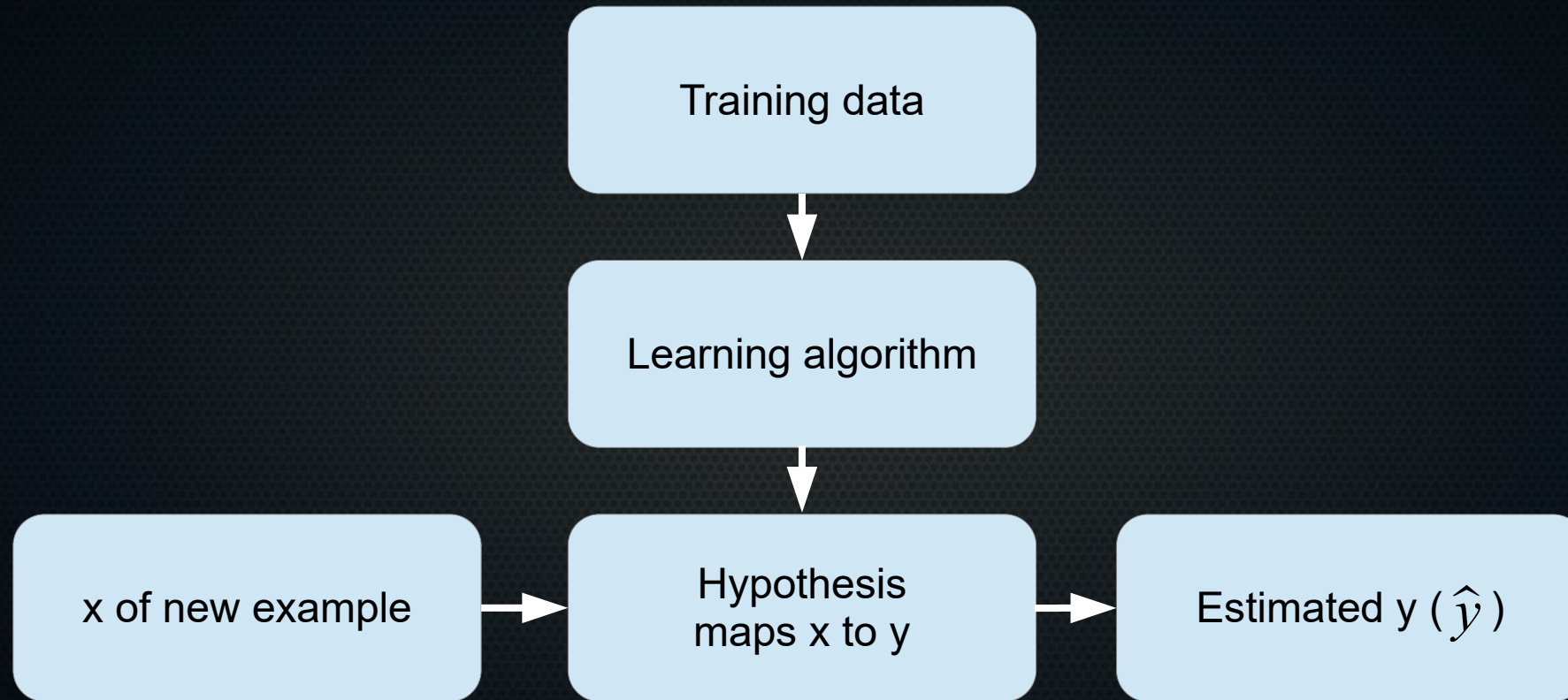


$$y = \theta_0 + \theta_1 x$$



# Process

$$y = ax + b$$

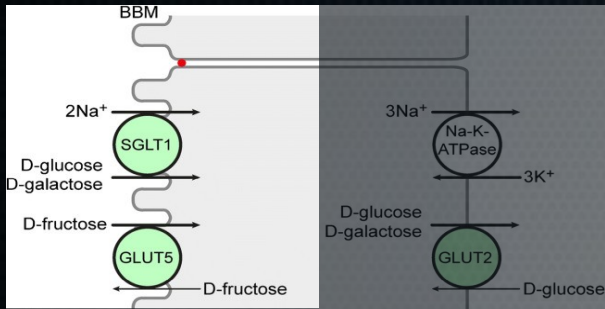


Source: Andrew Ng, Coursera

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

# Terminology

Training data

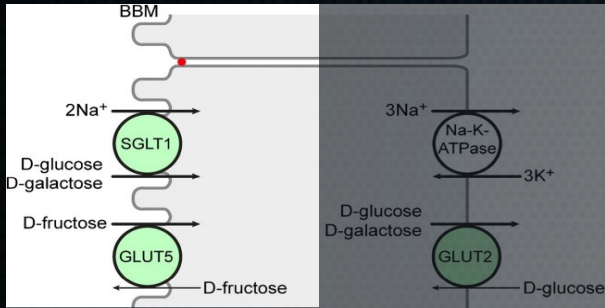


Sample #	SGLT1 expression level (arbitrary units relative to housekeeping gene)	Blood glucose level (mg/dL)
1	3	80
2	8	130
3	12	170
4	2	89



# Terminology

Training data



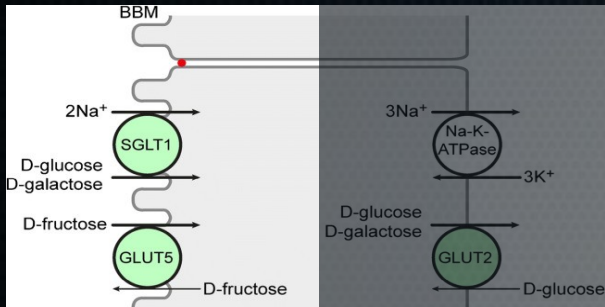
$m = \#$  of training examples

$m$

Sample #	SGLT1 expression level (arbitrary units relative to housekeeping gene)	Blood glucose level (mg/dL)
1	3	80
2	8	130
3	12	170
4	2	89

# Terminology

Training data



$m$  = # of training examples  
 $n$  = # of features/variables

$m$

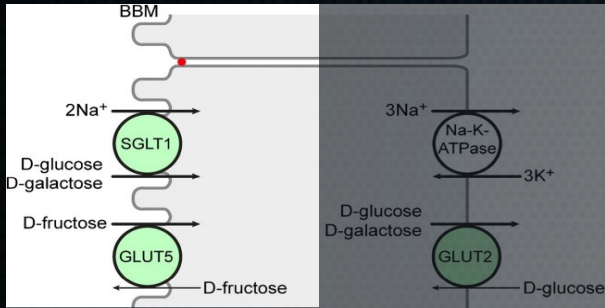
Sample #	SGLT1 expression level (arbitrary units relative to housekeeping gene)	Blood glucose level (mg/dL)
1	3	80
2	8	130
3	12	170
4	2	89

$n$



# Terminology

Training data



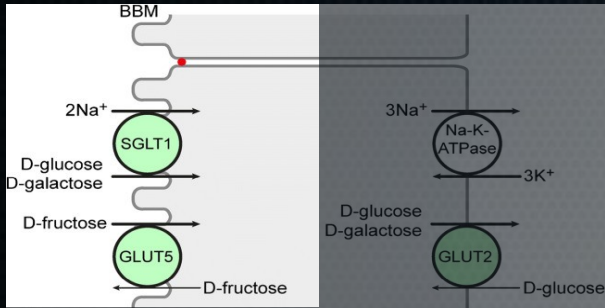
$m$  = # of training examples  
 $n$  = # of features/variables

Sample #	x1	x2	x3	Blood glucose level (mg/dL)
1	3	2	11	80
2	8	3	2	130
3	12	4	666	170
4	2	6	5	89

$n$

# Terminology

Training data



$m$  = # of training examples  
 $n$  = # of features/variables  
 $y$  = output variable/target variable or label (classification)

$m$

Sample #	SGLT1 expression level (arbitrary units relative to housekeeping gene)	Blood glucose level (mg/dL)
1	3	80
2	8	130
3	12	170
4	2	89

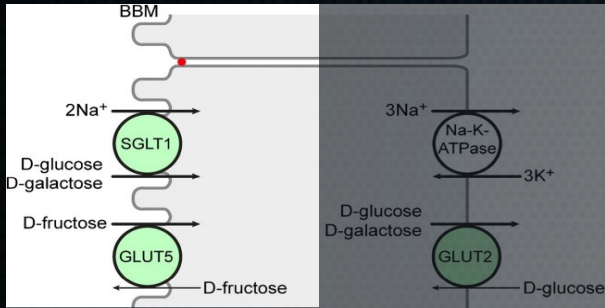
$n$

$y$



# Terminology

Training data



$m$  = # of training examples  
 $n$  = # of features/variables  
 $y$  = output variable/target variable or label (classification)  
 $(x^{(i)}, y^{(i)})$  =  $i$ -th training example

Sample #	SGLT1 expression level (arbitrary units relative to housekeeping gene)	Blood glucose level (mg/dL)
1	3	80
2	8	130
3	12	170
4	2	89

$(x^{(3)}, y^{(3)})$

$n$

$y$

# Cost function and gradient descent

---

- How to learn theta's from data?
- Two parts
  - How wrong are we for given parameters?
  - How do we update our parameters, given how wrong we are?

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



# Cost function and gradient descent

---

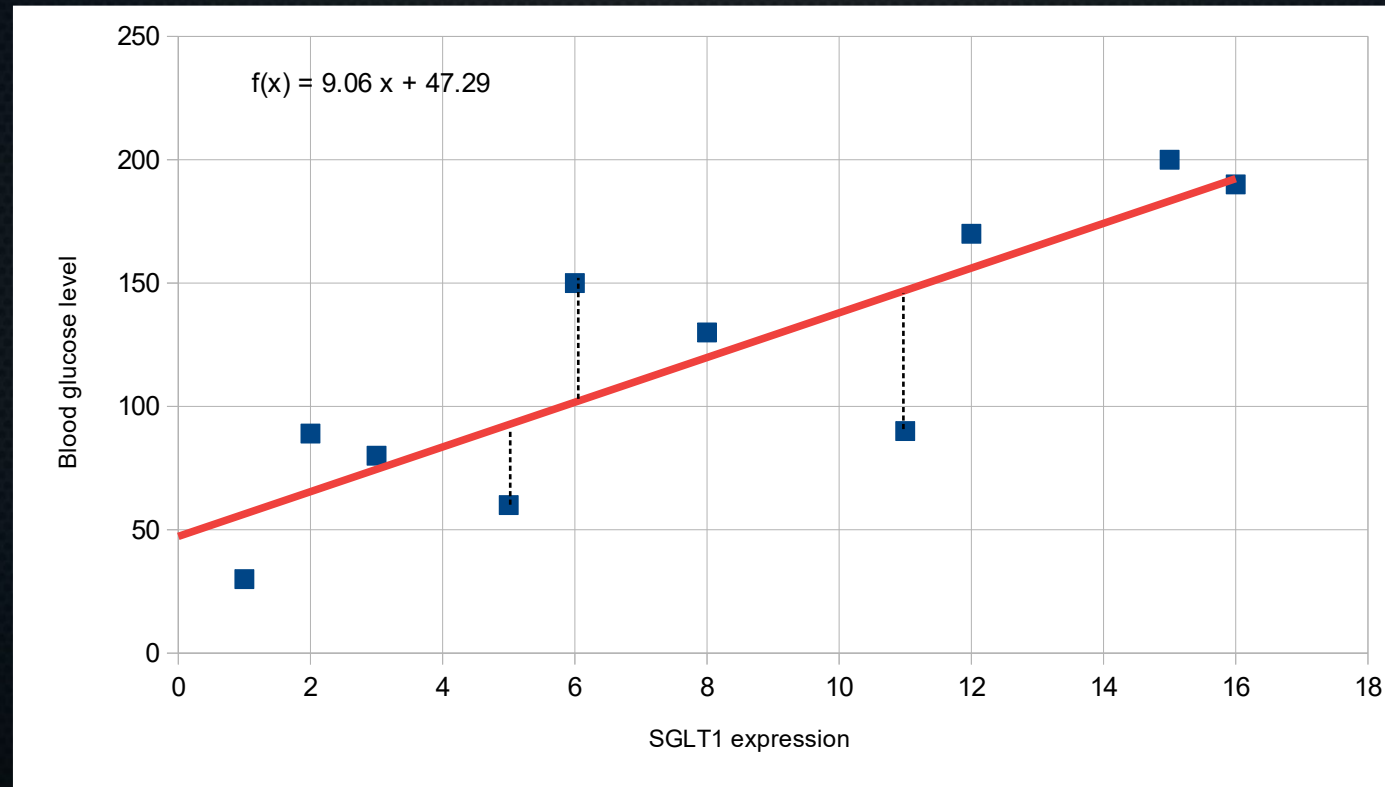
- How to learn theta's from data?
- Two parts
  - *How wrong are we for given parameters?*
  - How do we update our parameters, given how wrong we are?

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

# Cost function

- How wrong are we for given parameters?

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



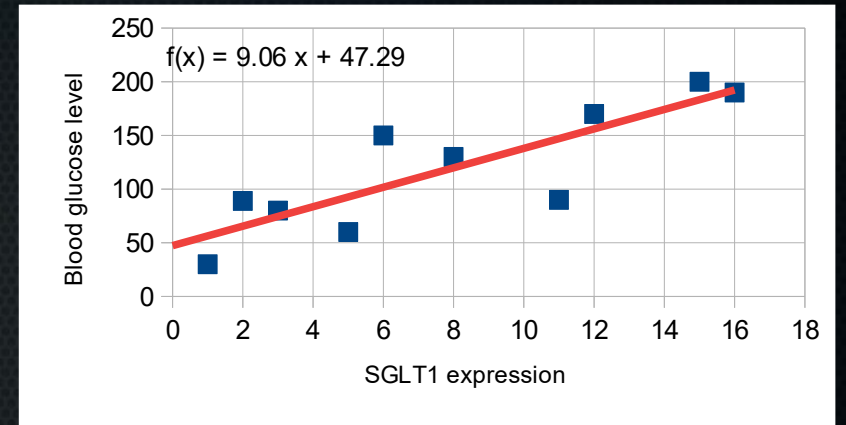


# Cost function

- How wrong are we for given parameters?
- Cost function:

$$J(\theta_0, \theta_1) = \frac{1}{m} \sum_i^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



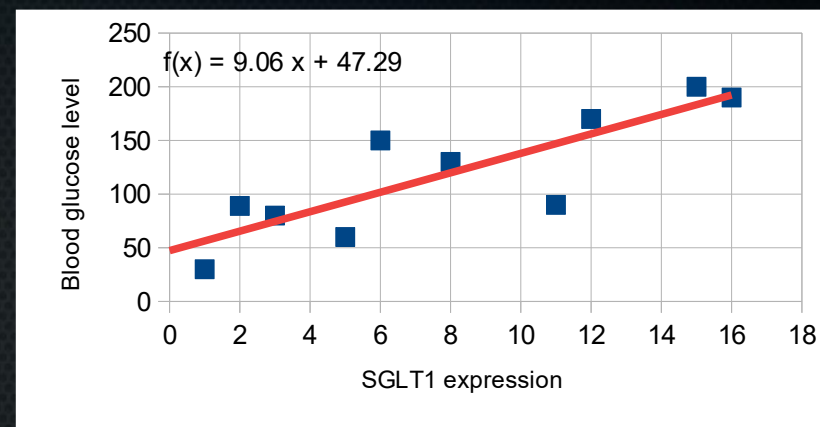
# Cost function

- How wrong are we for given parameters?
- Cost function:

$$J(\theta_0, \theta_1) = \frac{1}{m} \sum_i^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_i^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$





# Cost function

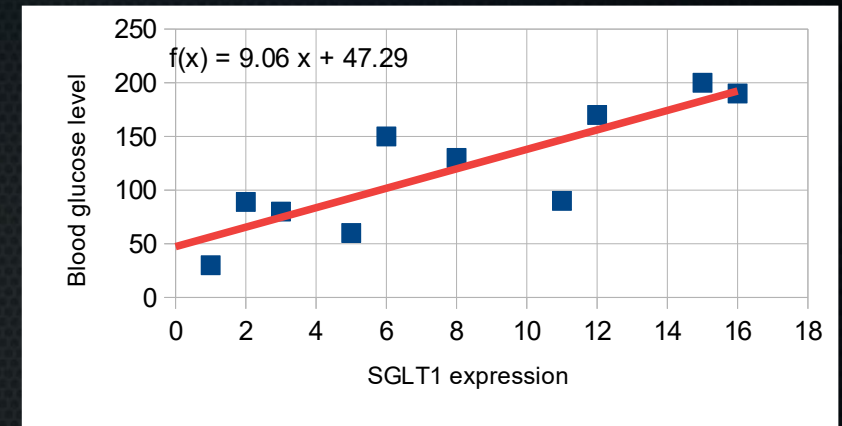
- How wrong are we for given parameters?
- Cost function:

$$J(\theta_0, \theta_1) = \frac{1}{m} \sum_i^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_i^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_i^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



# Cost function

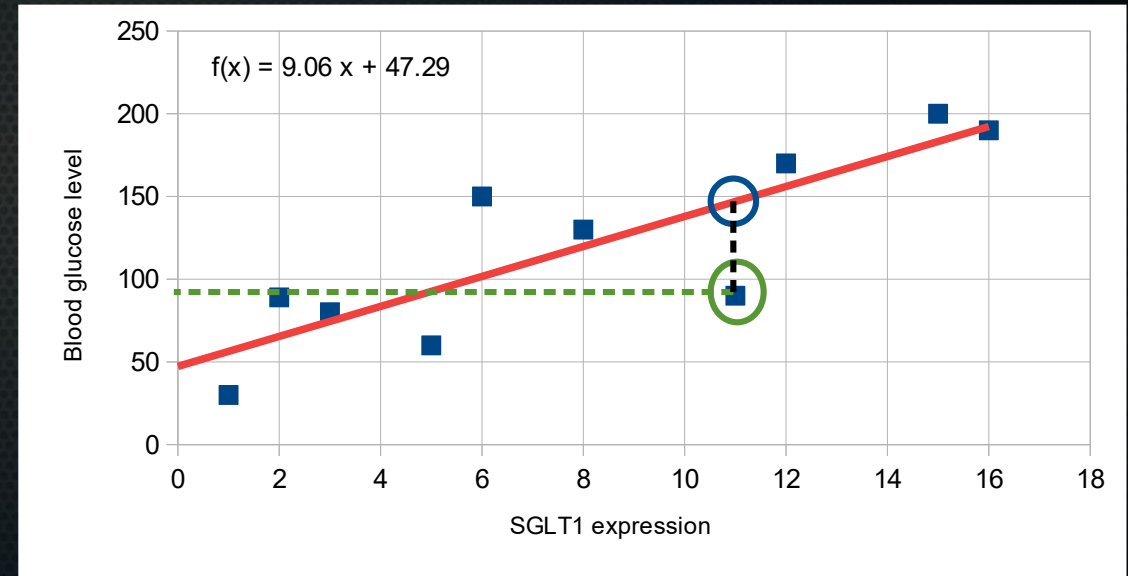
- How wrong are we for given parameters?
- Cost function:

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$





# Cost function conclusion

---

- If we want to be as correct as possible with our prediction, want to minimise:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

# Cost function conclusion

---

- If we want to be as correct as possible with our prediction, want to minimise:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

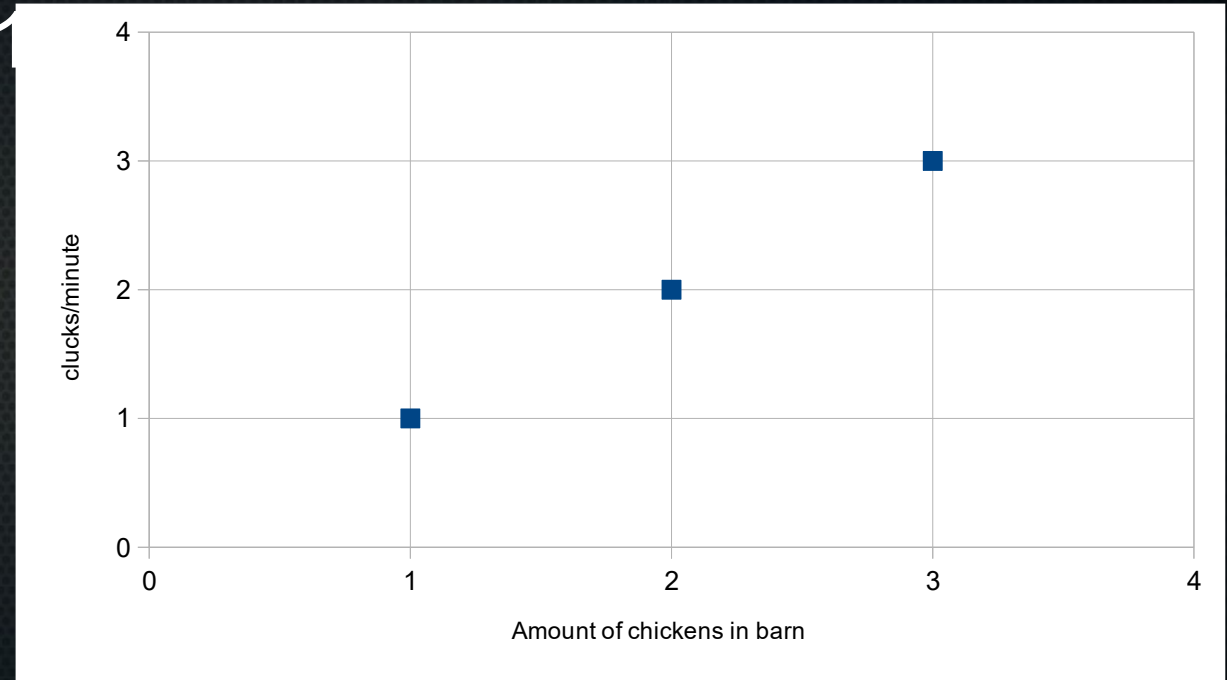
- Simplified example:  $\theta_0 = 0$ :

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_1 \cdot x^{(i)} - y^{(i)})^2$$



# Cost function conclusion illustration

- Let's say  $\theta_0 = 0$  (so the intercept is 0). What is  $J(\theta_1)$  for different values of  $\theta_1$ ?



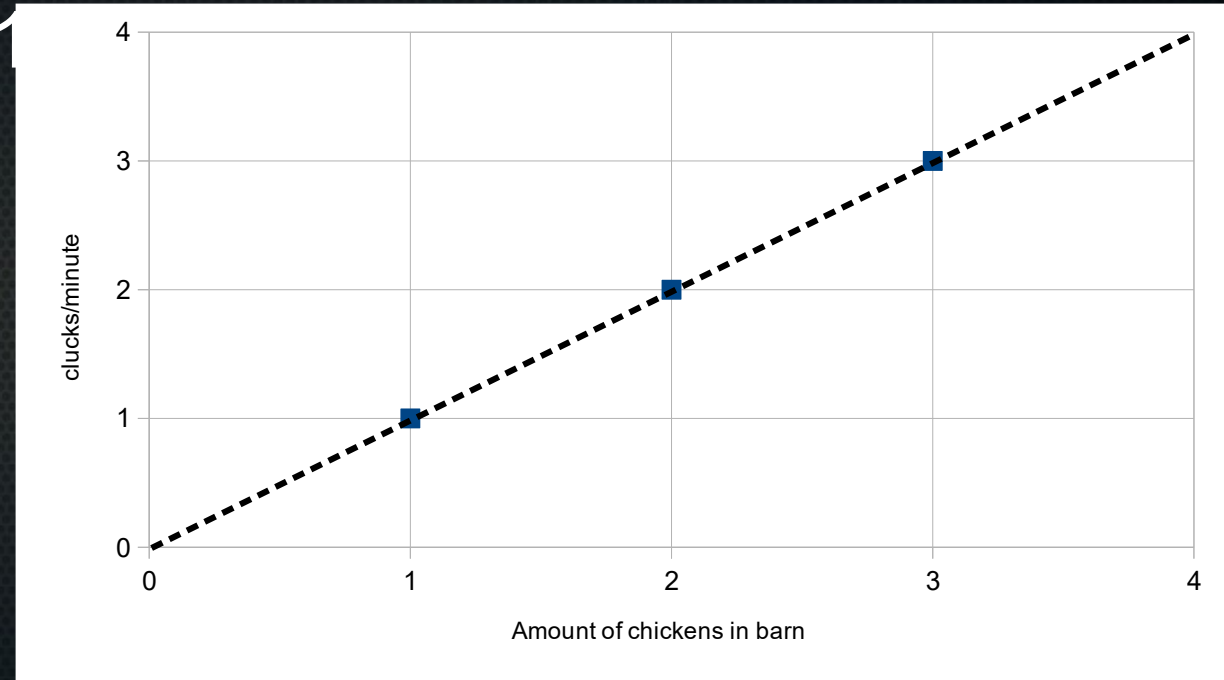
# Cost function illustration

- Let's say  $\theta_0 = 0$  (so the intercept is 0). What is  $J(\theta_1)$  for different values of  $\theta_1$ ?

- $\theta_1 = 1$

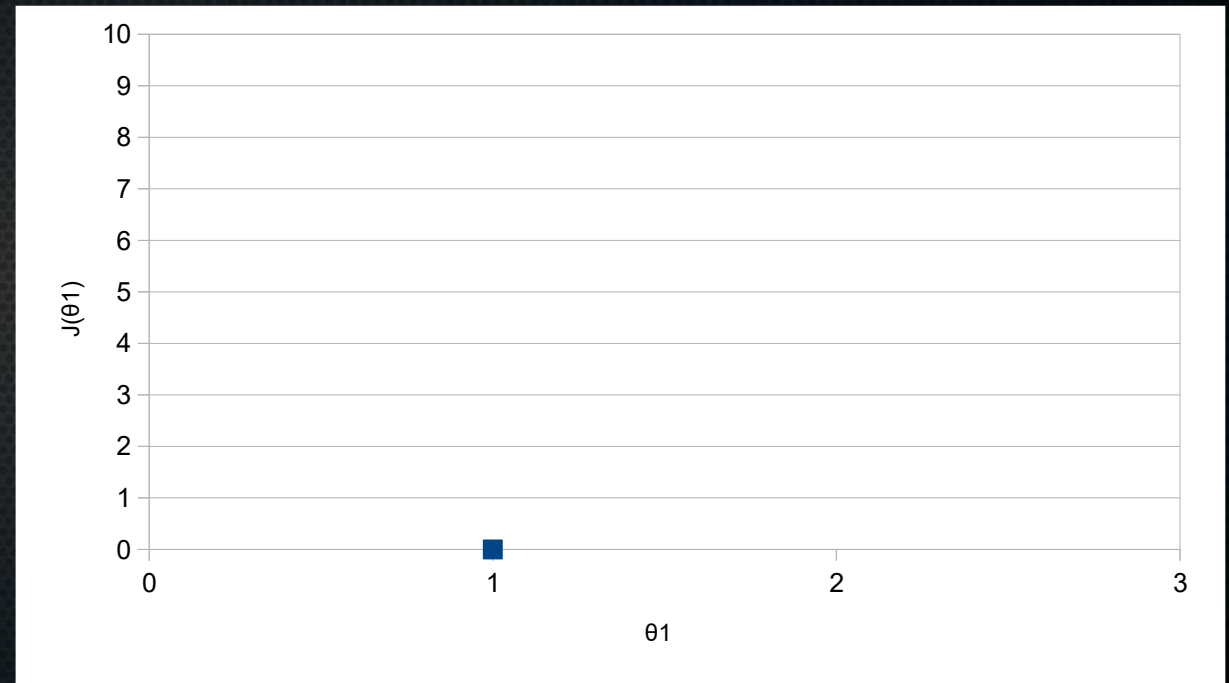
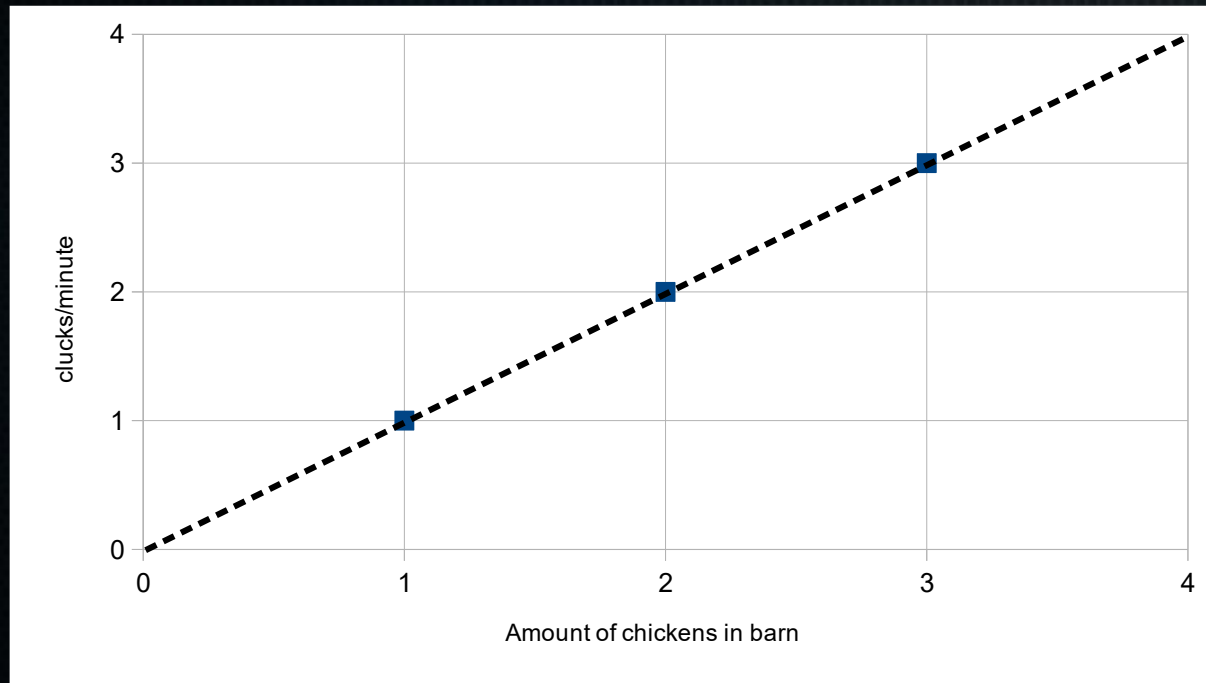
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^3 (\theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$J(\theta_1) = 0$$





# Cost function illustration

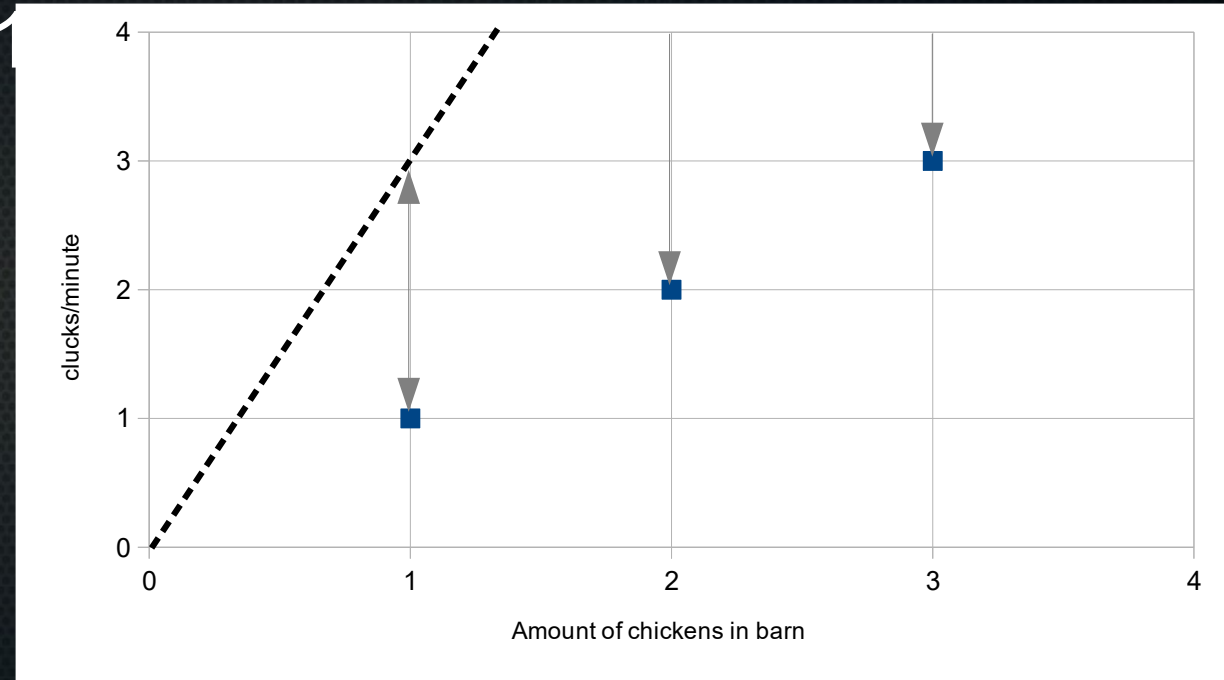


# Cost function illustration

- Let's say  $\theta_0 = 0$  (so the intercept is 0). What is  $J(\theta_1)$  for different values of  $\theta_1$ ?

- $\theta_1 = 3$

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^3 (\theta_1 \cdot x^{(i)} - y^{(i)})^2$$

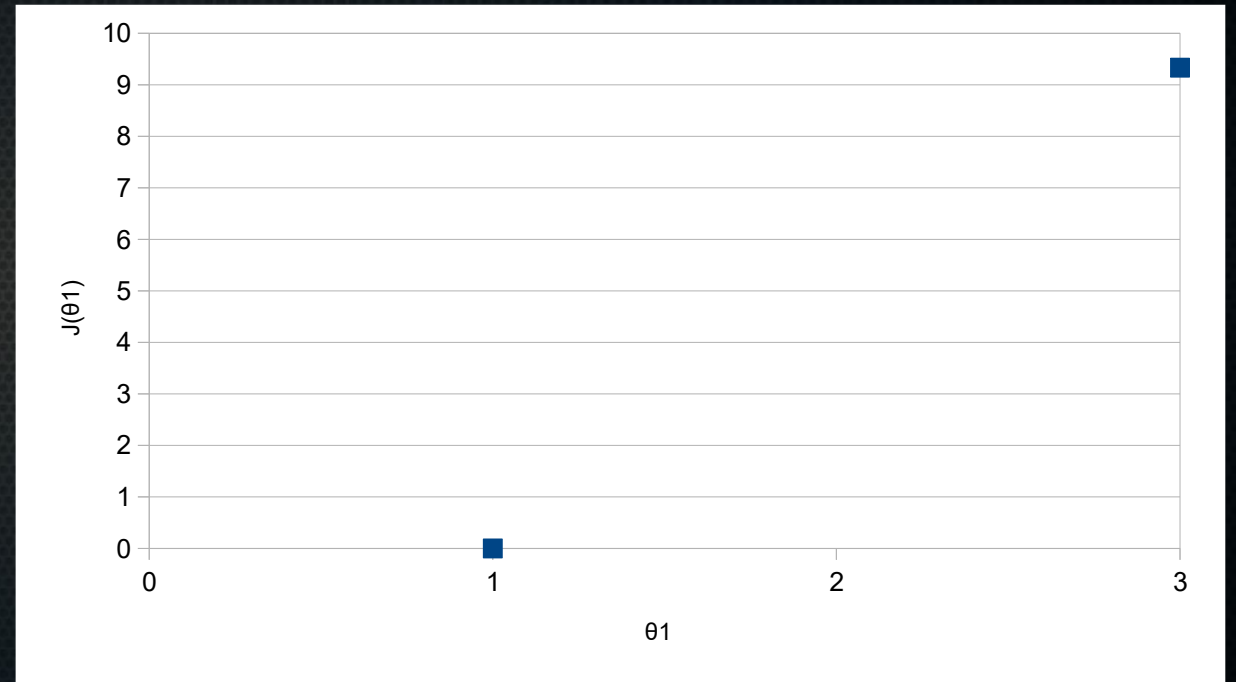
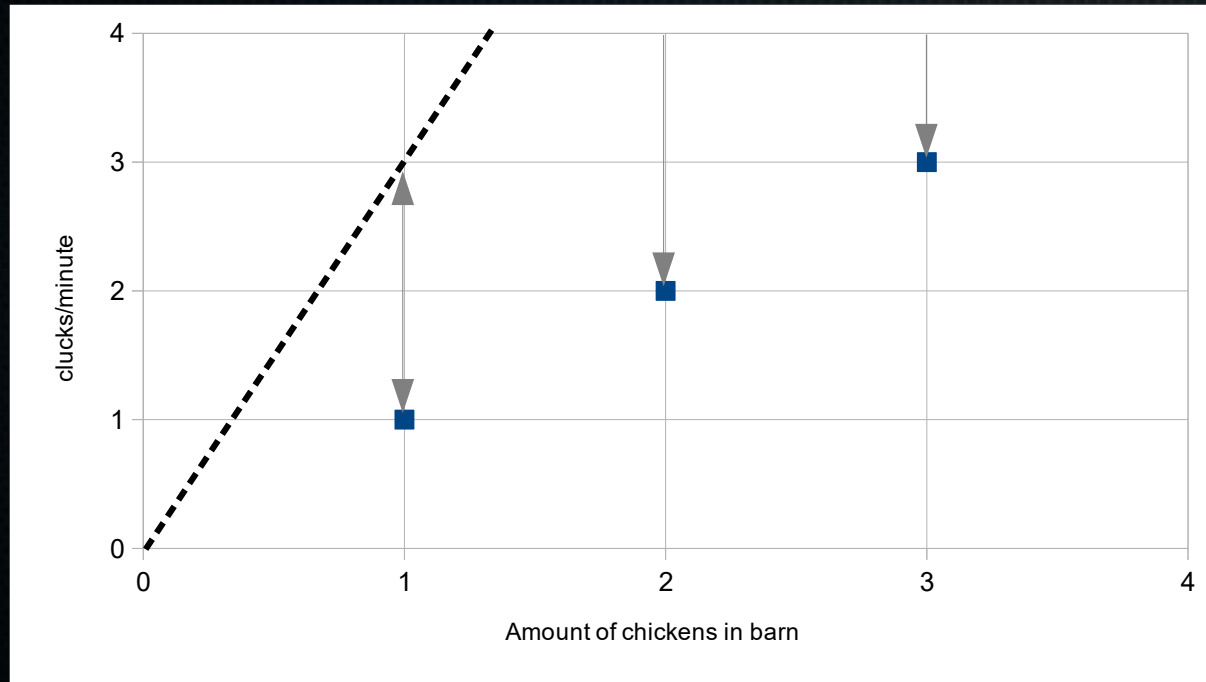


$$J(\theta_1) = \frac{1}{2 \cdot 3} \cdot ((3 - 1)^2 + (6 - 2)^2 + (9 - 3)^2) = 56/6 \approx 9.3$$





# Cost function illustration

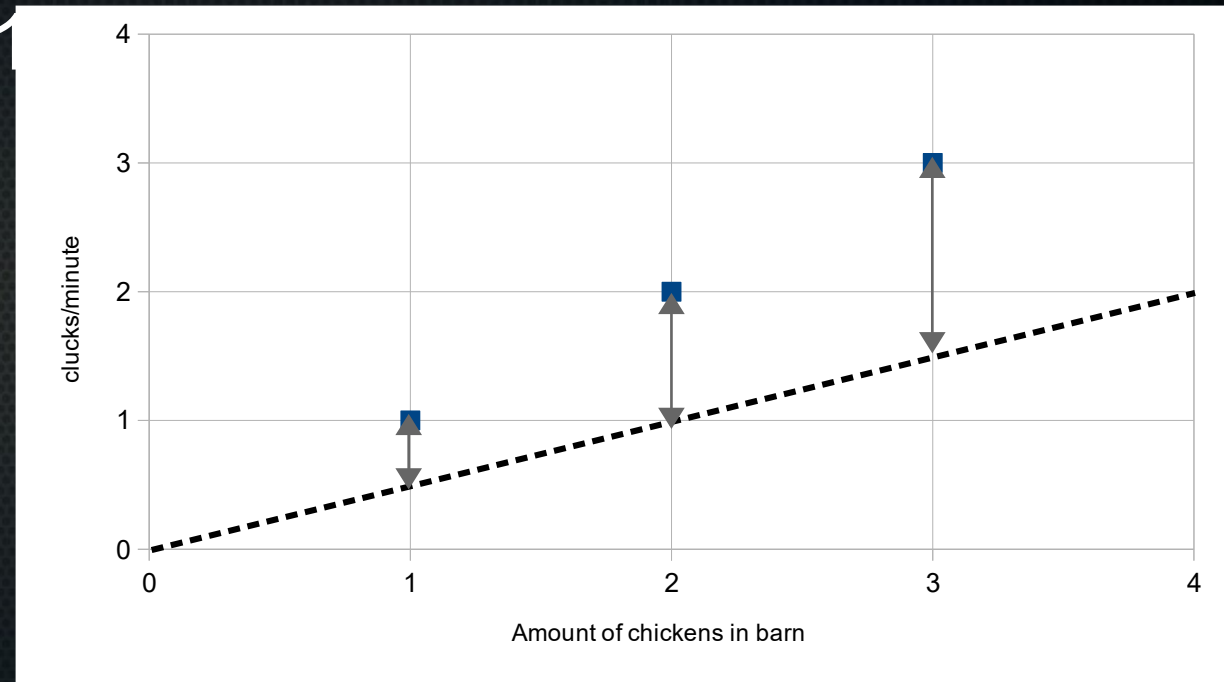


# Cost function illustration

- Let's say  $\theta_0 = 0$  (so the intercept is 0). What is  $J(\theta_1)$  for different values of  $\theta_1$ ?

- $\theta_1 = 0.5$

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^3 (\theta_1 \cdot x^{(i)} - y^{(i)})^2$$



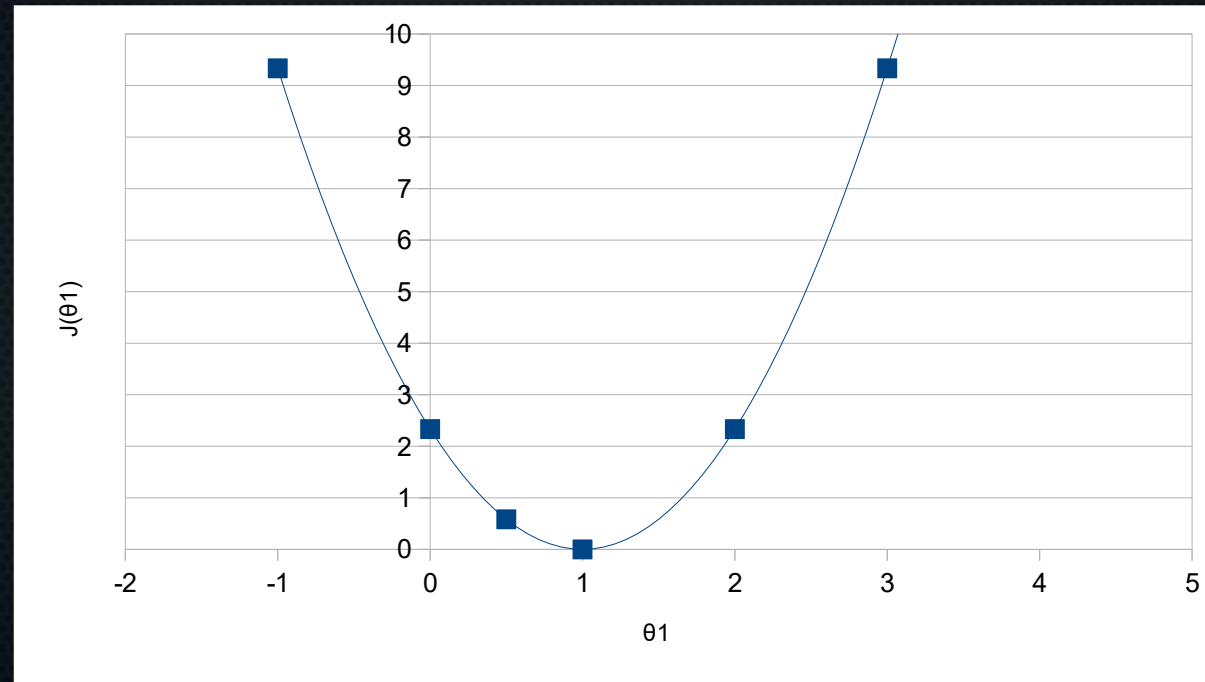
$$J(\theta_1) = \frac{1}{2 \cdot 3} \cdot ((0.5 - 1)^2 + (1 - 2)^2 + (1.5 - 3)^2) = 3.5 / 6 \approx 0.6$$





# Cost function illustration

---



# Cost function and gradient descent

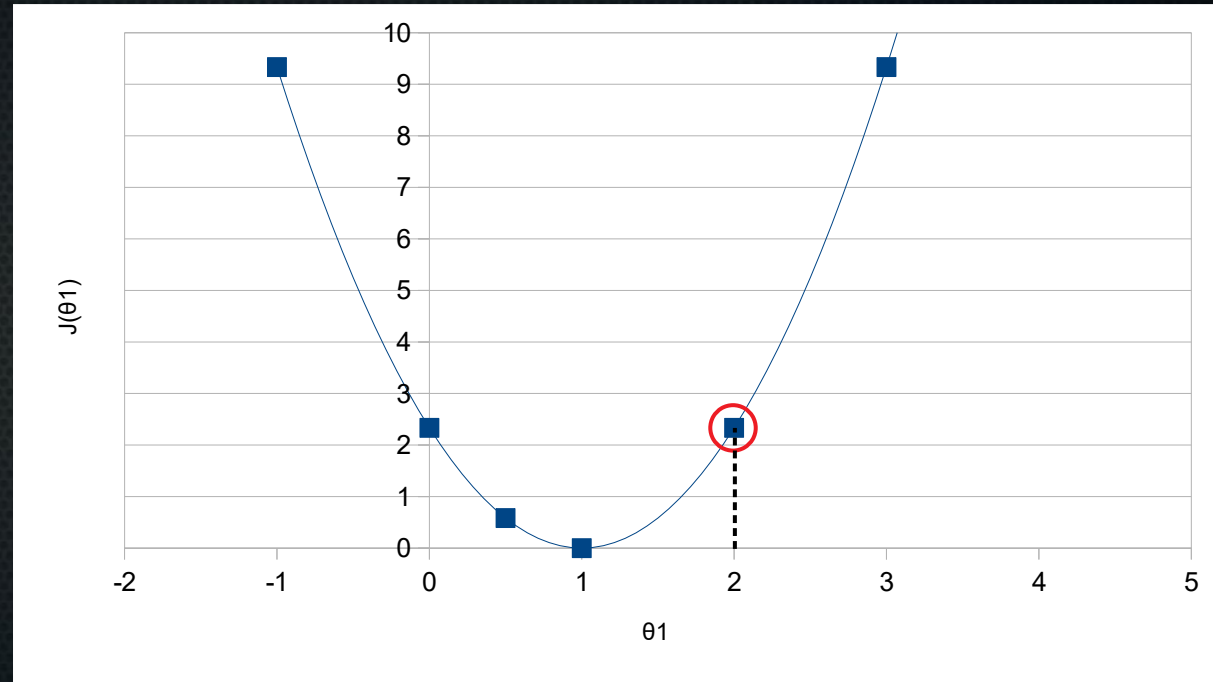
---

- How to learn theta's from data?
- Two parts
  - How wrong are we for given parameters?
  - *How do we update our parameters, given how wrong we are?*



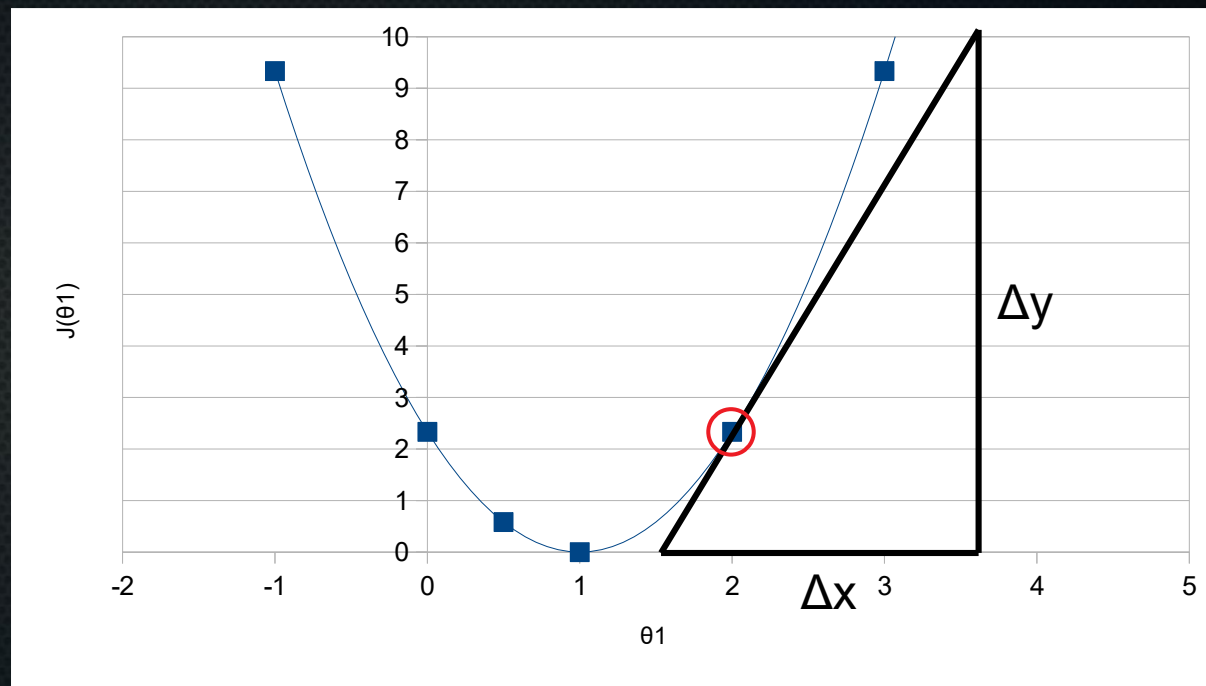
# Gradient descent

- Want to minimise
- Where to go?



# Gradient descent

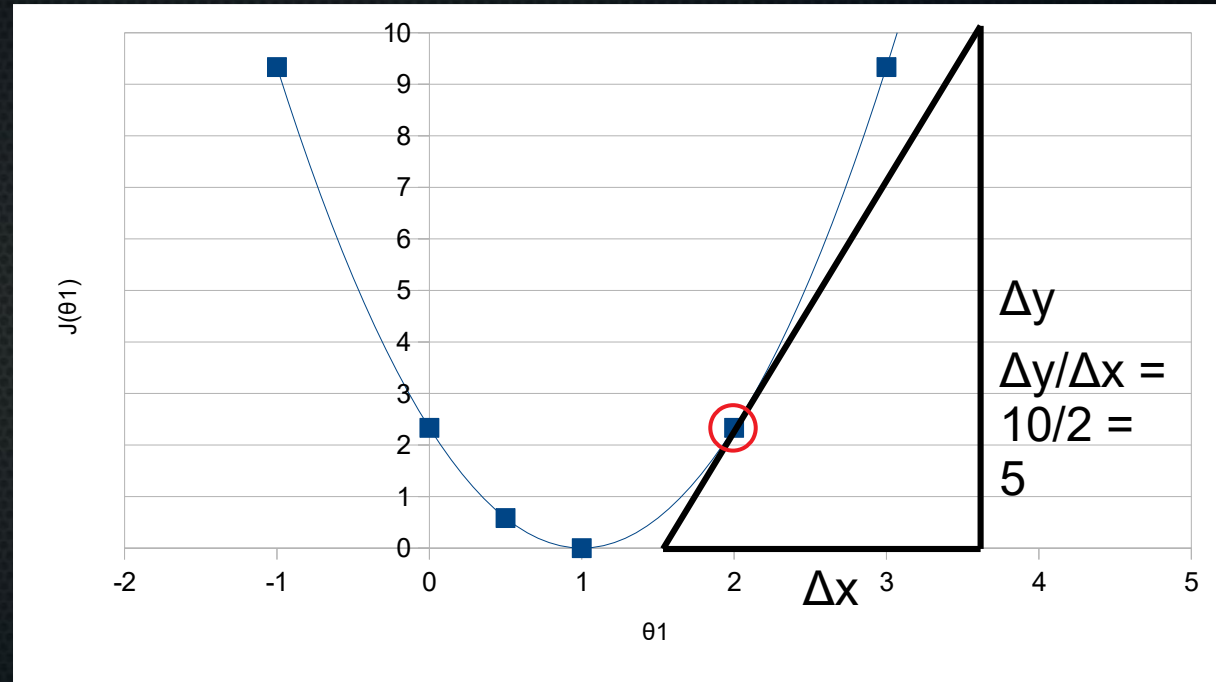
- Want to minimise
- Where to go?





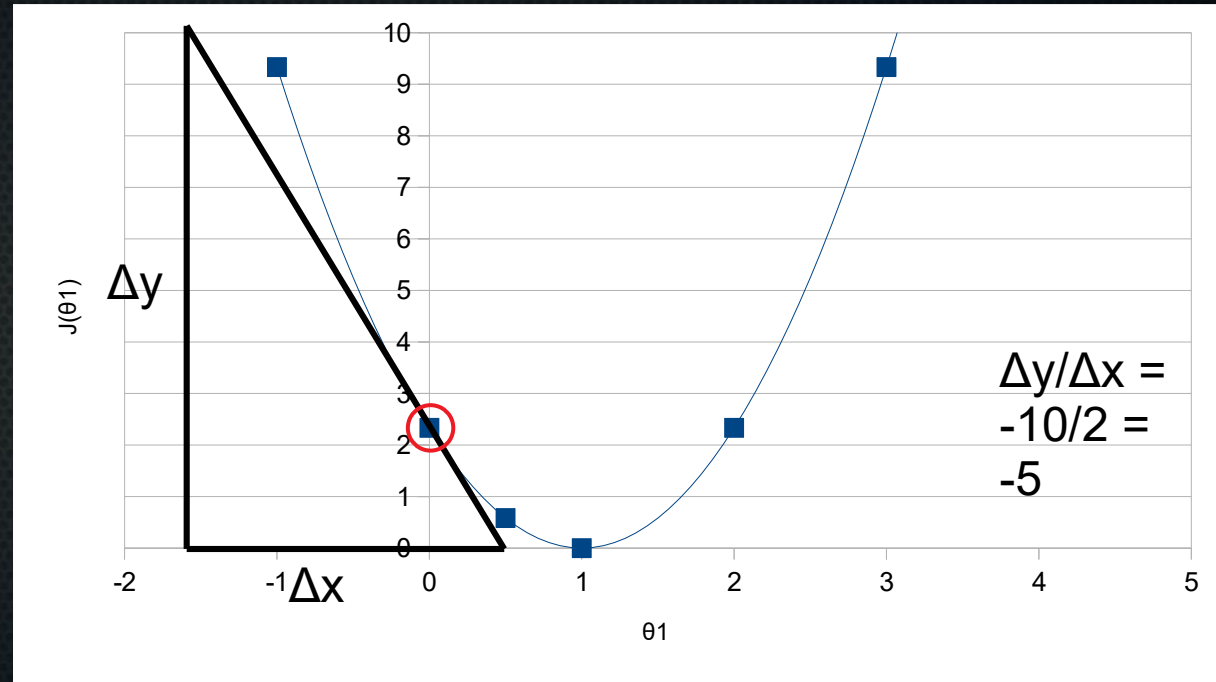
# Gradient descent

- Want to minimise
- Where to go?



# Gradient descent

- Want to minimise
- Where to go?

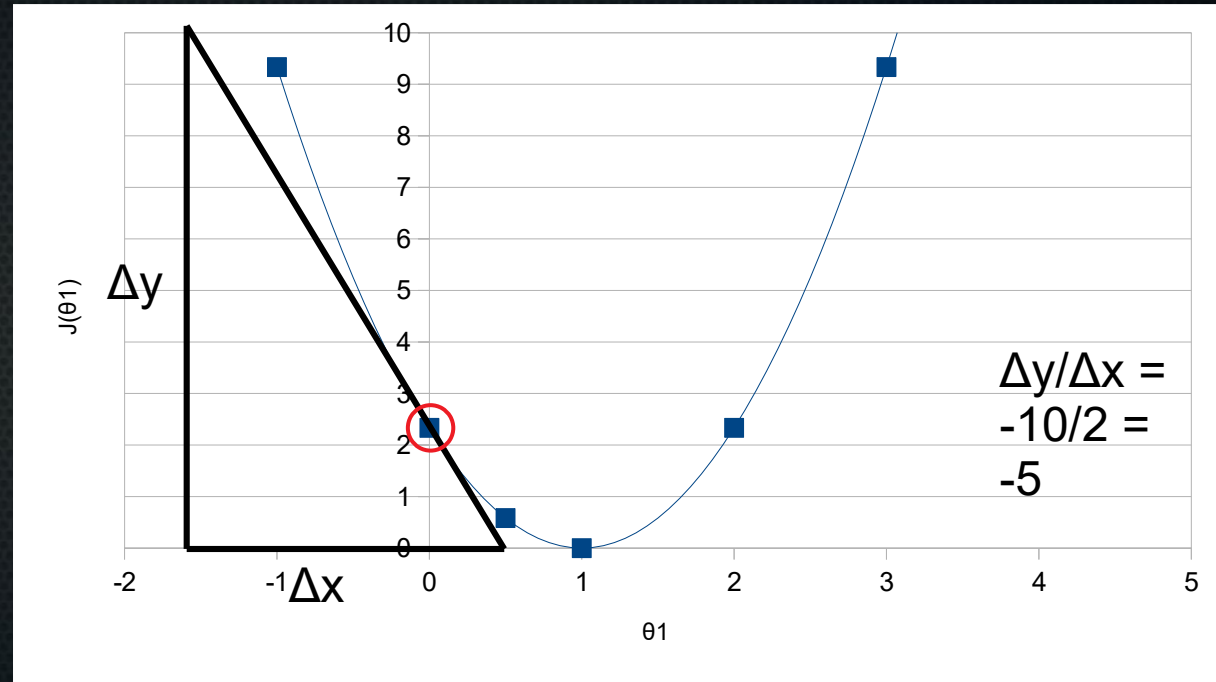




# Gradient descent

- Want to minimise
- Where to go?
- Change current theta1 as follows:

$$\theta_1 = \theta_1 - \alpha \cdot \frac{d}{d\theta_1} J(\theta_1)$$



# Gradient descent

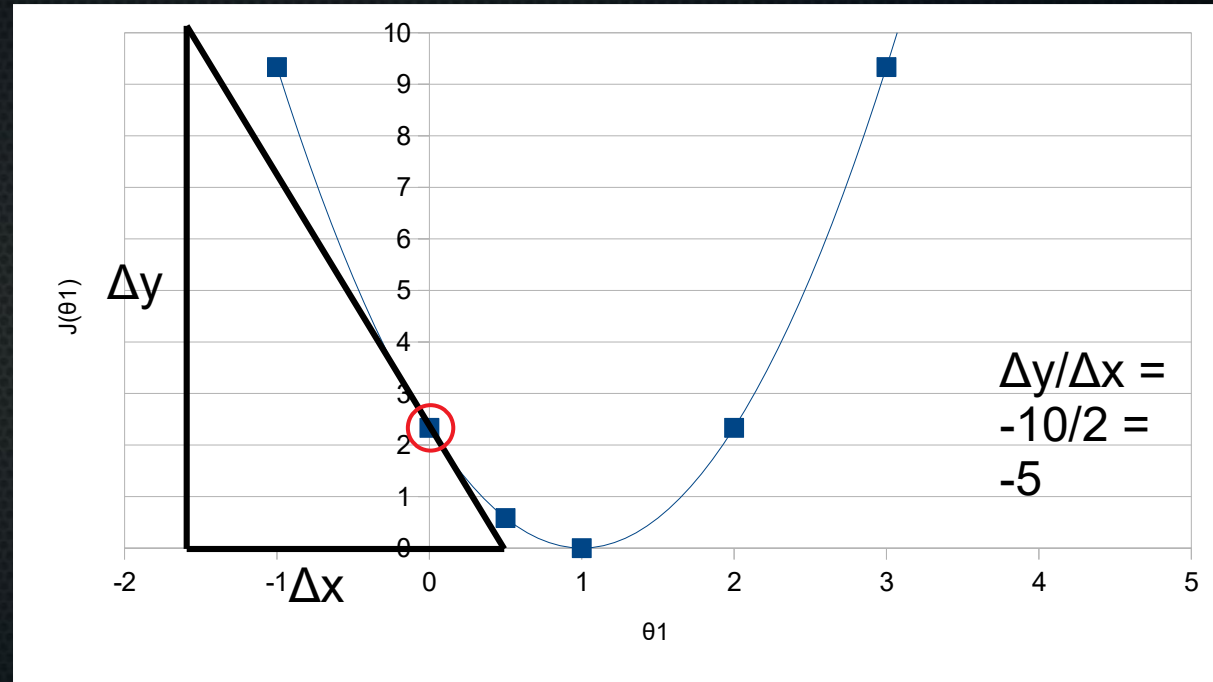
- Want to minimise
- Where to go?
- Change current theta1 as follows:

$$\theta_1 = \theta_1 - \alpha \cdot \frac{d}{d\theta_1} J(\theta_1)$$

$$\alpha = 0.2$$

$$\theta_1 = 0 - 0.2 \cdot -5$$

$$\theta_1 = 1$$





# Gradient descent

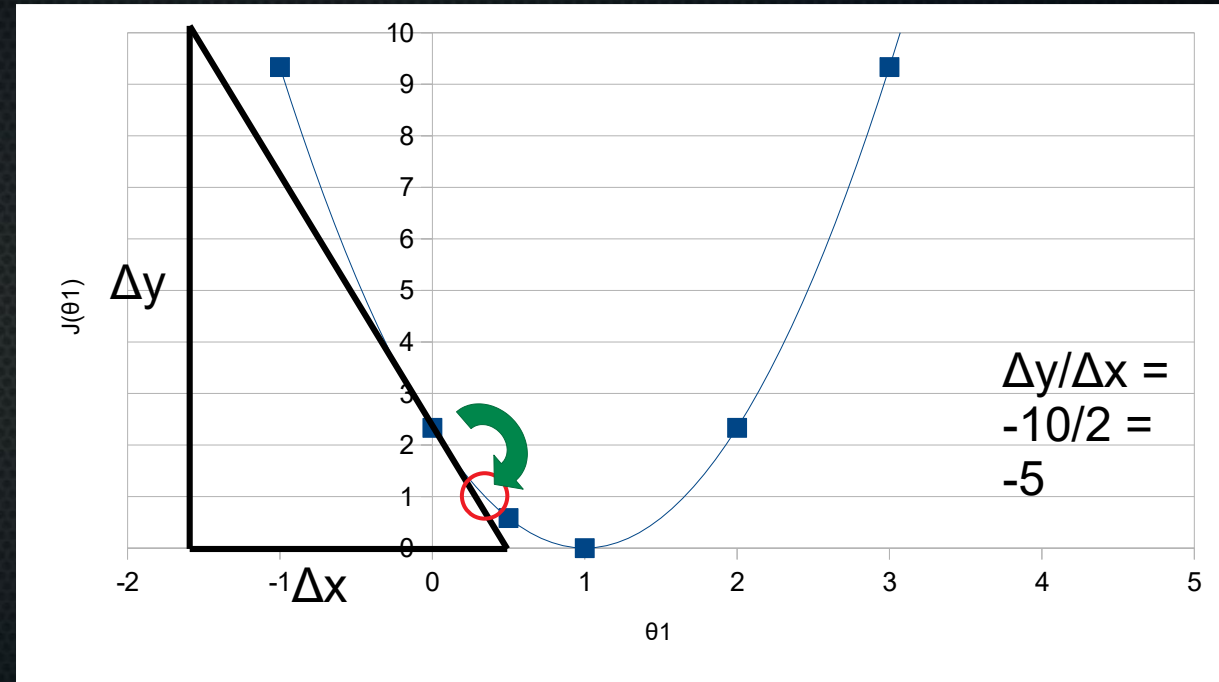
- Want to minimise
- Where to go?
- Change current theta1 as follows:

$$\theta_1 = \theta_1 - \alpha \cdot \frac{d}{d\theta_1} J(\theta_1)$$

$$\alpha = 0.02$$

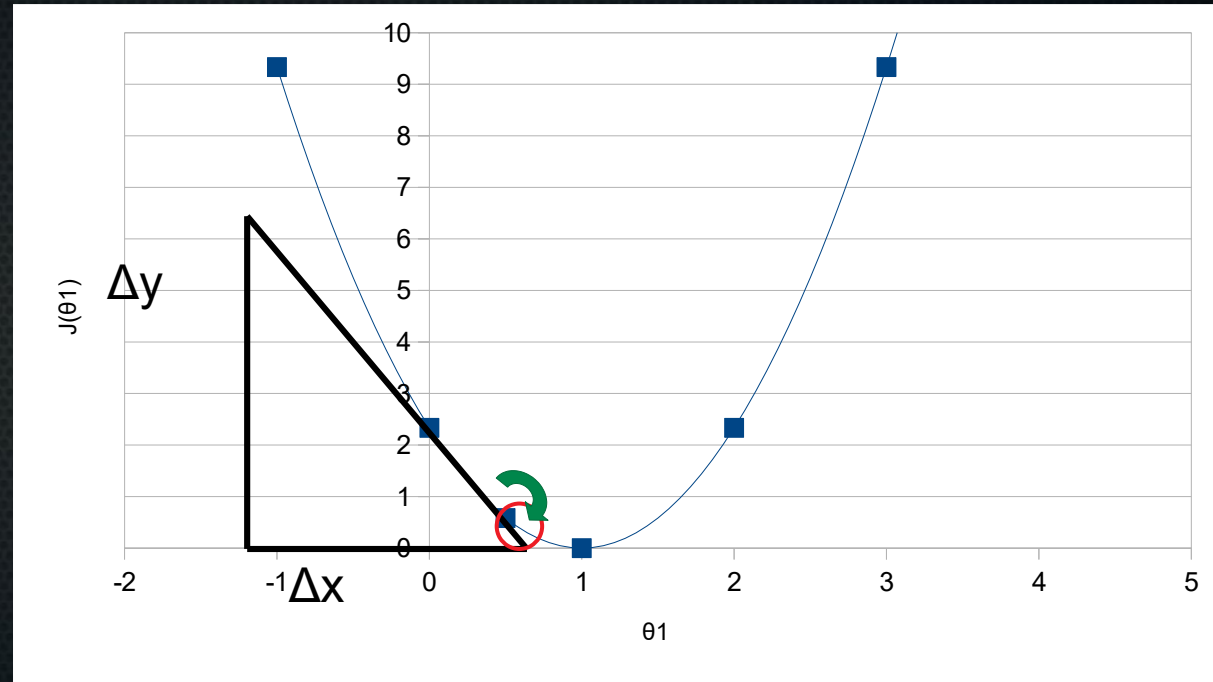
$$\theta_1 = 0 - 0.02 \cdot -5$$

$$\theta_1 = 0.1$$



# Gradient descent

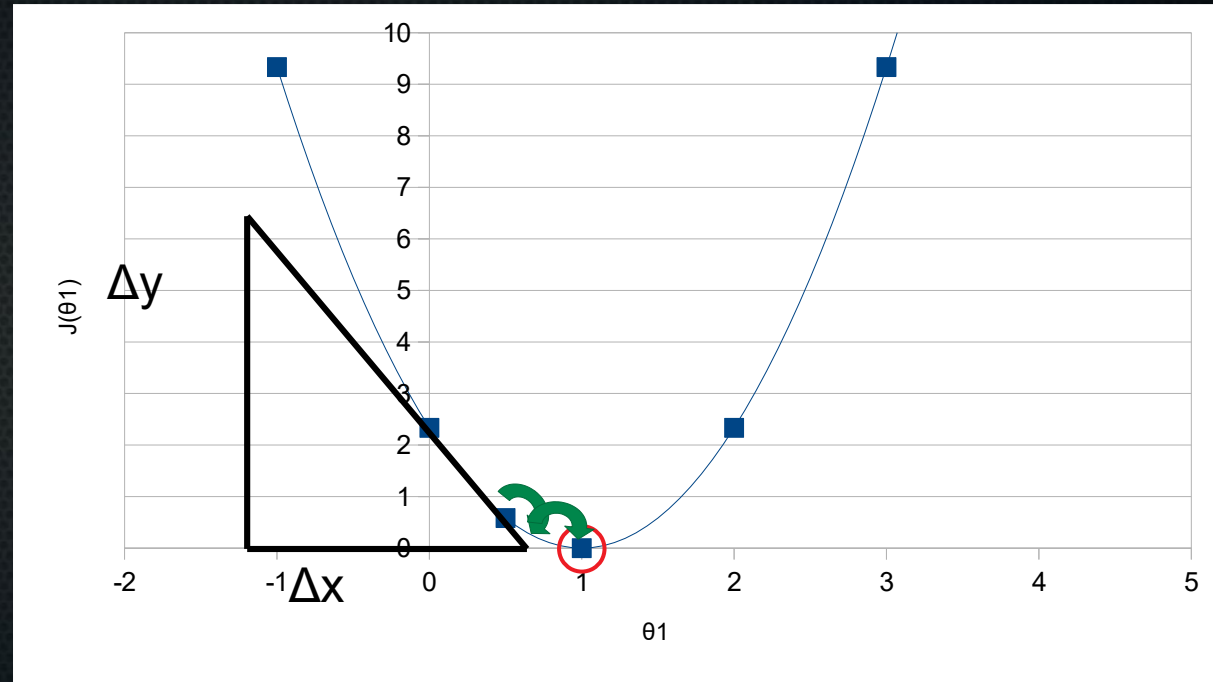
- Want to minimise
- Where to go?
- Change current  $\theta_1$
- Note: gradient becomes smaller closer to optimum, so can use fixed value for  $\alpha$





# Gradient descent

- Want to minimise
- Where to go?
- Change current  $\theta_1$
- Note: gradient becomes smaller closer to optimum, so can use fixed value for  $\alpha$



# Gradient descent

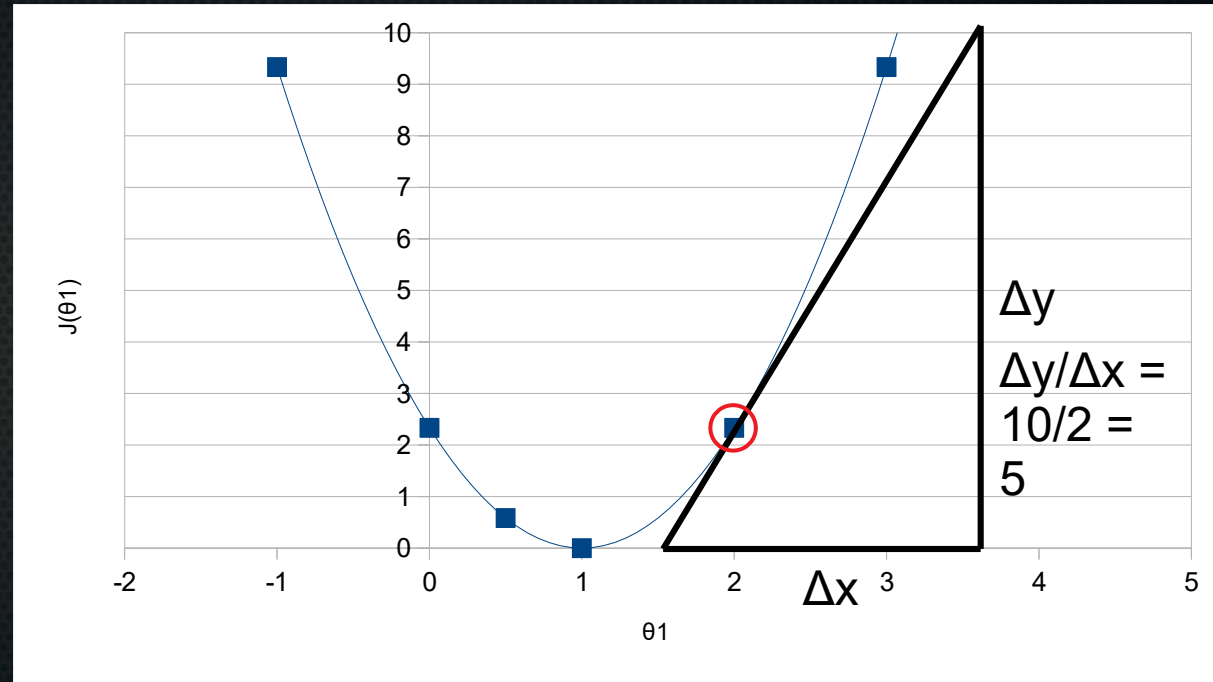
- Works also from other direction.

$$\theta_1 = \theta_1 - \alpha \cdot \frac{d}{d\theta_1} J(\theta_1)$$

$$\alpha = 0.2$$

$$\theta_1 = 2 - 0.2 \cdot 5$$

$$\theta_1 = 1$$



# Gradient descent

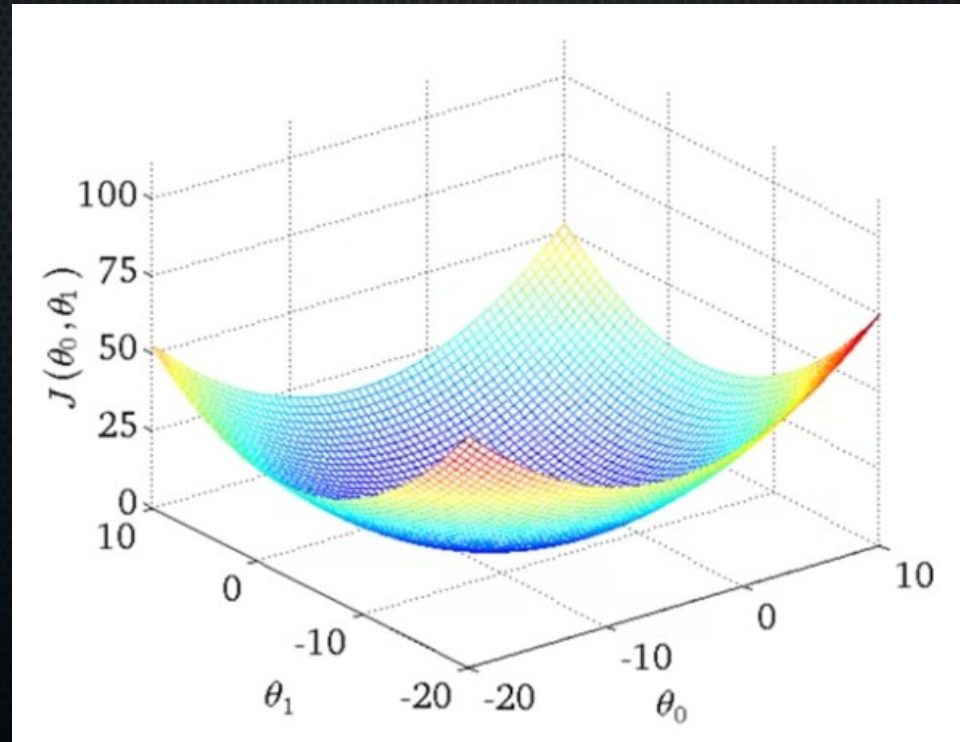
---

- Iteratively descend down the gradient of the cost function until convergence → optimal parameters!



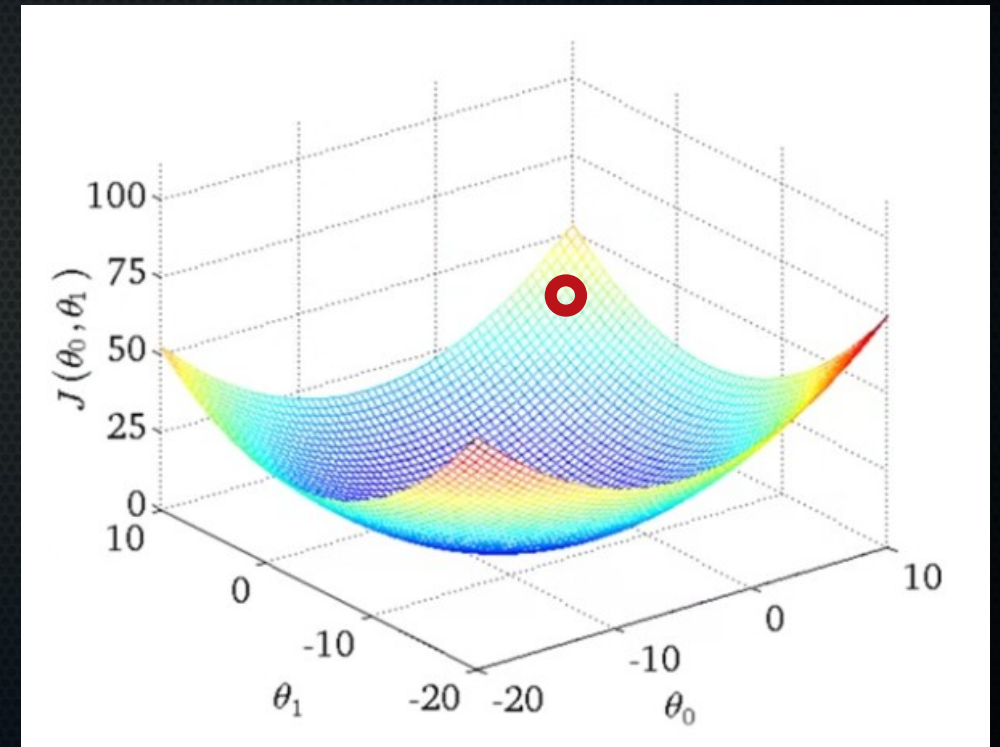
# Gradient descent

- Iteratively descend down the gradient of the cost function until convergence → optimal parameters!
- In reality: not one-dimensional:



# Gradient descent: partial derivatives

- Iteratively descend down the gradient of the cost function until convergence → optimal parameters!
- In reality: not one-dimensional. Want to fit intercept *and* slope.
- Use partial derivatives instead:

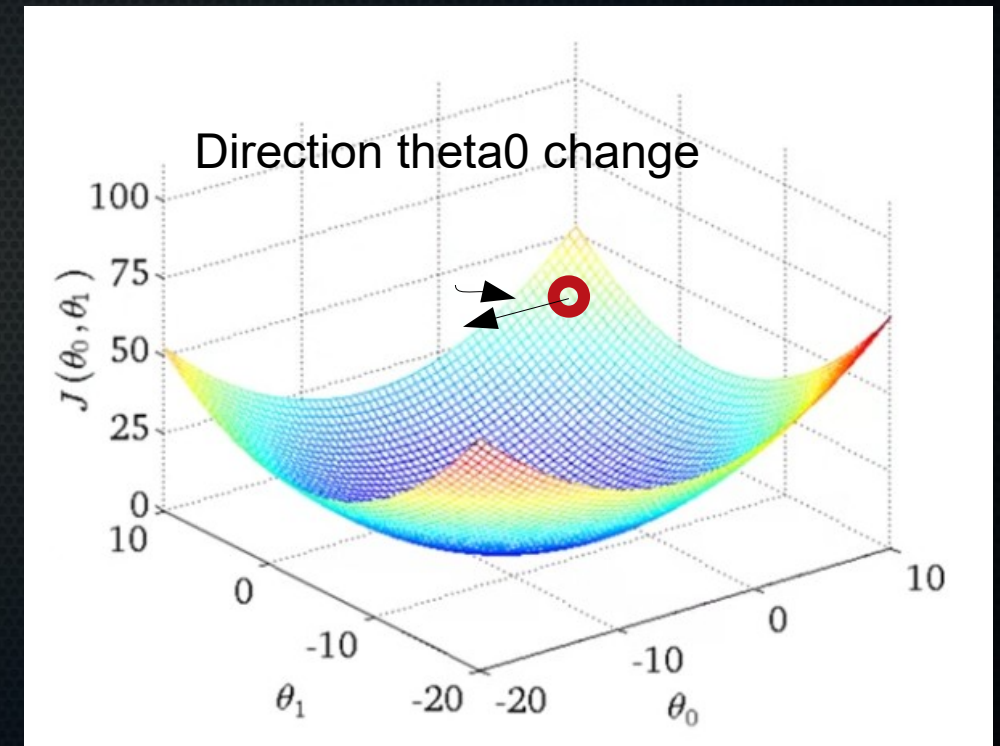


Source: Andrew Ng, Coursera



# Gradient descent: partial derivatives

- Iteratively descend down the gradient of the cost function until convergence → optimal parameters!
- In reality: not one-dimensional. Want to fit intercept *and* slope.
- Use partial derivatives instead:

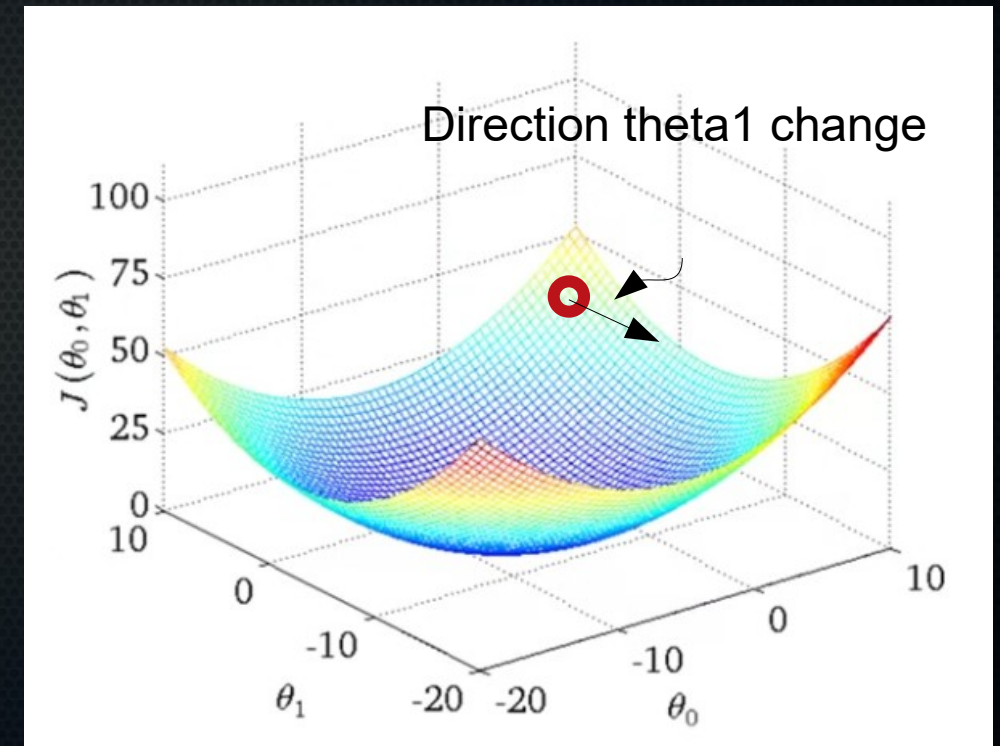


Source: Andrew Ng, Coursera



# Gradient descent: partial derivatives

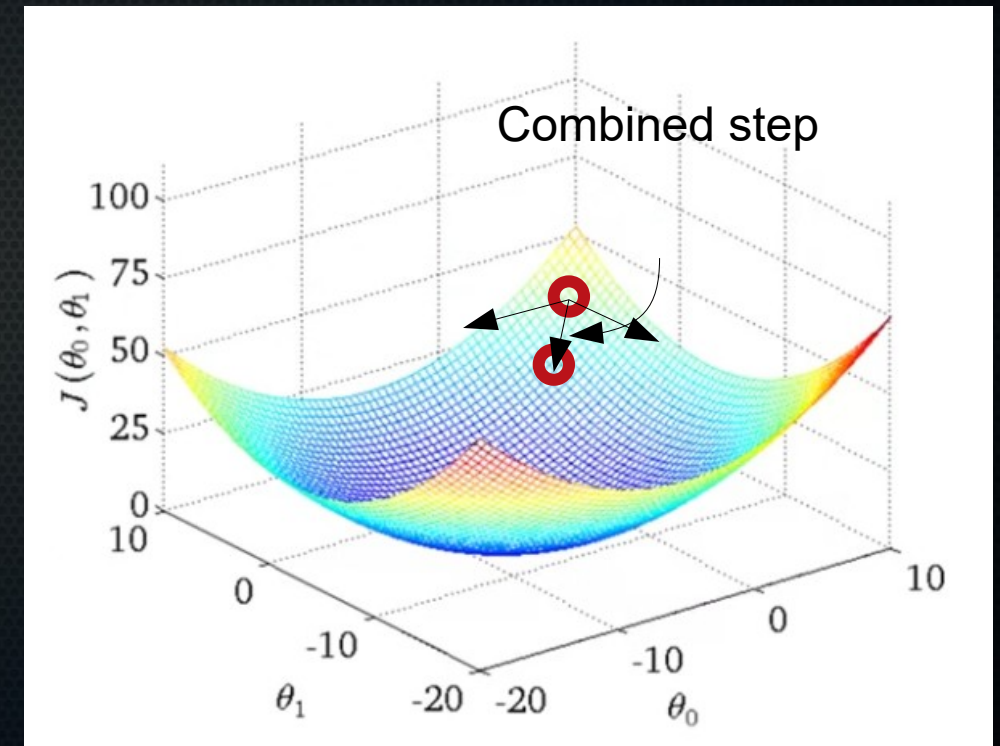
- Iteratively descend down the gradient of the cost function until convergence → optimal parameters!
- In reality: not one-dimensional. Want to fit intercept *and* slope.
- Use partial derivatives instead:



Source: Andrew Ng, Coursera

# Gradient descent: partial derivatives

- Iteratively descend down the gradient of the cost function until convergence → optimal parameters!
- In reality: not one-dimensional. Want to fit intercept *and* slope.
- Use partial derivatives instead:



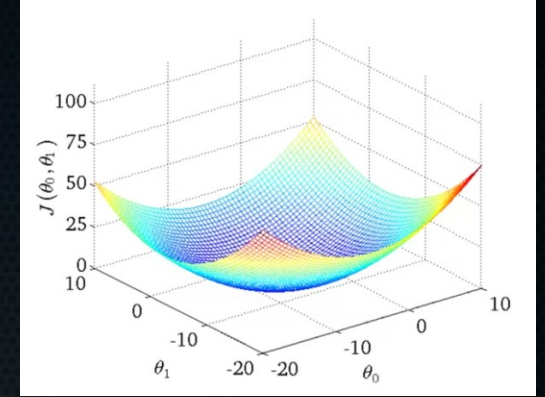
Source: Andrew Ng, Coursera



# Gradient descent: partial derivatives

- Partial derivatives:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$



Source: Andrew Ng, Coursera



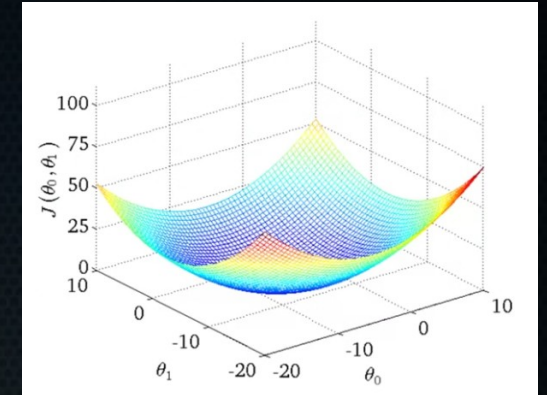
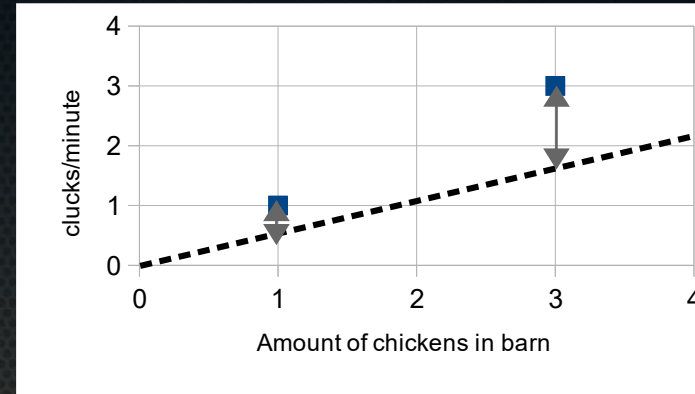
# Gradient descent: partial derivatives

- Partial derivatives:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$m=2$$

$$J(\theta_0, \theta_1) = \frac{1}{2 \cdot 2} ((\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)})^2 + (\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)})^2)$$



Source: Andrew Ng, Coursera

$$f(g(x))$$

$$\frac{dy}{dx} = f'(g(x)) \times g'(x)$$

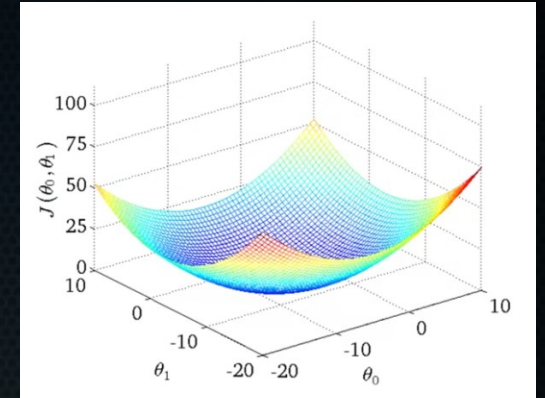
# Gradient descent: partial derivatives

- Partial derivatives:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$m=2$$

$$J(\theta_0, \theta_1) = \frac{1}{2 \cdot 2} \left( \underline{(\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)})^2} + \underline{(\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)})^2} \right)$$



Source: Andrew Ng, Coursera

$$f(\underline{g(x)})$$

$$\frac{dy}{dx} = f'(g(x)) \times g'(x)$$



# Gradient descent: partial derivatives

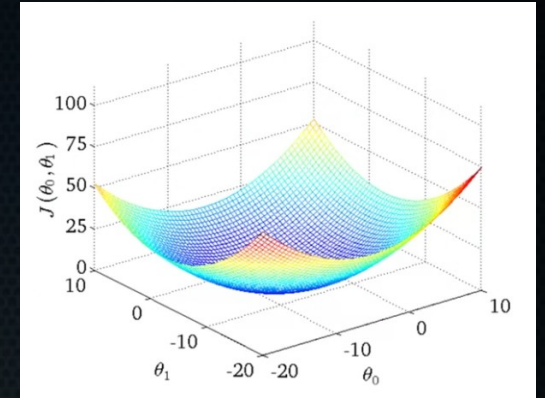
- Partial derivatives:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$m=2$$

$$J(\theta_0, \theta_1) = \frac{1}{2 \cdot 2} \left( \underline{(\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)})^2} + \underline{(\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)})^2} \right)$$

$$\begin{aligned} \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) &= \frac{1}{2 \cdot 2} \left( \underline{2(\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)})} * 1 \right. \\ &\quad \left. + \underline{2(\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)})} * 1 \right) \end{aligned}$$



Source: Andrew Ng, Coursera

$$f(\underline{g(x)})$$

$$\frac{dy}{dx} = \underline{f'(g(x))} \times g'(x)$$



# Gradient descent: partial derivatives

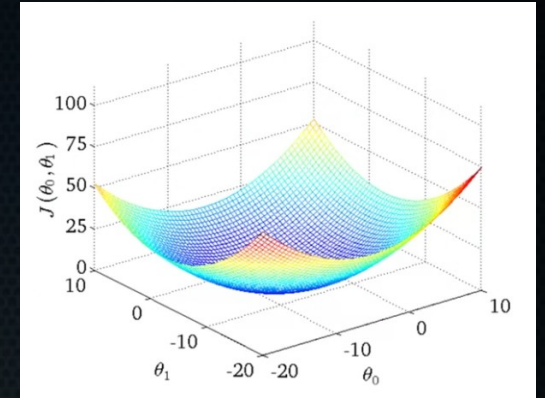
- Partial derivatives:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$m=2$$

$$J(\theta_0, \theta_1) = \frac{1}{2 \cdot 2} \left( \underline{(\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)})^2} + \underline{(\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)})^2} \right)$$

$$\begin{aligned} \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = & \frac{1}{2 \cdot 2} \left( \underline{2(\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)})} * \underline{1} \right. \\ & \left. + \underline{2(\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)})} * \underline{1} \right) \end{aligned}$$



Source: Andrew Ng, Coursera

$$f(\underline{g(x)})$$

$$\frac{dy}{dx} = \underline{f'(g(x))} \times \underline{g'(x)}$$

# Gradient descent: partial derivatives

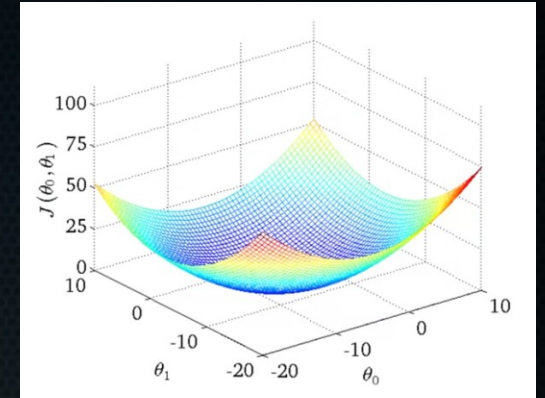
- Partial derivatives:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$m=2$$

$$J(\theta_0, \theta_1) = \frac{1}{2 \cdot 2} \left( \underline{(\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)})^2} + \underline{(\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)})^2} \right)$$

$$\underline{g'(x) = \frac{\partial}{\partial \theta_0} (1 \cdot \theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)}) = 1}$$



Source: Andrew Ng, Coursera

$$f(\underline{g(x)})$$

$$\frac{dy}{dx} = \underline{f'(g(x))} \times \underline{g'(x)}$$



# Gradient descent: partial derivatives

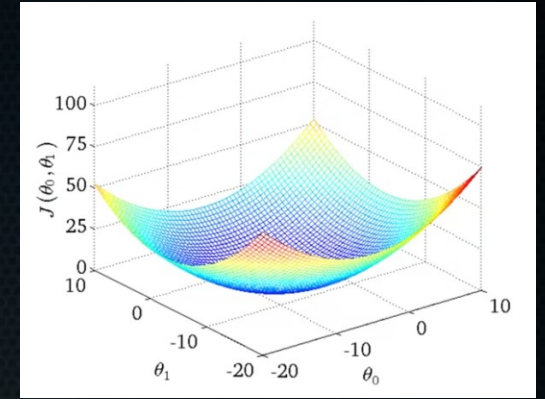
- Partial derivatives:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$m=2$$

$$J(\theta_0, \theta_1) = \frac{1}{2 \cdot 2} \left( \underline{(\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)})^2} + \underline{(\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)})^2} \right)$$

$$\underline{f'(g(x)) = \frac{\partial}{\partial \theta_0} (g(x))^2 = 2 \cdot g(x)}$$



Source: Andrew Ng, Coursera

$$f(\underline{g(x)})$$

$$\frac{dy}{dx} = \underline{f'(g(x))} \times \underline{g'(x)}$$



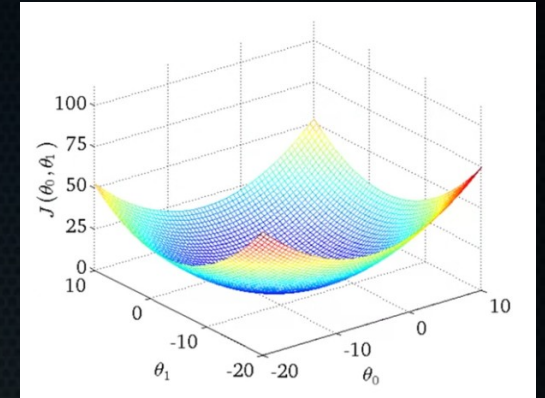
# Gradient descent: partial derivatives

- Partial derivatives:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$m=2$$

$$\begin{aligned} \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = & \frac{1}{2 \cdot 2} (2(\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)}) * 1 \\ & + 2(\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)}) * 1) \end{aligned}$$



Source: Andrew Ng, Coursera

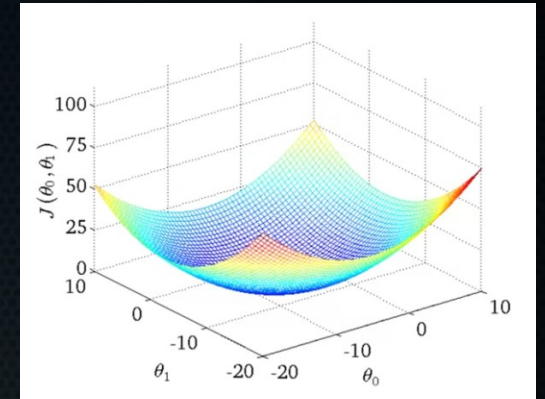
# Gradient descent: partial derivatives

- Partial derivatives:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$m=2$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{2 \cdot 2} \left( \underbrace{2(\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)})}_{\text{red}} * 1 + \underbrace{2(\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)})}_{\text{red}} * 1 \right)$$



Source: Andrew Ng, Coursera



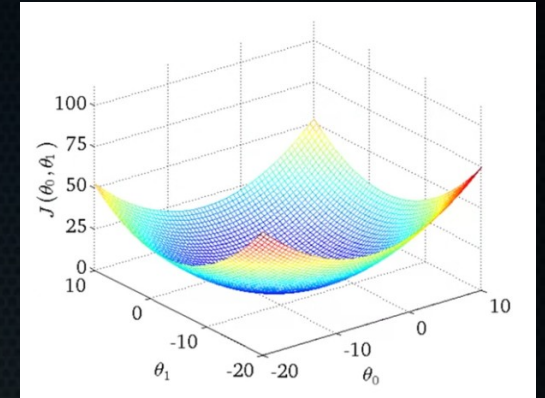
# Gradient descent: partial derivatives

- Partial derivatives:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$m=2$$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{2 \cdot 2} (2((\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)}) * 1 + (\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)}) * 1))$$



Source: Andrew Ng, Coursera



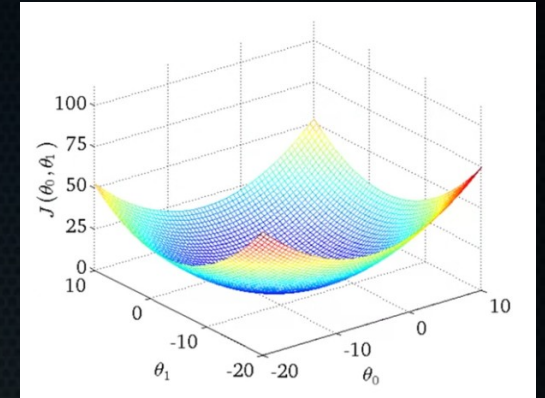
# Gradient descent: partial derivatives

- Partial derivatives:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$m=2$$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{\cancel{2} \cdot 2} (\cancel{2}((\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)}) * 1 + (\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)}) * 1))$$



Source: Andrew Ng, Coursera

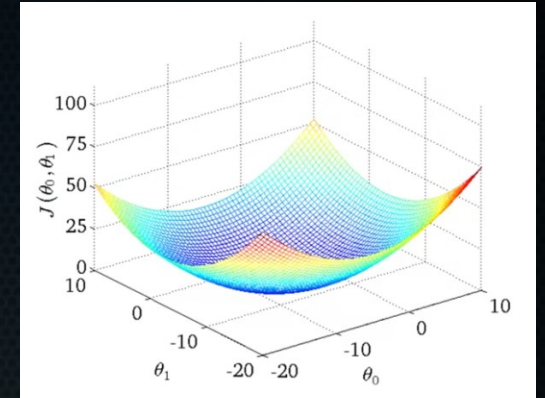
# Gradient descent: partial derivatives

- Partial derivatives:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$m=2$$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{2} ((\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)}) + (\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)}))$$



Source: Andrew Ng, Coursera



# Gradient descent: partial derivatives

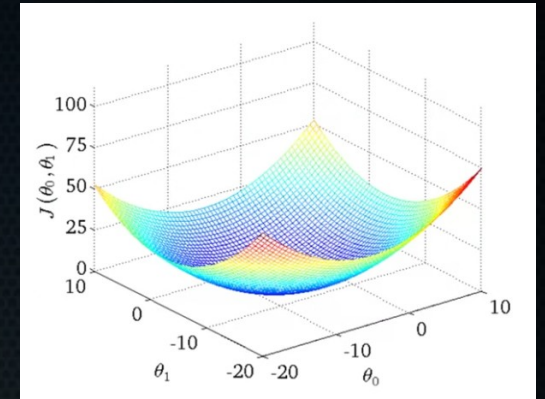
- Partial derivatives:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$m=2$$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{2} ((\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)}) + (\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)}))$$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})$$



Source: Andrew Ng, Coursera



# Gradient descent: partial derivatives

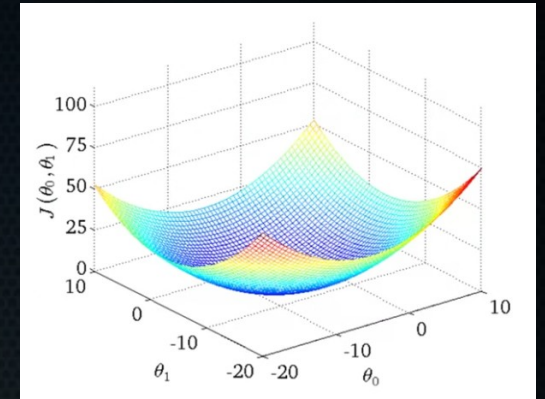
- Partial derivatives:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$m=2$$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{2} ((\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)}) + (\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)}))$$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)}) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})$$



Source: Andrew Ng, Coursera

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

# Gradient descent: partial derivatives

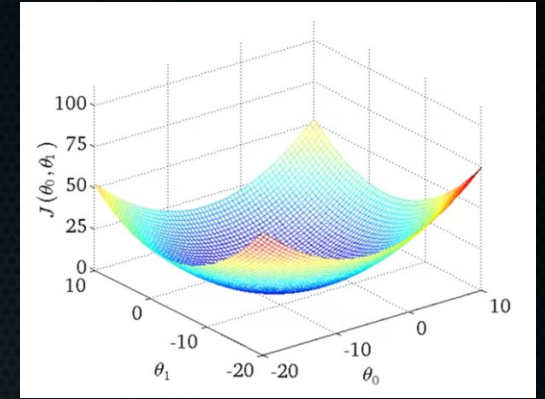
- Partial derivatives:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 \cdot x^{(i)} - y^{(i)})^2$$

$$m=2$$

$$J(\theta_0, \theta_1) = \frac{1}{2 \cdot 2} ((\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)})^2 + (\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)})^2)$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{2 \cdot 2} (2(\theta_0 + \theta_1 \cdot x^{(1)} - y^{(1)}) * \underline{x^{(1)}} + 2(\theta_0 + \theta_1 \cdot x^{(2)} - y^{(2)}) * \underline{x^{(2)}})$$



Source: Andrew Ng, Coursera

$$f(g(x))$$

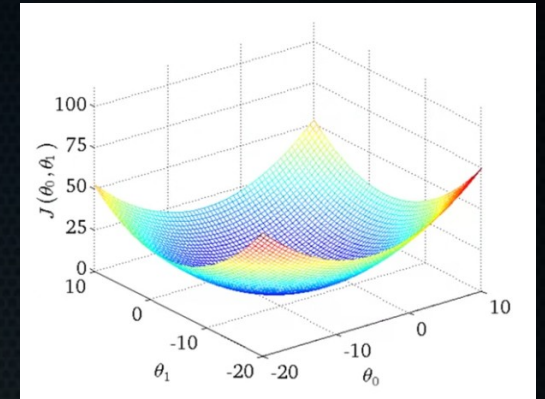
$$\frac{dy}{dx} = f'(g(x)) * \underline{g'(x)}$$



# Gradient descent: partial derivatives

- Partial derivative theta1:

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m ((h_{\theta}(x^{(i)}) - y^{(i)}) \cdot x^{(i)})$$



Source: Andrew Ng, Coursera



# Cost function and gradient descent

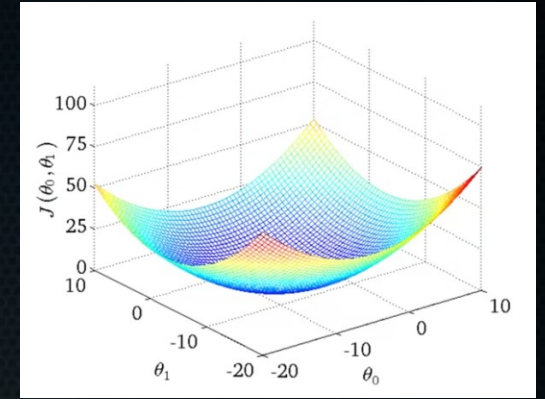
- Cost function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

- Partial derivatives for gradient descent:

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m ((h_{\theta}(x^{(i)}) - y^{(i)}) \cdot x^{(i)})$$



Source: Andrew Ng, Coursera

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

# Cost function and gradient descent

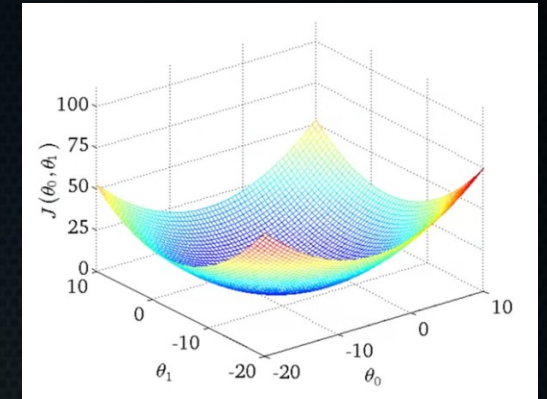
- Cost function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x) - y^{(i)})^2$$

- Gradient descent update:

$$\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \theta_0 - \frac{\alpha}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \theta_1 - \frac{\alpha}{m} \sum_{i=1}^m ((h_{\theta}(x^{(i)}) - y^{(i)}) \cdot x^{(i)})$$



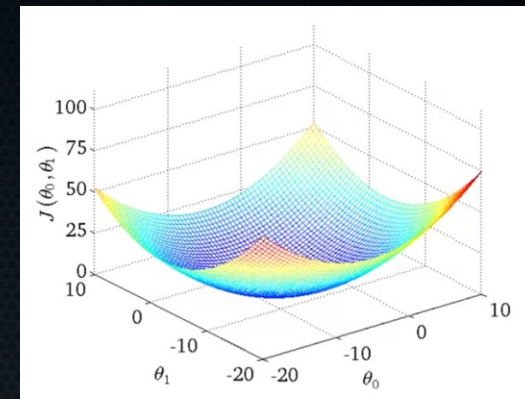
Source: Andrew Ng, Coursera

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



# What about $\alpha$ ?

- Learning rate, so-called hyperparameter.

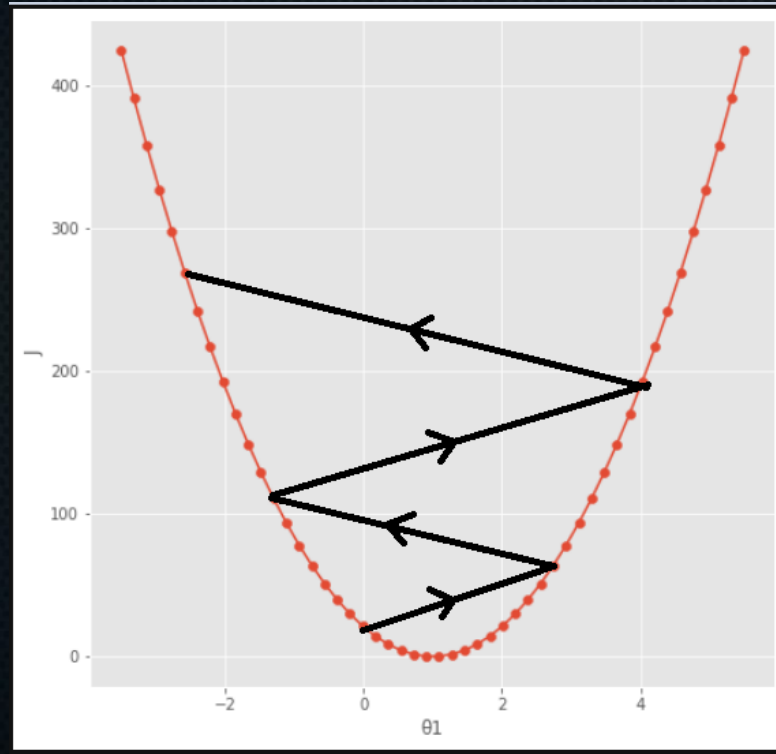


Source: Andrew Ng, Coursera

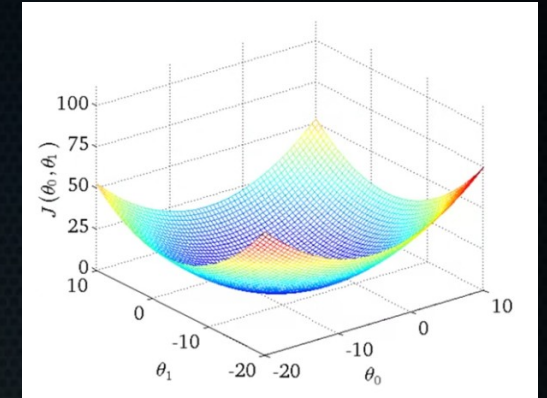
$$\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \theta_0 - \frac{\alpha}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})$$

# What about $\alpha$ ?

- Learning rate, so-called hyperparameter.
- Too high:



Source: <https://towardsdatascience.com/univariate-linear-regression-theory-and-practice-99329845e85d>



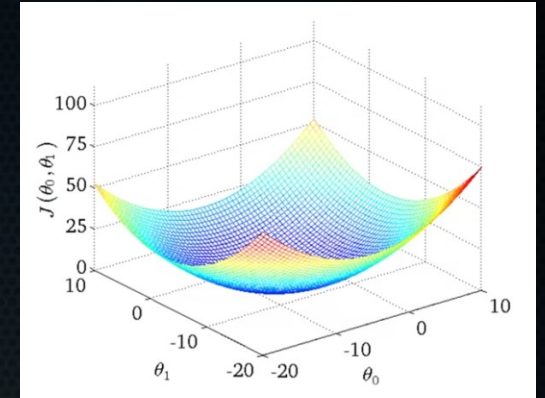
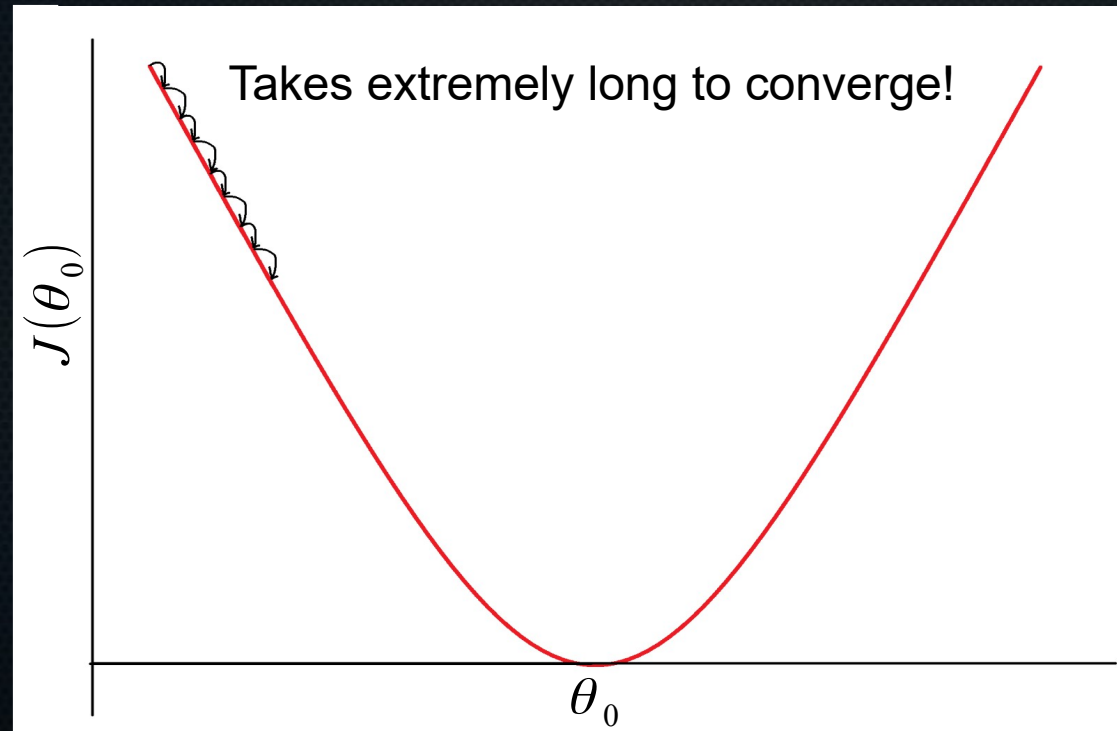
Source: Andrew Ng, Coursera

$$\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \theta_0 - \frac{\alpha}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})$$



# What about $\alpha$ ?

- Learning rate, so-called hyperparameter.
- Too low:

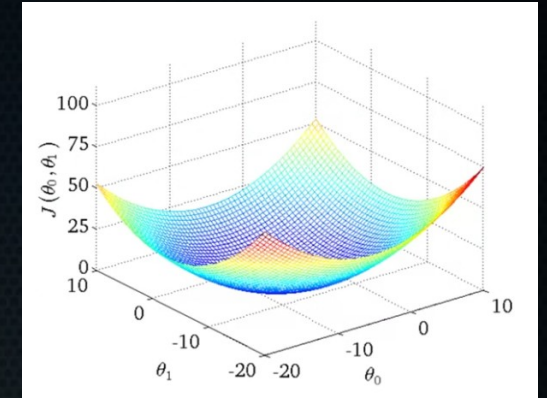


Source: Andrew Ng, Coursera

$$\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \theta_0 - \frac{\alpha}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})$$

# What about $\alpha$ ?

- Learning rate, so-called hyperparameter.
- Will discuss later how we pick it!



Source: Andrew Ng, Coursera



# Summary

---

- Defined a cost function for linear regression
- Showed how gradient descent can be used to minimise this cost function by changing the parameters
- Calculated partial derivatives for use with gradient descent
- Encountered our first hyperparameter,  $\alpha$ , which governs the size of update steps

# Practicals

---

- You should now open and do *Day1/Practical/PracticalMaterialDay1\_ShortPractical1.ipynb*
- Open Anaconda prompt, navigate to *Basic-Machine-Learning-for-Bioinformatics*, and type *jupyter notebook*. Then navigate to and open the correct file in this browser interface.
- DuckDuckGo and Google are your friend: numpy takes some getting used to, so search, search, search!
- I will start the next lecture in about an hour. If you're not finished: don't worry! Just continue where you left off after the lecture.



# Practicals

---

- Feel free to add code cells to experiment in, and always experiment before putting something in a function!
- Handy shortcuts:
  - Ctrl + Enter → run current code cell
  - Ctrl + b → add new cell below current cell
  - Ctrl + a → add new cell above current cell
  - Esc + m → turns selected cell into a markdown cell (for writing notes)
  - Shift tab: go one indentation level back on all selected lines
  - Tab: Go one indentation level deeper on all selected lines
  - You can toggle line numbers under View in the top menu bar.