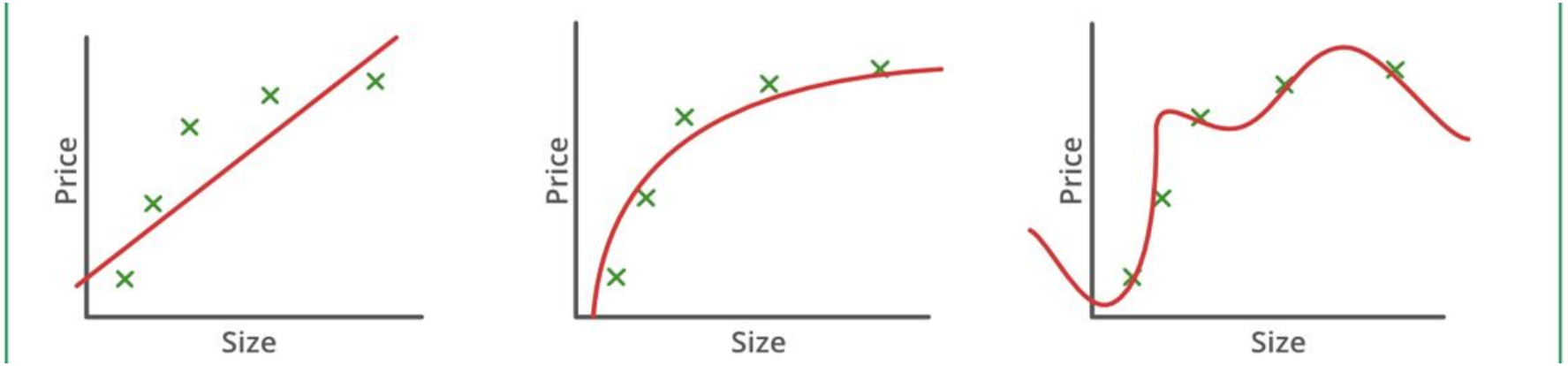
Learning to generalize. Bias-variance tradeoff. Regularization

Intuition

Lecture 16

by Marina Barsky

Housing price prediction: models



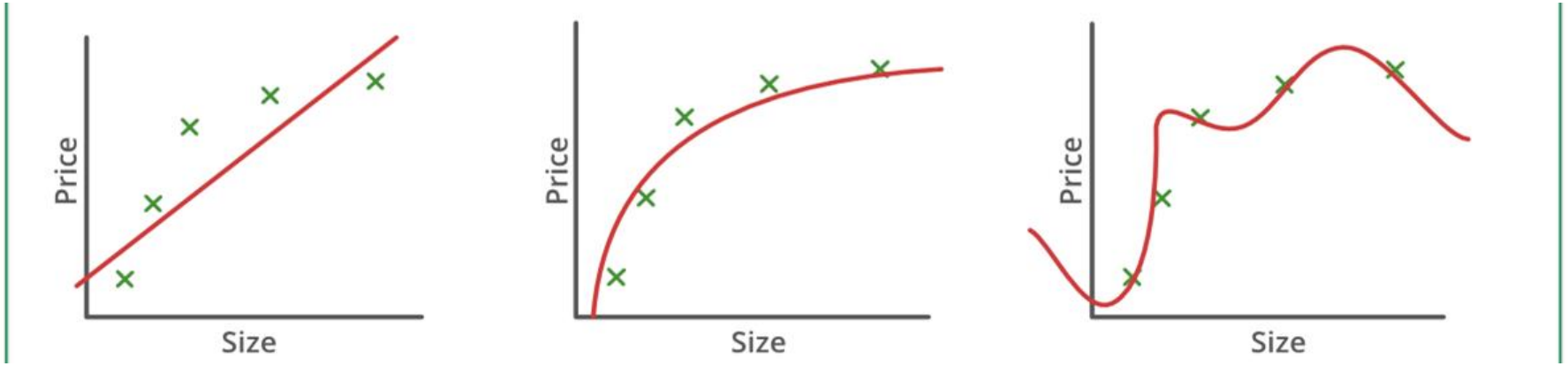
$y = wx + b$

$y = w_1x + w_2x^2 + b$

$y = w_1x + w_2x^2 + + w_2x^2 + w_2x^2 + b$

- We build a model by fitting the line to data
- We can fit many different models: linear model, square model or a very high-degree polynomial
- Which model is better?

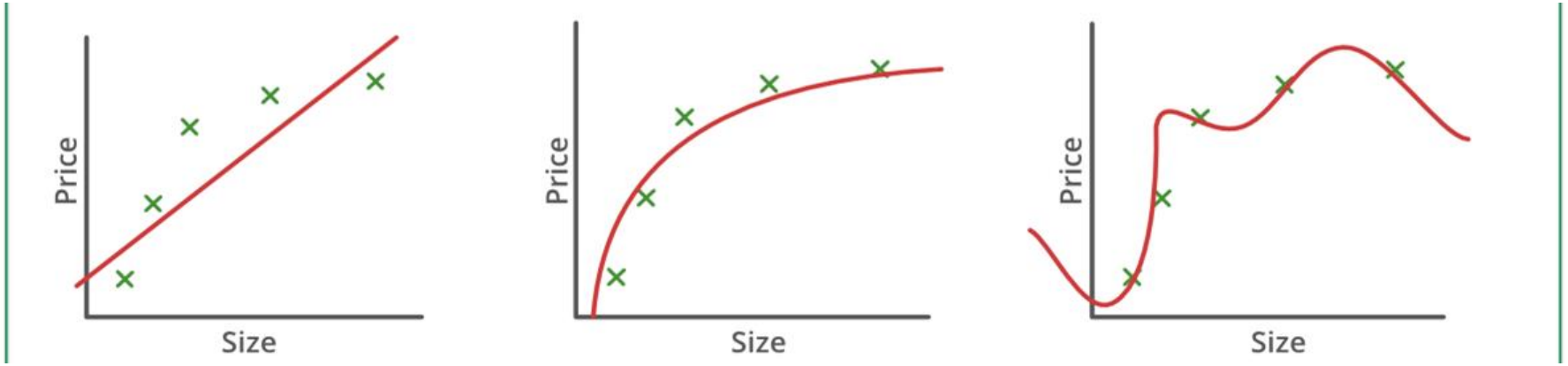
Housing price prediction: models



$y = wx + b$ $y = w_1x + w_2x^2 + b$ $y = w_1x + w_2x^2 + \ldots + w_3x^3 + w_4x^5 + b$

- In this example we can "see" that the shape of our data closely resembles a second degree polynomial
- But that is only clear if we can visualize the dataset (in two dimensions)
- It is not easy to visualize a multi-dimensional dataset!

Housing price prediction: models



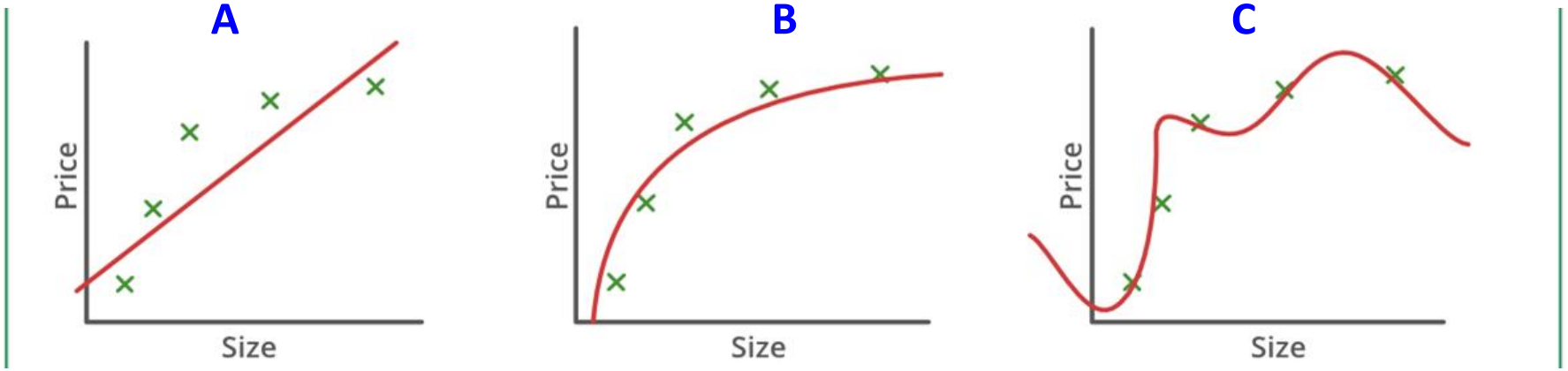
$y = wx + b$ $y = w_1x + w_2x^2 + b$ $y = w_1x + w_2x^2 + \ldots + w_3x^3 + w_4x^5 + b$

We try different models and choose the one with

- The best train set accuracy?
- The best test set accuracy?

or?

Simple models



$y = wx + b$

$y = w_1x + w_2x^2 + b$

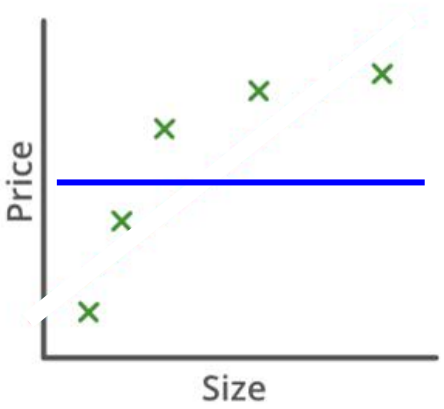
$y = w_1x + w_2x^2 + \ldots + w_3x^3 + w_4x^5 + b$

Simple

Simple models assume that the relationship between y and x is simple (for example, linear)

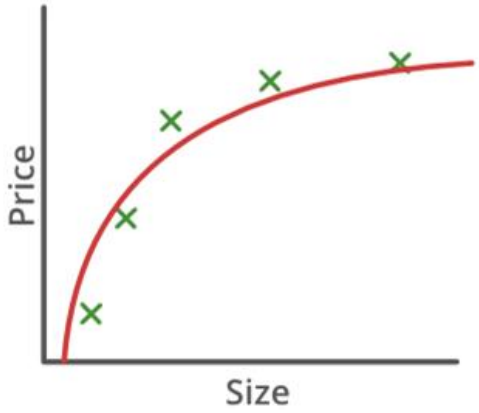
Simple models

A



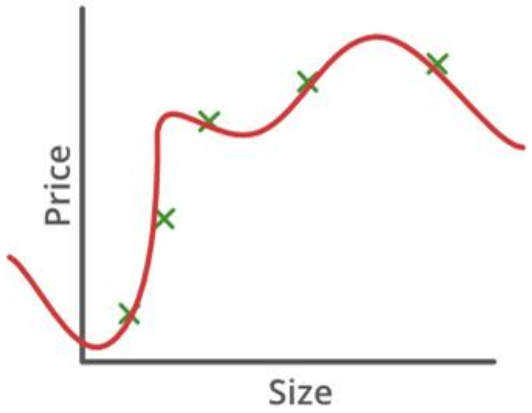
$y = wx + b$

B



$y = w_1x + w_2x^2 + b$

C



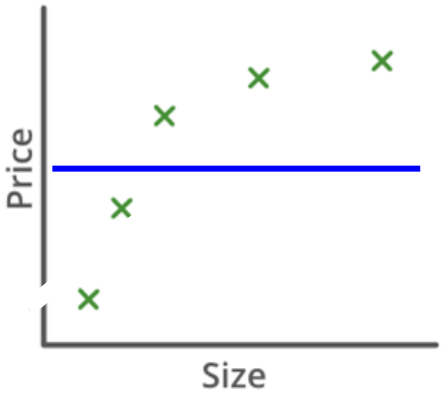
$y = w_1x + w_2x^2 + \ldots + w_3x^3 + w_4x^5 + b$

Extremely simple

In the extreme case, simple models assume that for each x the value of y is exactly the same (ignoring real data altogether)

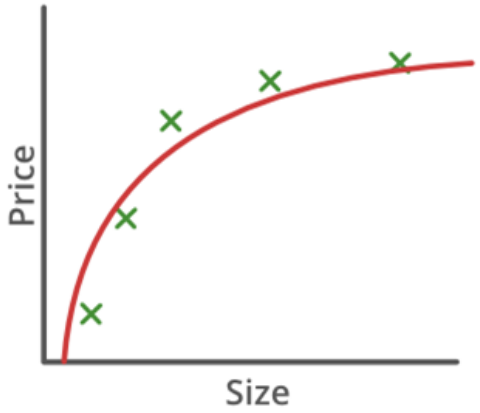
Simple models

A



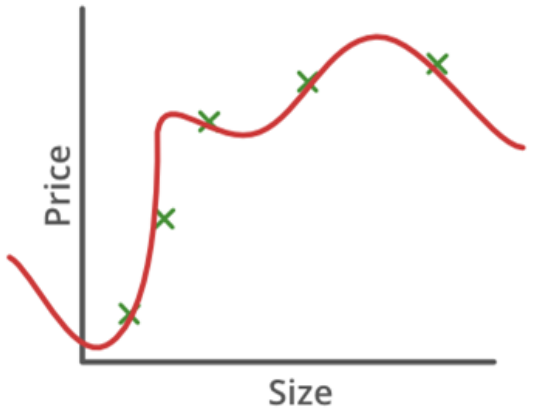
$y = wx + b$

B



$y = w_1x + w_2x^2 + b$

C



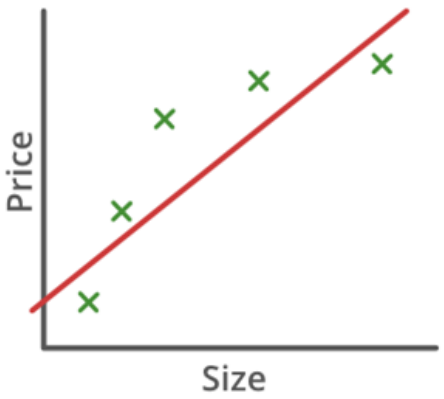
$y = w_1x + w_2x^2 + \ldots + w_3x^3 + w_4x^5 + b$

Extremely Simple

We say that the model ignores the data and is ***biased towards a specific value***

Simple models

A



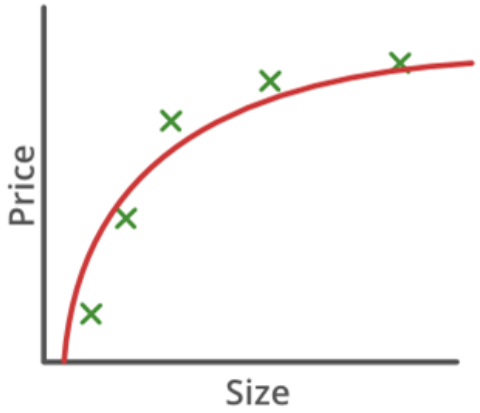
$y = wx + b$

Simple

High bias

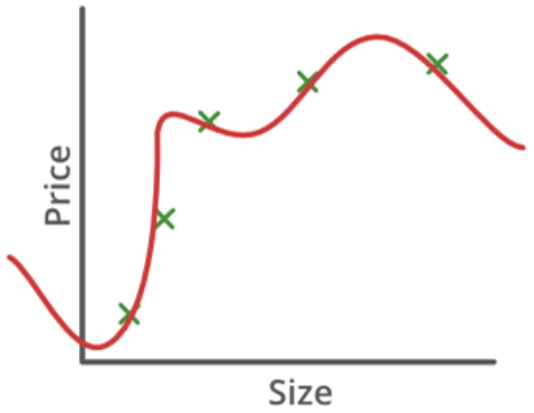
Underfitting

B



$y = w_1x + w_2x^2 + b$

C



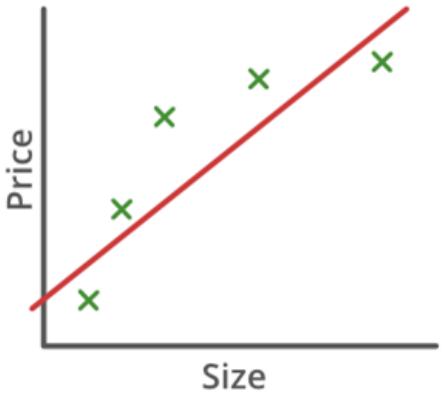
$y = w_1x + w_2x^2 + \ldots + w_3x^3 + w_4x^5 + b$

The model is **biased** towards a specific value or towards a specific function - ignoring the data and causing **underfitting**

What will be the training error? And testing error?

Simple models

A



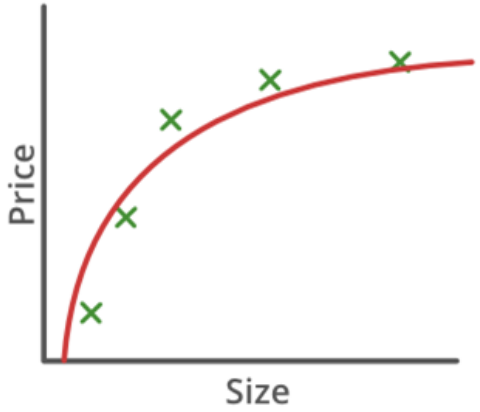
$y = wx + b$

Simple

High bias, low variance

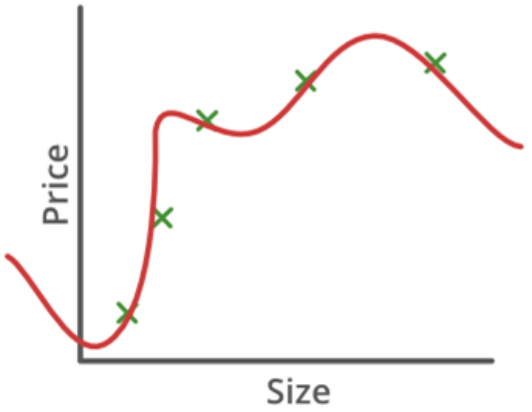
Underfitting

B



$y = w_1x + w_2x^2 + b$

C



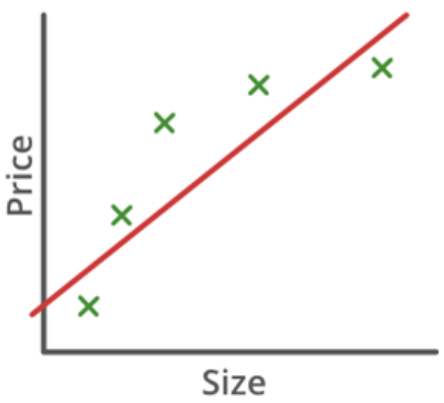
$y = w_1x + w_2x^2 + \ldots + w_3x^3 + w_4x^5 + b$

Training error is high, and if the data is really non-linear - the testing error will also be very high

Solution: make model more complex by forcing the function vary in accordance with data – increase model's ***variance***

Overcomplicated models

A



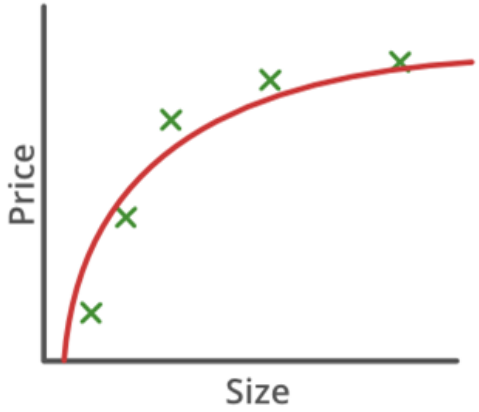
$y = wx + b$

Simple

High bias, low variance

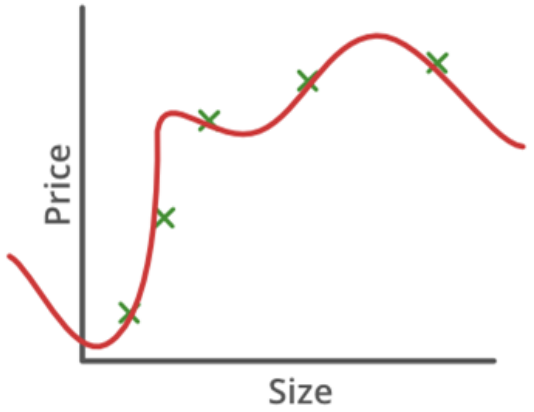
Underfitting

B



$y = w_1x + w_2x^2 + b$

C



$y = w_1x + w_2x^2 + \ldots + w_3x^3 + w_4x^5 + b$

Overcomplicated

Low bias, high variance

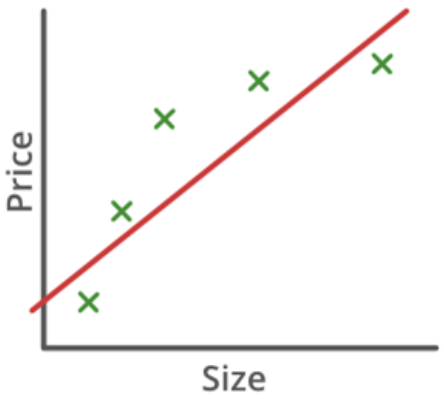
Overfitting

We build more and more complex models which closely align with the data (reflect the variance in data), and at some point we fit the data perfectly (C)

What would be the training error? And the testing error?

Overcomplicated models

A



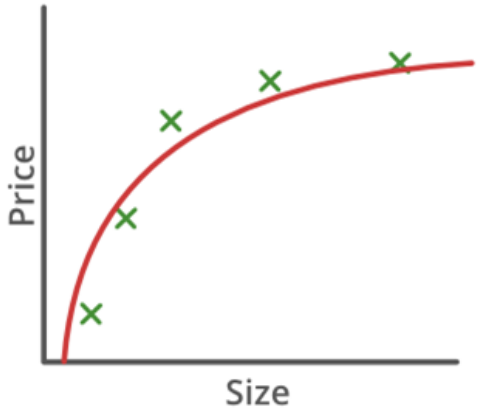
$y = wx + b$

Simple

High bias, low variance

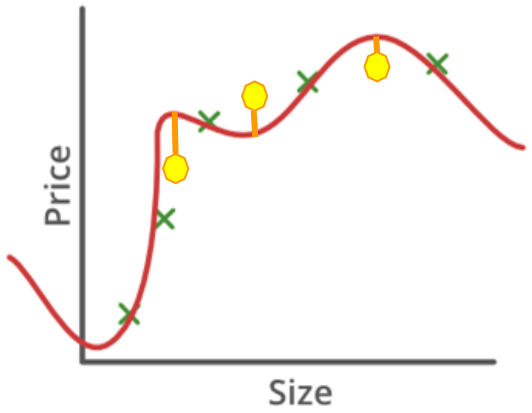
Underfitting

B



$y = w_1x + w_2x^2 + b$

C



$y = w_1x + w_2x^2 + \ldots + w_3x^3 + w_4x^5 + b$

Overcomplicated

Low bias, high variance

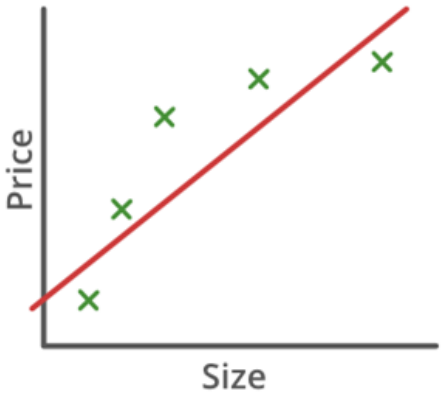
Overfitting

We build more and more complex models which closely vary with the data, and at some point we fit the data perfectly

The training error is close to 0, and the testing error is very high

Overcomplicated models

A



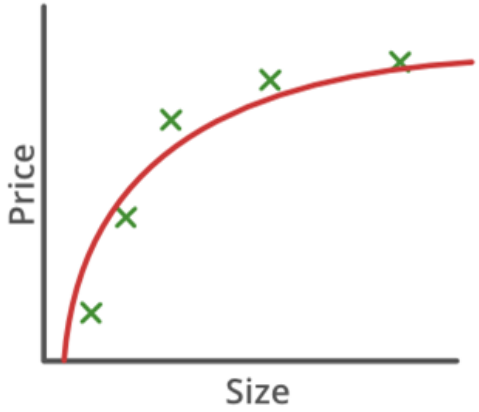
$y = wx + b$

Simple

High bias, low variance

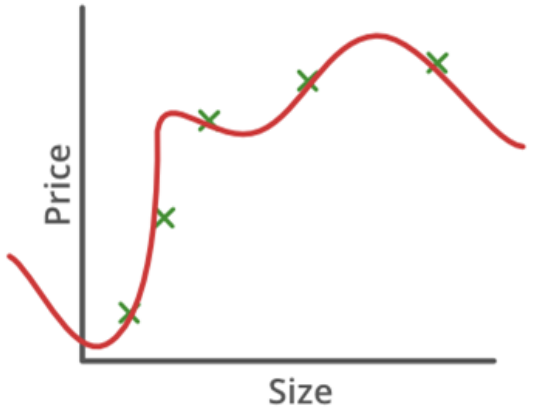
Underfitting

B



$y = w_1x + w_2x^2 + b$

C



$y = w_1x + w_2x^2 + \ldots + w_3x^3 + w_4x^5 + b$

Overcomplicated

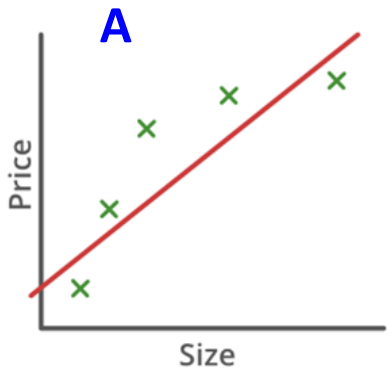
Low bias, high variance

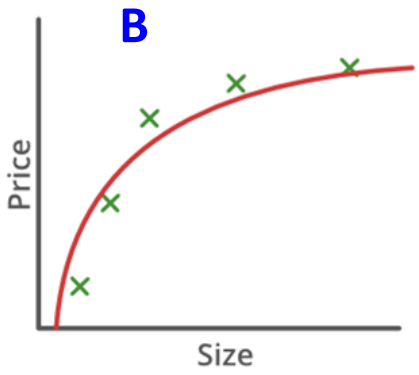
Overfitting

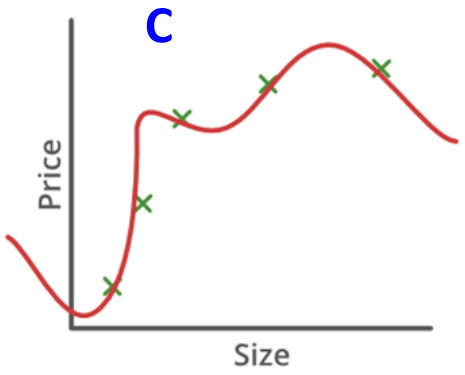
We say that our model just **memorized** the training points and

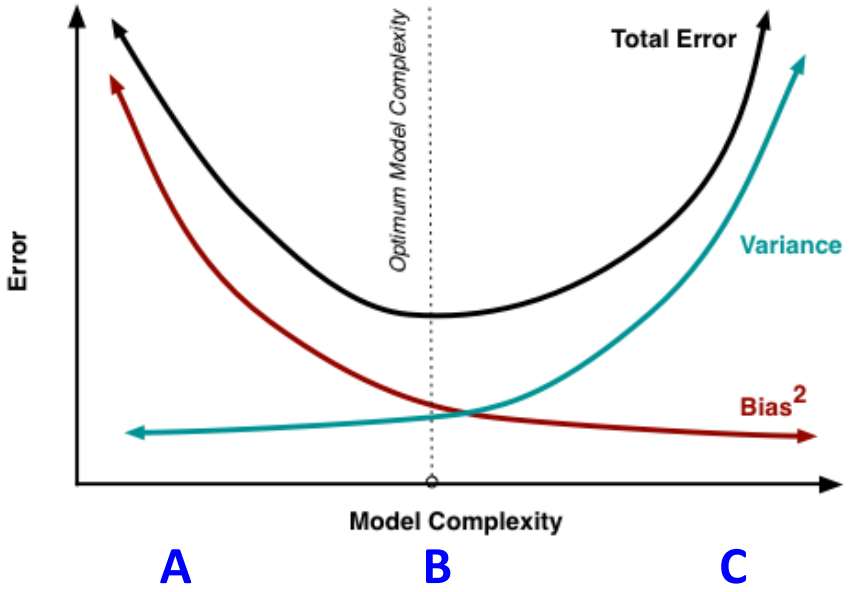
failed to generalize

Model error always has 3 parts









E = **Bias^2**+**Variance**+**Irreducible Error**

The goal is to select a model with the best predictive power possible - model **B**

Bias-variance tradeoffs

large bias/small variance:

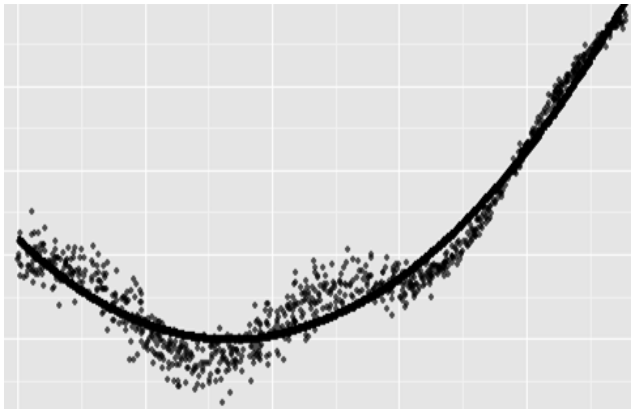
- too few features
- highly pruned decision trees
- large-k k-NN

small bias/high variance:

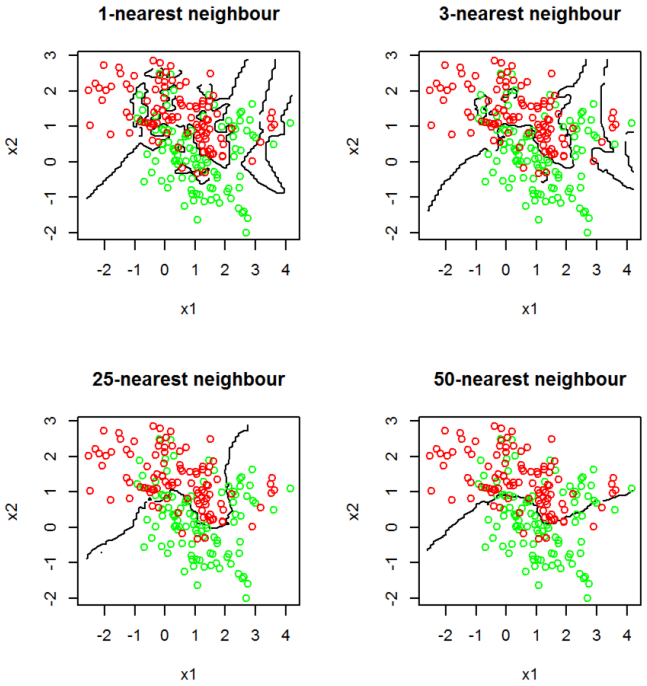
- too many features
- unpruned trees
- small-k k-NN

Click me

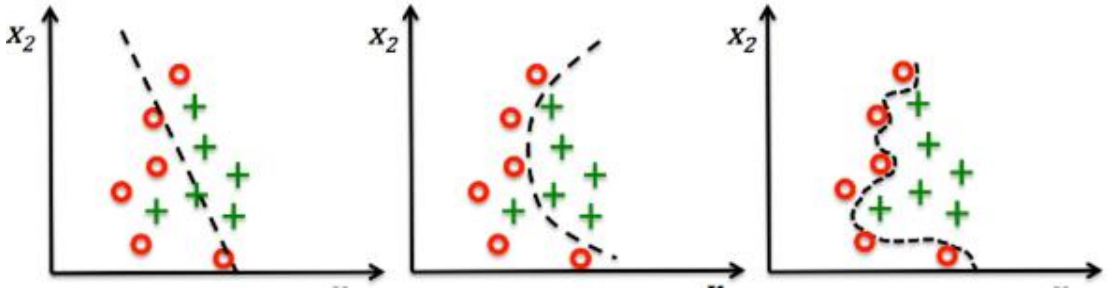
Identify which of these models have high/low bias/variance

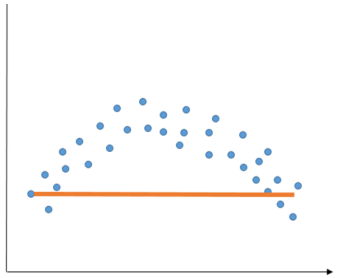


Polynomial Regression



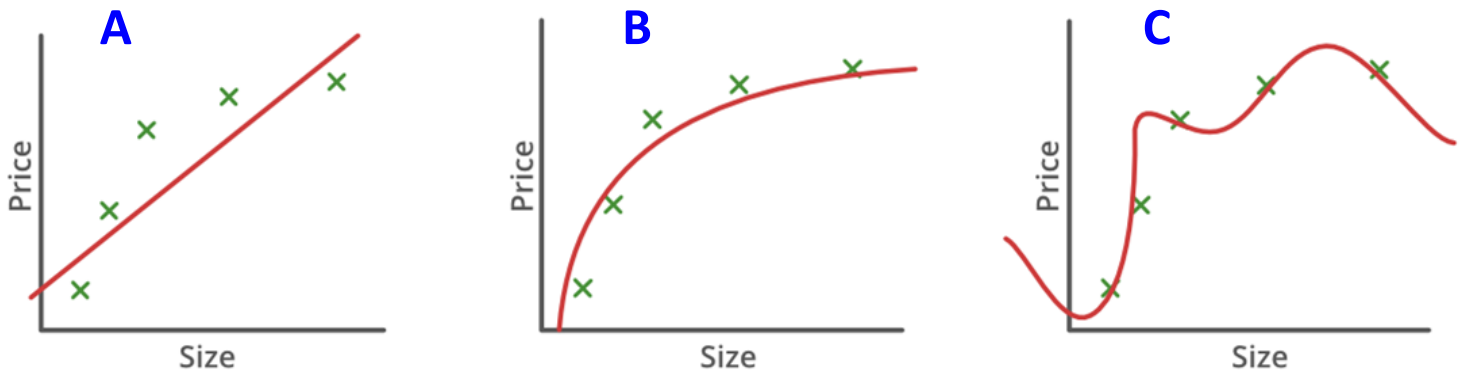
Decision Boundary (Logistic Regression, SVM)





Linear Regression

So, how do we discover model B?



We start from the simplest possible model and evaluate the **training error**

We gradually increase complexity gradually, the training error becomes smaller: we lowering bias and increasing variance

Each time we also test the model on the validation set (not used for learning):

- If **test error** is low and close to the train error - we found a good model!
- If test error is much higher than the train error - **model failed to generalize.** We followed data too closely and arrived at overfitting

Dealing with high-variance models (overfitting)

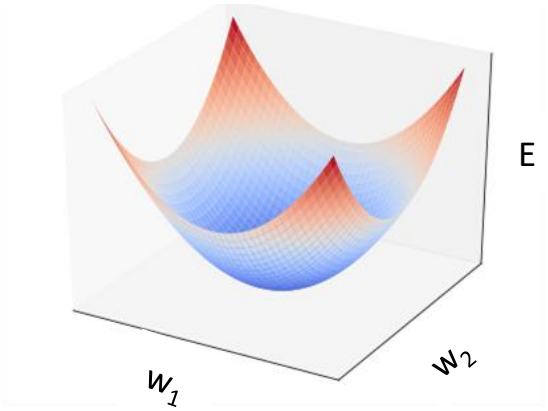
Solutions:

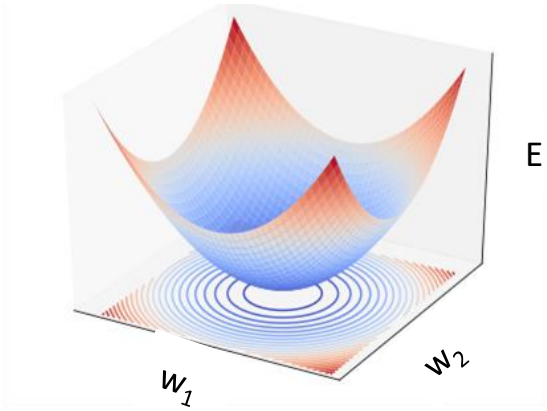
- Remove some features
  - Select “important” features manually
  - Use a feature selection algorithm

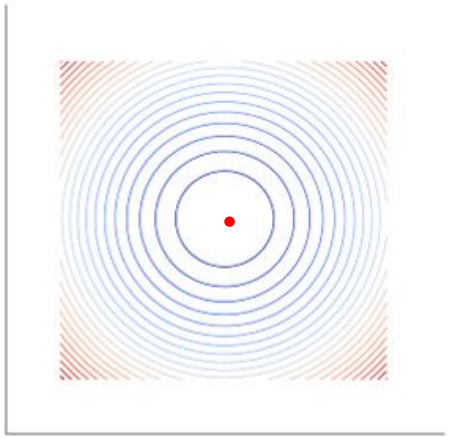
- Use regularization
  - Keep all features **but reduce their weights**

Regularization - a subset of methods used to encourage generalization in learned models, mainly by making it difficult for a model to concentrate on the fine-grained details of training data

Regularization for linear regression







Gradient descent finds the weights for the global minimum of SSR: the bottom of the bowl - the red dot in the projection (contours) of the hyperplane

By making error too close to 0, we can overfit the training data

L2 Norm or *Ridge* Regression

L2 Norm is an Euclidean distance norm of the form $\mathbf{|w_1|^2 + |w_2|^2}$

Let say we have *n* two-dimensional feature vectors, each vector $\mathbf{x}_i$ with its target variable t_i We then add an Euclidean distance norm to the modified cost function:

$$Error_{Ridge} = \sum_{i=1}^{n} (t_i - y_i)^2 + \lambda(|w_1|^2 + |w_2|^2)$$

where $y(\mathbf{x}) = w_1\mathbf{x}_1 + w_2\mathbf{x}_2 + b$ is an equation of the line (hyperplane) we are trying to fit to data using gradient descent

A particular value of (w_1, w_2) adds a new "bias" component to the error function and will **force to keep weights smaller**

The equation with smaller weights (smaller correlation between some features and the output) will have smaller variance (varying with data less)

Looking for a global minimum of a new objective cost function

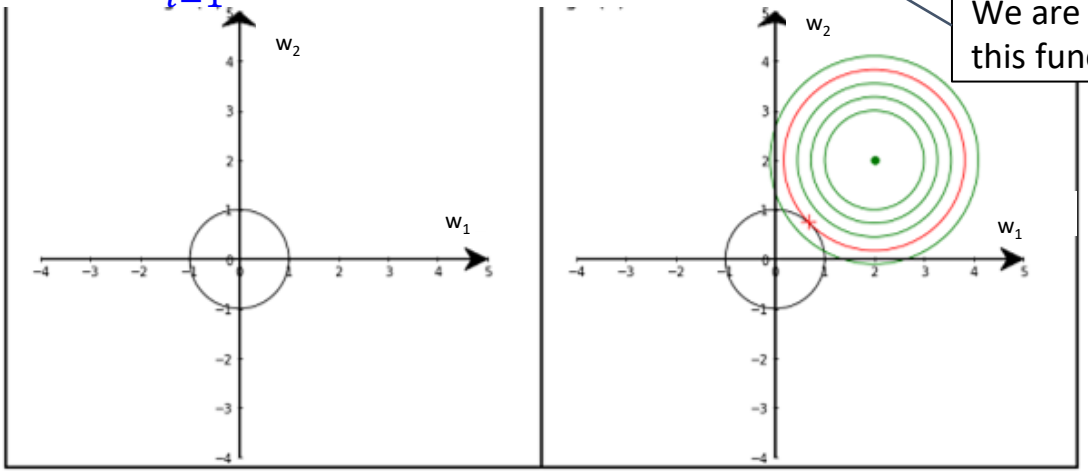
Modified Cost function with L2 Regularization:

$$Error_{Ridge} = \sum_{i=1}^{n} (t_i - y_i)^2 + \lambda(|w_1|^2 + |w_2|^2)$$

We now search for a global minimum of this cost function by taking partial derivatives with respect to b, w_1 and w_2

L2 Regularization: visually

$$Error_{Ridge} = \sum_{i=1}^{n} (t_i - y_i)^2 + \lambda(|w_1|^2 + |w_2|^2)$$



We are trying to minimize this function

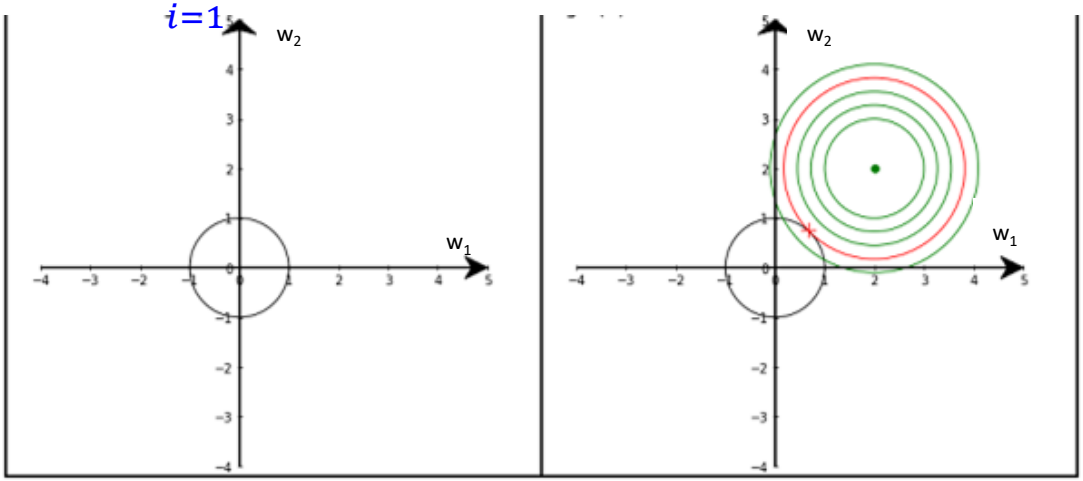
Let's say $\boldsymbol{\lambda}$= 1 and the minimum value of L2 norm is also 1.

Then there could be many possible values of (w_1, w_2) that give the value 1. All these combinations of (w_1, w_2) make a circle with radius 1

We are trying to minimize both the SSR and the bias - at the same time

L2 Regularization: visually

$$Error_{Ridge} = \sum_{i=1}^{n} (t_i - y_i)^2 + \lambda(|w_1|^2 + |w_2|^2)$$

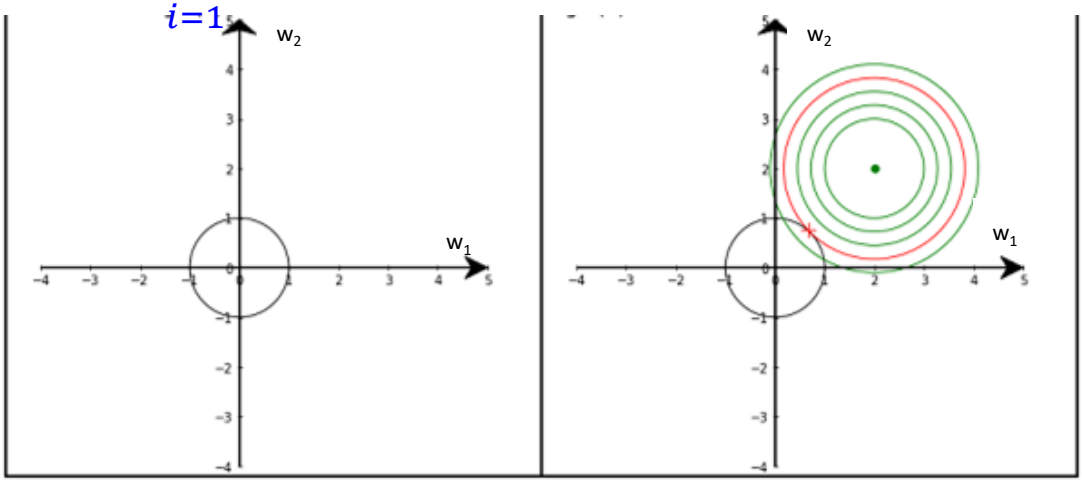


We are trying to minimize both the SSR and the bias - thus there are two forces pulling into different directions:

- Force 1: Bias term pulling w_1, and w_2 to lie somewhere on the black circle only
- Force 2: Gradient trying to descent to the global minimum indicated by green dot

L2 Regularization: visually

$$Error_{Ridge} = \sum_{i=1}^{n} (t_i - y_i)^2 + \lambda(|w_1|^2 + |w_2|^2)$$

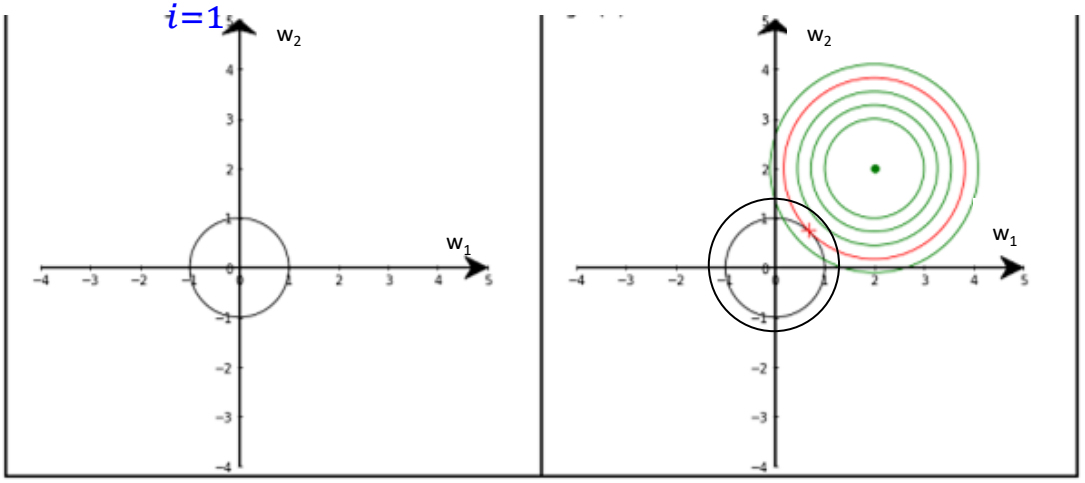


Both forces pull and eventually settle near the point of intersection indicated by ‘Red cross’

Descending further towards the green dot will increase the bias term. We can lower the bias term moving towards smaller weights (inside the circle), but that will increase the SSR

L2 Regularization: visually

$$Error_{Ridge} = \sum_{i=1}^{n} (t_i - y_i)^2 + \lambda(|w_1|^2 + |w_2|^2)$$



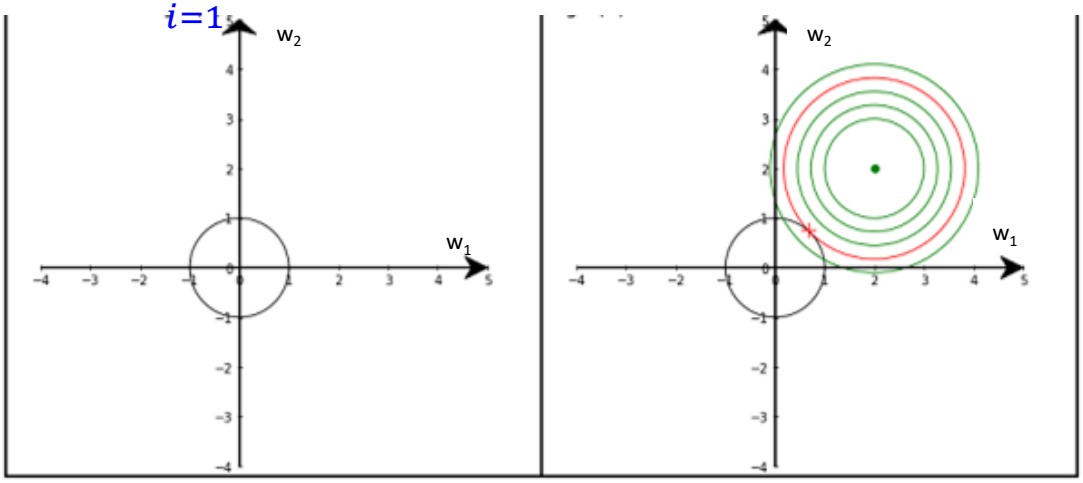
We assumed that $\boldsymbol{\lambda}$= 1 and the value of L2 norm is 1.

Let's keep $\boldsymbol{\lambda}$= 1 and try to minimize the overall bias term

The values of the norm then will be a set of different circles centered at (0,0) and the result of optimization will be somewhere at the intersection of one of these circles and the contours - where the overall function is minimized - for a given $\boldsymbol{\lambda}$

L2 Regularization: visually

$$Error_{Ridge} = \sum_{i=1}^{n} (t_i - y_i)^2 + \boldsymbol{\lambda}(|w_1|^2 + |w_2|^2)$$



Now try a different value - say λ = 0.5

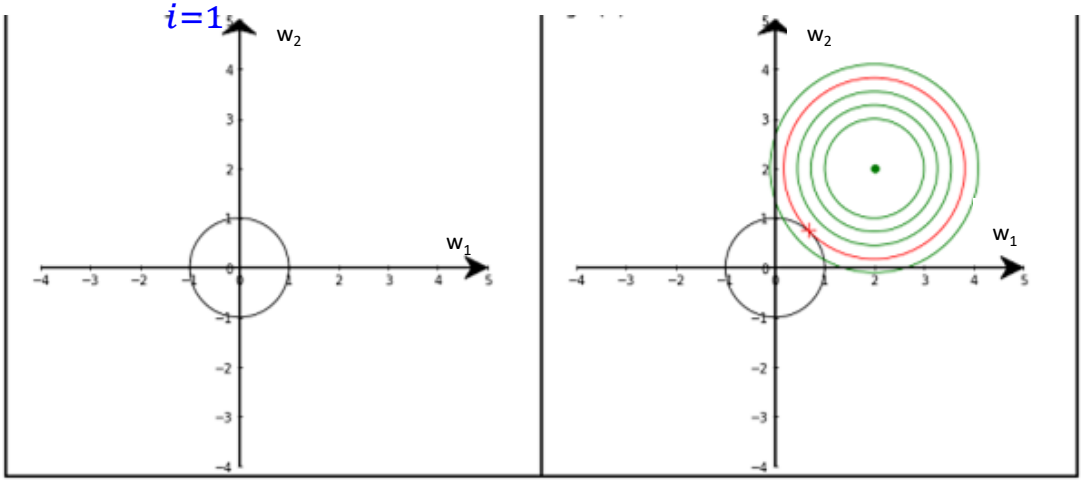
Repeat the Gradient Descent process for λ = 0.5 and test on validation set

This process is repeated for various values of λ - to identify the optimum value of λ

λ is called a ***hyperparameter***: the parameter that is not learned by the algorithm but is set by an experimenter

L2 Regularization: visually

$$Error_{Ridge} = \sum_{i=1}^{n} (t_i - y_i)^2 + \boldsymbol{\lambda}(|w_1|^2 + |w_2|^2)$$



The overall objective is always to have low cost after the Gradient Descent → the values of λ, w1,w2 are identified keeping that as objective

By the end of this process, the Variance is reduced - i.e. all weights are reduced in magnitude

L1 norm or Lasso Regression

In this case we minimize the following error function:

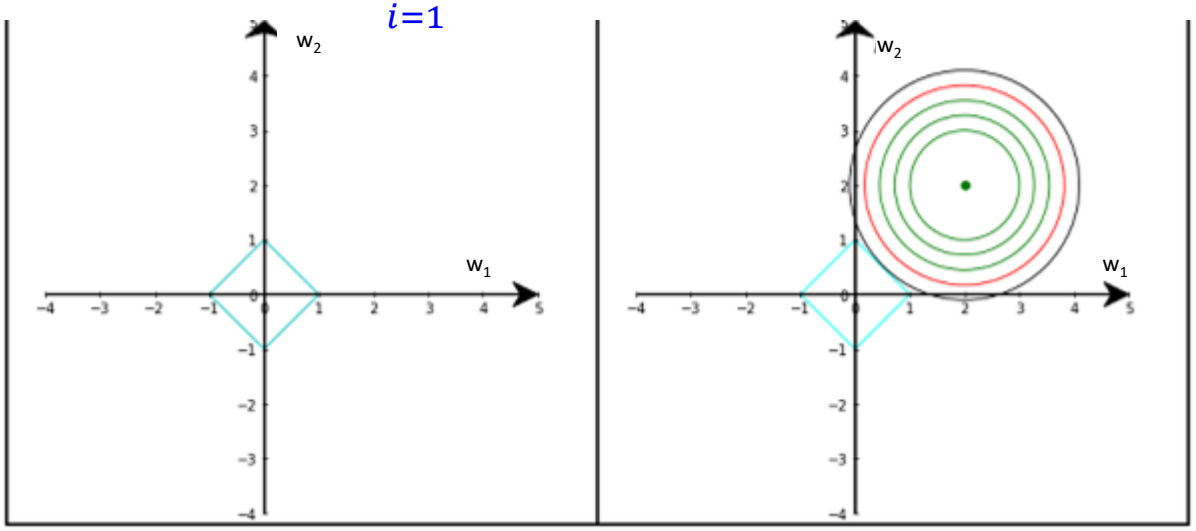
$$Error_{Lasso} = \sum_{i=1}^{n} (t_i - y_i)^2 + \boldsymbol{\lambda}(|\boldsymbol{w_1}| + |\boldsymbol{w_2}|)$$

We still try to keep the weights from getting too big - but the weights are now not squared

This is less restrictive: the bias term is a sum of weights, not the sum of squared weights

L1 Regularization: visually

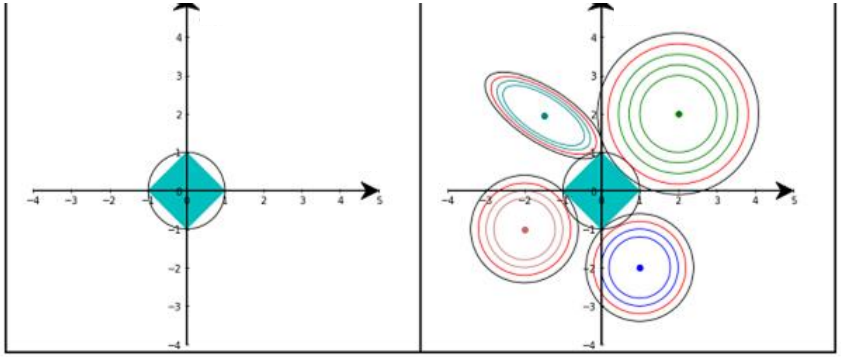
$$Error_{Lasso} = \sum_{i=1}^{n} (t_i - y_i)^2 + \boldsymbol{\lambda}(|\boldsymbol{w_1}| + |\boldsymbol{w_2}|)$$



Every sum of weights we are trying to minimize would lie on a diamond defined by line equations

Again, we are trying to minimize both L1 norm and SSR and the best solution is on the intersection of the contours with one of the diamonds

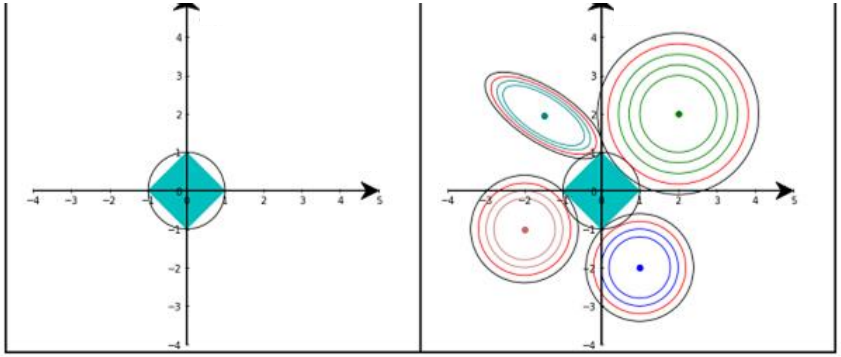
Property of L1 Regularization



For the same value of the bias term, the area occupied by L1 Norm is smaller than by L2 Norm

The point of intersection between the L1 Norm and Gradient Descent Contours **tend to converge near the axes**

Property of L1 Regularization



It means that some of the weights become zero!

L1 regularization leaves only important features and produces zeros for some weights → it can be used for **feature selection**

Regularizations for linear regression: summary

Linear regression: [prediction]

Ridge regression: [prediction] [bias-variance tradeoff]

Lasso regression: [prediction] [bias-variance tradeoff] [feature selection]

There is also Elastic Net: uses both L1 and L2 norms

Fascinating math behind regularization

https://youtu.be/u73PU6Qwl1I

https://www.youtube.com/watch?v=KvtGD37Rm5I

https://www.youtube.com/watch?v=qbvRdrd0yJ8

Intuition behind regularization

Regularization artificially discourages complex or extreme explanations of the world even if they fit our current dataset

The idea is that **complex explanations are unlikely to generalize well to the future** - they may happen to perfectly explain data points from the past, but this may just be because of the properties of the current dataset