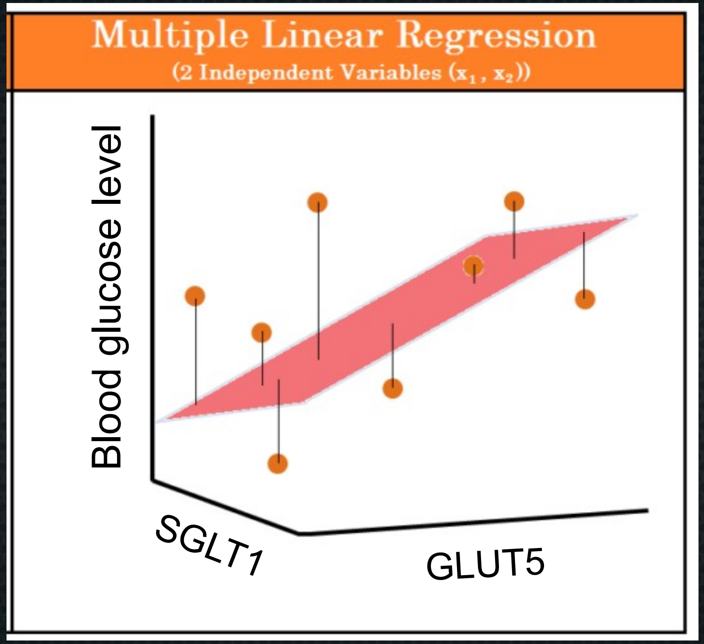
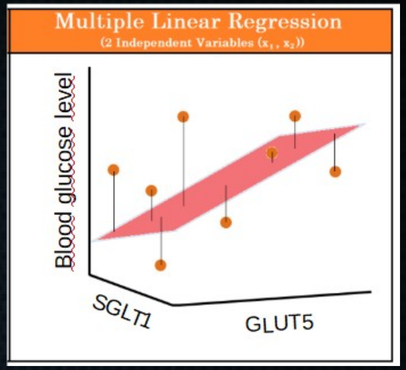
Multiple variables



Multiple variables

- Simple:

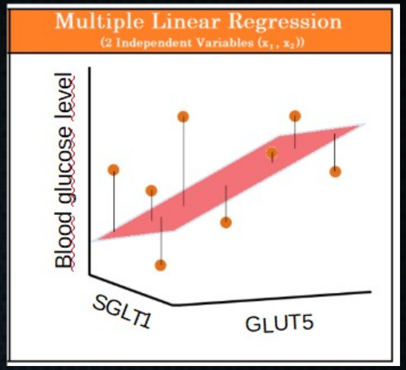
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$



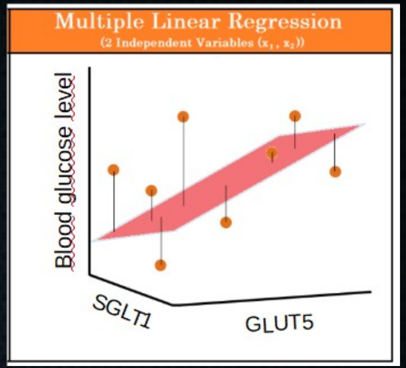
Multiple variables

- Simple:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$



Multiple variables

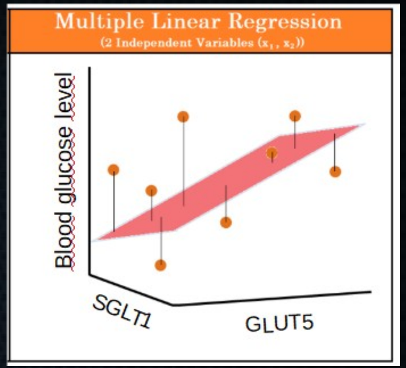


- Simple:

$$h_\theta(x)=\theta_0+\theta_1 x_1+\theta_2 x_2+\ldots+\theta_n x_n$$

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

Multiple variables



- Simple:

$$h_\theta(x)=\theta_0+\theta_1 x_1+\theta_2 x_2+\ldots+\theta_n x_n$$

$$\theta_0=\theta_0-\alpha\frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1,\theta_2,\ldots,\theta_n)=\theta_0-\frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)$$

$$\theta_1=\theta_1-\alpha\frac{\partial}{\partial\theta_1}J(\theta_0,\theta_1,\theta_2,\ldots,\theta_n)=\theta_1-\frac{\alpha}{m}\sum_{i=1}^{m}\left(\left(h_\theta(x^{(i)})-y^{(i)}\right)\cdot x_1^{(i)}\right)$$

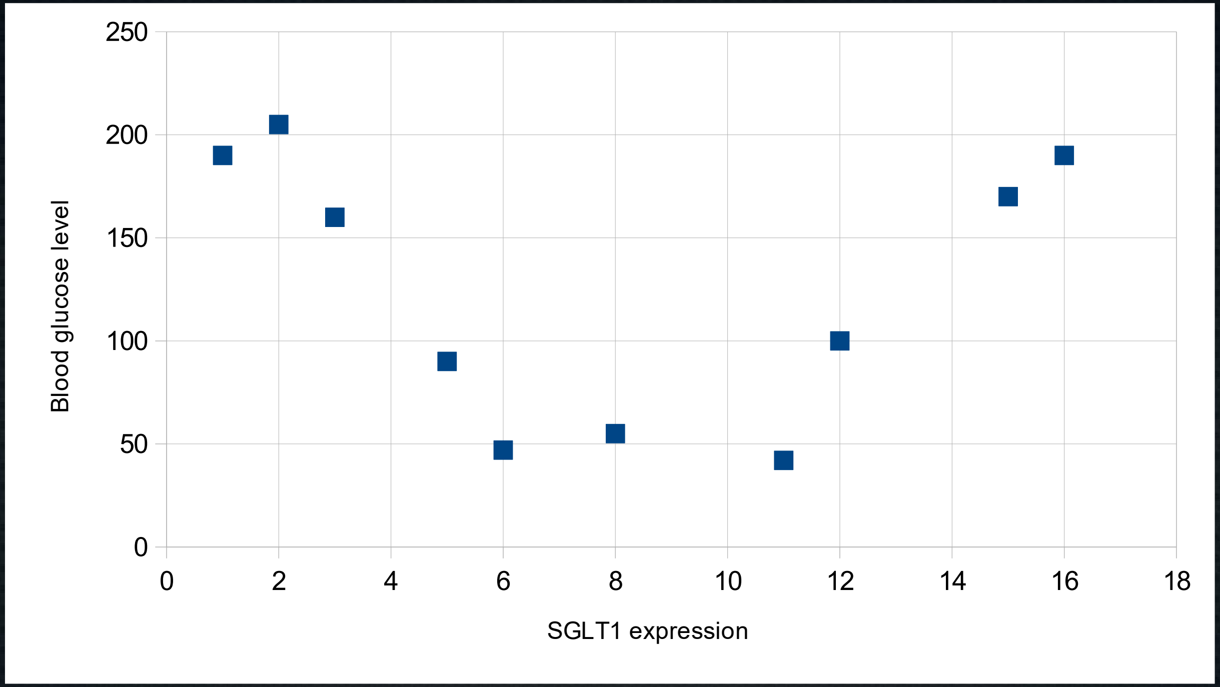
.

.

.

$$\theta_n=\theta_n-\alpha\frac{\partial}{\partial\theta_n}J(\theta_0,\theta_1,\theta_2,\ldots,\theta_n)=\theta_1-\frac{\alpha}{m}\sum_{i=1}^{m}\left(\left(h_\theta(x^{(i)})-y^{(i)}\right)\cdot x_n^{(i)}\right)$$

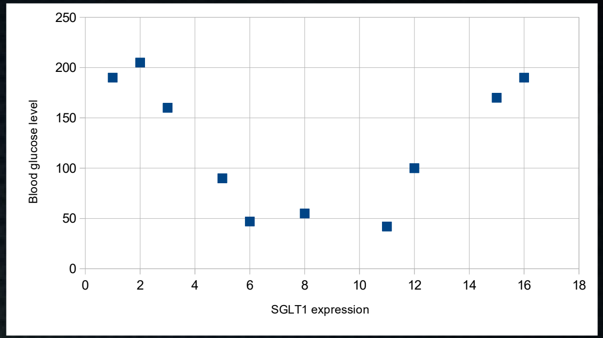
Polynomials/power functions



Polynomials/power functions

- Simple:

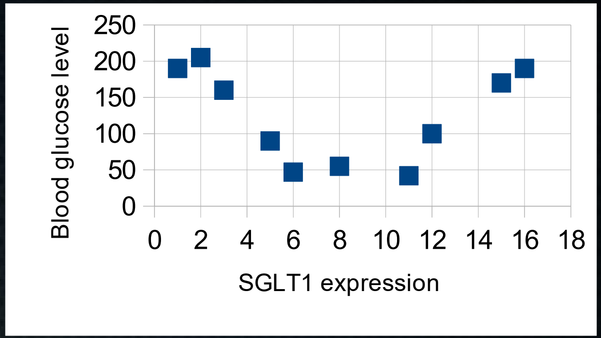
$$h_\theta(x)=\theta_0+\theta_1 x_1+\theta_2 x_2$$



Polynomials/power functions

- Simple:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$



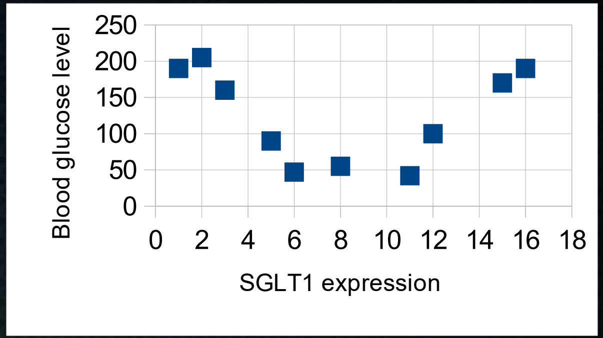
Sample #	SGLT1_linear (x1)	Blood glucose level (mg/dL)
1	3	155
2	8	55
3	12	101
4	2	200

Polynomials/power functions

- Simple:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Square

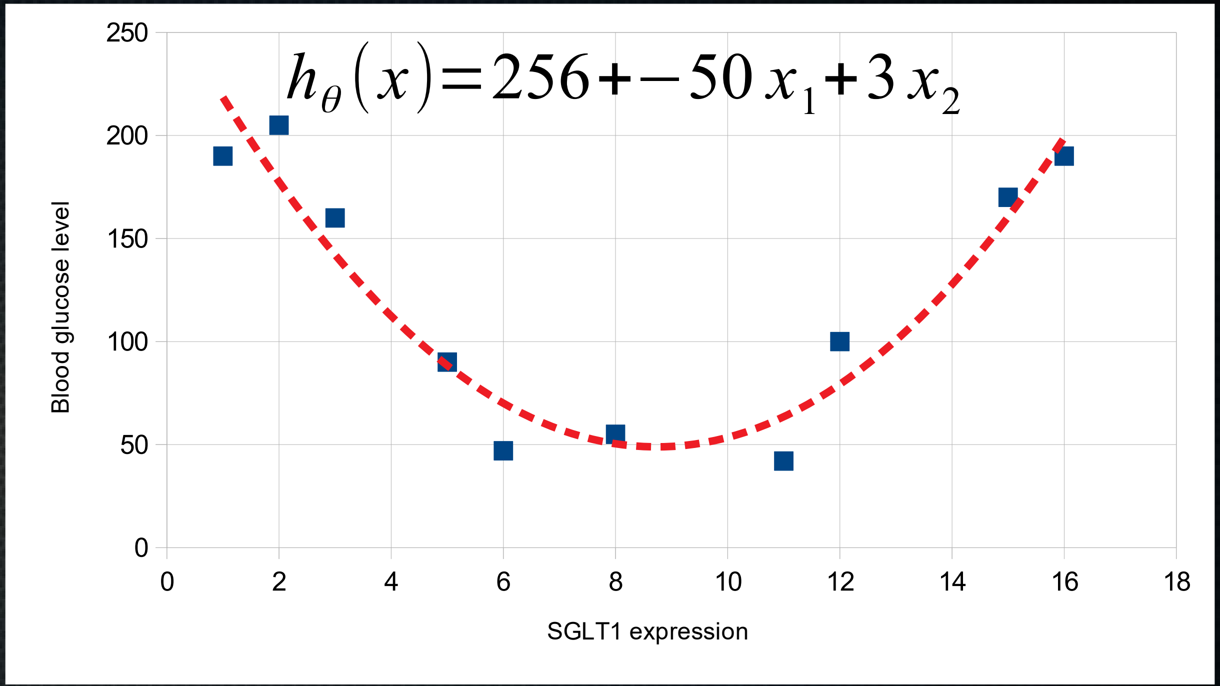


Sample #	SGLT1_linear (x1)	SGLT1_square (x2)	Blood glucose level (mg/dL)
1	3	9	155
2	8	64	55
3	12	144	101
4	2	4	200

Polynomials/power functions

- Simple:

$$h_\theta(x)=\theta_0+\theta_1 x_1+\theta_2 x_2$$



Sample #	SGLT1_linear (x1)	SGLT1_square (x2)	Blood glucose level (mg/dL)
1	3	9	155
2	8	64	55
3	12	144	101
4	2	4	200

Note on feature scaling

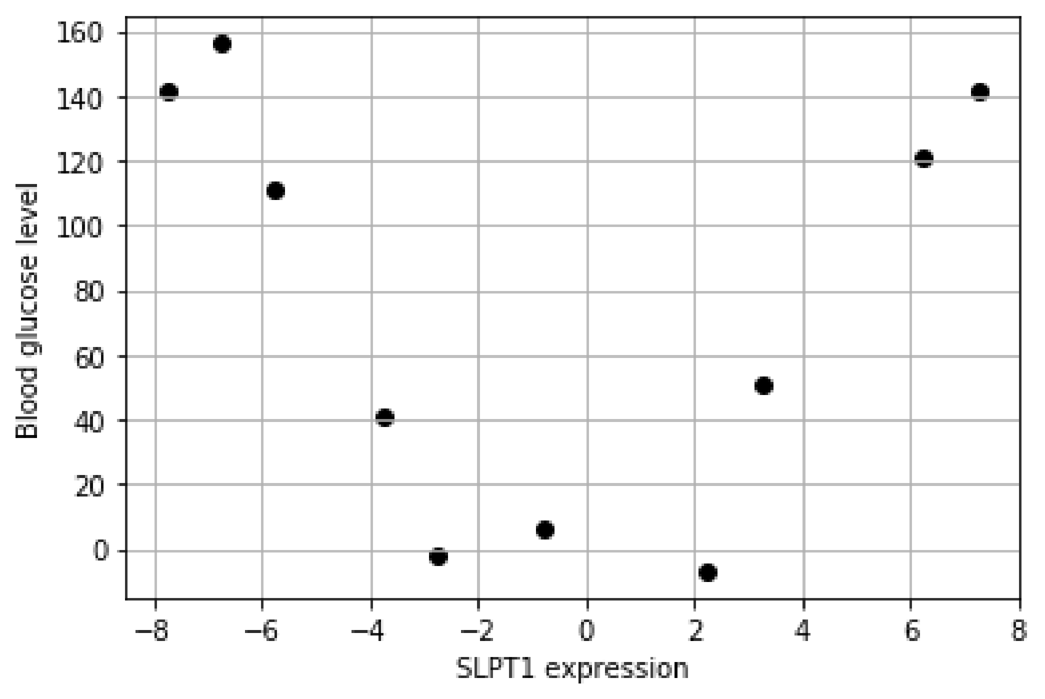
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Sample #	SGLT1_linear (x1)	SGLT1_square (x2)	Blood glucose level (mg/dL)
1	3	9	155
2	8	64	55
3	12	144	101
4	2	4	200

Note on feature scaling

Sample #	SGLT1_linear (x1)	SGLT1_square (x2)	Blood glucose level (mg/dL)
1	3	9	155
2	8	64	55
3	12	144	101
4	2	4	200

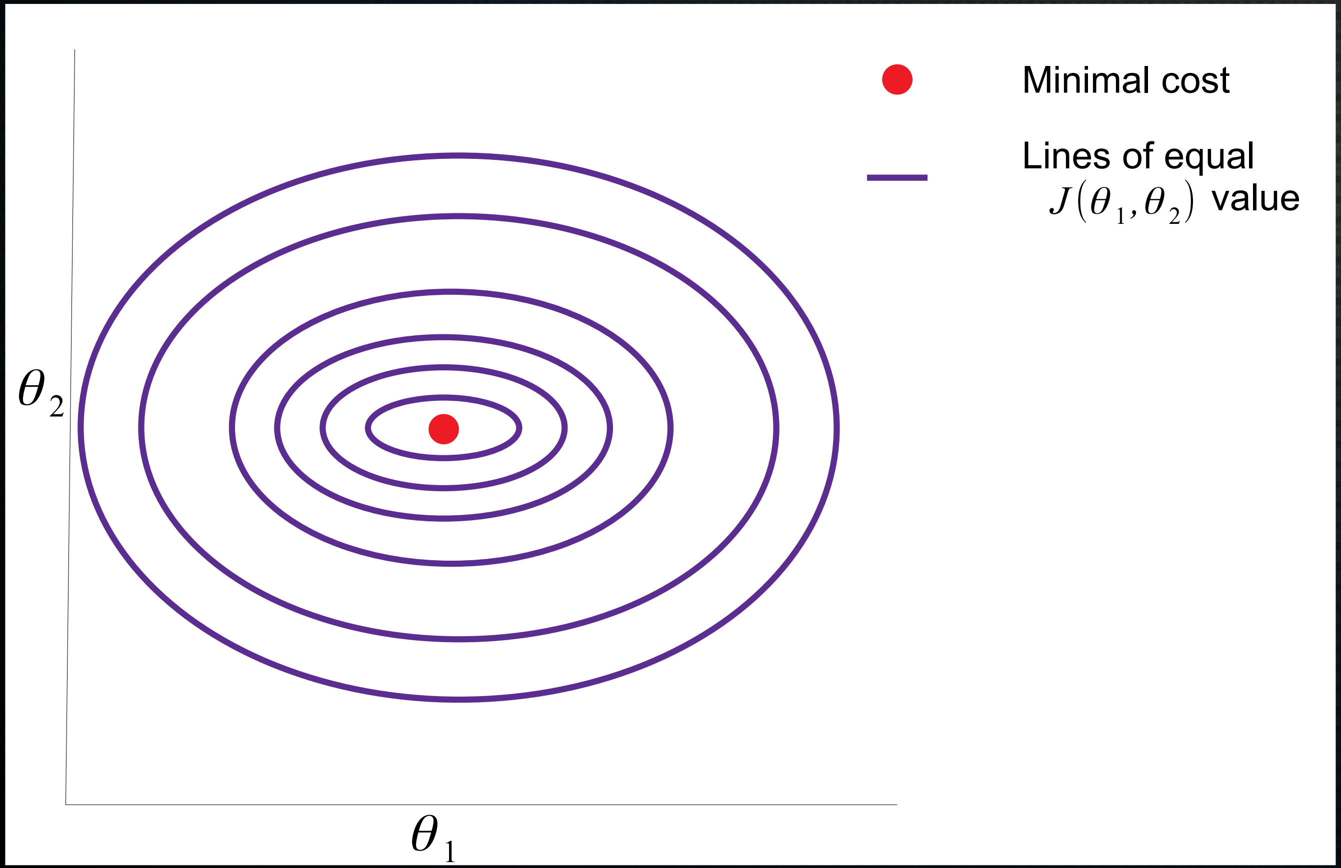
$$h_\theta(x) = \cancel{\theta_0} + \theta_1 x_1 + \theta_2 x_2$$

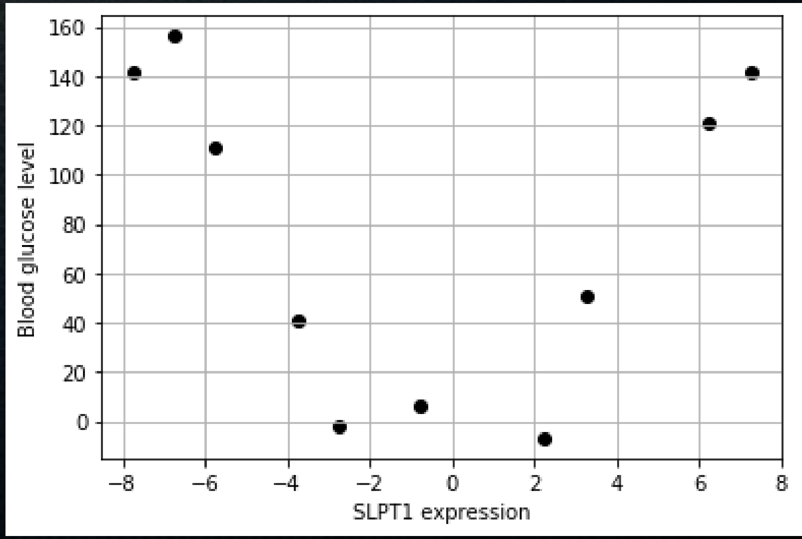


Note on feature scaling

Sample #	SGLT1_linear (x1)	SGLT1_square (x2)	Blood glucose level (mg/dL)
1	3	9	155
2	8	64	55
3	12	144	101
4	2	4	200

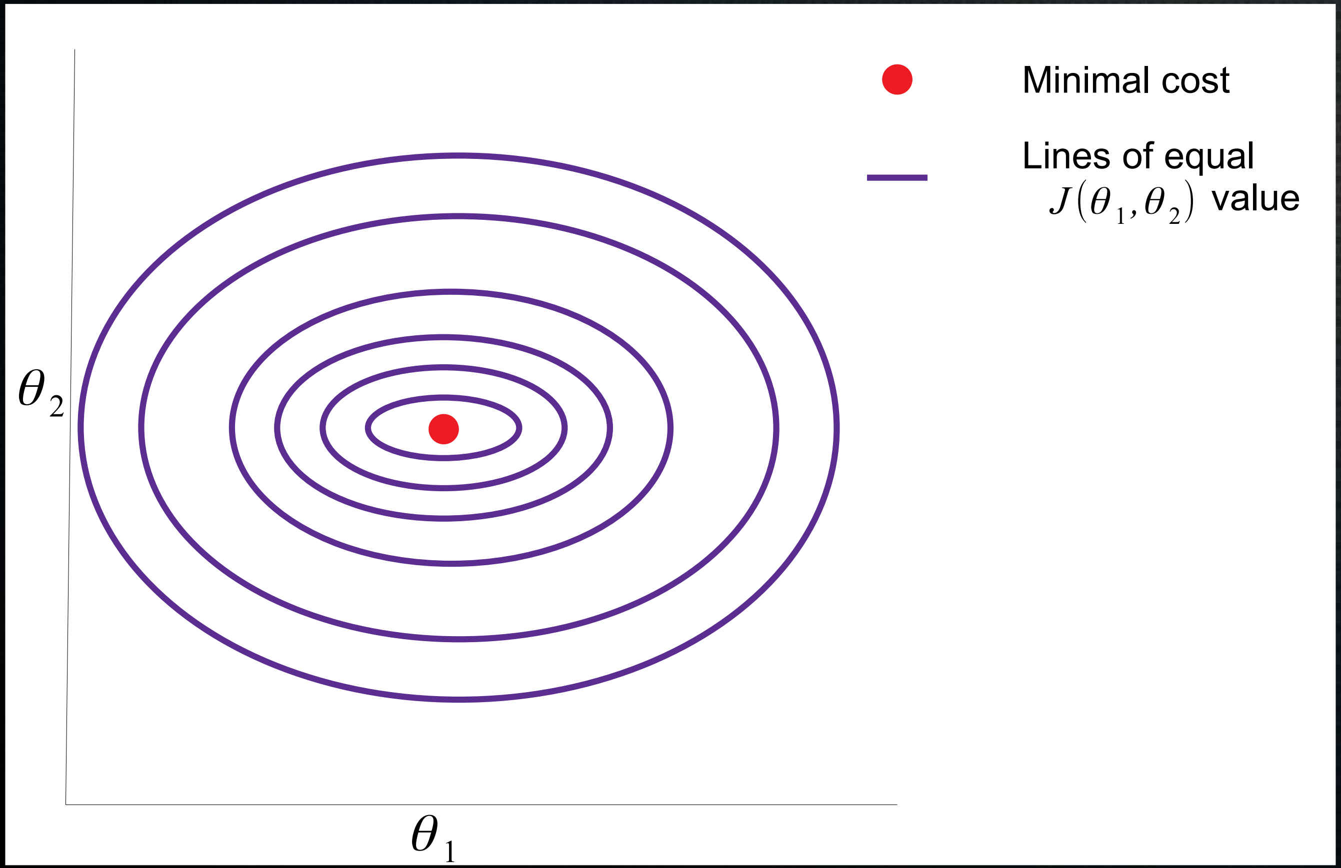
$$h_\theta(x) = \cancel{\theta_0} + \theta_1 x_1 + \theta_2 x_2$$





Note on feature scaling

$$h_\theta(x) = \cancel{\theta_0} + \theta_1 x_1 + \theta_2 x_2$$



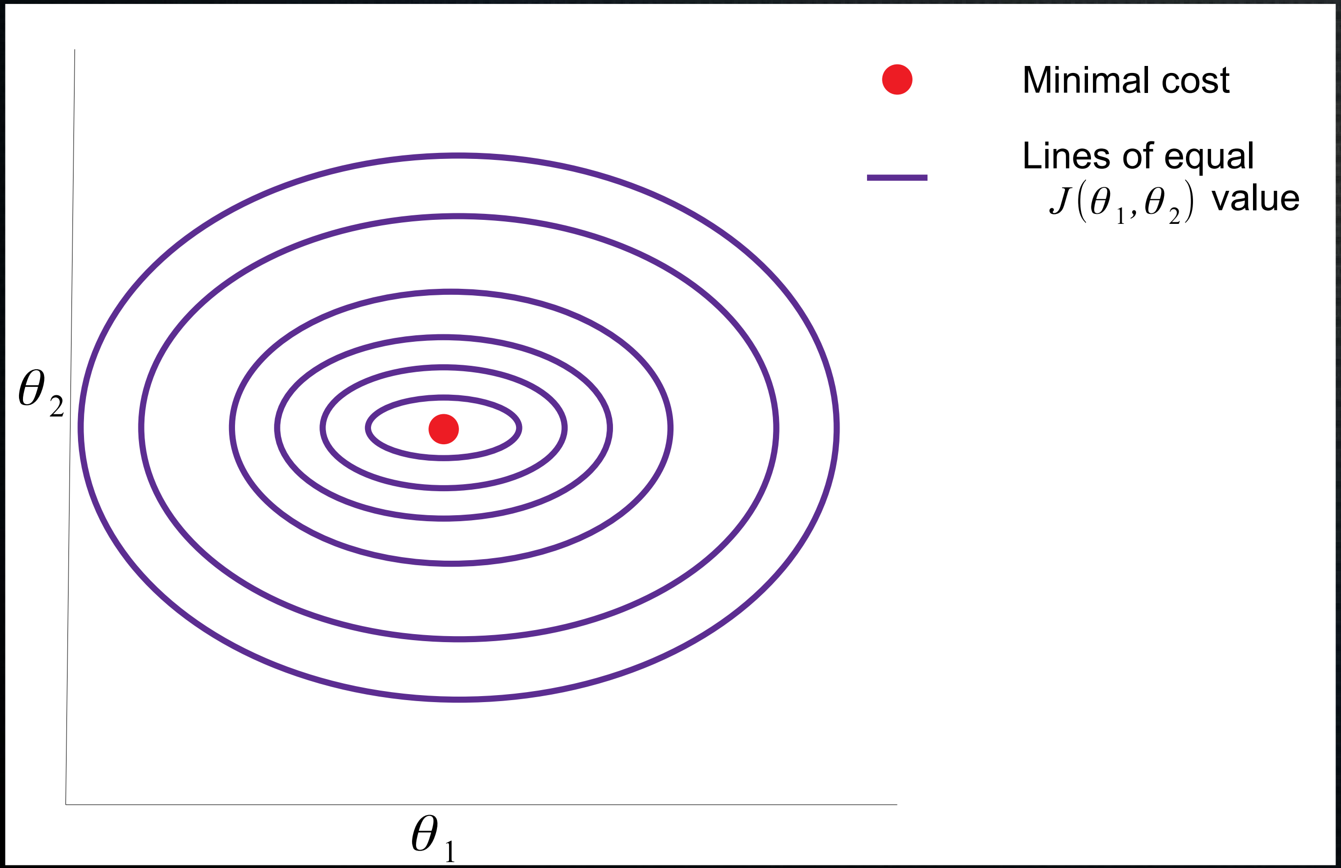
Sample #	SGLT1_linear (x1)	SGLT1_square (x2)	Blood glucose level (mg/dL)
1	3	9	155
2	8	64	55
3	12	144	101
4	2	4	200

Range: 2-12 Range: 4-144

$$\theta_n = \theta_n - \frac{\alpha}{m} \sum_{i=1}^{m} \left(\left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x_n^{(i)} \right)$$

Note on feature scaling

$$h_\theta(x) = \cancel{\theta_0} + \theta_1 x_1 + \theta_2 x_2$$



Sample #	SGLT1_linear (x1)	SGLT1_square (x2)	Blood glucose level (mg/dL)
1	3	9	155
2	8	64	55
3	12	144	101
4	2	4	200

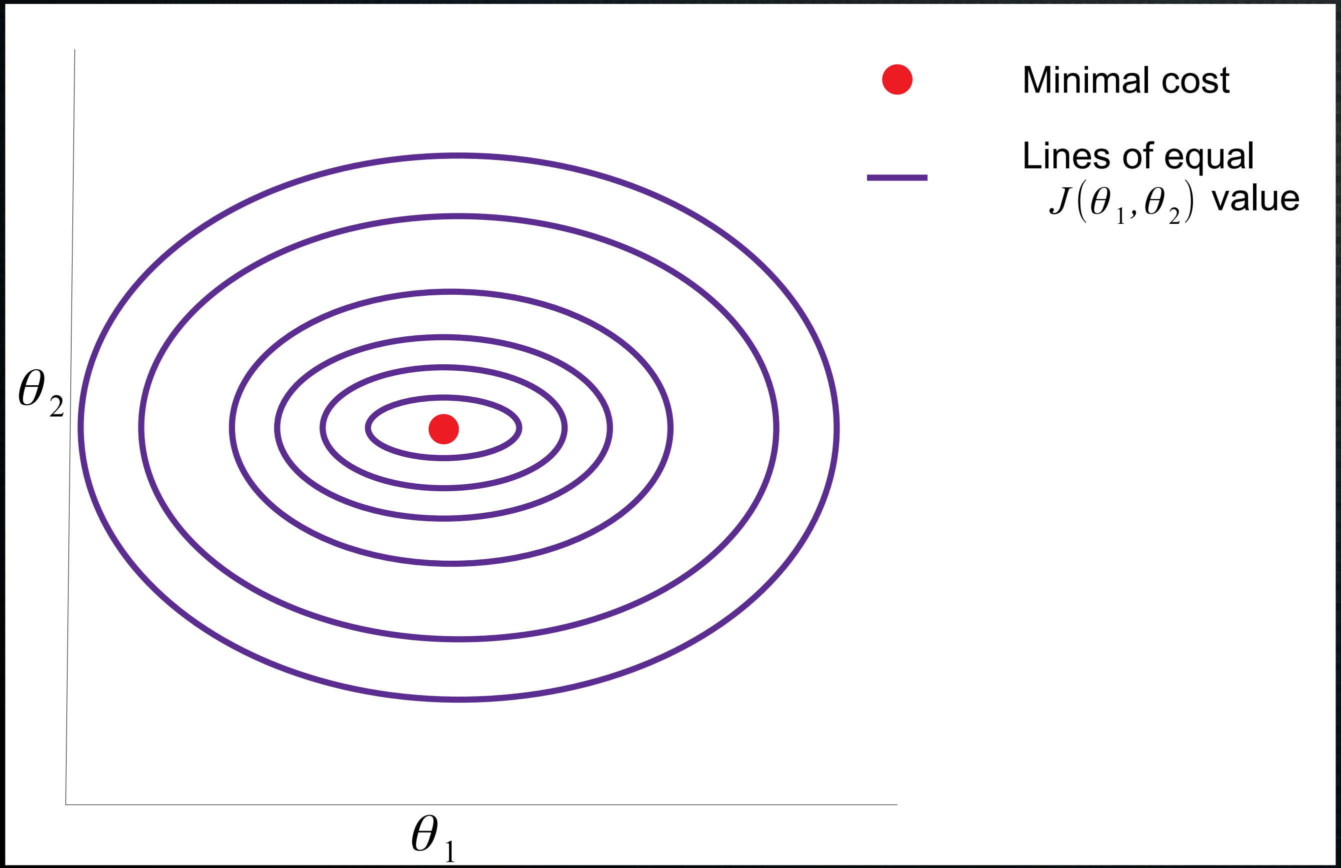
Range: 2-12 Range: 4-144

$$\theta_n = \theta_n - \frac{\alpha}{m} \sum_{i=1}^{m} \left(\left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x_n^{(i)} \right)$$

Small steps

Note on feature scaling

$$h_\theta(x)=\cancel{\theta_0}+\theta_1 x_1+\theta_2 x_2$$



Sample #	SGLT1_linear (x1)	SGLT1_square (x2)	Blood glucose level (mg/dL)
1	3	9	155
2	8	64	55
3	12	144	101
4	2	4	200

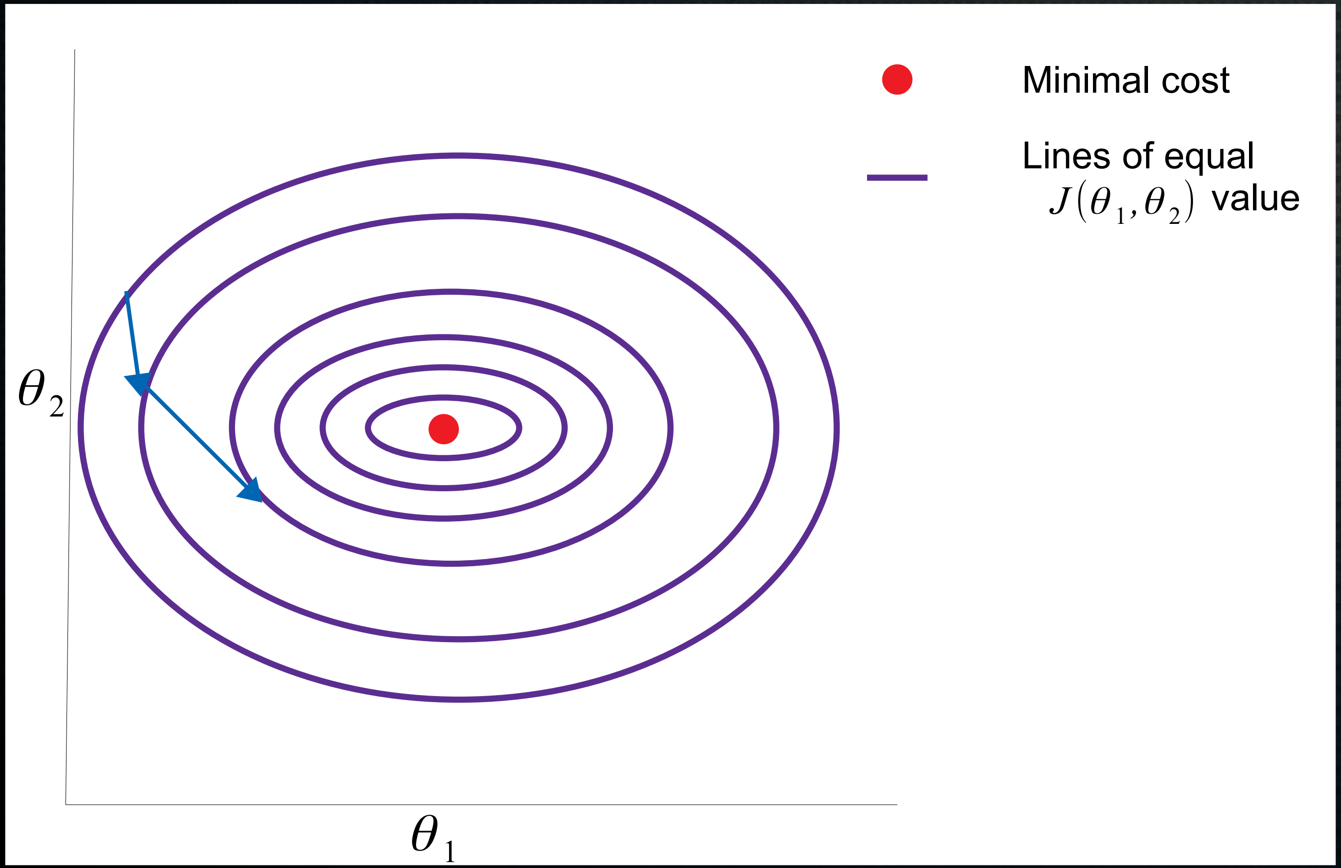
Range: 2-12 Range: 4-144

$$\theta_n=\theta_n-\frac{\alpha}{m}\sum_{i=1}^{m}\left(\left(h_\theta(x^{(i)})-y^{(i)}\right)\cdot x_n^{(i)}\right)$$

Large steps

Note on feature scaling

$$h_\theta(x) = \cancel{\theta_0} + \theta_1 x_1 + \theta_2 x_2$$



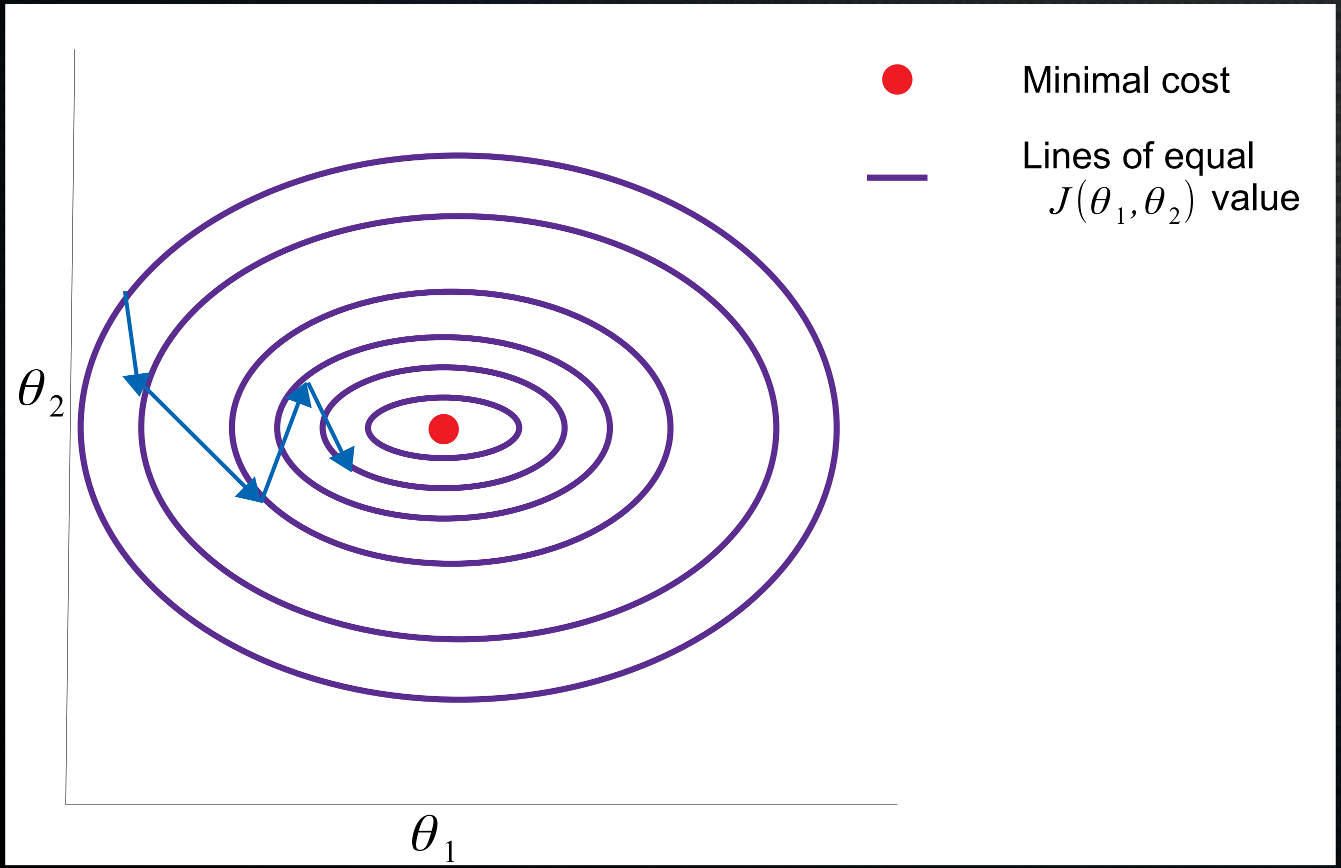
Sample #	SGLT1_linear (x1)	SGLT1_square (x2)	Blood glucose level (mg/dL)
1	3	9	155
2	8	64	55
3	12	144	101
4	2	4	200

Range: 2-12 Range: 4-144

$$\theta_n = \theta_n - \frac{\alpha}{m} \sum_{i=1}^{m} \left(\left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x_n^{(i)} \right)$$

Note on feature scaling

$$h_\theta(x) = \cancel{\theta_0} + \theta_1 x_1 + \theta_2 x_2$$



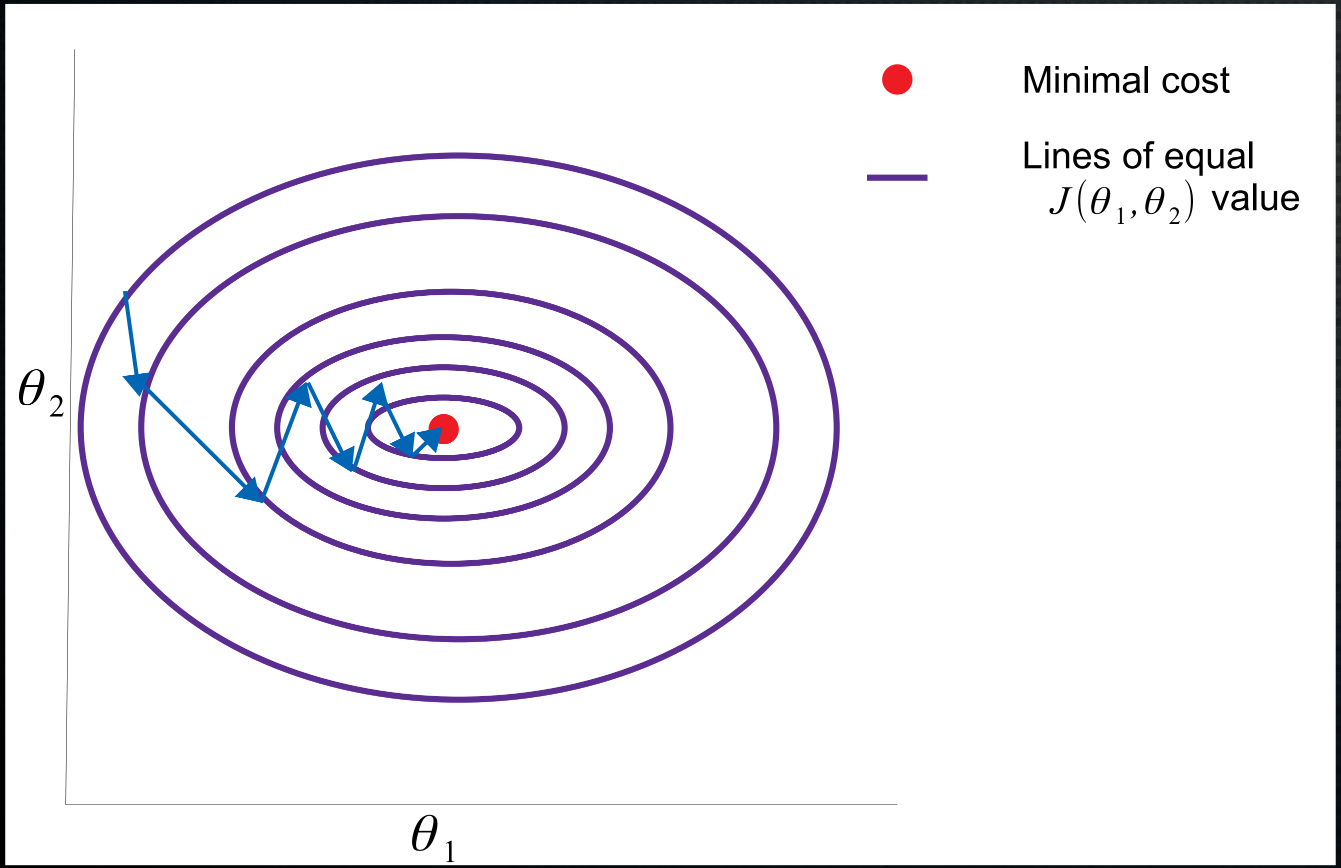
Sample #	SGLT1_linear (x1)	SGLT1_square (x2)	Blood glucose level (mg/dL)
1	3	9	155
2	8	64	55
3	12	144	101
4	2	4	200

Range: 2-12 Range: 4-144

$$\theta_n = \theta_n - \frac{\alpha}{m} \sum_{i=1}^{m} ((h_\theta(x^{(i)}) - y^{(i)}) \cdot x_n^{(i)})$$

Note on feature scaling

$$h_\theta(x)=\cancel{\theta_0}+\theta_1 x_1+\theta_2 x_2$$



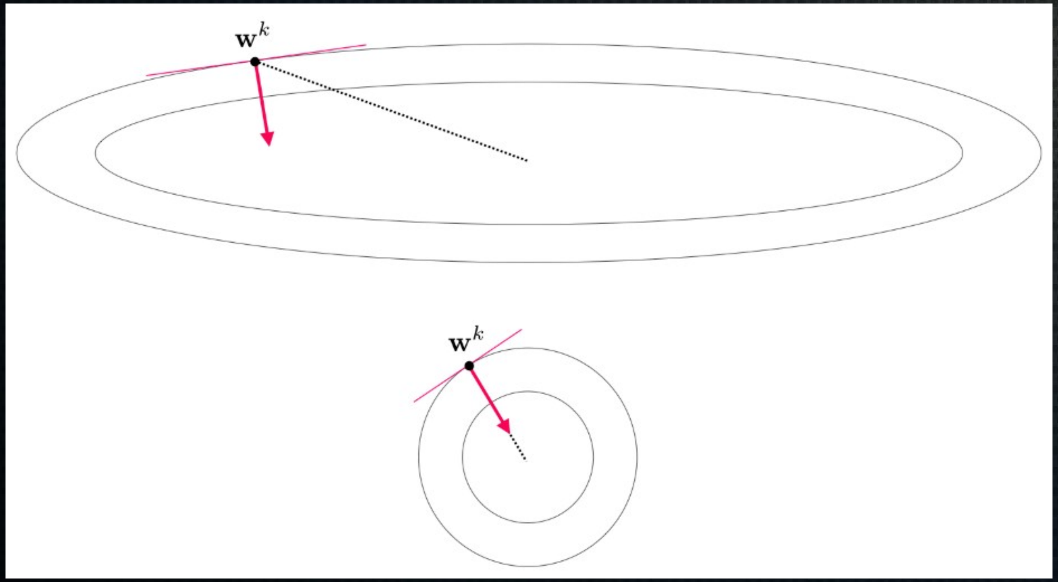
Sample #	SGLT1_linear (x1)	SGLT1_square (x2)	Blood glucose level (mg/dL)
1	3	9	155
2	8	64	55
3	12	144	101
4	2	4	200

Range: 2-12 Range: 4-144

$$\theta_n=\theta_n-\frac{\alpha}{m}\sum_{i=1}^{m}\left(\left(h_\theta(x^{(i)})-y^{(i)}\right)\cdot x_n^{(i)}\right)$$

Note on feature scaling

$$h_\theta(x) = \cancel{\theta_0} + \theta_1 x_1 + \theta_2 x_2$$



Sample #	SGLT1_linear (x1)	SGLT1_square (x2)	Blood glucose level (mg/dL)
1	3	9	155
2	8	64	55
3	12	144	101
4	2	4	200

Range: 2-12 Range: 4-144

Note on feature scaling

$$h_\theta(x) = \cancel{\theta_0} + \theta_1 x_1 + \theta_2 x_2$$

$$x_j = \frac{x_j - mean(x_j)}{std.dev(x_j)}$$

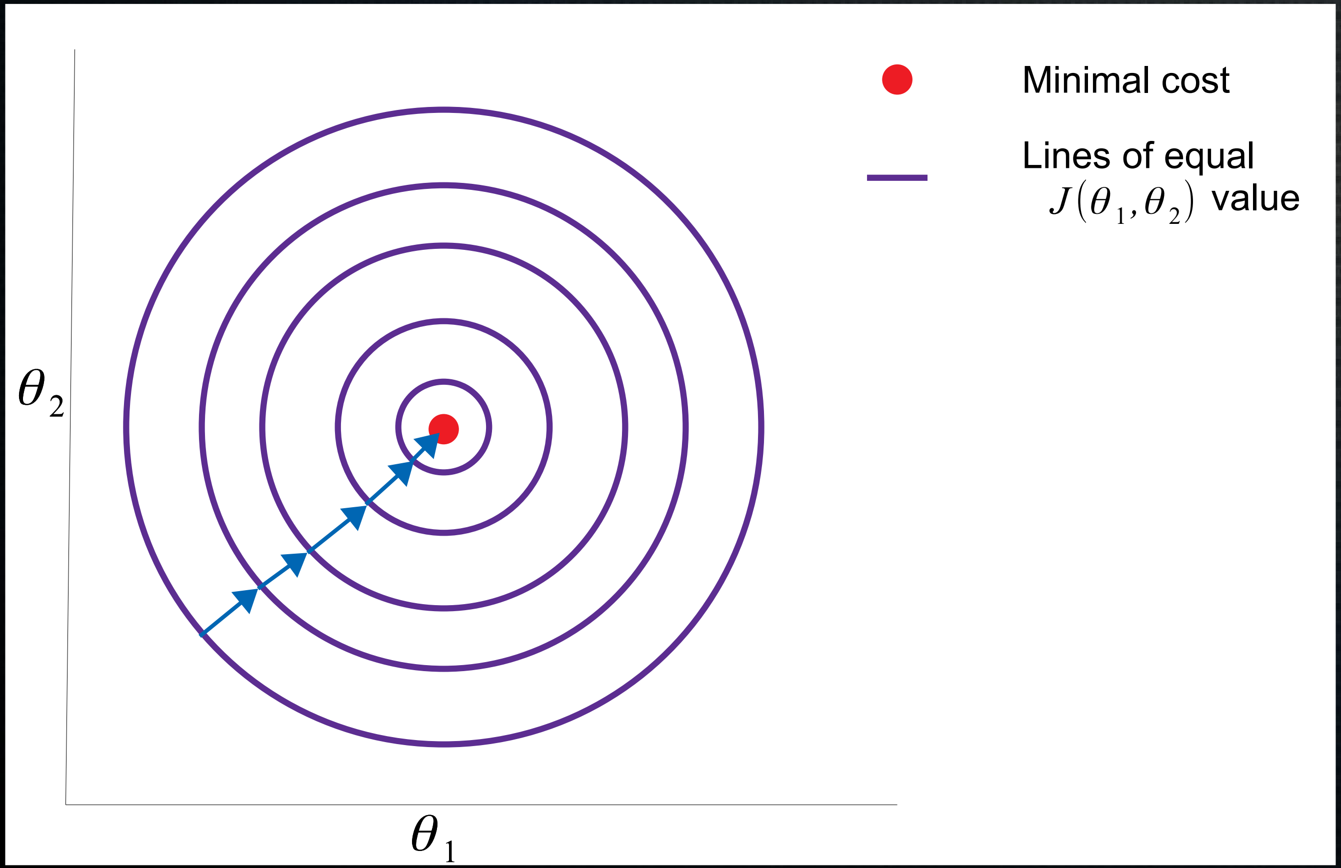
Sample #	SGLT1_linear (x1)	SGLT1_square (x2)	Blood glucose level (mg/dL)
1	-0,70	-0,71	155
2	0,38	0,13	55
3	1,24	1,36	101
4	-0,91	-0,79	200

Note on feature scaling

$$h_\theta(x) = \cancel{\theta_0} + \theta_1 x_1 + \theta_2 x_2$$

$$x_j = \frac{x_j - mean(x_j)}{std.dev(x_j)}$$

Sample #	SGLT1_linear (x1)	SGLT1_square (x2)	Blood glucose level (mg/dL)
1	-0,70	-0,71	155
2	0,38	0,13	55
3	1,24	1,36	101
4	-0,91	-0,79	200

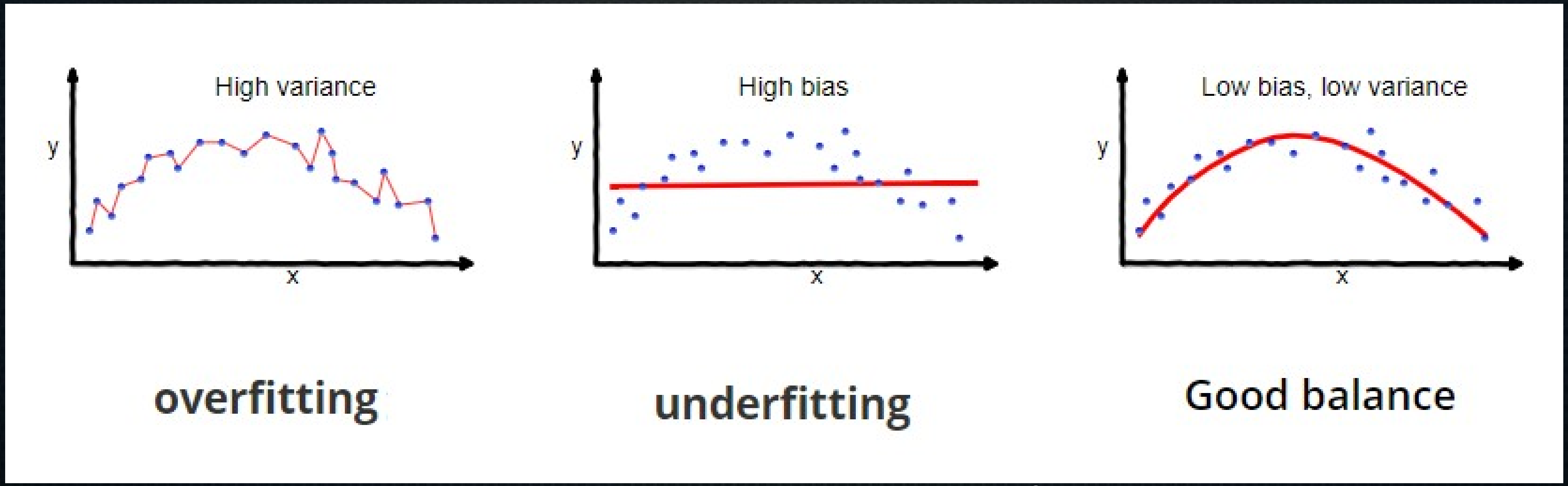


Summary

- Easily extendable to multiple features and polynomials
- Normalisation to help gradient descent converge faster

How to generalise well

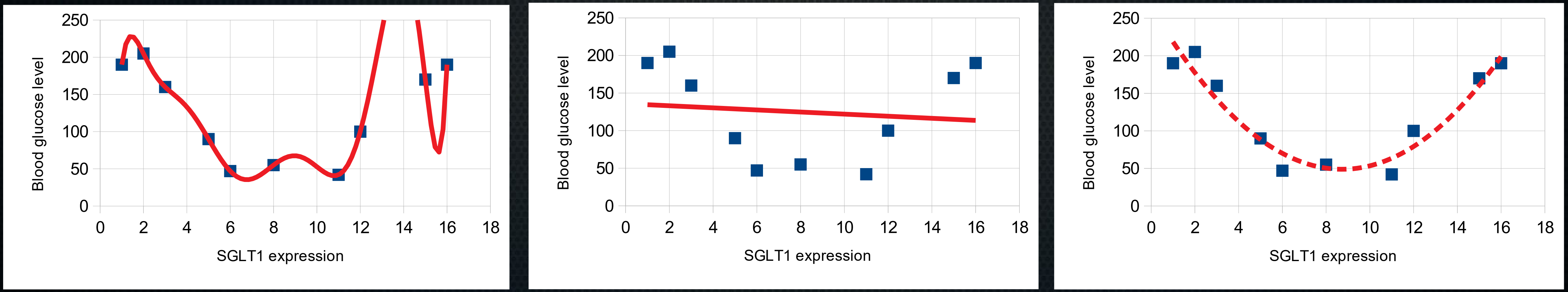
- Goal is not to fit the training data perfectly, but to generalise well



Source: https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

How to generalise well

- Goal is not to fit the training data perfectly, but to generalise well

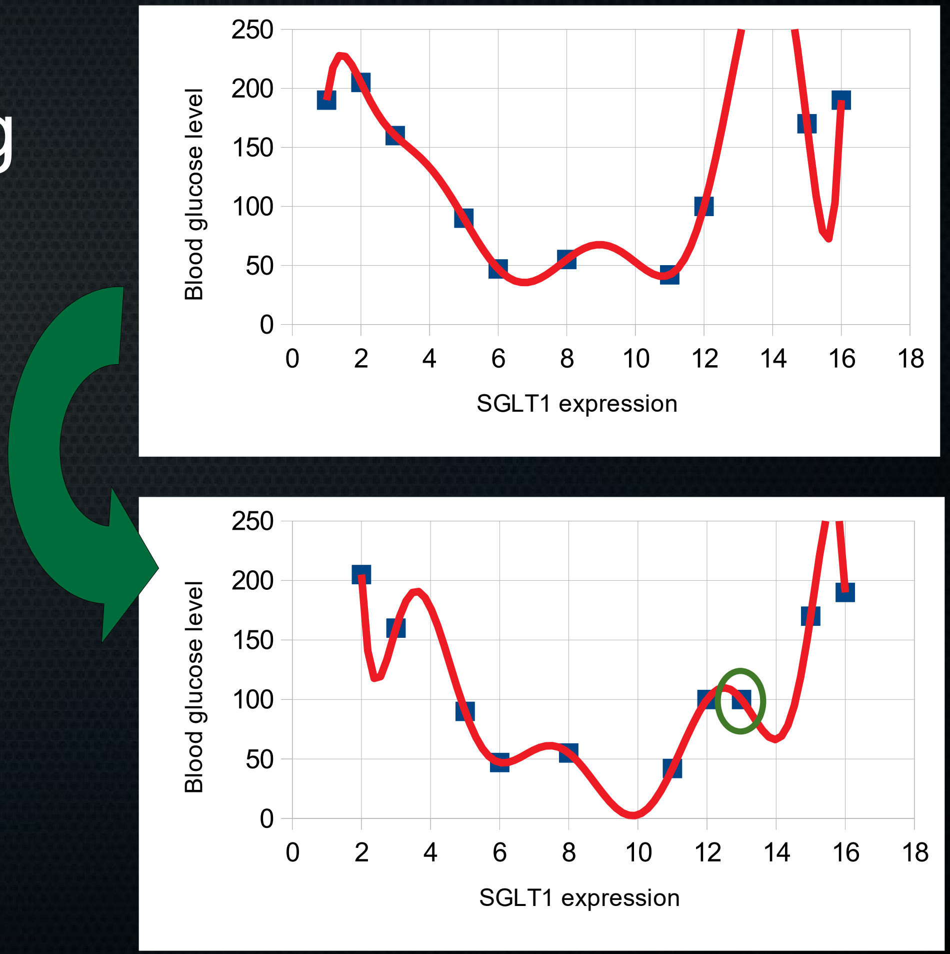


How to generalise well

- High **variance**:
  -how much our hypothesis function would change if we changed our training set
  -used x^1-x^9 as features

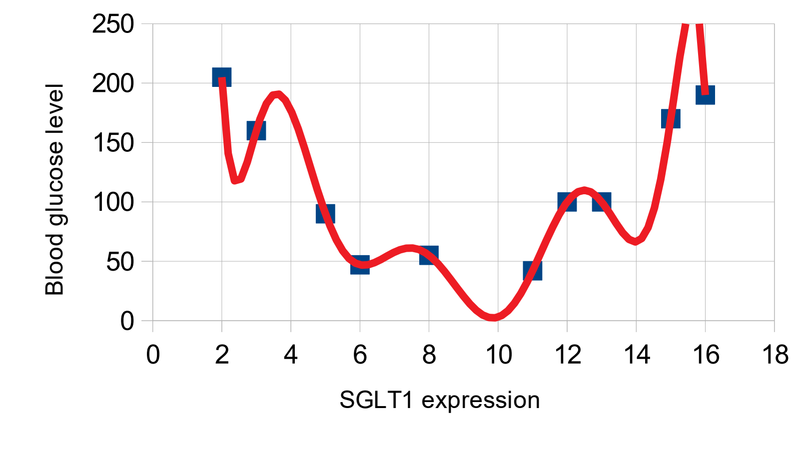
How to generalise well

- High **variance**:
  -how much our hypothesis function would change if we changed our training set
  -used x^1-x^9 as features
  -If we add one point:



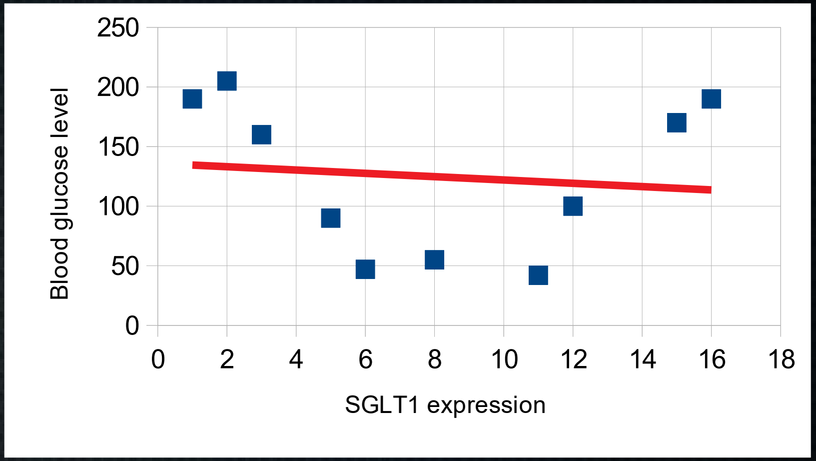
How to generalise well

- High **variance**:
  -Huge amount of possible functions that pass through all points: large hypothesis space, can basically fit all the training data we give perfectly!
  -Do great on this data, but will fare poorly when predicting unseen data



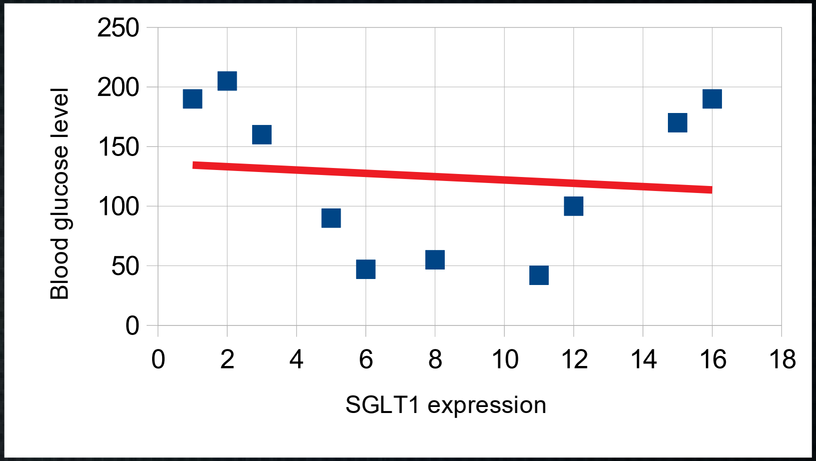
How to generalise well

- High **bias**:
  -error introduced by approximating a complex process with a simple model.
  -Only 1 feature (intercept + theta1)



How to generalise well

- High **bias**:
  -error introduced by approximating a complex process with a simple model.
  -Only 1 feature (intercept + theta1)
  -Data clearly shows that a linear curve is not the best fit, yet we keep our preconception, or *bias*, that it should adhere to a univariate linear regression
  -Does poorly on this data and will also fare poorly when predicting unseen data



How to generalise well

- Just right:
  -Not fit too closely to known examples, probably generalises well.

What can we do to find a good model?

- Find a way to approximate generalisation error: how well do you do on unseen data?
- See how error on seen and unseen data changes with amount of training data (plot learning curves)
- Reduce dimensionality by using only certain features
- Automatically constrain the fitting by penalising the cost function for too many/too large parameters

What can we do to find a good model?

- ***<u>Find a way to approximate generalisation error: how well do you do on unseen data?</u>***
- See how error on seen and unseen data changes with amount of training data (plot learning curves)
- Automatically constrain the fitting by penalising the cost function for too many/too large parameters

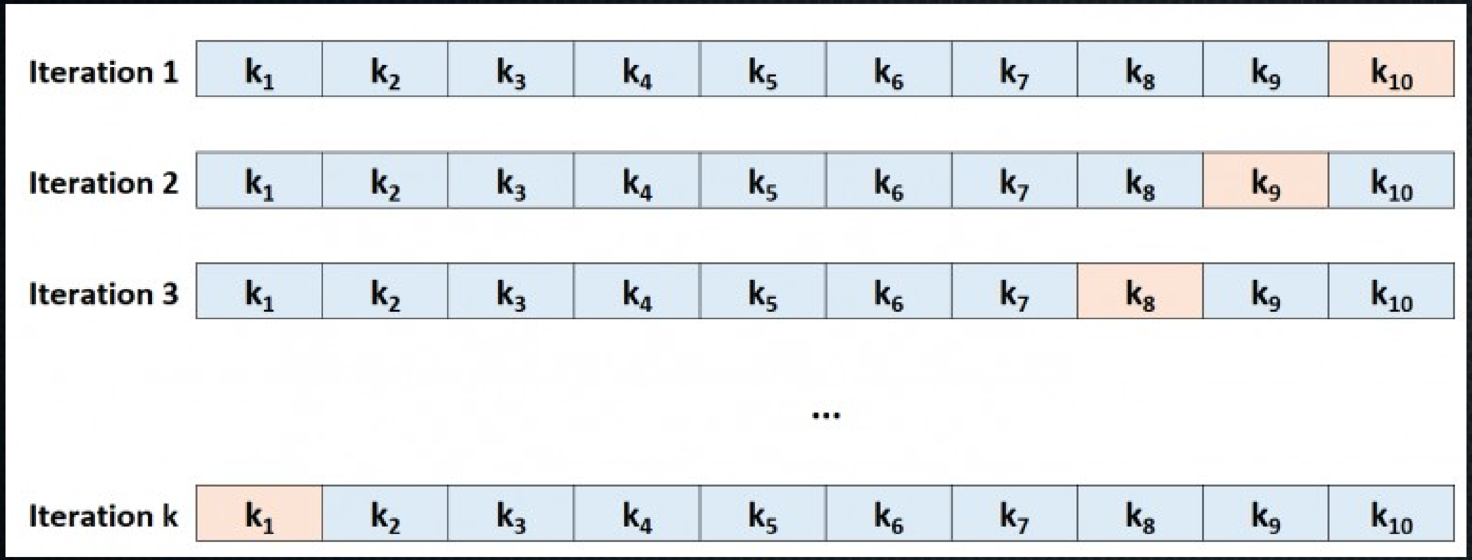
Approximate generalisation error: split data

- Split data into training data, validation data, and a test set
- Test set: completely untouched until you are done training
- Train set: train your classifier on this
- Validation set: test your trained classifier on this

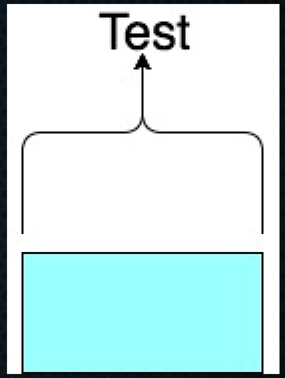
Source: https://stackoverflow.com/questions/56099495/what-should-be-passed-as-input-parameter-when-using-train-test-split-function-tw

Approximate generalisation error: split data

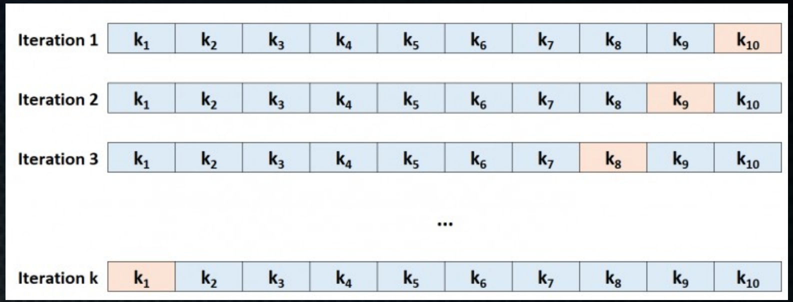
- K-fold cross-validation (often 10-fold):

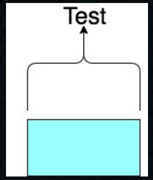


Source:https://www.statology.org/k-fold-cross-validation/



Approximate generalisation error: split data

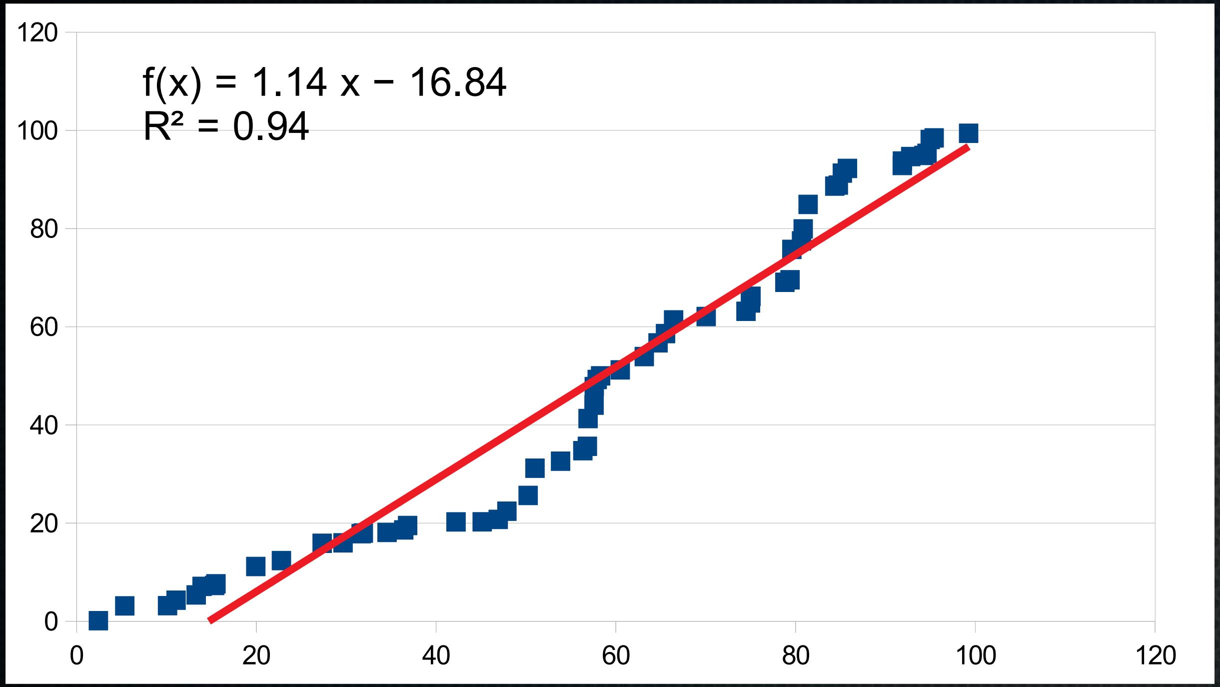




- Procedure:
  -Shuffle the data
  -Divide into k folds (e.g. 10 folds of 100 training examples each)
  - For each fold:
    -find parameters by minimising cost function on training set with gradient descent
    -predict on validation data
    -calculate cost on validation data (you know the true values)

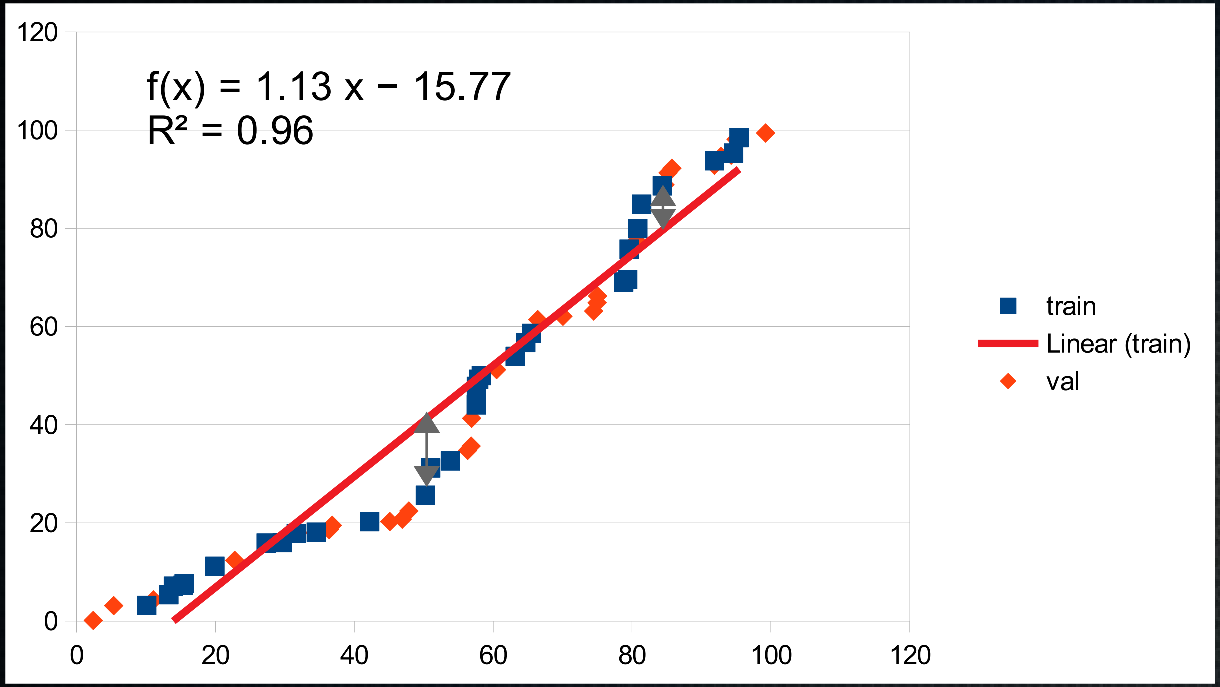
  -Average validation cost over folds ~ generalisation error

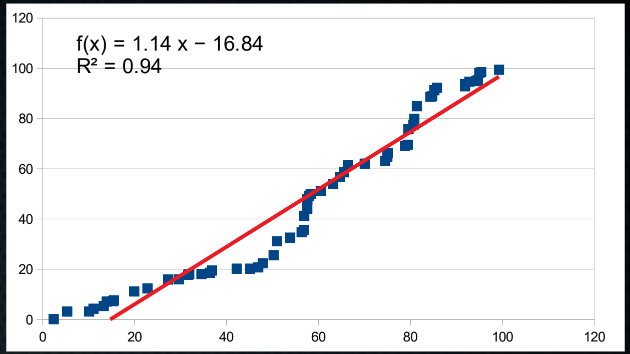
Example 2-fold cv on linear regression



All data

Example 2-fold cv on linear regression: fold 1

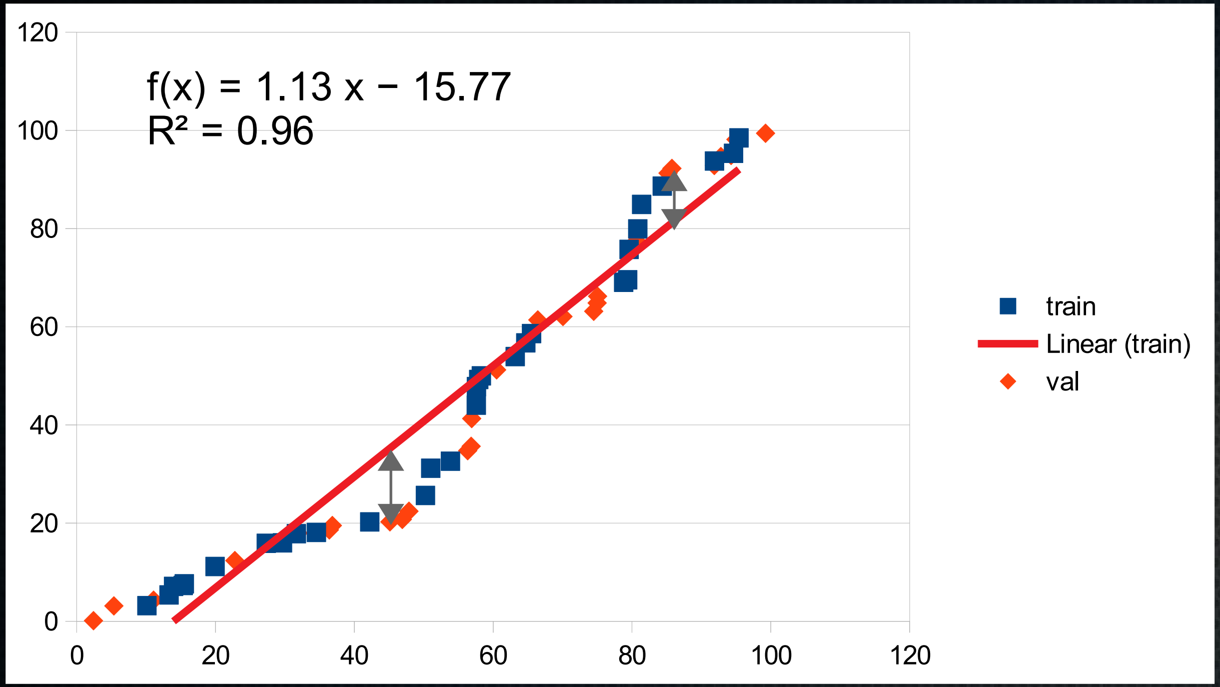


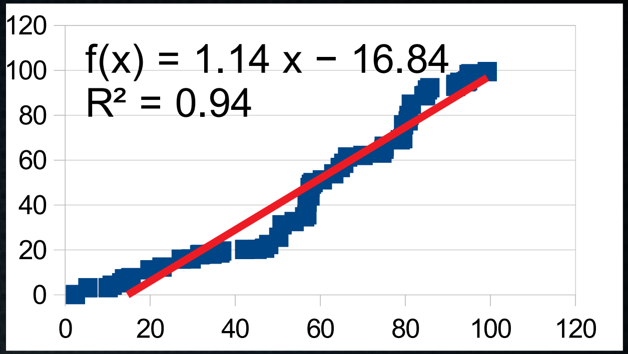


All data

$$J(\theta_0, \theta_1)_{train} = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 = 41{,}66$$

Example 2-fold cv on linear regression: fold 1



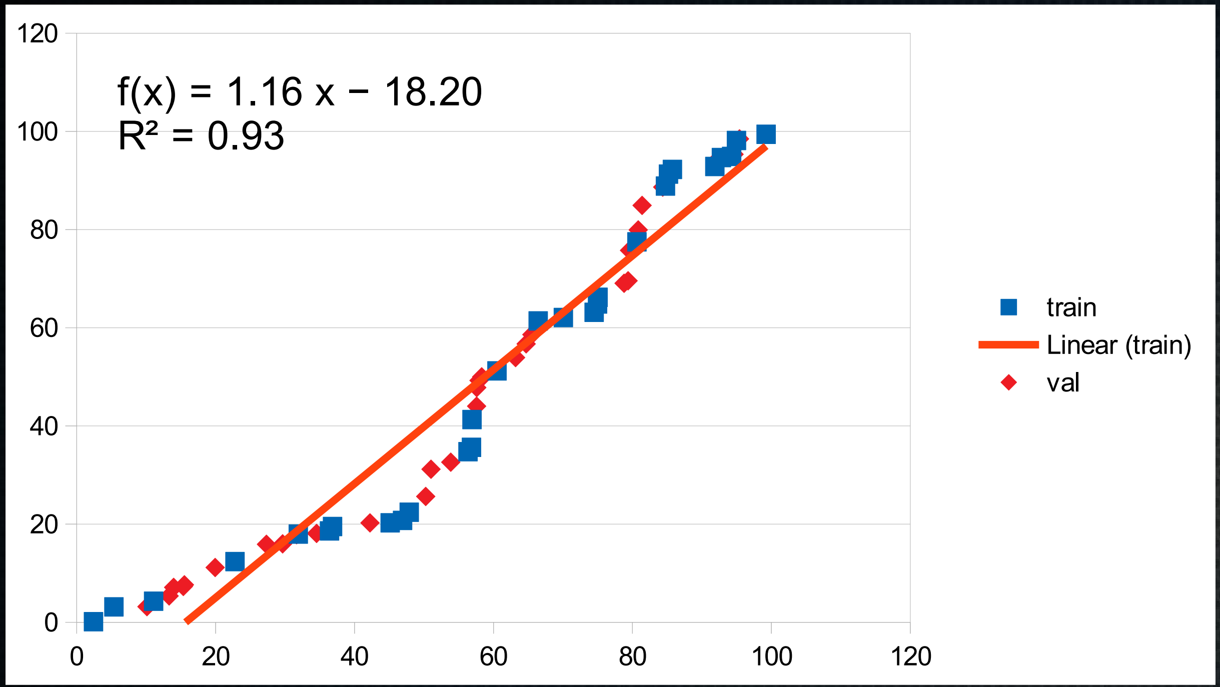


All data

$$J(\theta_0,\theta_1)_{train}=\frac{1}{2m_{train}}\sum_{i=1}^{m_{train}}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2=41{,}66$$

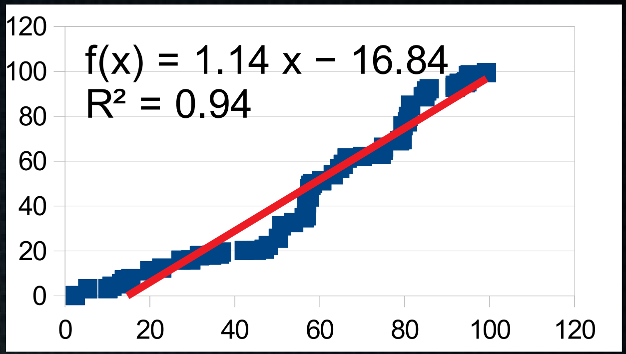
$$J(\theta_0,\theta_1)_{val}=\frac{1}{2m_{val}}\sum_{i=1}^{m_{val}}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2=52{,}34$$

Example 2-fold cv on linear regression: fold 2

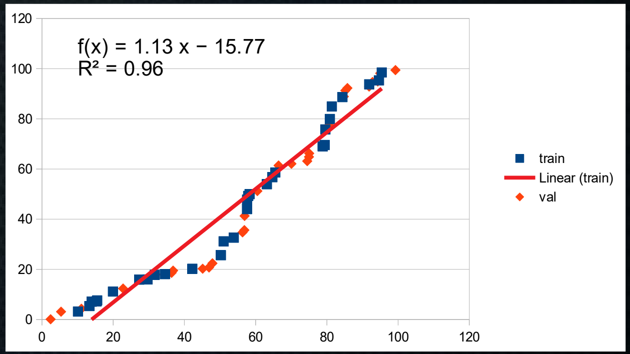


$$J(\theta_0,\theta_1)_{train}=\frac{1}{2m_{train}}\sum_{i=1}^{m_{train}}\left(h_\theta(x^{(i)})-y^{(i)}\right)^2=75{,}25$$

$$J(\theta_0,\theta_1)_{val}=\frac{1}{2m_{val}}\sum_{i=1}^{m_{val}}\left(h_\theta(x^{(i)})-y^{(i)}\right)^2=43$$



All data

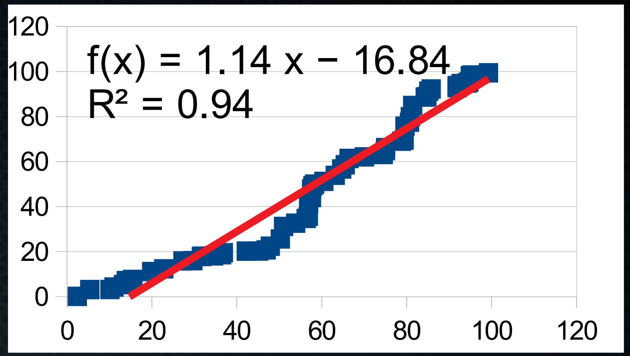


$$J(\theta_0,\theta_1)_{train}=41{,}66$$
$$J(\theta_0,\theta_1)_{val}=52{,}34$$

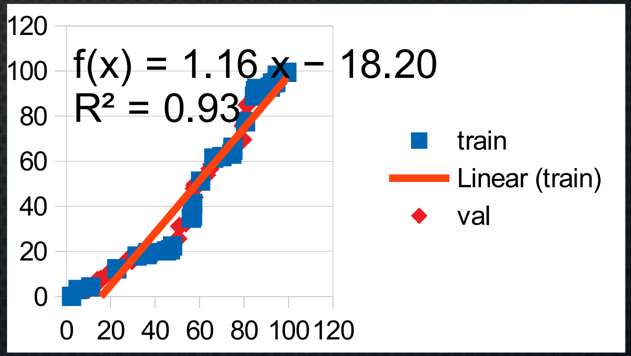
Fold 1

Example 2-fold cv on linear regression

- Avg. train error: 58,5
- Avg. Validation error: 47,7
- Perform better on unseen than seen data: generalises well.
- Finally: would train on all data and test that on test set.

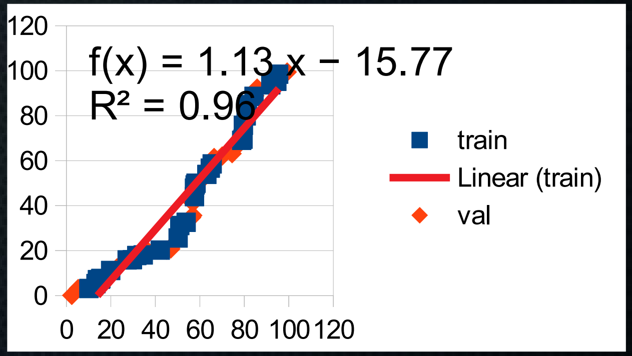


All data



$$J(\theta_0,\theta_1)_{train}=75,25$$
$$J(\theta_0,\theta_1)_{val}=43$$

Fold 2



$$J(\theta_0,\theta_1)_{train}=41,66$$
$$J(\theta_0,\theta_1)_{val}=52,34$$

Fold 1

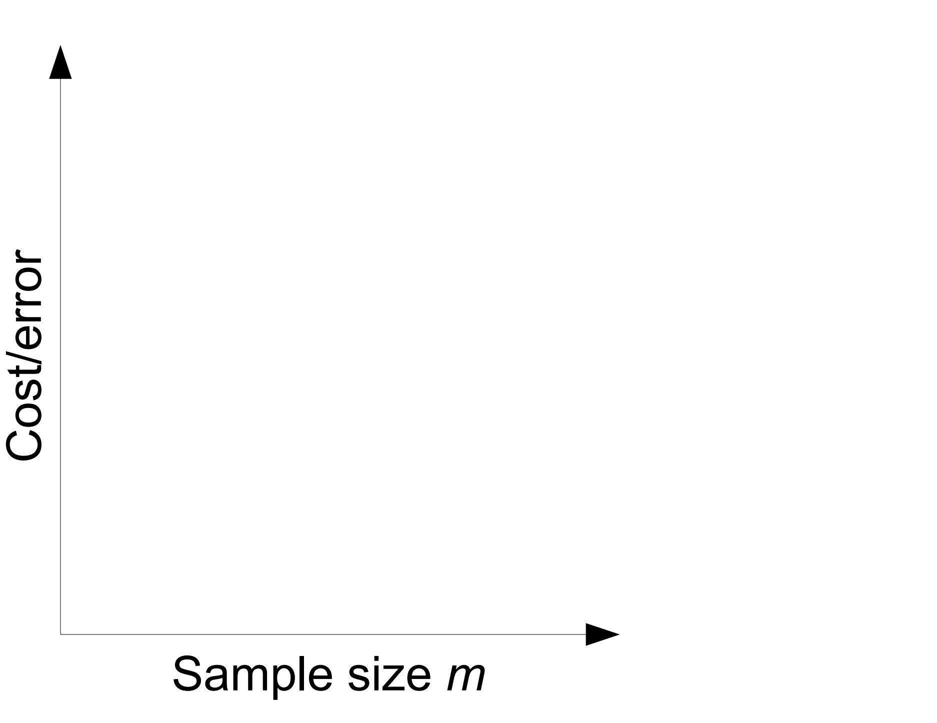
What can we do to find a good model?

- ~~Find a way to approximate generalisation error: how well do you do on unseen data?~~
- ***<u>See how error on seen and unseen data changes with amount of training data (plot learning curves)</u>***
- Automatically constrain the fitting by penalising the cost function for too many/too large parameters

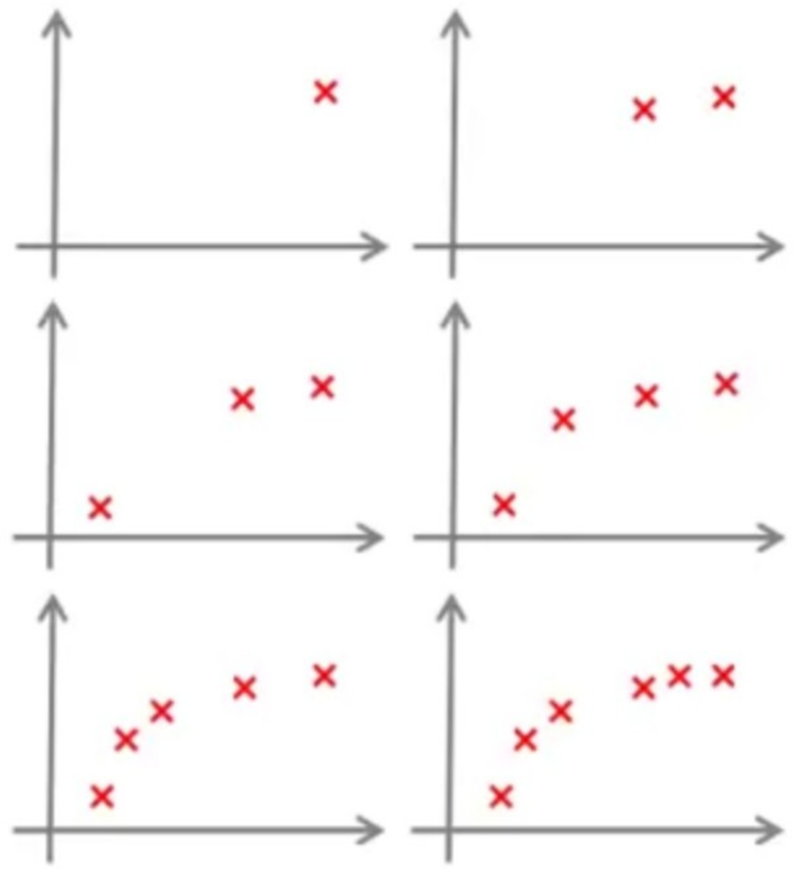
Learning curves

$$J_{train} = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$J_{val} = \frac{1}{2m_{val}} \sum_{i=1}^{m_{val}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$



$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

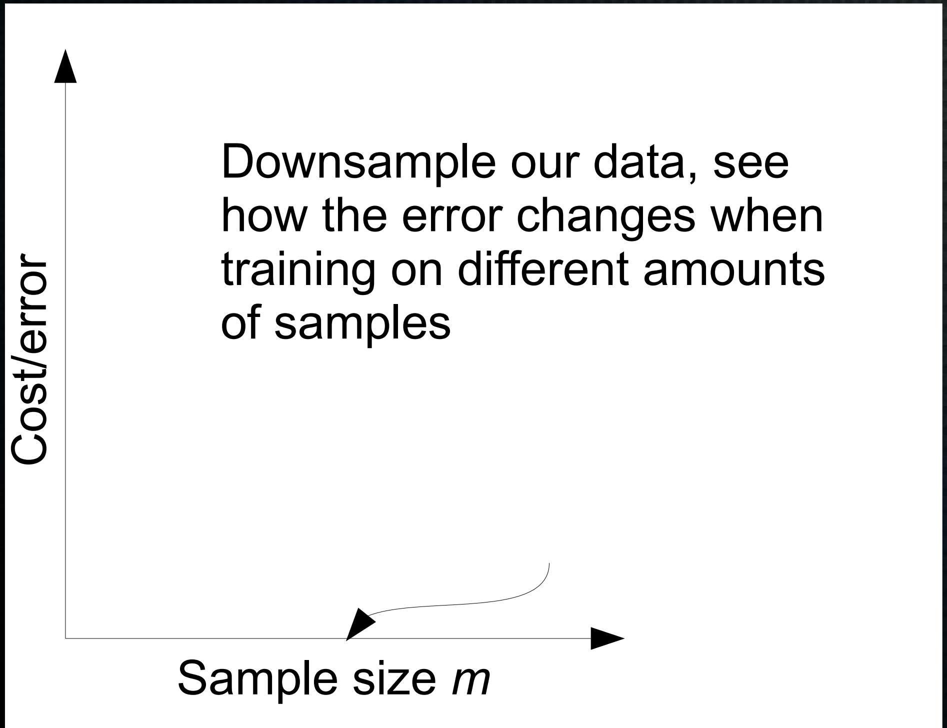


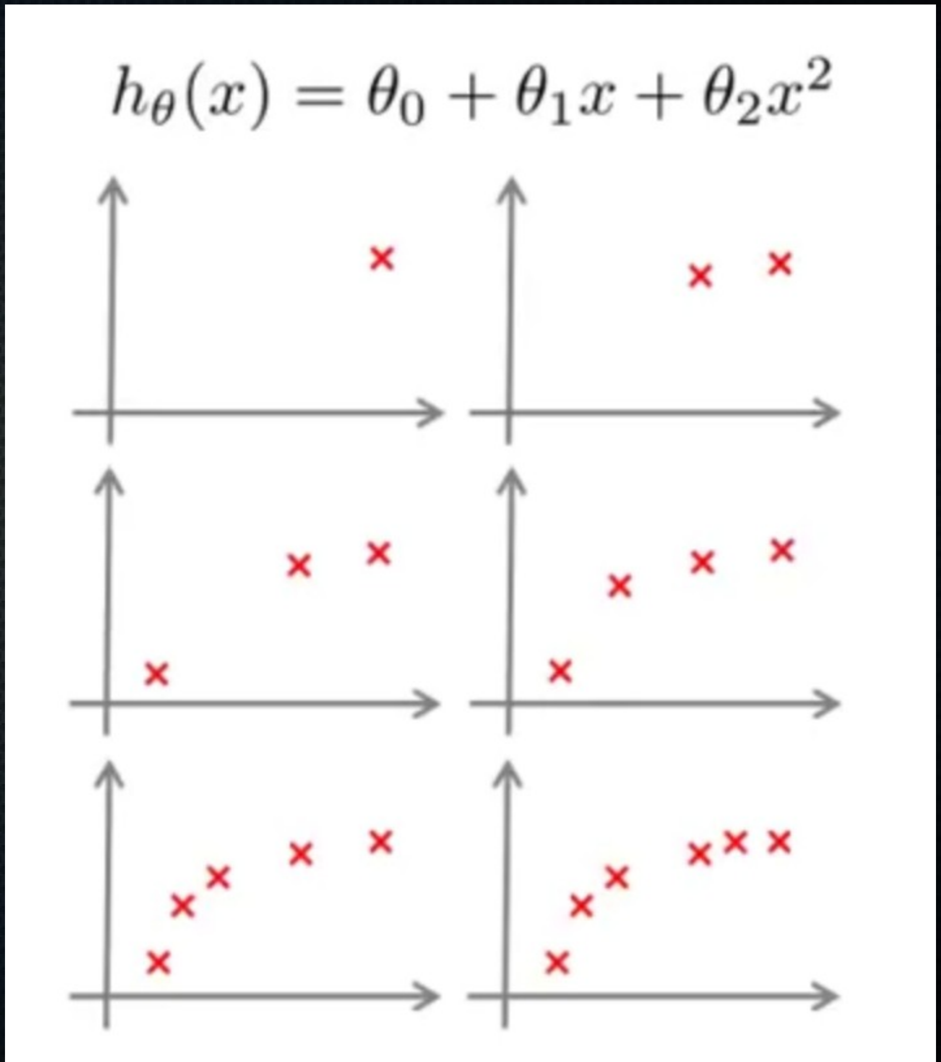
Source: Andrew Ng, Coursera

Learning curves

$$J_{train} = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$J_{val} = \frac{1}{2m_{val}} \sum_{i=1}^{m_{val}} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$



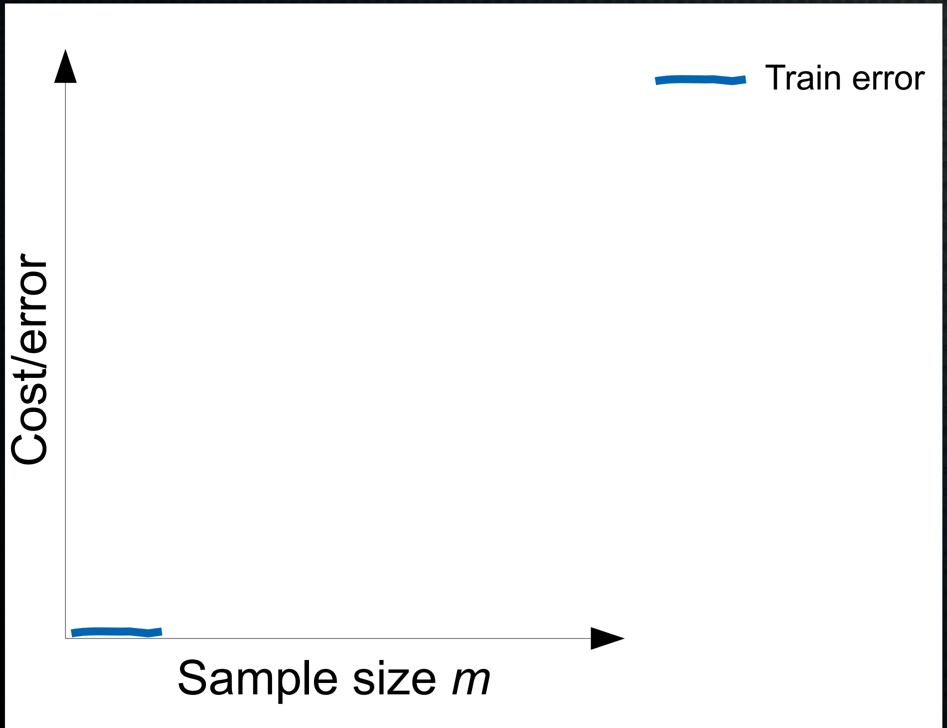


Source: Andrew Ng, Coursera

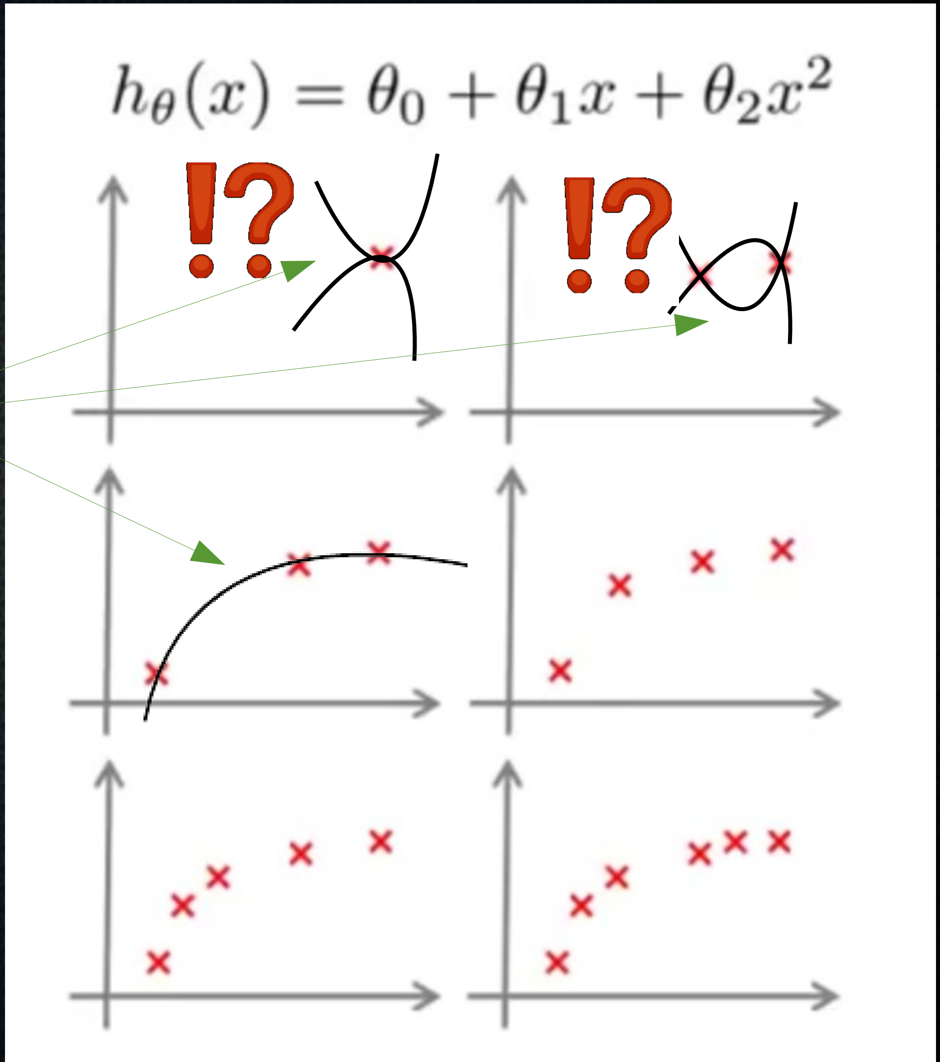
Learning curves

$$J_{train} = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$J_{val} = \frac{1}{2m_{val}} \sum_{i=1}^{m_{val}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$



Easy to fit few datapoints perfectly

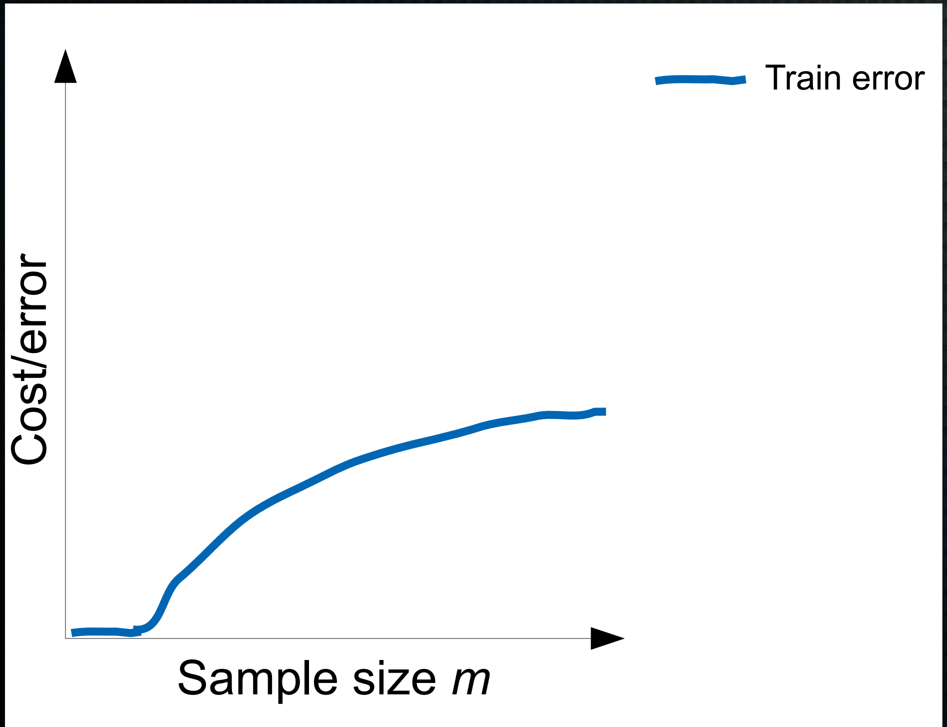


Source: Andrew Ng, Coursera

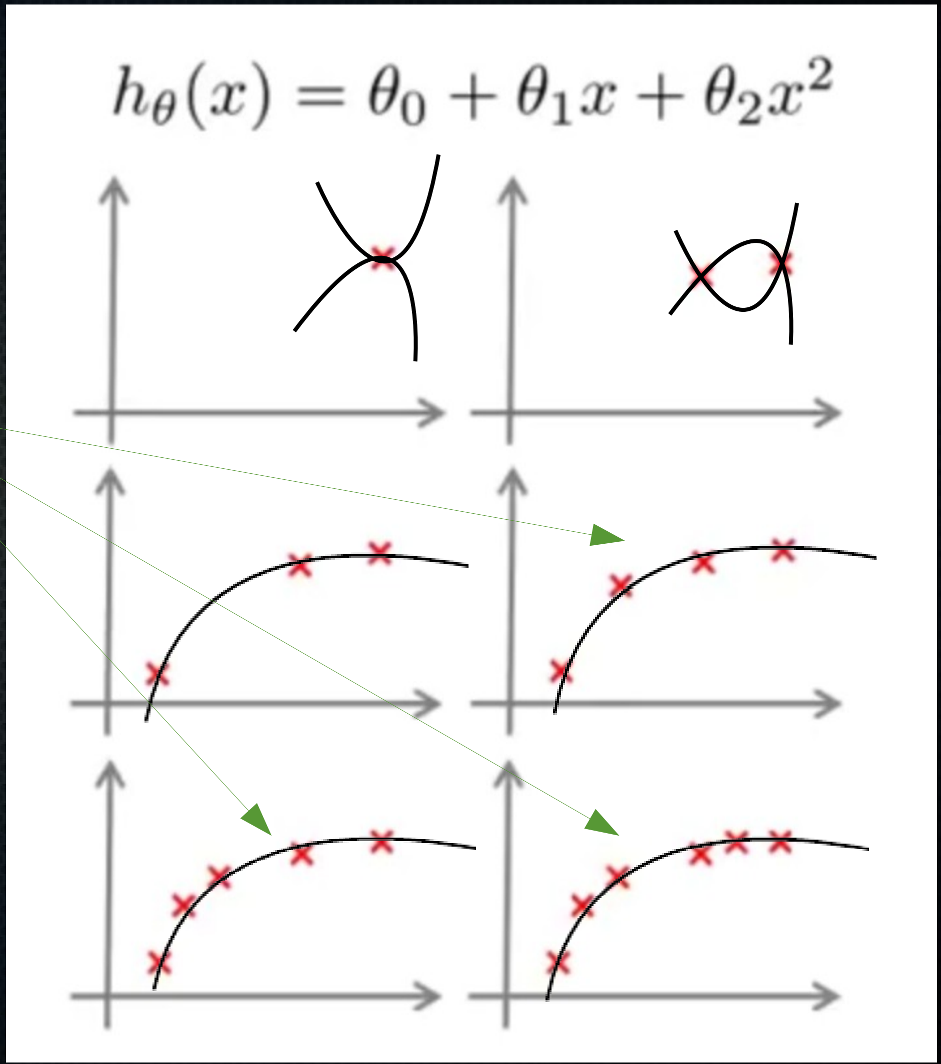
Learning curves

$$J_{train} = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$J_{val} = \frac{1}{2m_{val}} \sum_{i=1}^{m_{val}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$



Harder and harder to fit everything perfectly

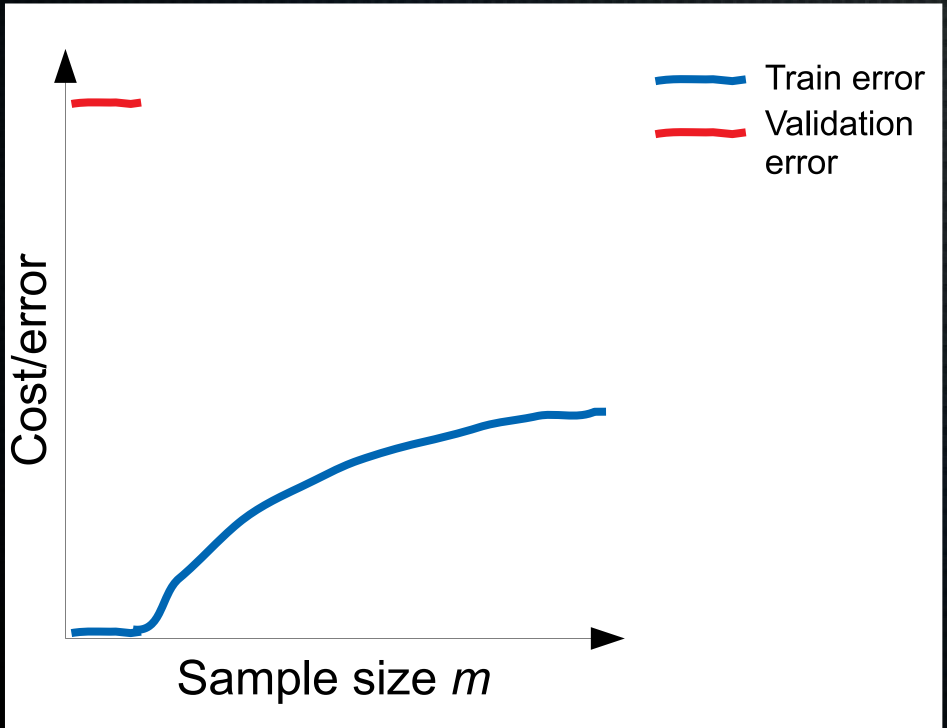


Source: Andrew Ng, Coursera

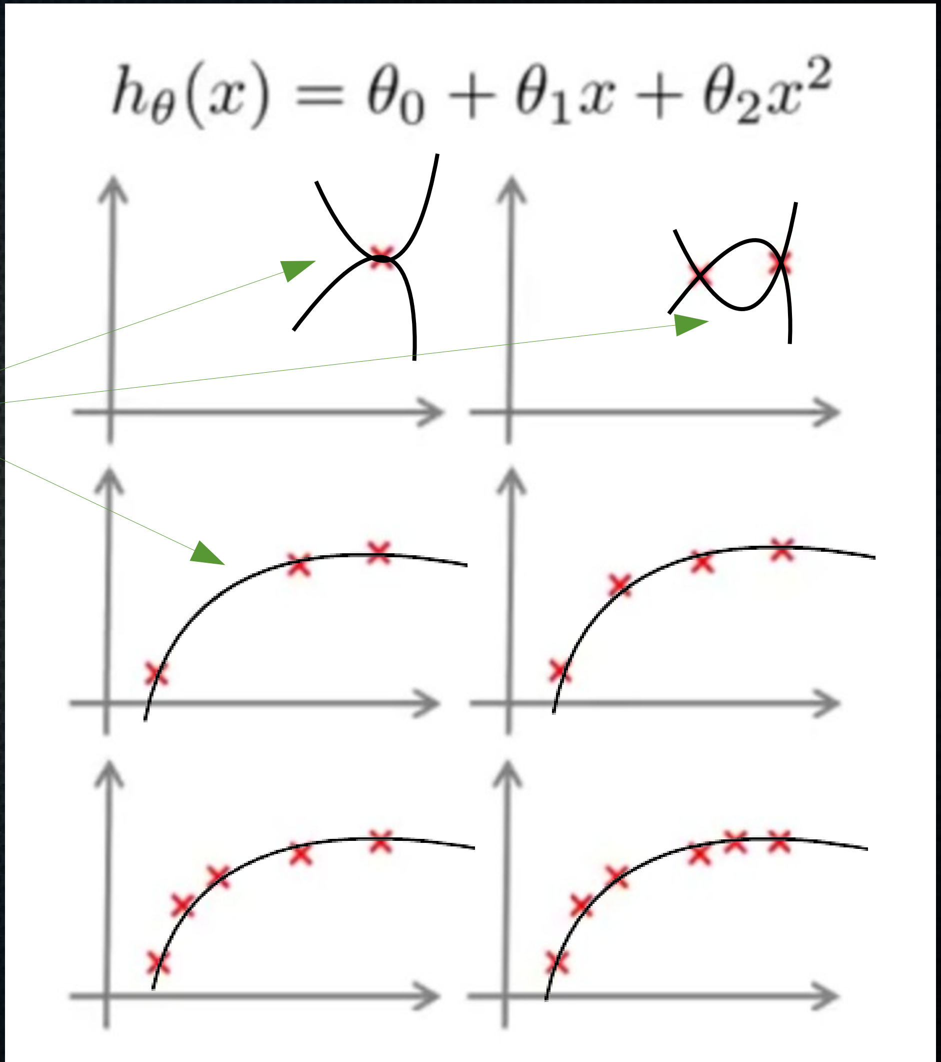
Learning curves

$$J_{train} = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$J_{val} = \frac{1}{2m_{val}} \sum_{i=1}^{m_{val}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$



Generalises poorly to new data

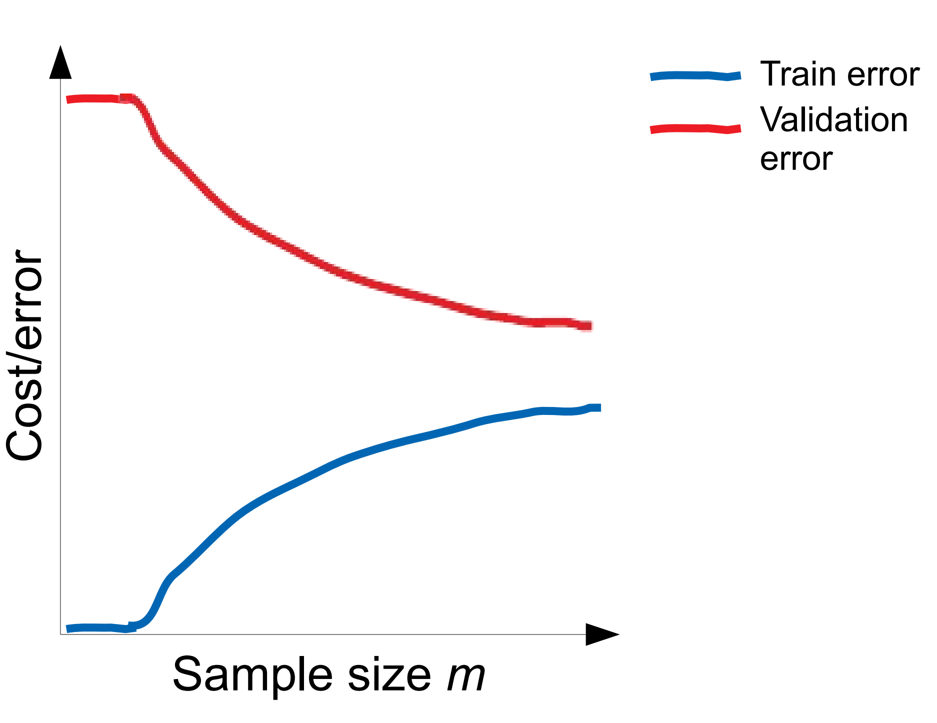


Source: Andrew Ng, Coursera

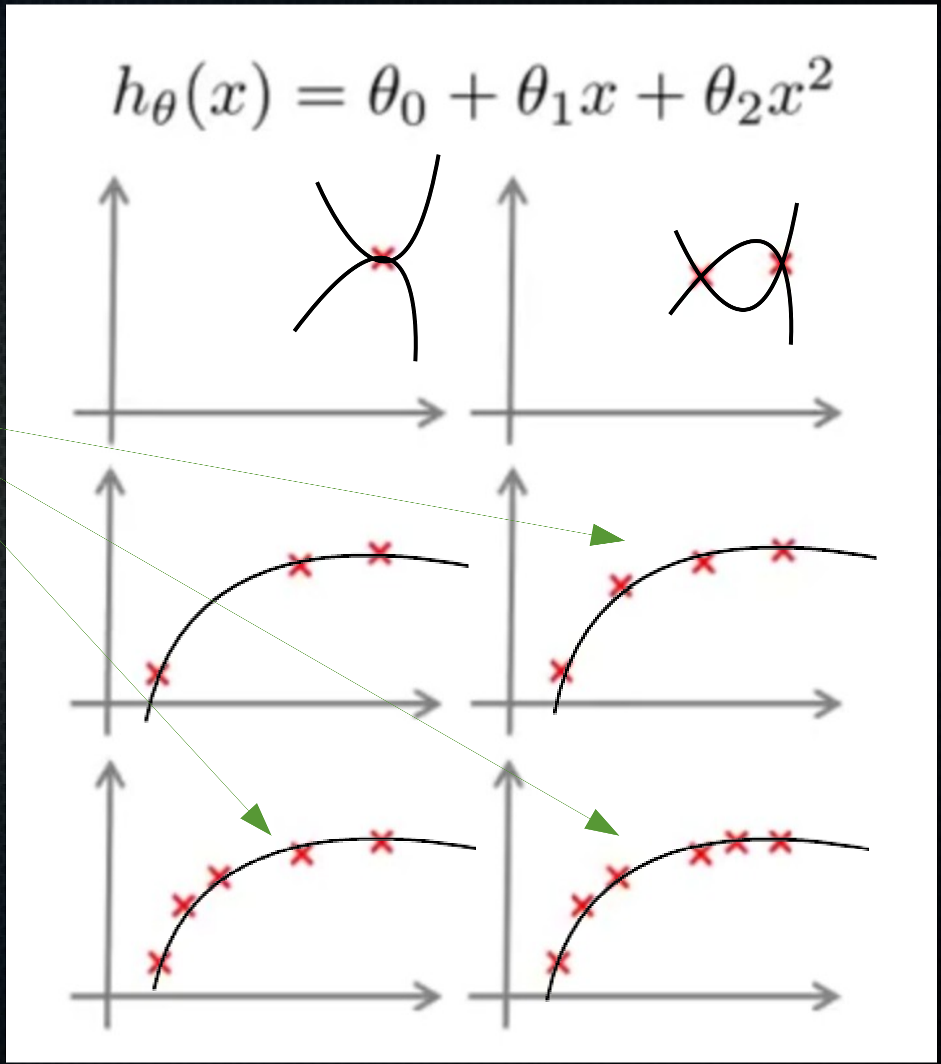
Learning curves

$$J_{train} = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{val} = \frac{1}{2m_{val}} \sum_{i=1}^{m_{val}} (h_\theta(x^{(i)}) - y^{(i)})^2$$



Generalises better to new data

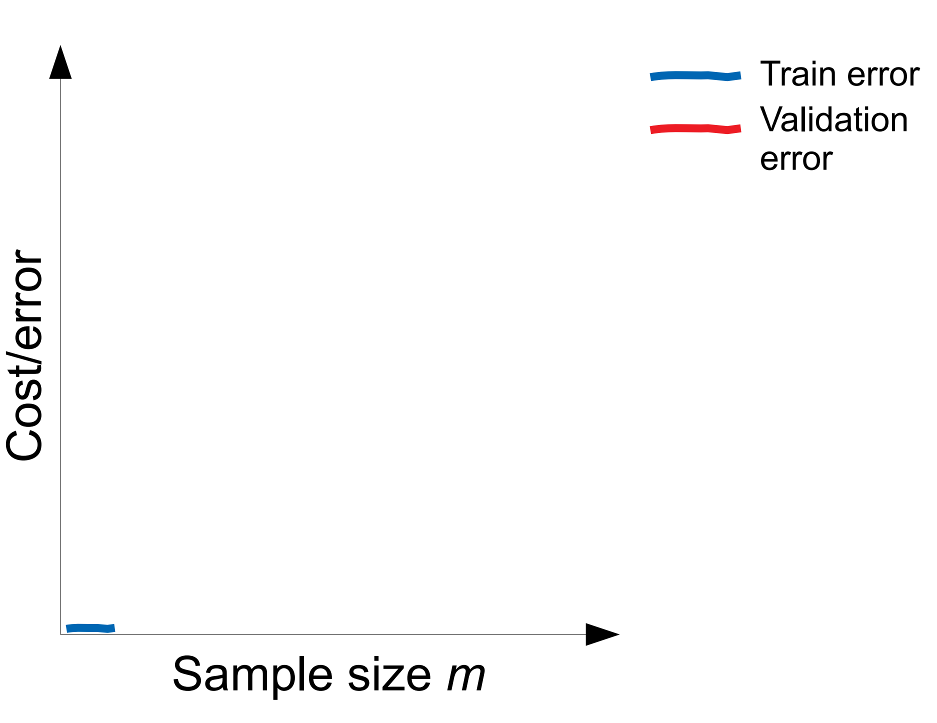


Source: Andrew Ng, Coursera

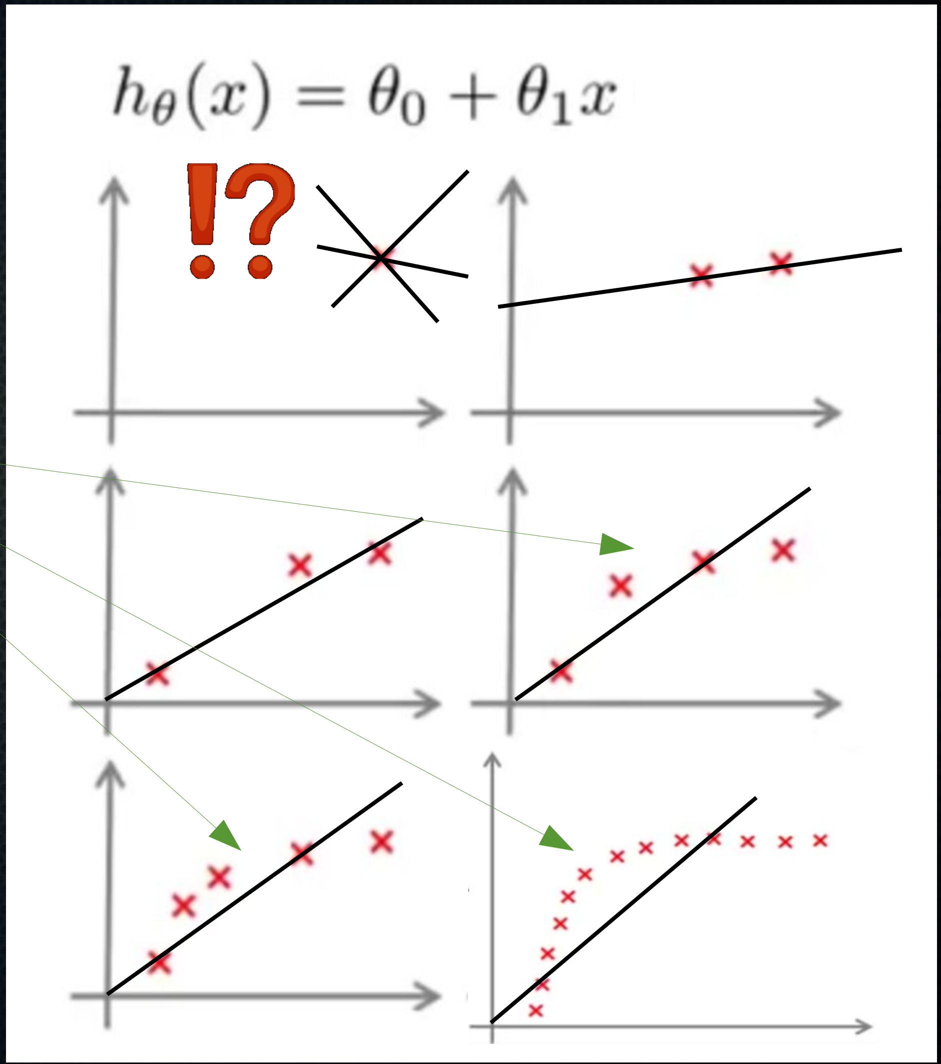
Learning curves: high bias

$$J_{train} = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$J_{val} = \frac{1}{2m_{val}} \sum_{i=1}^{m_{val}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$



Keep pretty much the same line even as data increases

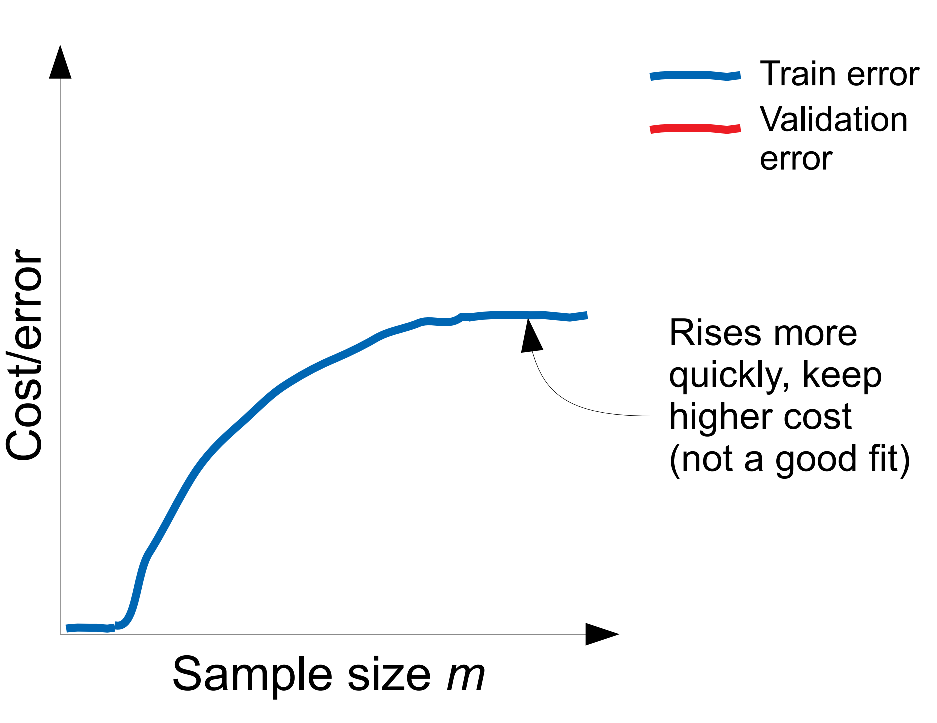


Source: Andrew Ng, Coursera

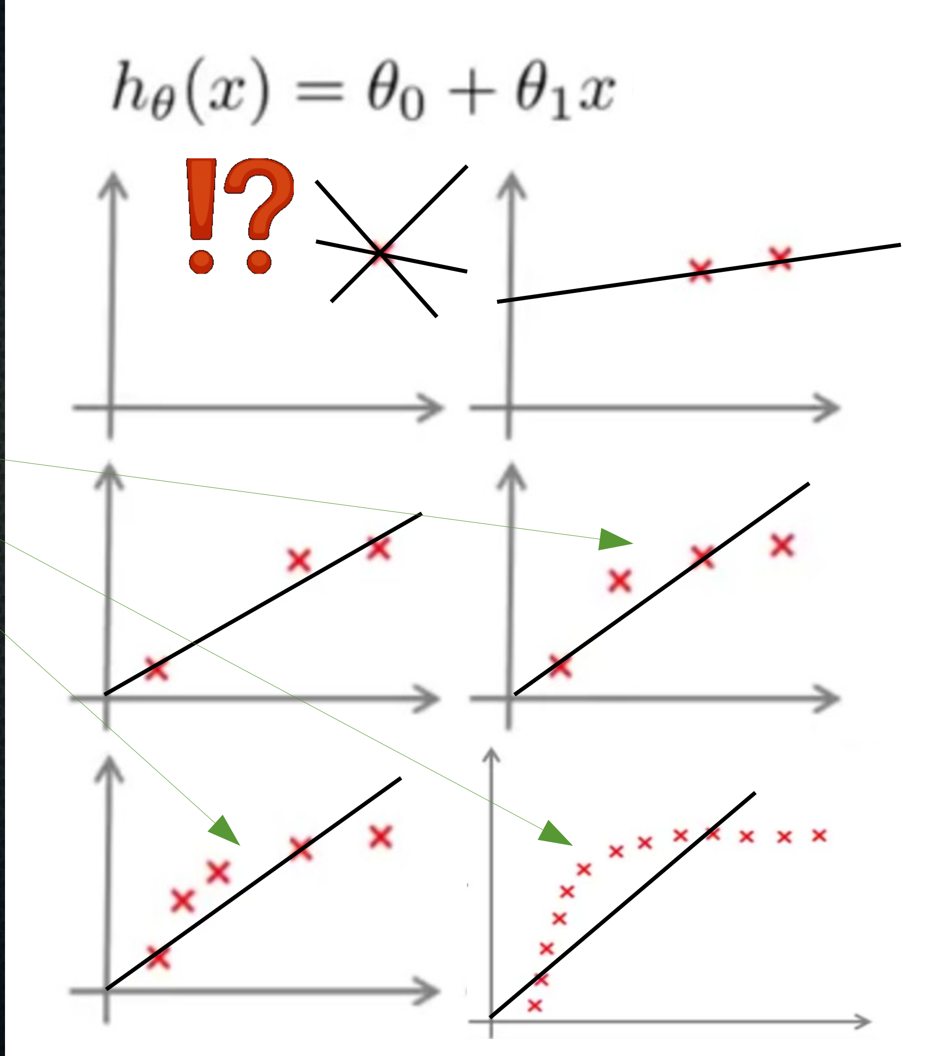
Learning curves: high bias

$$J_{train} = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$J_{val} = \frac{1}{2m_{val}} \sum_{i=1}^{m_{val}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$



Keep pretty much the same line even as data increases

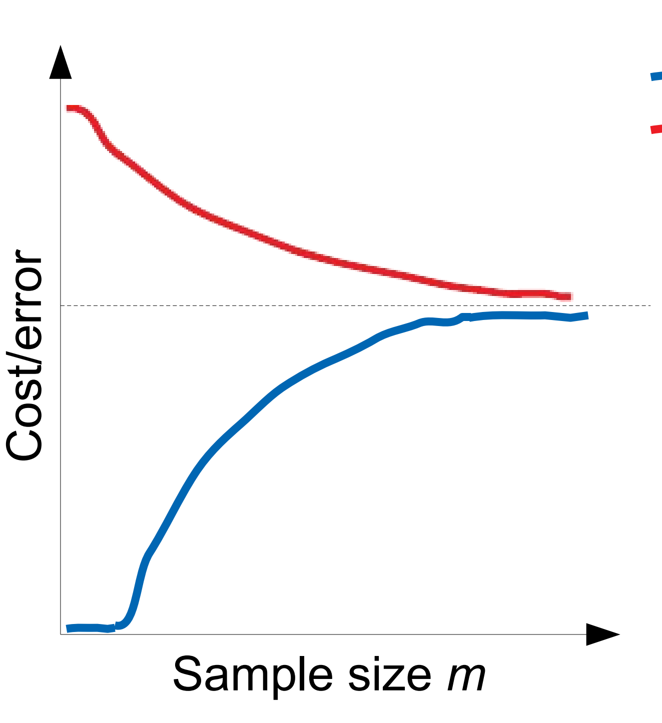


Source: Andrew Ng, Coursera

Learning curves: high bias

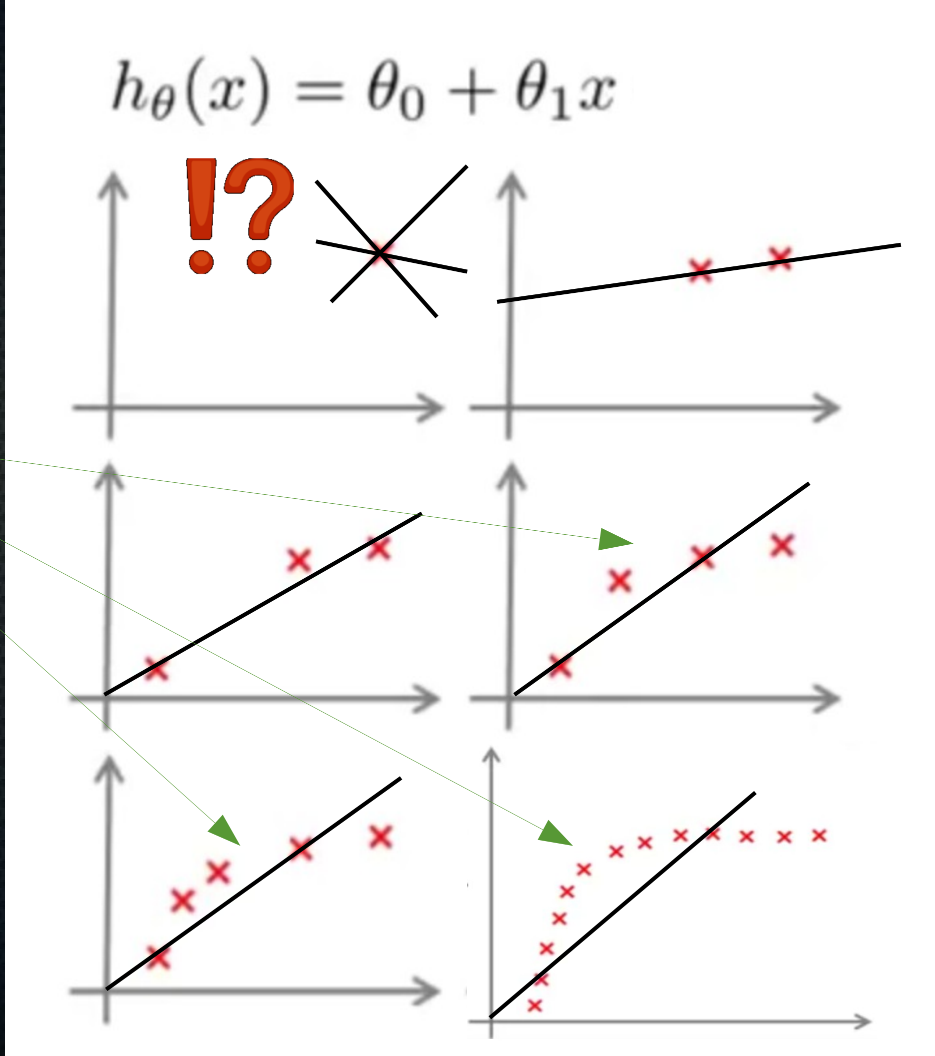
$$J_{train} = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$J_{val} = \frac{1}{2m_{val}} \sum_{i=1}^{m_{val}} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$



Errors close together, keep high error after training.

Keep pretty much the same line even as data increases



Source: Andrew Ng, Coursera

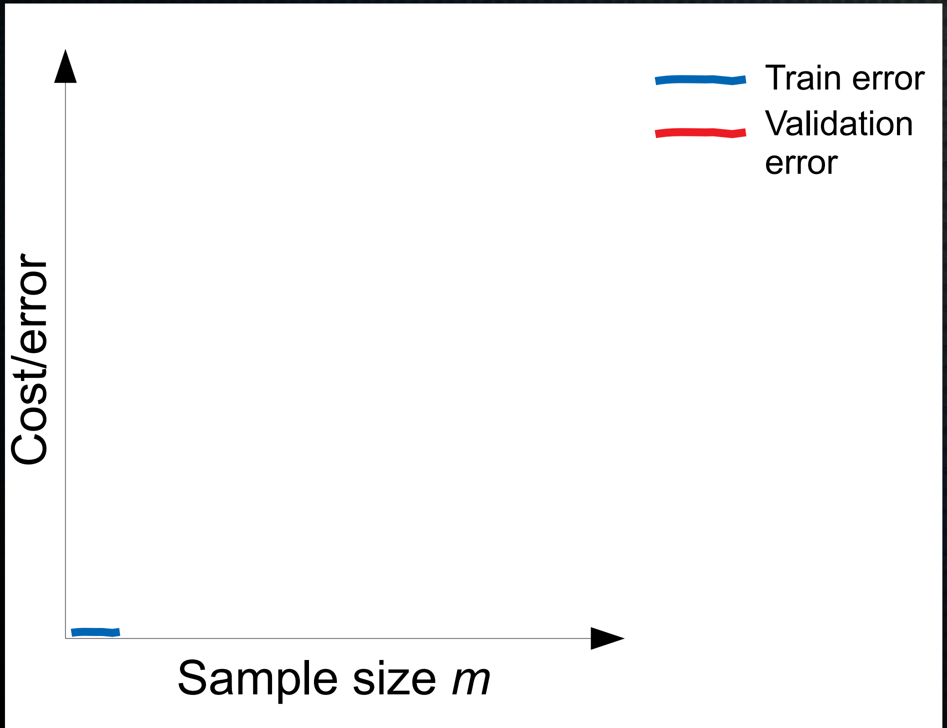
Learning curves: high bias

- If your learning algorithm is biased, getting more training data will not help!

Learning curves: high variance

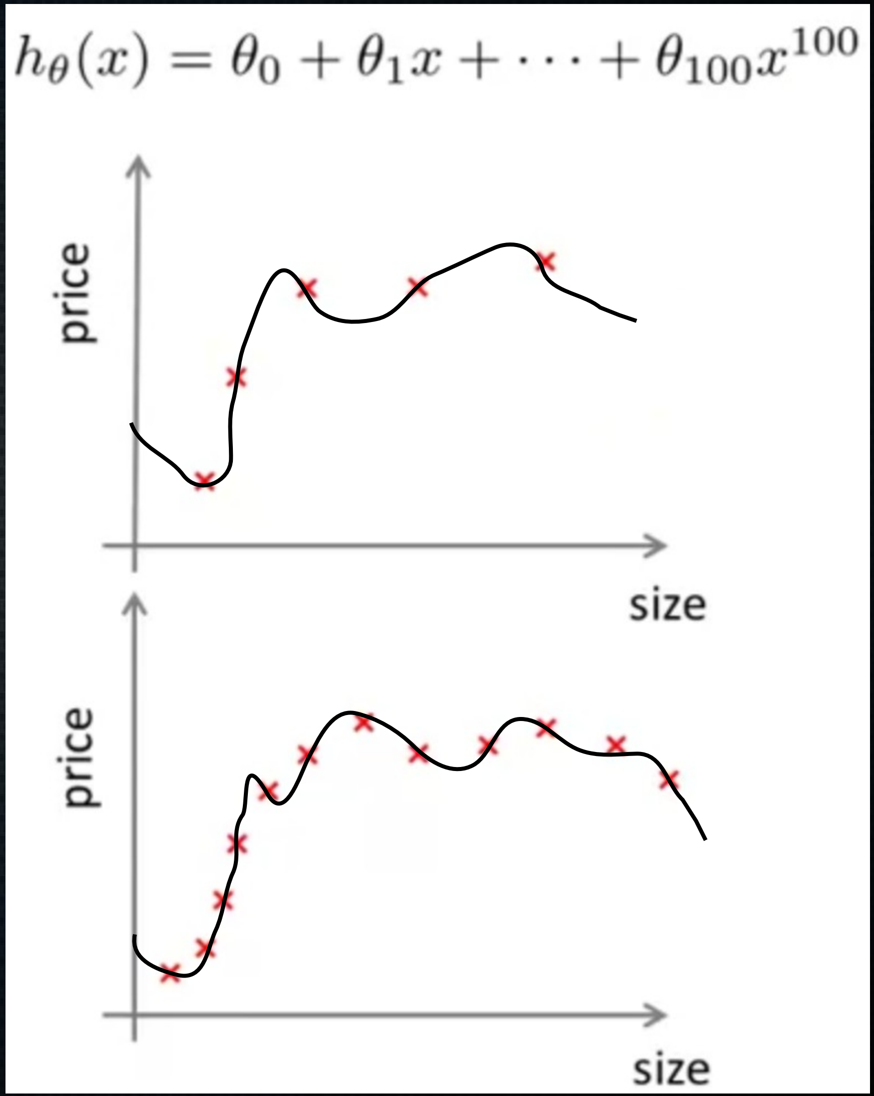
$$J_{train} = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$J_{val} = \frac{1}{2m_{val}} \sum_{i=1}^{m_{val}} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$



Fit perfectly

Fit very well, but not perfect

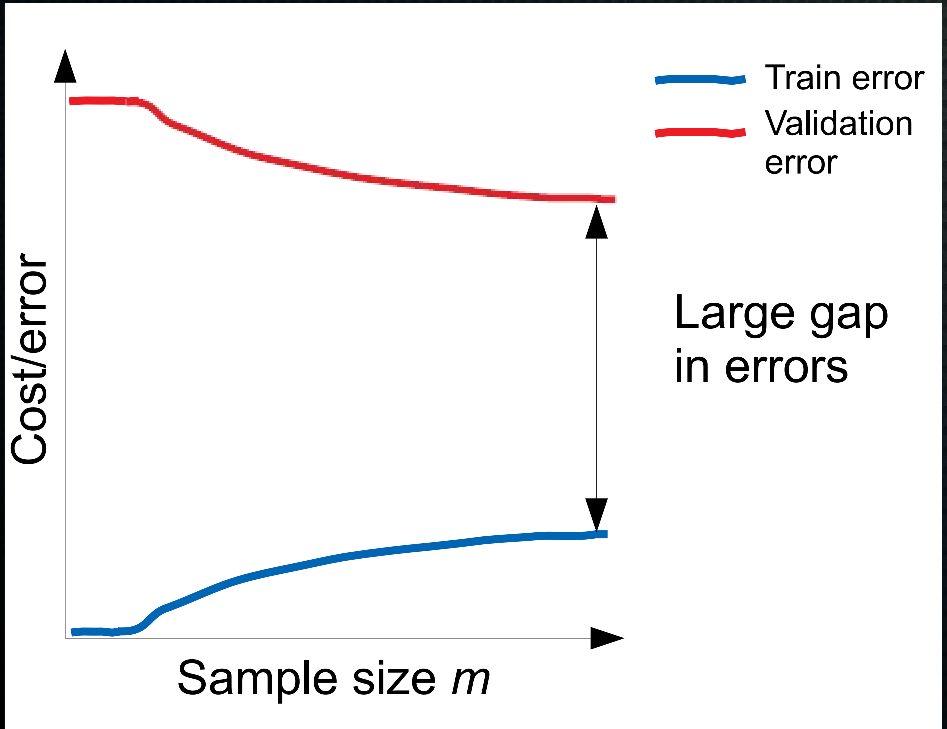


Source: Andrew Ng, Coursera

Learning curves: high variance

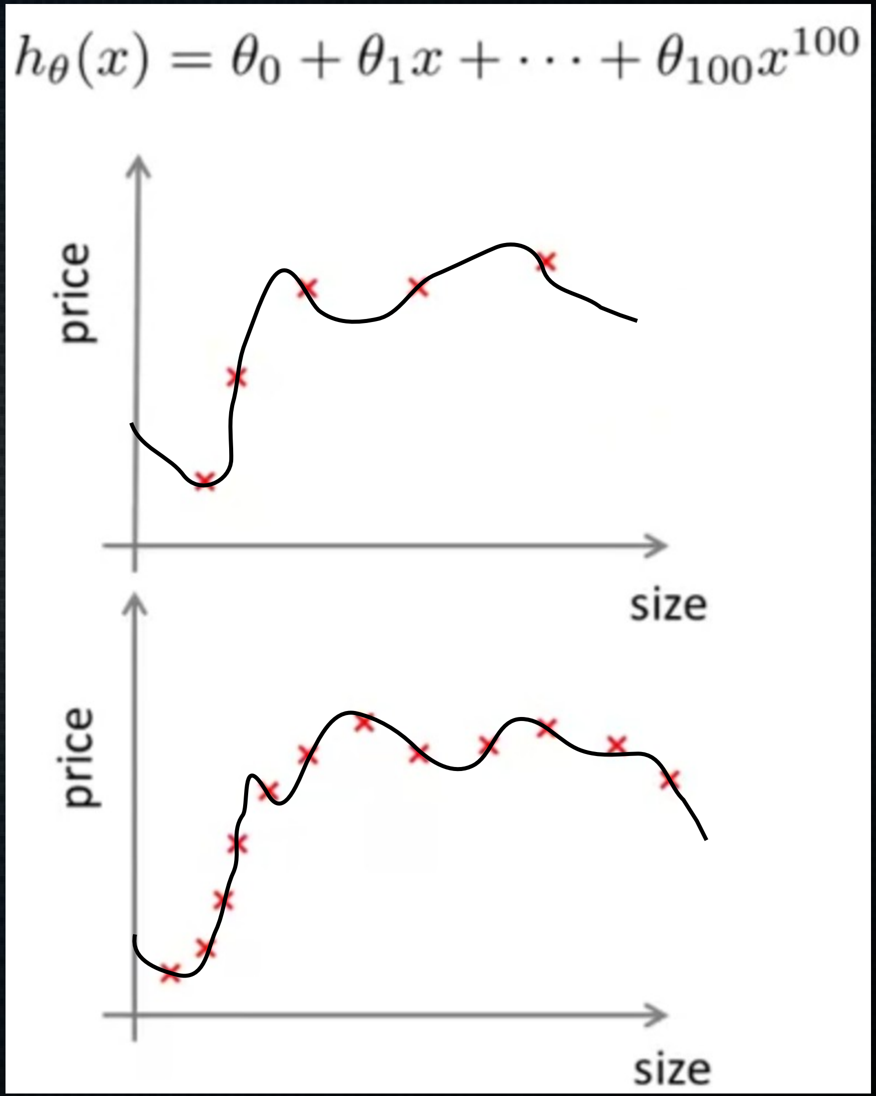
$$J_{train} = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$J_{val} = \frac{1}{2m_{val}} \sum_{i=1}^{m_{val}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$



Fit perfectly

Fit very well, but not perfect

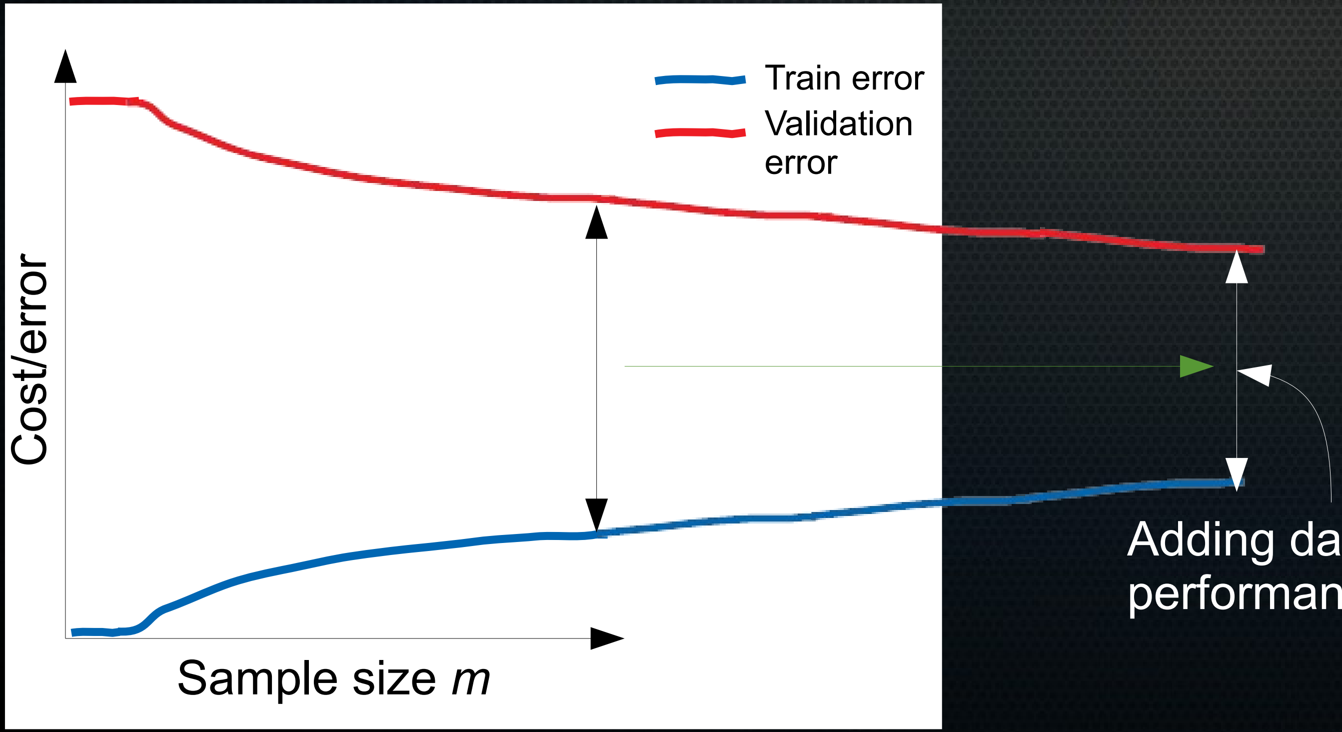


Source: Andrew Ng, Coursera

Learning curves: high variance

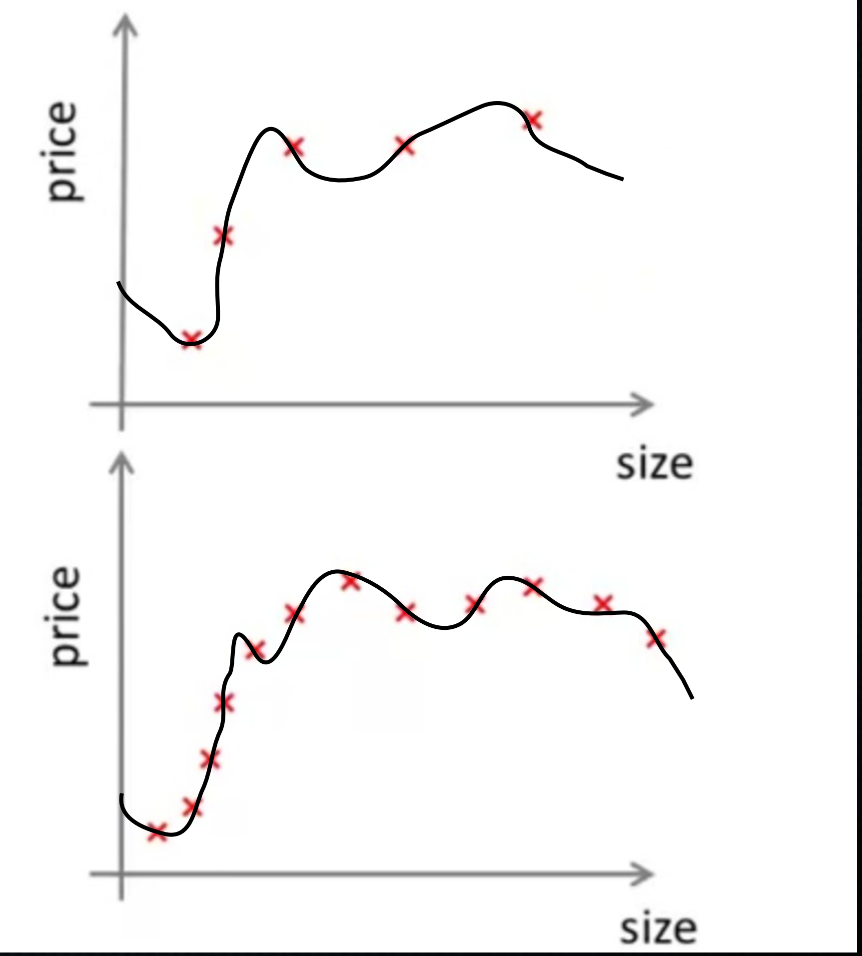
$$J_{train} = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$J_{val} = \frac{1}{2m_{val}} \sum_{i=1}^{m_{val}} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$



Adding data: better performance

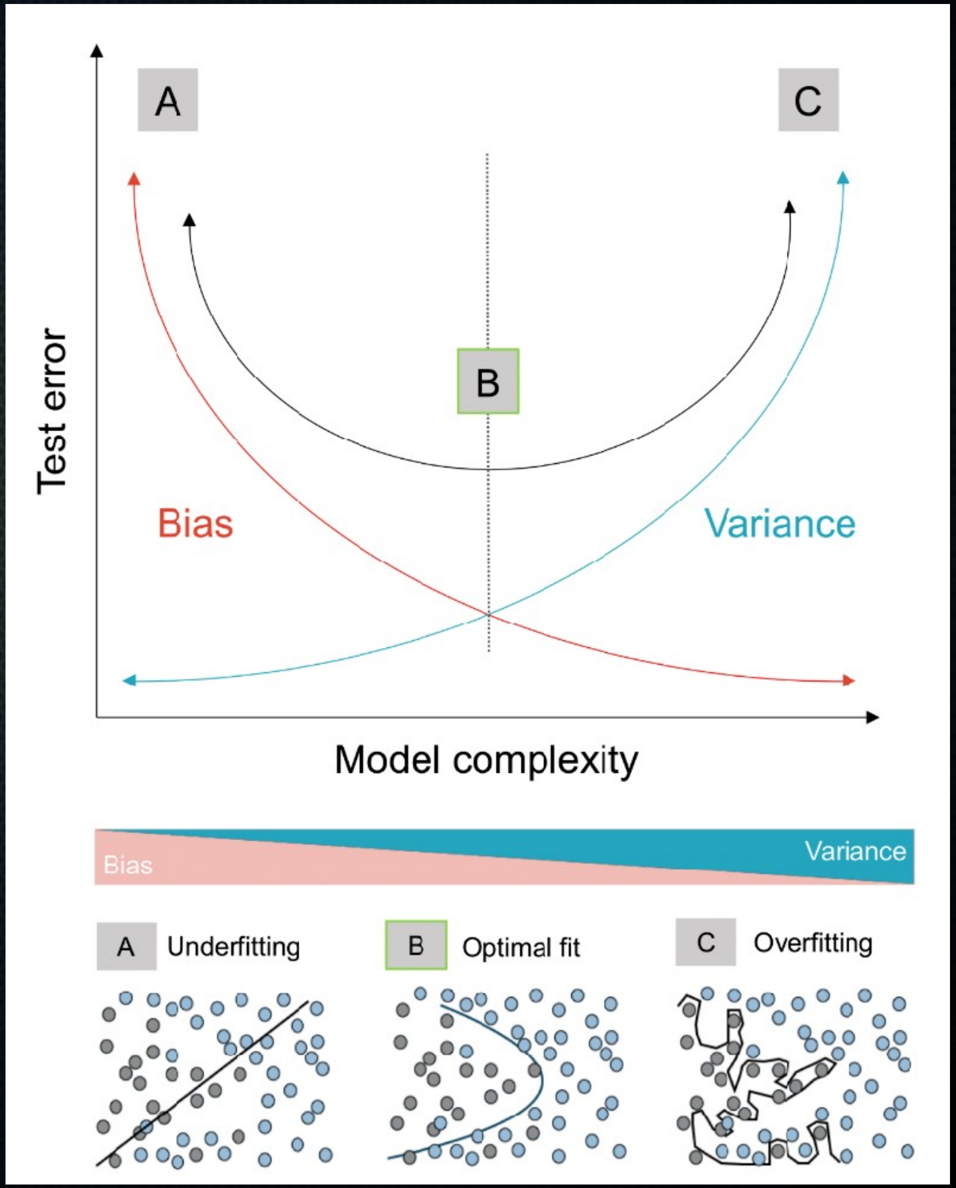
$$h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{100} x^{100}$$



Source: Andrew Ng, Coursera

Learning curves

- There‘s a trade-off between bias and variance
- Learning curves allow you to diagnose what your algorithm might be suffering from



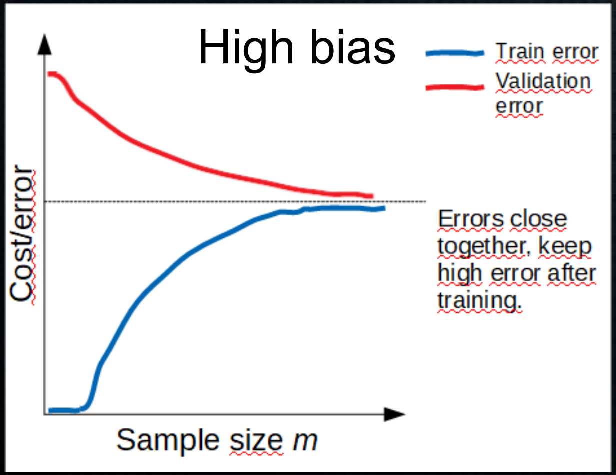
Source: Kernbach, J. M., & Staartjes, V. E. (2020). Machine learning-based clinical prediction modeling--A practical guide for clinicians. arXiv preprint arXiv:2006.15069.

Summary: cross-validation and learning curves

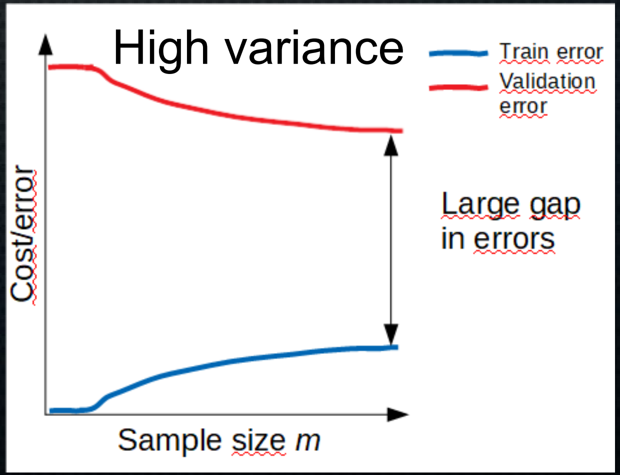
- Data is split into data to train on and a test set that you don‘t touch at all
- Training data is split into k folds, with (average) cross-validation error as proxy for how your algorithm performs on unseen data

Summary: cross-validation and learning curves

- Data is split into data to train on and a test set that you don‘t touch at all
- Training data is split into k folds, with (average) cross-validation error as proxy for how your algorithm performs on unseen data
- Learning curves allow you to diagnose whether your algorithm suffers from high bias or high variance



Underfitting: use more complex model



Overfitting: use less complex model or supply more training data

What can we do to find a good model?

- ~~Find a way to approximate generalisation error: how well do you do on unseen data?~~
- ~~See how error on seen and unseen data changes with amount of training data (plot learning curves)~~
- <u>***Automatically constrain the fitting by penalising the cost function for too many/too large parameters***</u>

Regularisation

- Change the cost function to apply a cost for complexity

Regularisation

- Change the cost function to apply a cost for complexity

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

Regularisation: ridge regression

- Change the cost function to apply a cost for complexity

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

$$J(\theta_0, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} (\theta_j)^2$$

- Make J a function of both the error of predictions given some parameters *and* the magnitude of those parameters themselves

Regularisation: ridge regression

- Change the cost function to apply a cost for complexity

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

$$\left. J(\theta_0, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} (\theta_j)^2 \right\}$$

By convention: don't shrink bias/intercept term

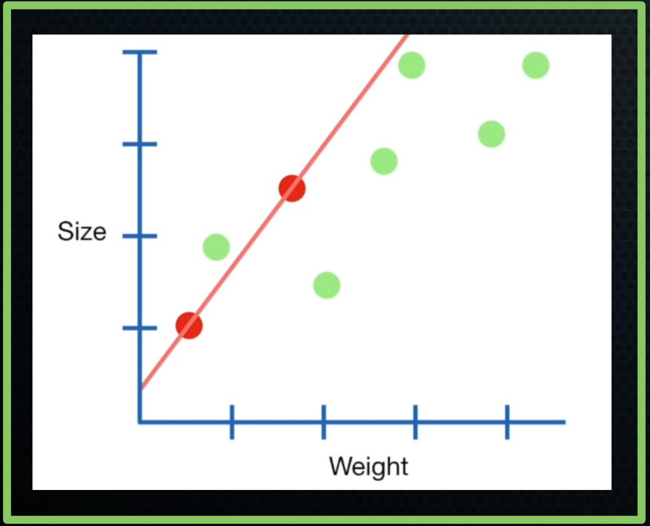
- Make J a function of both the error of predictions given some parameters *and* the magnitude of those parameters themselves

Regularisation: ridge regression

- Change the cost function to apply a cost for complexity

$$J(\theta_0, \ldots, \theta_n) = \boxed{\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x) - y^{(i)}\right)^2} + \lambda \sum_{j=1}^{n}(\theta_j)^2 \qquad h_\theta(x) = \theta_0 + \theta_1 x$$

- Add some **bias** (constrain hypothesis to a set with small parameter values) but reduces **variance**:



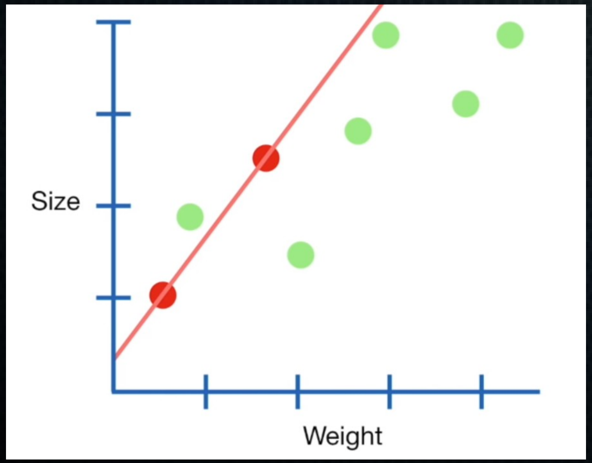
Source: StatQuest. “Regularization Part 1: Ridge (L2) Regression” YouTube, Joshua Starmer, 24 Sep. 2018,https://www.youtube.com/watch?v=Q81RR3yKn30

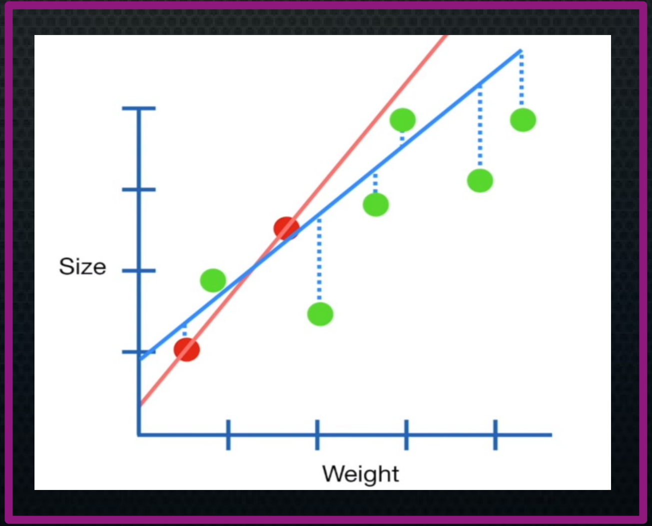
Regularisation: ridge regression

- Change the cost function to apply a cost for complexity

$$J(\theta_0, \dots, \theta_n) = \boxed{\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}(\theta_j)^2} \qquad h_\theta(x) = \theta_0 + \theta_1 x$$

- Add some **bias** (constrain hypothesis to a set with small parameter values) but reduces **variance**:





Constrained how much the line may increase with Weight (biased) → generalises better to test set

Source: StatQuest. “Regularization Part 1: Ridge (L2) Regression” YouTube, Joshua Starmer, 24 Sep. 2018,https://www.youtube.com/watch?v=Q81RR3yKn30

Regularisation: LASSO regression

- Same idea, slightly different execution:

$$J(\theta_0, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} (\theta_j)^2$$

Regularisation: LASSO regression

- Same idea, slightly different execution:

$$J(\theta_0, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \cancel{(\theta_j)^2}$$

Regularisation: LASSO regression

- Same idea, slightly different execution:

$$J(\theta_0, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} |\theta_j|$$

Regularisation: LASSO regression

- Same idea, slightly different execution:

$$J(\theta_0, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} |\theta_j|$$

- Ridge regression *shrinks* parameters/weights to 0, LASSO can make them 0 outright, i.e. simply removes uninformative features.

Regularisation: LASSO regression

- Same idea, slightly different execution:

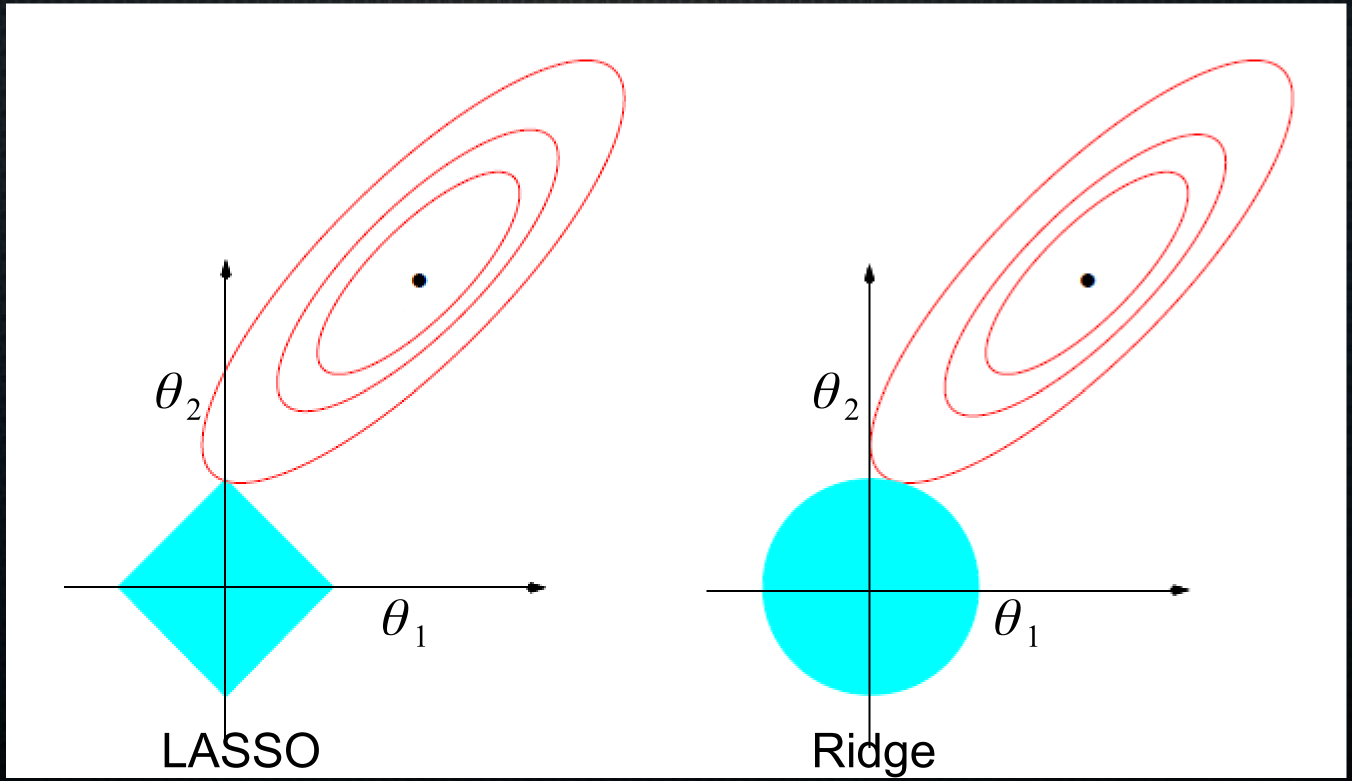
$$J(\theta_0, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} |\theta_j|$$

- Ridge regression *shrinks* parameters/weights to 0, LASSO can make them 0 outright, i.e. simply removes uninformative features. → Why?

Regularisation: Ridge vs. LASSO regression

- Ridge regression *shrinks* parameters/weights to 0, LASSO can make them 0 outright, i.e. simply removes uninformative features. → Why?

$$\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x)-y^{(i)}\right)^2$$

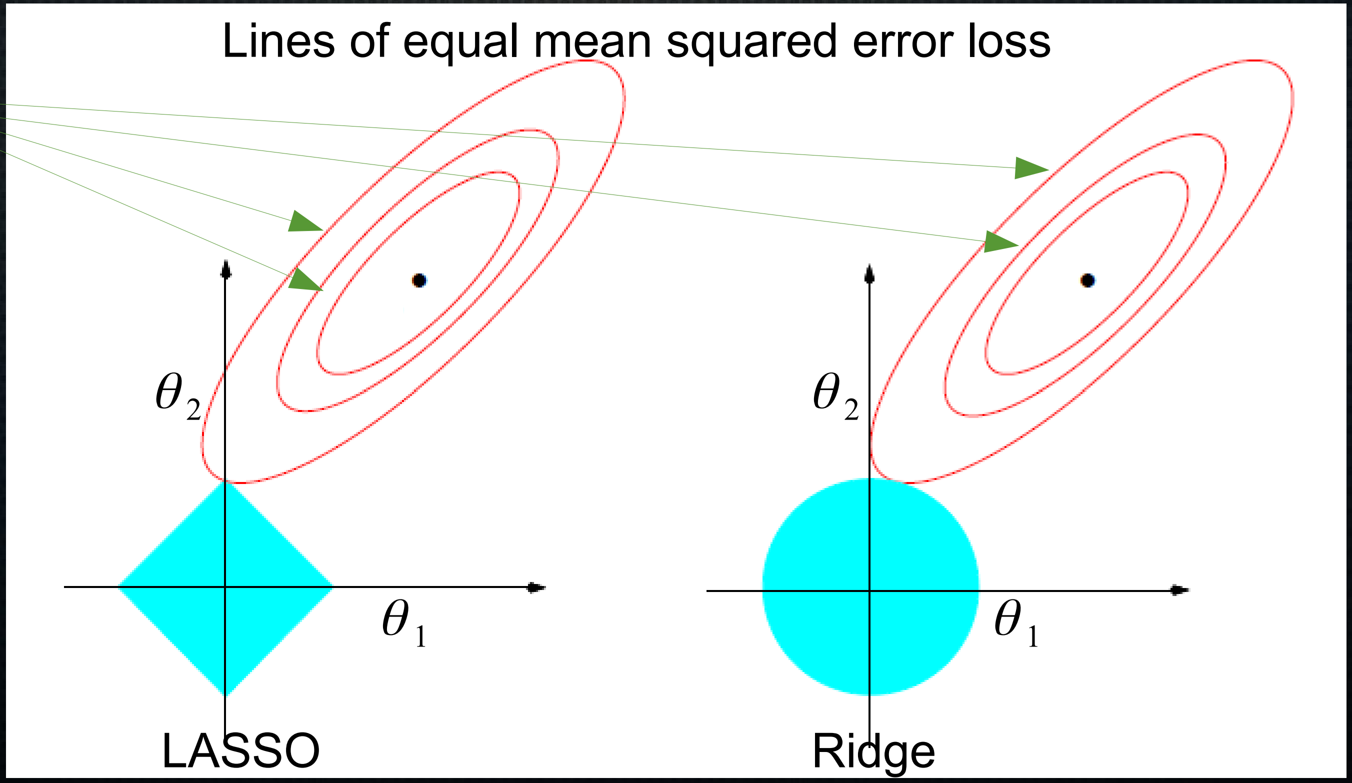


Source: Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

Regularisation: Ridge vs. LASSO regression

- Ridge regression *shrinks* parameters/weights to 0, LASSO can make them 0 outright, i.e. simply removes uninformative features. → Why?

$$\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x)-y^{(i)}\right)^2$$

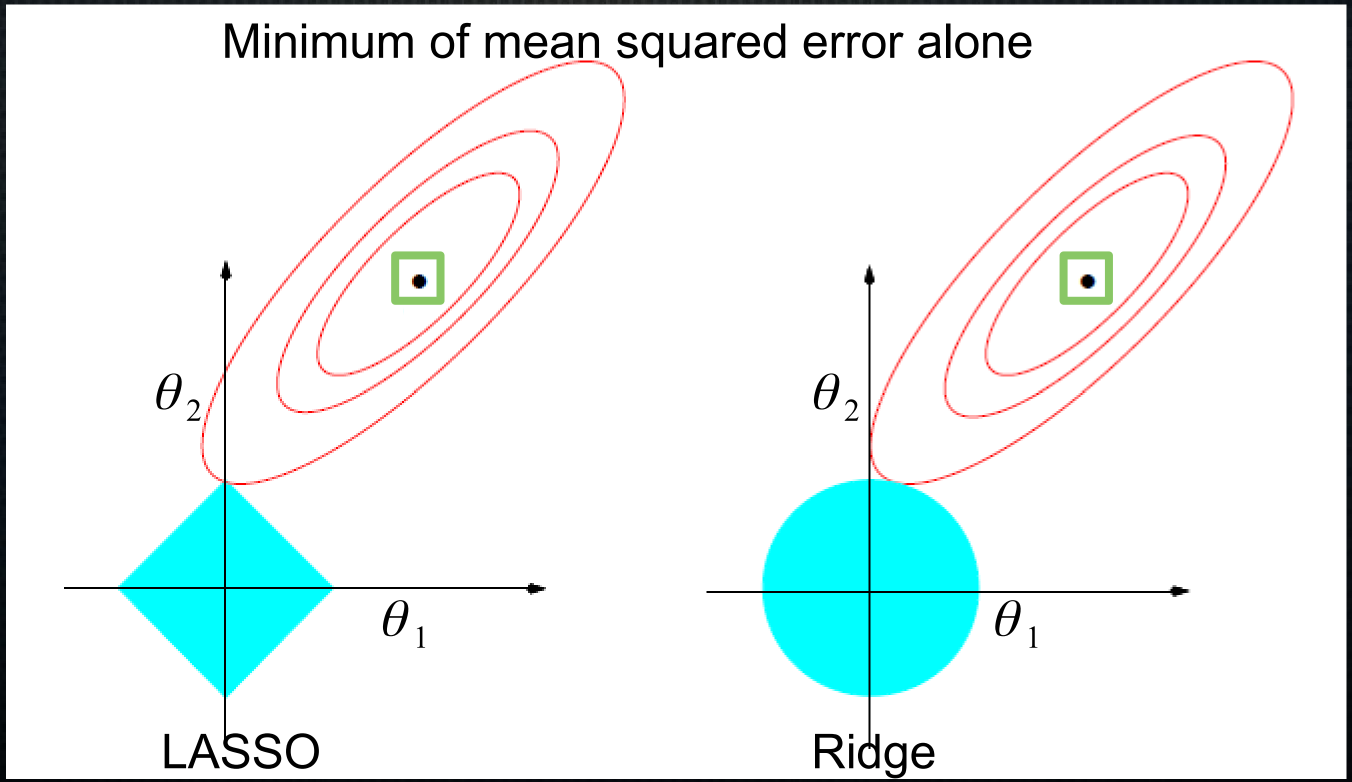


Source: Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

Regularisation: Ridge vs. LASSO regression

- Ridge regression *shrinks* parameters/weights to 0, LASSO can make them 0 outright, i.e. simply removes uninformative features. → Why?

$$\boxed{\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x)-y^{(i)}\right)^2}$$



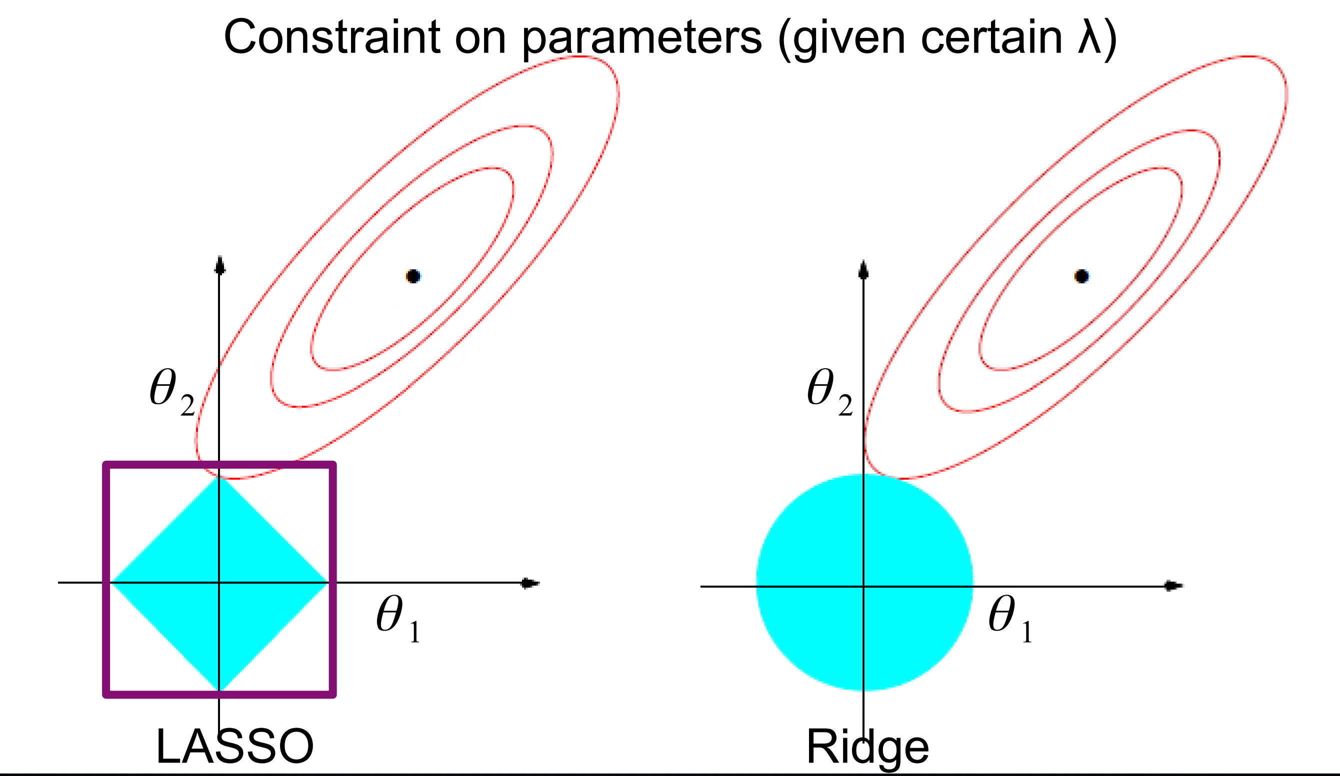
Source: Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

Regularisation: Ridge vs. LASSO regression

- Ridge regression *shrinks* parameters/weights to 0, LASSO can make them 0 outright, i.e. simply removes uninformative features. → Why?

$$\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x)-y^{(i)}\right)^2$$

$$\boxed{\lambda\sum_{j=1}^{n}|\theta_j|}$$



Source: Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

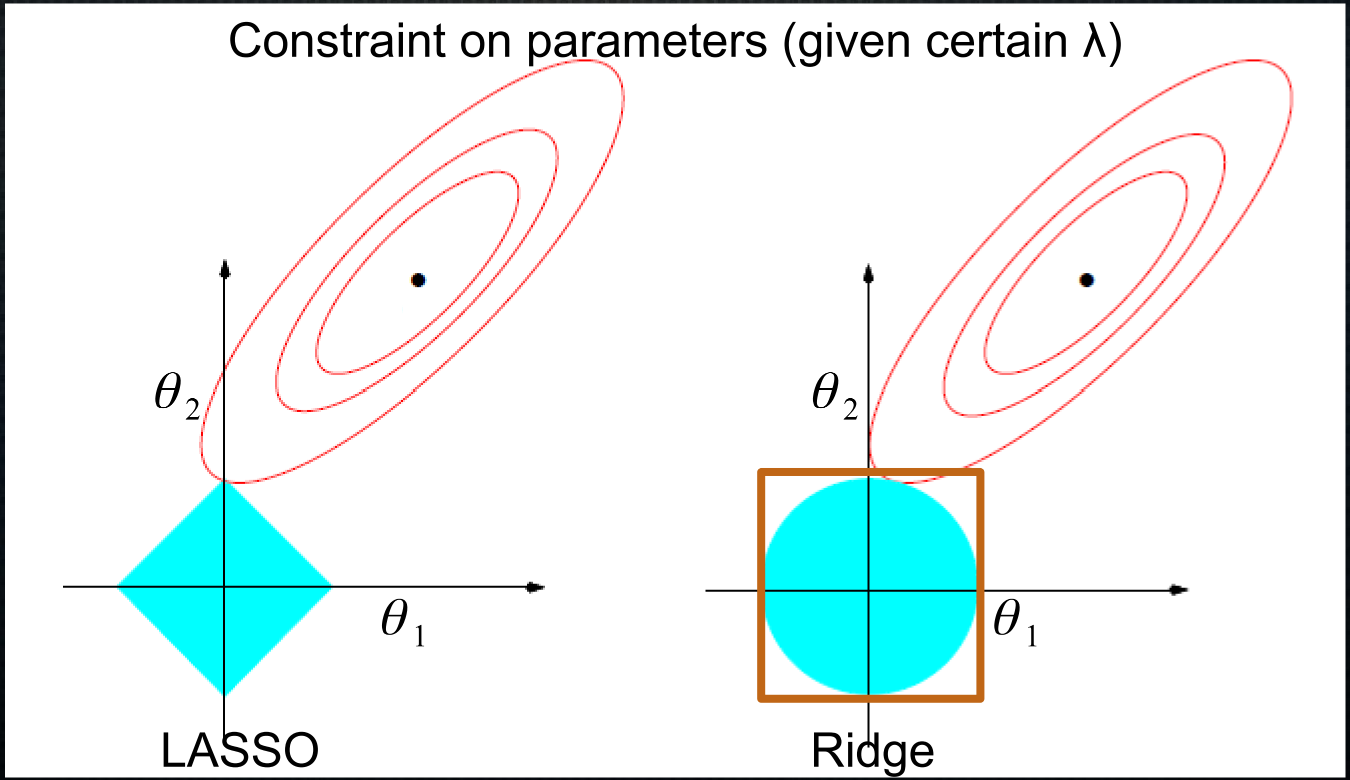
Regularisation: Ridge vs. LASSO regression

- Ridge regression *shrinks* parameters/weights to 0, LASSO can make them 0 outright, i.e. simply removes uninformative features. → Why?

$$\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x)-y^{(i)}\right)^2$$

$$\lambda\sum_{j=1}^{n}|\theta_j|$$

$$\boxed{\lambda\sum_{j=1}^{n}(\theta_j)^2}$$



Source: Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

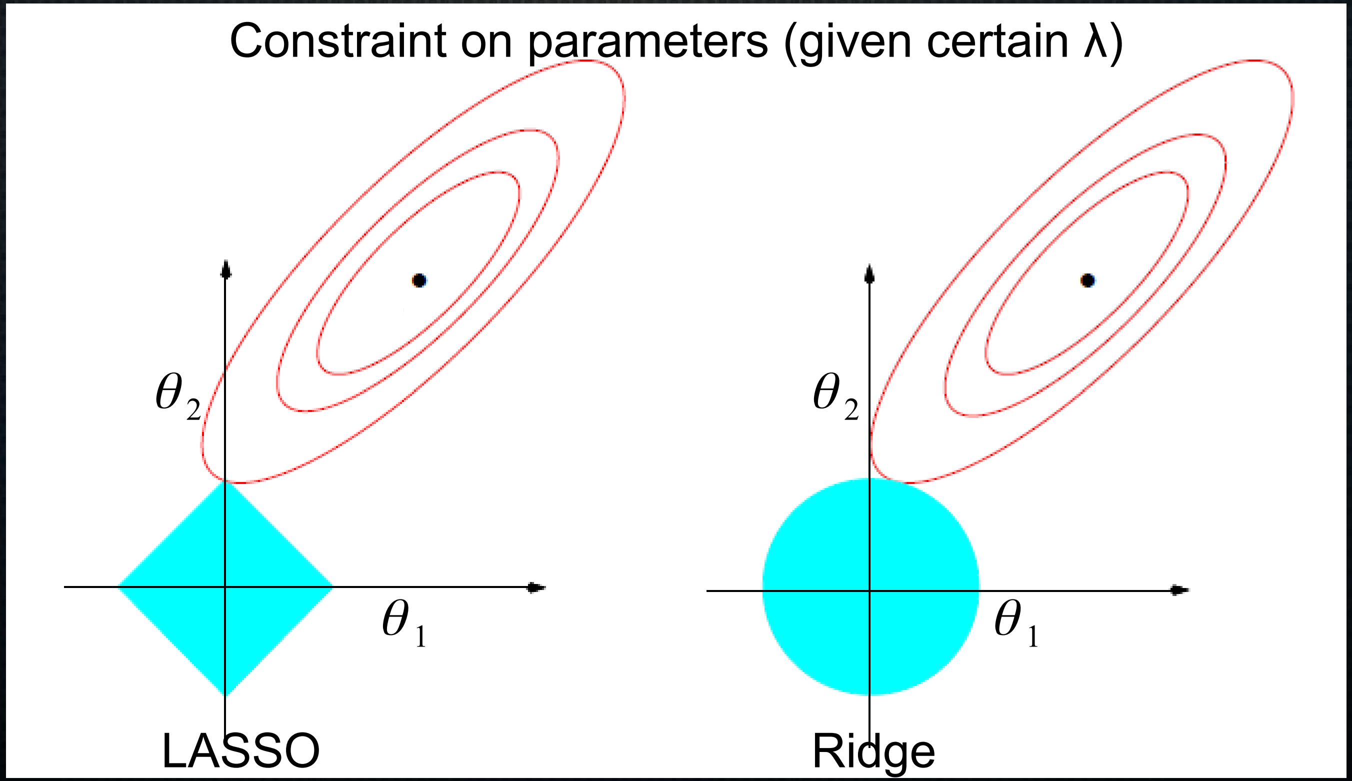
Regularisation: Ridge vs. LASSO regression

- Optimum = best least squares fit given constraint = intersect red lines with blue area.

$$\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x)-y^{(i)}\right)^2$$

$$\lambda\sum_{j=1}^{n}|\theta_j|$$

$$\lambda\sum_{j=1}^{n}(\theta_j)^2$$



Source: Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

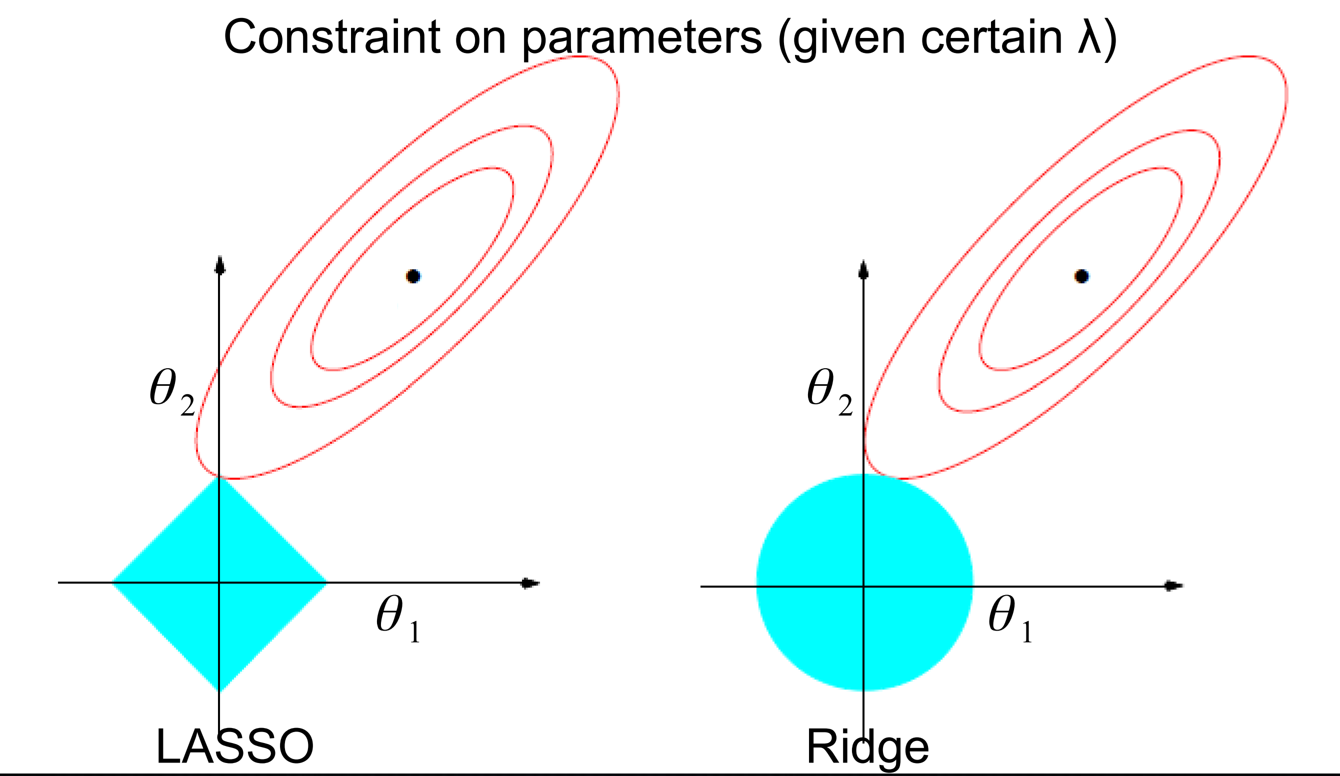
Regularisation: Ridge vs. LASSO regression

- Optimum = best least squares fit given constraint = intersect red lines with blue area.
- LASSO: can be at tip of rhombus, where theta1 = 0.
- Ridge: intersection ellips with circle never where either = 0

$$\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x)-y^{(i)}\right)^2$$

$$\lambda\sum_{j=1}^{n}|\theta_j|$$

$$\lambda\sum_{j=1}^{n}(\theta_j)^2$$



Source: Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

What about λ?

$$J(\theta_0, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} |\theta_j|$$

- Lambda is another hyperparameter
- Intuition: higher lambda → constrain parameters more, i.e. increase **bias**.
  lower lambda → constrain parameters less, i.e. increase **variance**

What about λ?

$$J(\theta_0, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} |\theta_j|$$

- How to pick a good value?

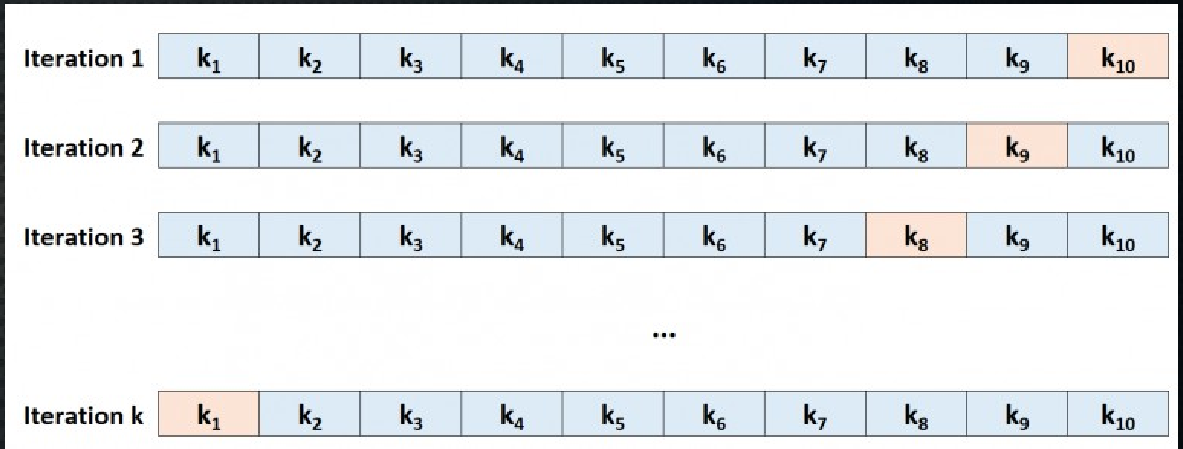
What about λ?

$$J(\theta_0, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} |\theta_j|$$

- How to pick a good value?
- Cross-validation!

What about λ?

$$J(\theta_0, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} |\theta_j|$$

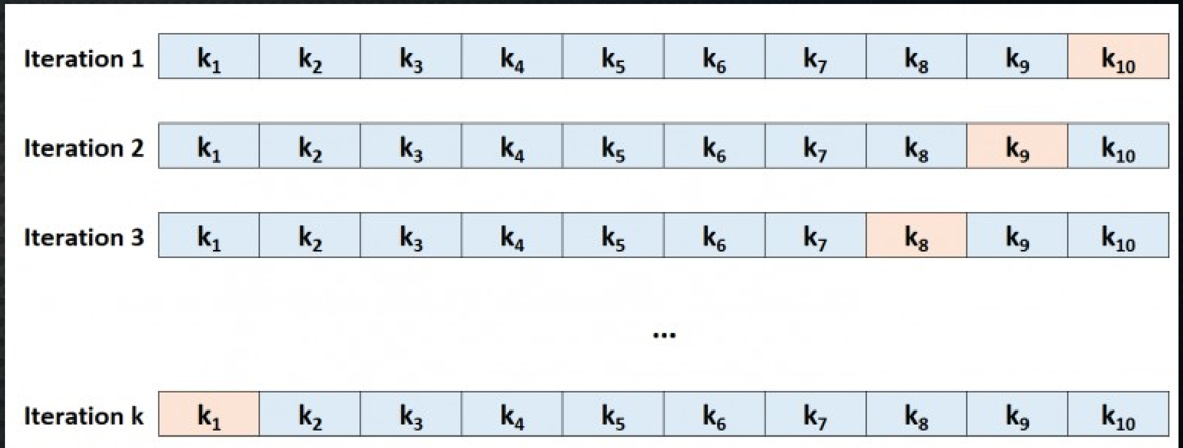


- How to pick a good value?
- Cross-validation!
  - Vary lambda over orders of magnitude (0.3, 1, 3, 7,10, 30, 100)
  - For each lambda:
    - Train on training set using regularisation term

What about λ?

$$J(\theta_0, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} |\theta_j|$$

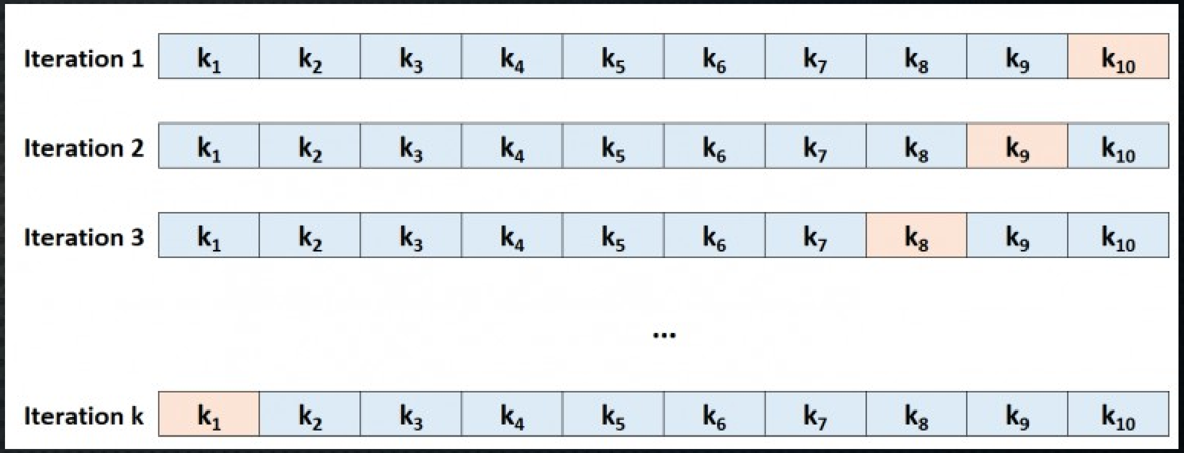
- How to pick a good value?
- Cross-validation!
  - Vary lambda over orders of magnitude (0.3, 1, 3, 7,10, 30, 100)
  - For each lambda:
    - Train on training set using regularisation term
    - Calculate validation error ***without regularisation term***



Don‘t care about parameter sizes, just care about how good your predictions are!

What about λ?

$$J(\theta_0, \ldots, \theta_n) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}|\theta_j|$$



- How to pick a good value?
- Cross-validation!
  - Vary lambda over orders of magnitude (0.3, 1, 3, 7,10, 30, 100)
  - For each lambda:
    - Train on training set using regularisation term
    - Calculate validation error ***without regularisation term***
  - Pick lambda that gives best (average) validation error!

What can we do to find a good model?

- ~~Find a way to approximate generalisation error: how well do you do on unseen data?~~
- ~~See how error on seen and unseen data changes with amount of training data (plot learning curves)~~
- ~~Automatically constrain the fitting by penalising the cost function for too many/too large parameters~~

All done!

Summary

- We want our models to generalise well but have to contend with **bias** and **variance**
- We can measure (by proxy) how well we generalise using cross-validation
- We can plot learning curves for different subsamplings of the data to diagnose bias and variance
- To avoid overfitting we can use regularisation terms in the cost function, (slightly) increasing bias but decreasing variance
- Hyperparameters (such as λ) can be chosen by trying different ones and seeing what gives best results on the validation data