# REGRESSION

Lasso and Elastic Net

# How do we learn from data?

1
5
3.6
1
3
-1

Training set

Learning algorithm

Model

h

Desired output **y**

Training phase


# Let’s look at another example


https://soclaimon.wordpress.com/2015/07/24/%E0%B9%82%E0%B8%A1%E0%B9%80%E0%B8%94%E0%B8%A5%E0%B8%99%E0%B9%89%E0%B8%B33%E0%B8%A2%E0%B8%B8%E0%B8%84%E0%B8%A1%E0%B8%B2%E0%B8%A3%E0%B9%8C%E0%B8%84-%E0%B8%9B%E0%B8%B9/

# Predicting amount of rainfall


https://esan108.com/%E0%B8%9E%E0%B8%A3%E0%B8%B0%E0%B9%82%E0%B8%84%E0%B8%81%E0%B8%B4%E0%B8%99%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3-%E0%B8%AB%E0%B8%A1%E0%B8%B2%E0%B8%A2%E0%B8%96%E0%B8%B6%E0%B8%87.html

# Predicting amount of rainfall

| Cloth | Corn | Grass | Water | Beer | Rainfall |
|---|---|---|---|---|---|
| 4 | 6 | 3 | 10 | 0 | **76950** |
| 5 | 1 | 0 | 0 | 7 | **30234** |
| 6 | 0 | 3 | 5 | 7 | **123456** |
| 5 | 0 | 3 | 12 | 0 | **89301** |
| 4 | 3 | 0 | 6 | 7 | **?** |

We assume the input features have some correlation with the amount of rainfall.

Can we create a model that predict the amount of rainfall?

What is the output?

What is the input (features)?

# Predicting the amount of rainfall

- The correlation can be positive or negative

# Predicting the amount of rainfall

| Cloth | Corn | Grass | Water | Beer | Rainfall |
|---|---|---|---|---|---|
| 4 | 6 | 3 | 10 | 0 | **76950** |
| 5 | 1 | 0 | 0 | 7 | **30234** |
| 6 | 0 | 3 | 5 | 7 | **123456** |
| 5 | 0 | 3 | 12 | 0 | **89301** |
| 4 | 3 | 0 | 6 | 7 | **?** |

Can we create a model that predict the amount of rainfall?

- What is the output?
- What is the input (features)?

# Predicting the amount of rainfall

| Cloth | Corn | Grass | Water | Beer | Rainfall |
|---|---|---|---|---|---|
| 4 | 6 | 3 | 10 | 0 | **76950** |
| 5 | 1 | 0 | 0 | 7 | **30234** |
| 6 | 0 | 3 | 5 | 7 | **123456** |
| 5 | 0 | 3 | 12 | 0 | **89301** |
| 4 | 3 | 0 | 6 | 7 | **?** |

- $h_{\boldsymbol{\theta}}(\mathbf{x_1}) = \theta_0 + \theta_1 x_{1,1} + \theta_2 x_{1,2} + \theta_3 x_{1,3} + \theta_4 x_{1,4} + \theta_5 x_{1,5}$

  1 refers to index of the data (the first in the training/test set)
- Where θs are the parameter (weights) of the model
- Xs are values in the table

# (Linear) Regression

- $h_{\boldsymbol{\theta}}(\mathbf{x_1}) = \theta_0 + \theta_1 x_{1,1} + \theta_2 x_{1,2} + \theta_3 x_{1,3} + \theta_4 x_{1,4} + \theta_5 x_{1,5}$
- θs are the parameter (or weights)

Assume $x_0$ is always 1

- We can rewrite

n is dimension of x

$$h_\theta(\mathbf{x}_i) = \Sigma_{j=0}^{n} \theta_j x_{i,j} = \theta^T \mathbf{x}_i$$

h is parameterized by θ

- Notation: vectors are bolded
- Notation: vectors are column vectors

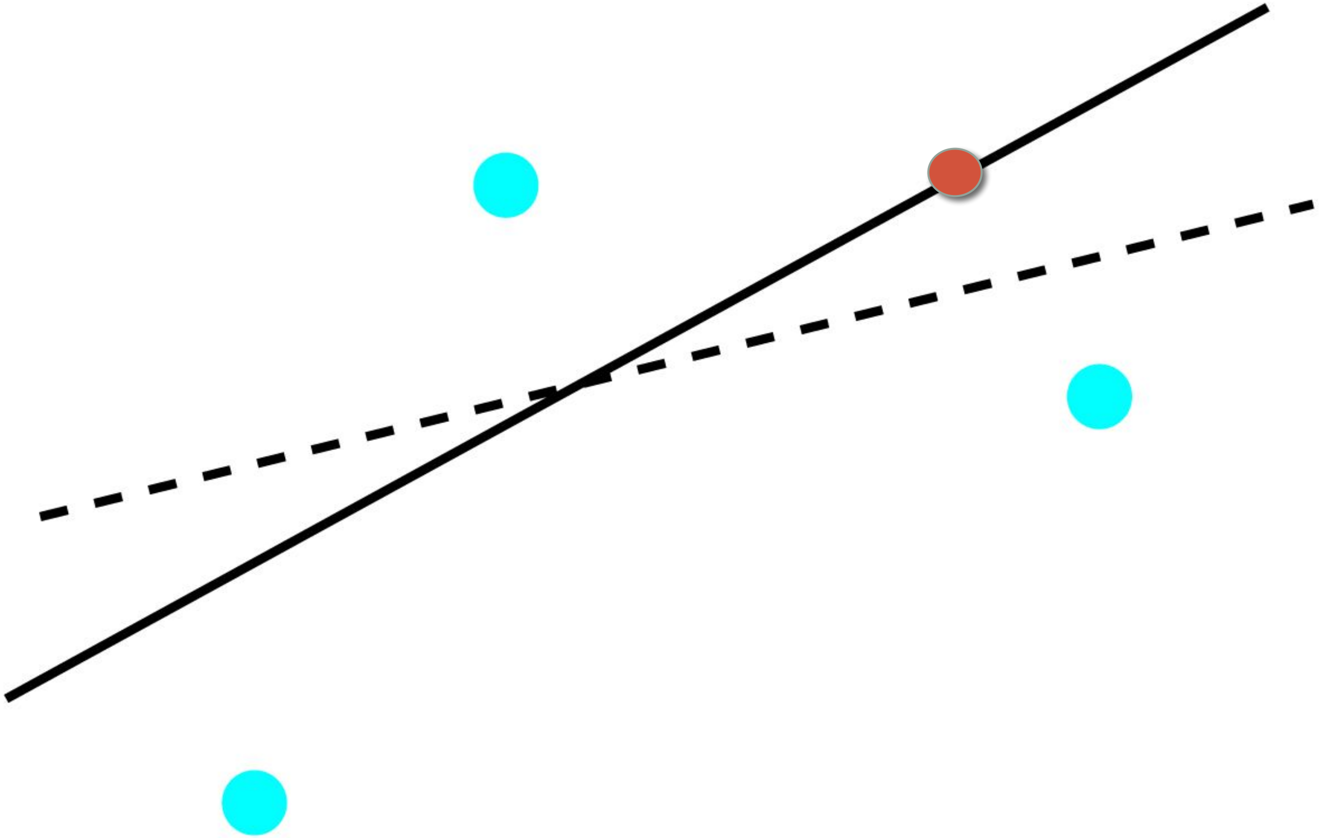
# Picking $\theta$

- Random until you get the best performance?

- How to quantify best performance?

$$h_\theta(\mathbf{x}_i) = \Sigma_{j=0}^{n} \theta_j x_{i,j} = \theta^T \mathbf{x}_i$$

$$h_\theta(\mathbf{x}_i) = \Sigma_{j=0}^{n} \theta_j x_{i,j} = \theta^T \mathbf{x}_i$$



https://chem.libretexts.org/Textbook_Maps/Analytical_Chemistry_Textbook_Maps/Map%3A_Analytical_Chemistry_2.0_(Harvey)/05_Standardizing_Analytical_Methods/5.4%3A_Linear_Regression_and_Calibration_Curves

# Cost function (Loss function)

- Let's use the mean square error (MSE)

m is the number of training examples

$$J(\theta) = \frac{1}{m}\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x_i})^2$$

i here is the index of the training example

Note how **x** is bolded

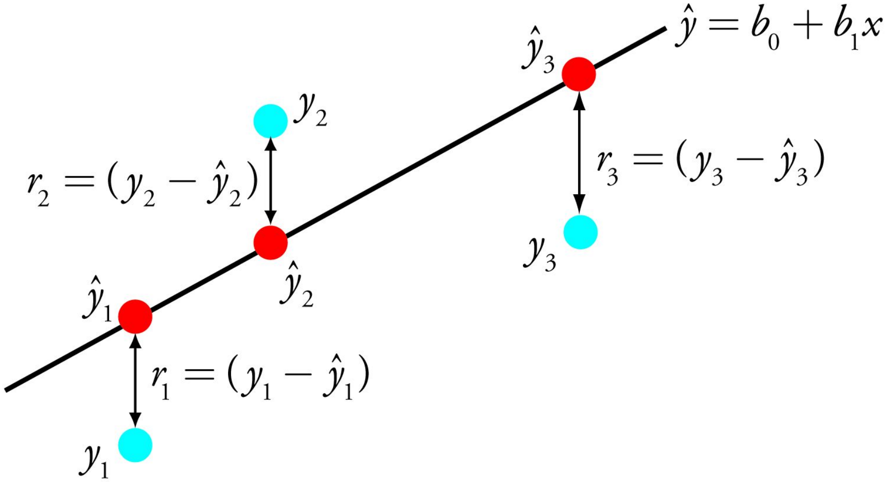
We want to pick **θ** that minimize the loss

# Cost function (Loss function)

- Let's use the mean square error (MSE)

$$J(\theta) = \frac{1}{m}\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x_i})^2$$

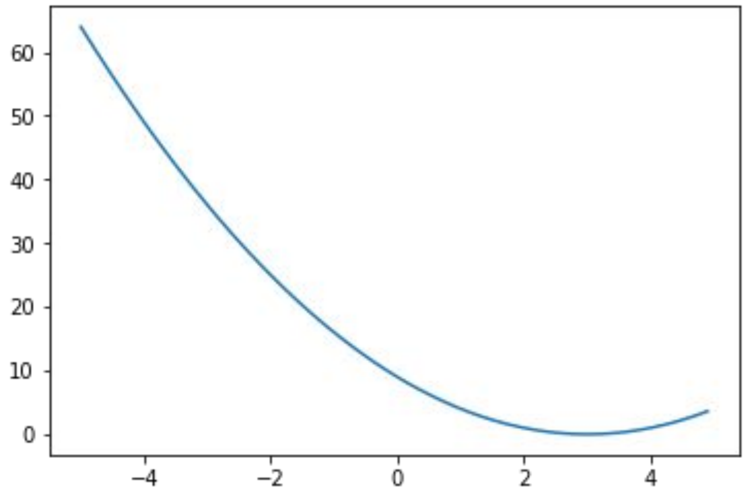
We want to pick **θ** that minimize the loss



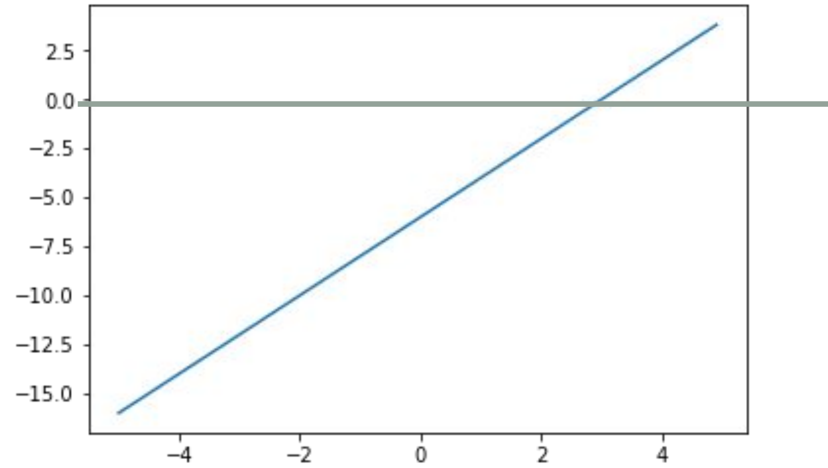
# Minimizing a function

- You have a function
  - $y = (x - a)^2$
- You want to minimize Y with respect to x
  - $dy/dx = 2x - 2a$
  - Take the derivative and set the derivative to 0
    - (And maybe check if it's a minima, maxima or saddle point)
- We can also go with an iterative approach

# Gradient descent



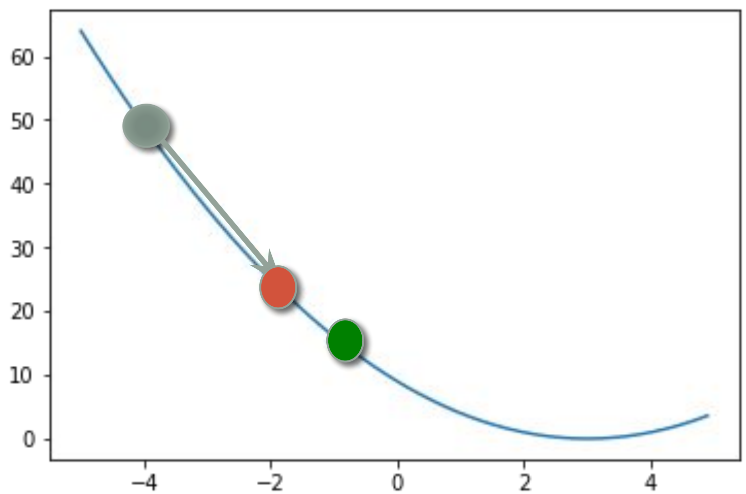

y



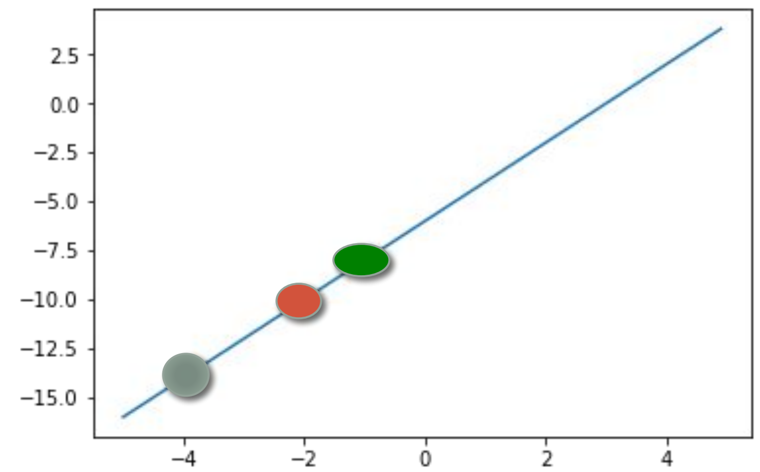

dy/dx

First what does dy/dx means?

# Gradient descent



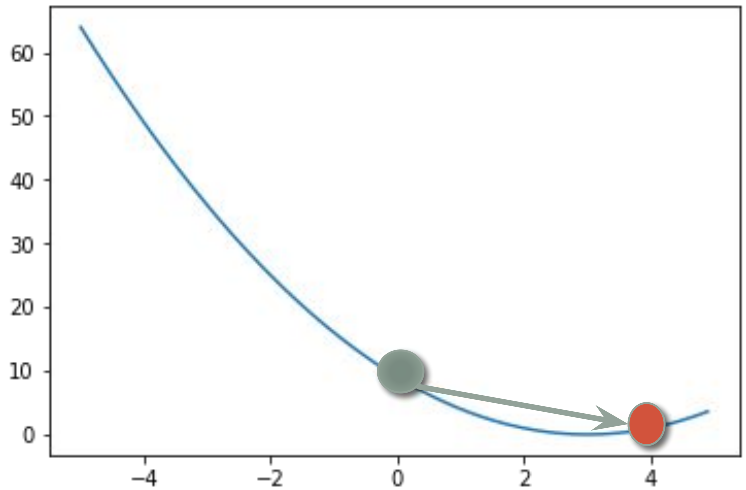

y



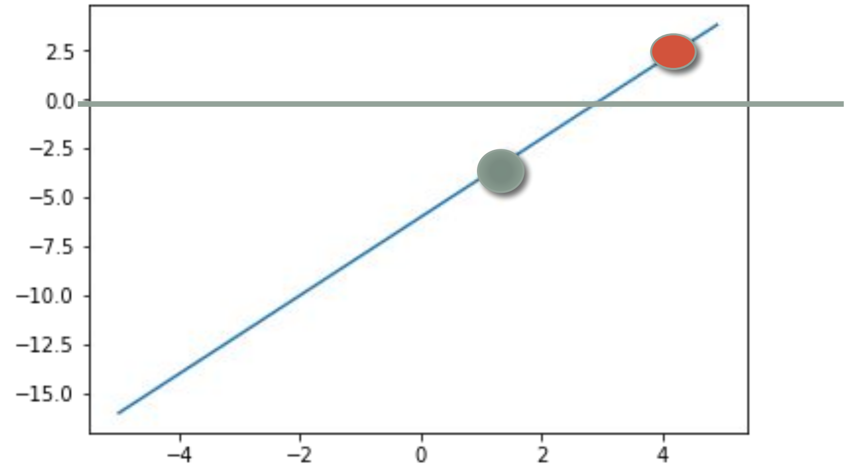

dy/dx

Move along the negative direction of the gradient
The bigger the gradient the bigger step you move

# Gradient descent



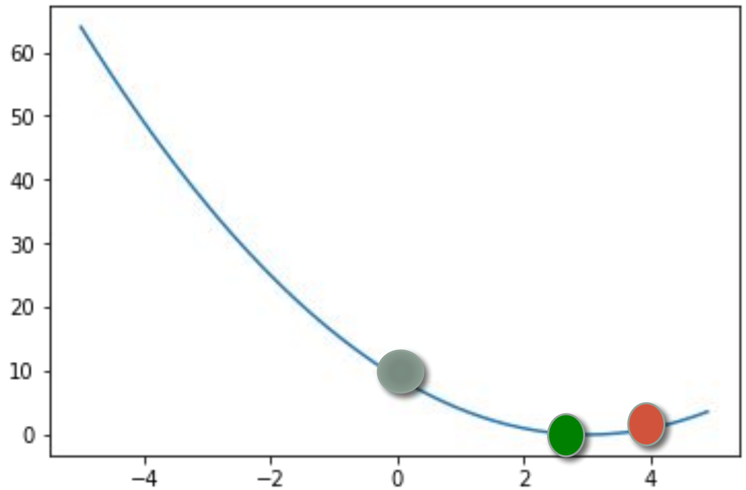

y



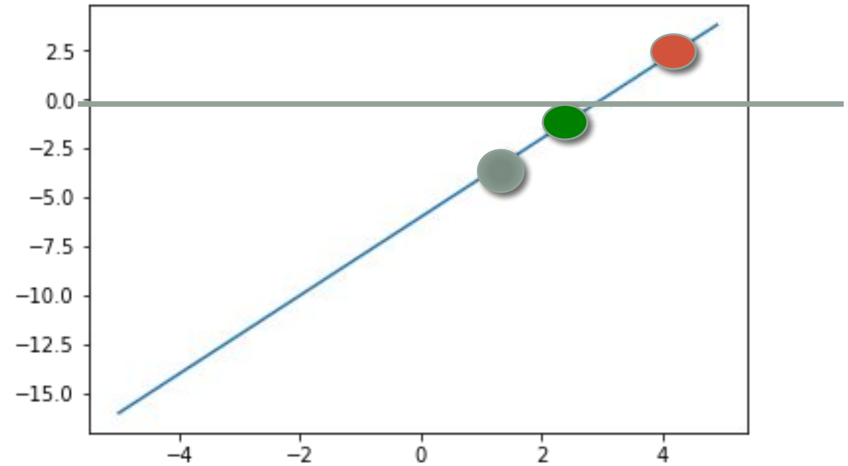

dy/dx

What happens when you overstep?

# Gradient descent




y




dy/dx

If you over step you can move back

# Formal definition

- $y = f(x)$

- Pick a starting point $x_0$

- Moves along $-dy/dx$
- $x_{n+1} = x_n - r * dy/dx$
- Repeat till convergence
- $r$ is the learning rate

  Big $r$ means you might overstep

  Small $r$ and you need to take more steps

# Other loss functions

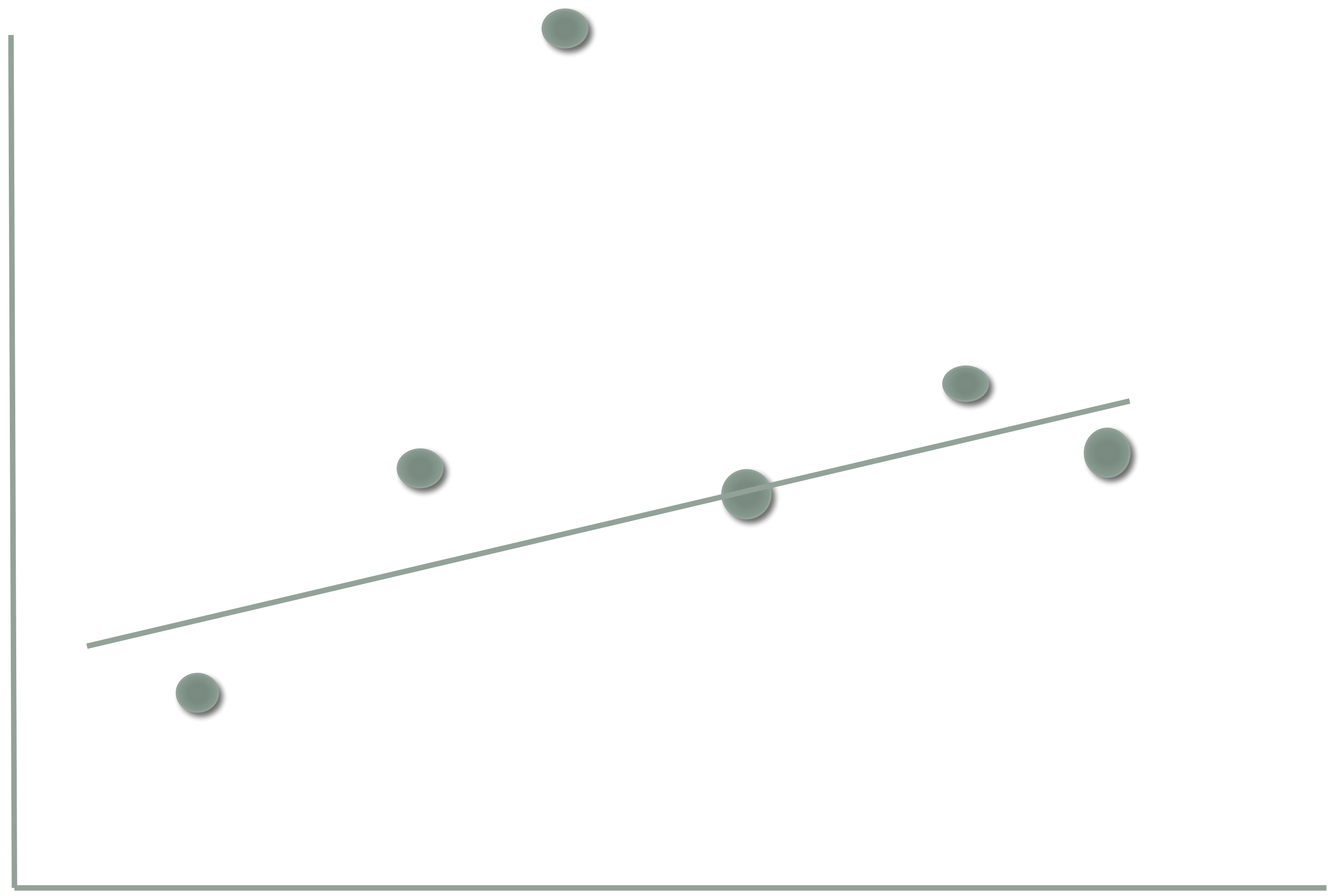
- MSE

$$J(\theta) = \frac{1}{m} \Sigma_{i=1}^{m} (y_i - \theta^T \mathbf{x_i})^2$$

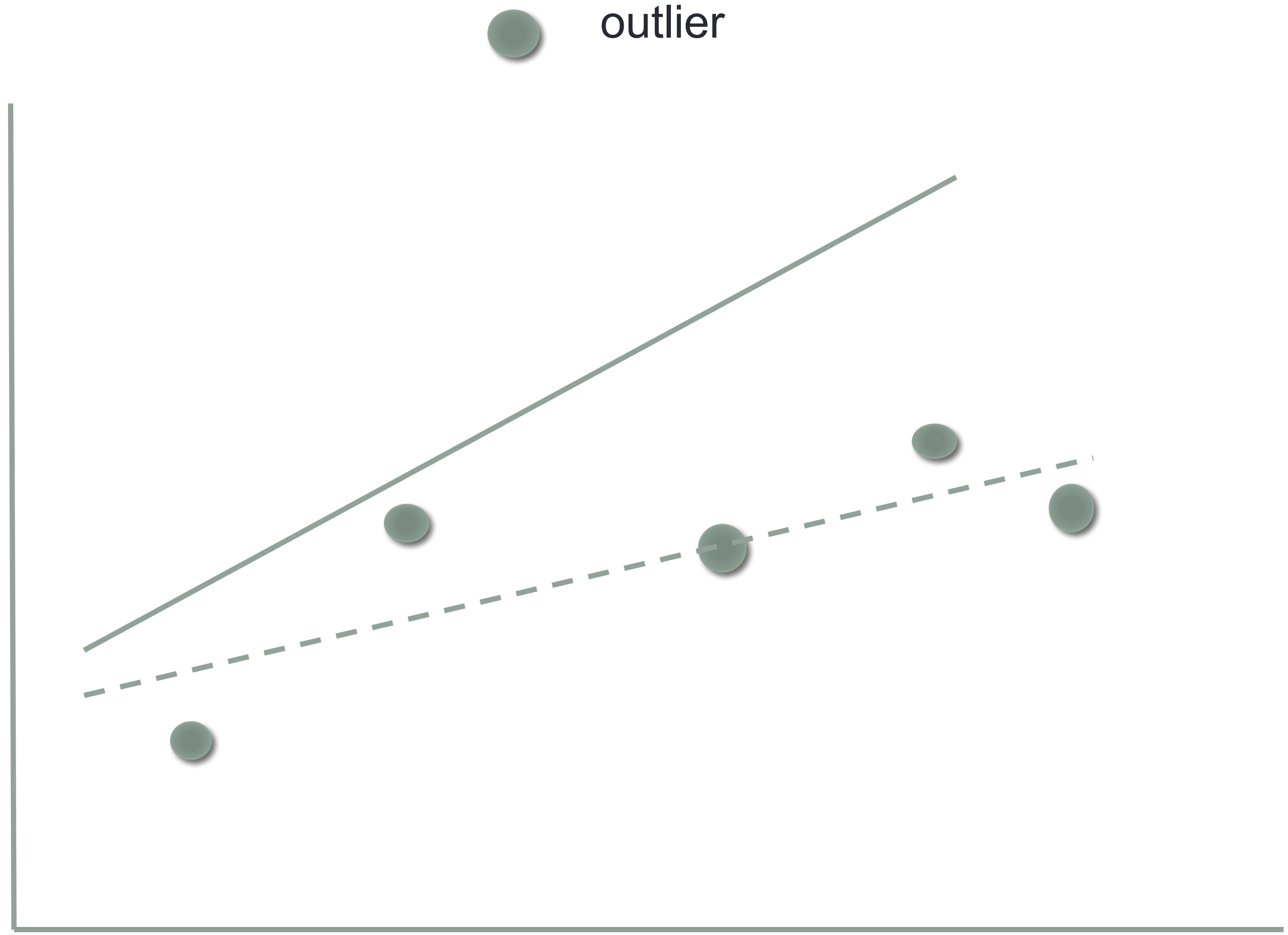
  - Also called L2 loss
- L1 loss

$$\frac{1}{m} \Sigma_{i=1}^{m} |y_i - \theta^T \mathbf{x_i}|$$

# L2 vs L1 loss



# L2 vs L1 loss




Outlier frequently happens in the real world

# Weights

The weights of the model usually implies importance of the feature for prediction.

- Higher value higher importance

  This is important in a linear model.

$$h_\theta(\mathbf{x}_i) = \Sigma_{j=0}^{n} \boxed{\theta_j} x_{i,j}$$

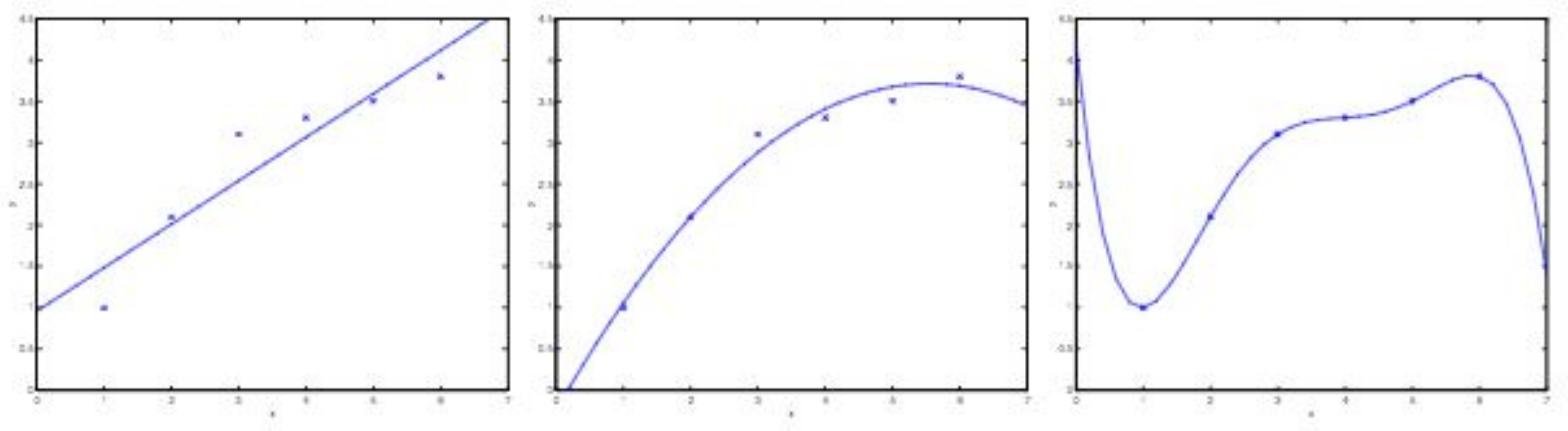
# Regression with non-linear features

- If we add extra features that are non-linear
  - For example $x^2$

| Cloth | Corn | Grass | Water | Beer | Rainfall |
|---|---|---|---|---|---|
| 4 | 6 | 3 | 10 | 0 | **76950** |
| 5 | 1 | 0 | 0 | 7 | **30234** |
| 6 | 0 | 3 | 5 | 7 | **123456** |
| 5 | 0 | 3 | 12 | 0 | **89301** |
| 4 | 3 | 0 | 6 | 7 | **?** |

- $h_{\boldsymbol{\theta}}(\mathbf{x}_1) = \theta_0 + \theta_1 x_{1,1} + \theta_2 x_{1,2} + \theta_3 x_{1,3} + \theta_4 x_{1,4} + \theta_5 x_{1,5} + \theta_6 x_{1,1}^2 + \ldots$
- These can be considered as additional features
- We can now have a line that is non-linear

# Overfitting Underfitting



Adding more non-linear features makes the line more curvy

(Adding more features also means more model parameters)

The curve can go directly to the outliers with enough parameters.

We call this effect overfitting

For the opposite case, having not enough parameters to model the data is called underfitting

# Bias-Variance tradeoff

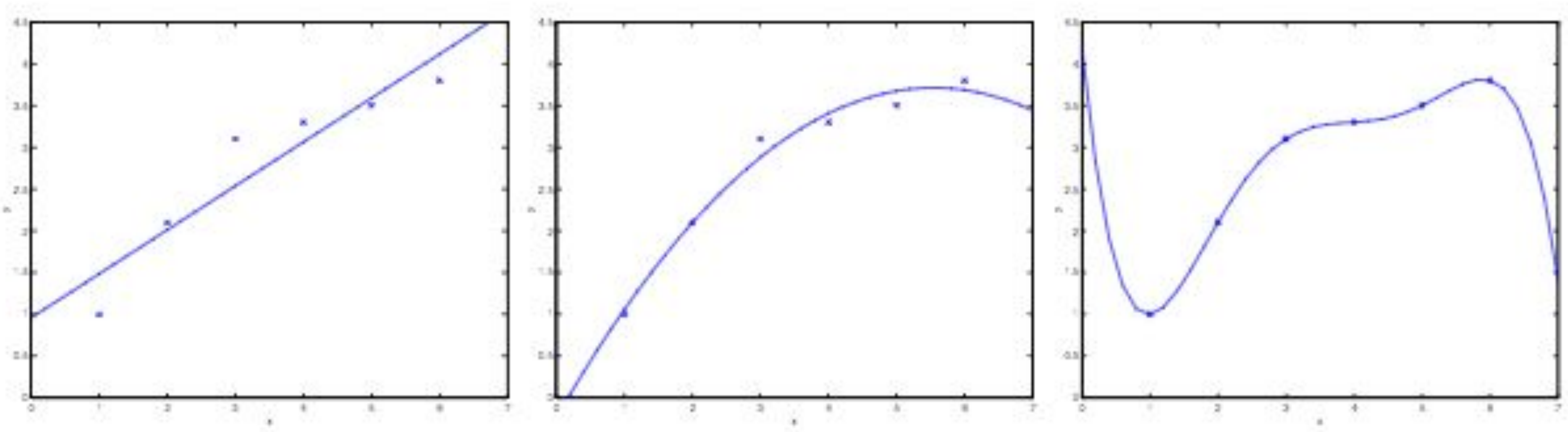
The bias-variance trade off refers to tradeoff between validation error and training error.

Overfitting (High variance)

High error in validation

Underfitting (High bias)

High error in training



# Reducing overfitting by regularization

- What?
  - Regularization is a method to lower the model variance (and thereby increasing the model bias)
- Why?
  - Gives more generalizability (lower variance)
  - Better for lower amounts of data (reduce overfitting)
- How?
  - Introducing regularizing terms in the original loss function

# Famous types of regularization

- L1 regularization: Regularizing term is a sum

Regularization strength

$$J(\theta) = \frac{1}{m}\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x}_i)^2 + \alpha \Sigma_d |\theta_d|$$

Training data index

feature index

- L2 regularization: Regularizing term is a sum of squares

$$J(\theta) = \frac{1}{m}\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x}_i)^2 + \alpha \Sigma_d |\theta_d|^2$$

# How does it work?

Let’s say we want

$10 = a\ x_1 + b\ x_2$ where $x_1 = 2$ and $x_2 = 3$

Find the weights a and b

$$J(\theta) = \frac{1}{m}\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x}_i)^2 + \alpha\Sigma_d|\theta_d| \qquad J(\theta) = \frac{1}{m}\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x}_i)^2 + \alpha\Sigma_d|\theta_d|^2$$

| a | b | Loss with no regularization | L1 reg | L2 reg |
|---|---|---|---|---|
| 3 | 2 | $(10-12)^2 = 4$ | 4+5 = 9 | 4+13 = 17 |
| 3 | 1.33 | $(10-10)^2 = 0$ | 0+4.33 = 4.33 | 0+10.77 = 10.77 |
| 5 | 0 | $(10-10)^2 = 0$ | 0+5 = 5 | 0+25 = 25 |
| 0 | 3.33 | $(10-10)^2 = 0$ | 3.33 | 11.09 |
| 2 | 2 | $(10-10)^2 = 0$ | 4 | 8 |

L1 regularization prefers picking a few features -> automatically selects good features
L2 regularization prefers spreading out the weights -> more robust to noise

# Lasso and Elastic Net

Lasso = linear regression with L1 regularization

Elastic Net = linear regression with L1 and L2 regularization

| L2 regularization | L1 regularization |
|---|---|
| Computational efficient due to having analytical solutions | Computational inefficient on non-sparse cases |
| Non-sparse outputs | Sparse outputs |
| No feature selection | Built-in feature selection |

http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/

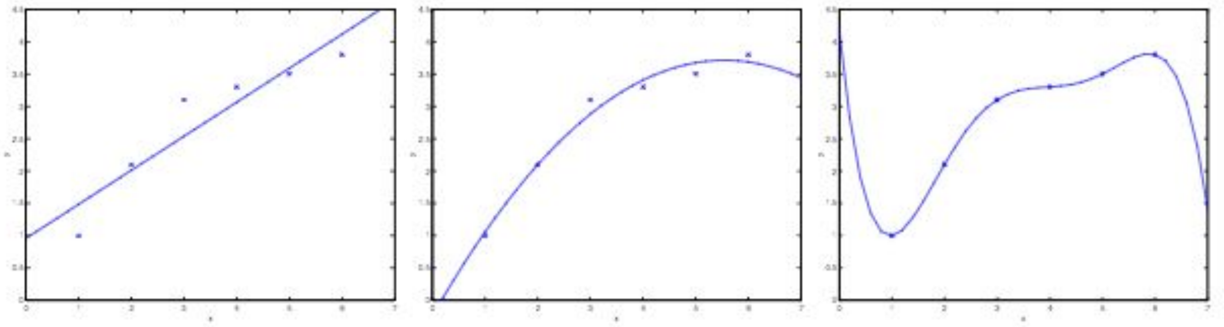
# Summary

Linear regression

Loss function and minimizing the loss function

Overfitting and underfitting

Regularization



$$J(\theta) = \frac{1}{m}\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x}_i)^2 + \alpha \Sigma_d |\theta_d|^2$$

# Lab

Housing price prediction

Regression model

- + non-linear features
  - overfitting vs underfitting

Feature selection

- Using correlation coefficients (univariate analysis)
- Using model weights (multivariate analysis)
- Lasso