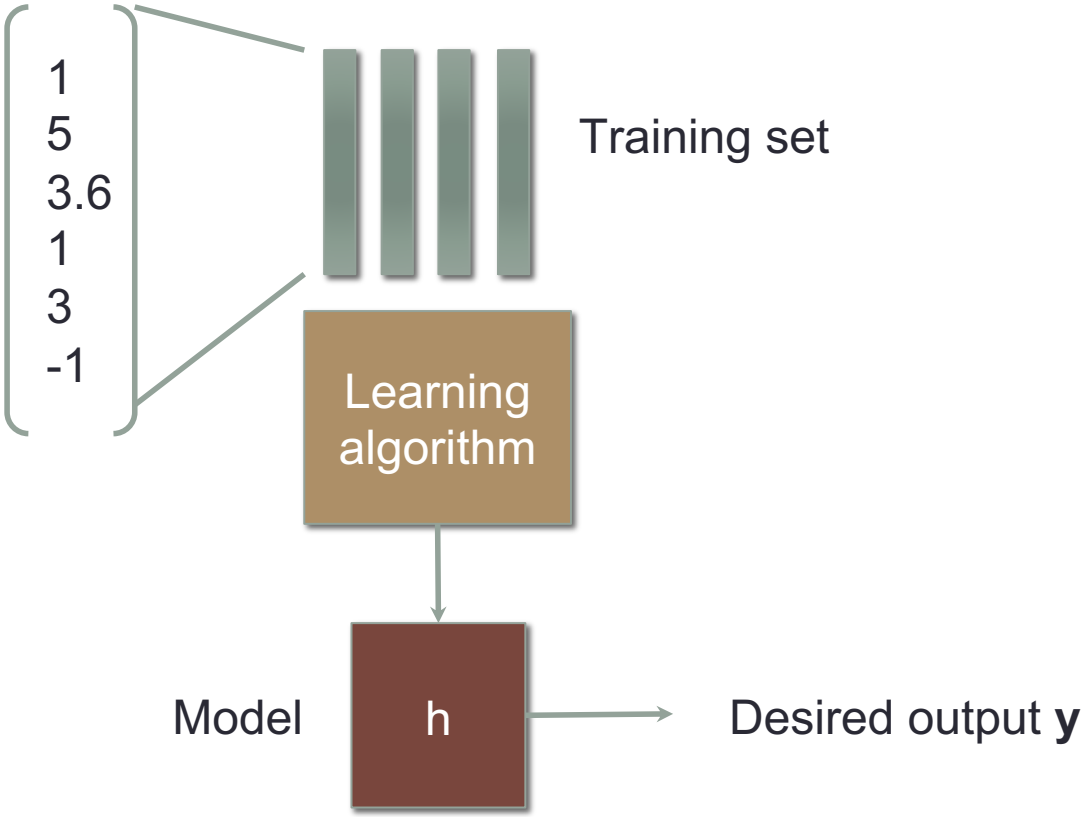
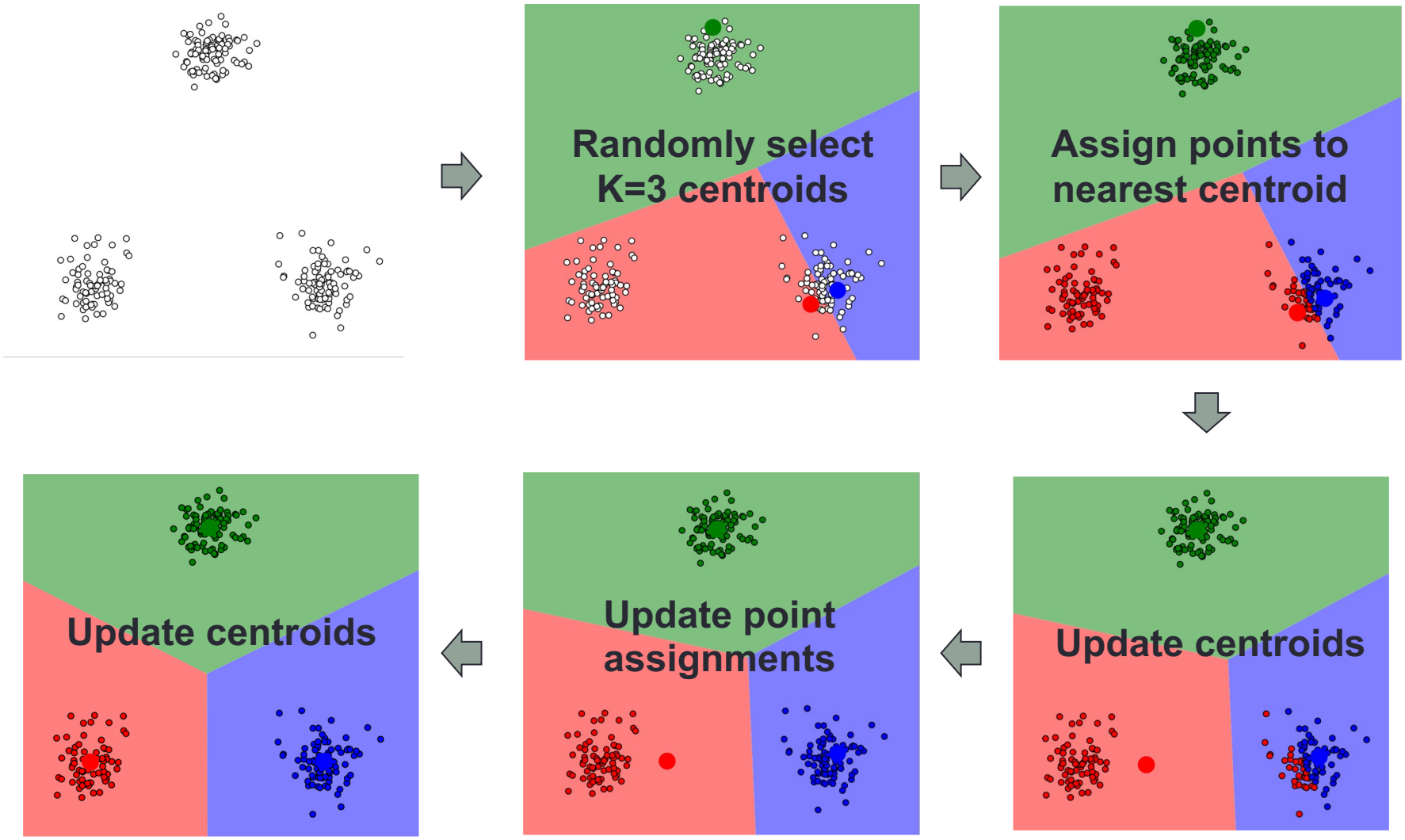
REGRESSION

with some MLE

How do we learn from data?



An Illustration Of K-Mean Clustering



From https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Types of machine learning

1. Supervised learning

   Learn a model F from pairs of (**x**,y)

2. Unsupervised learning

   Discover the hidden structure in unlabeled data **x** (no y)

3. Reinforcement learning

   Train an agent to take appropriate actions in an environment by maximizing rewards

REGRESSION

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 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	**?**

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 of the model
- Xs are values in the table

(Linear) Regression

- $h_\theta(\mathbf{x}_1) = \theta_0 + \theta_1 x_{1,1} + \theta_2 x_{1,2} + \theta_3 x_{1,3} + \theta_4 x_{1,4} + \theta_5 x_{1,5}$
- θs are the parameter (or weights)
- We can rewrite

Assume x_0 is always 1

n is dimension of x

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

h is parameterized by θ

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

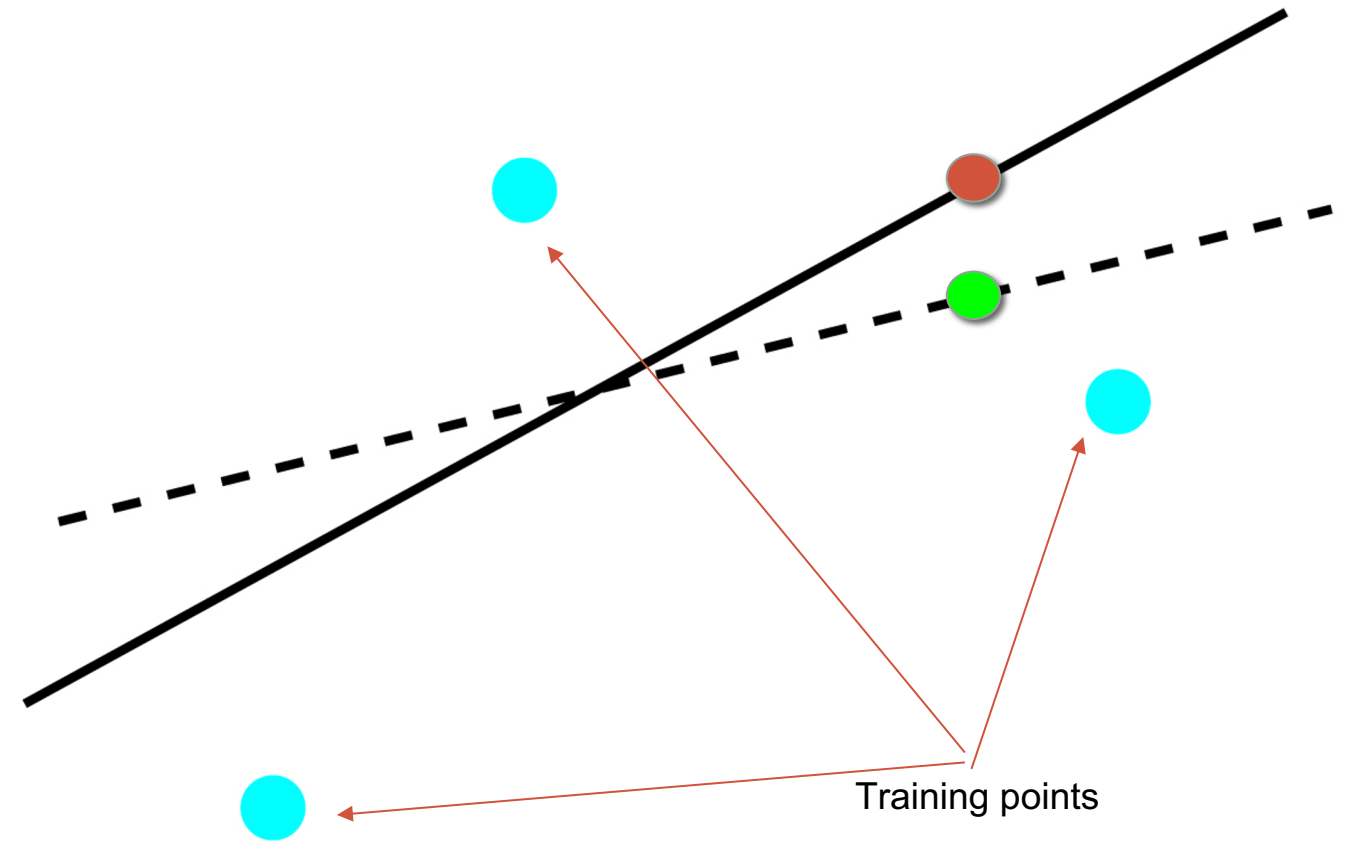
Picking θ

- How to quantify best performance?

- Random until you get the 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

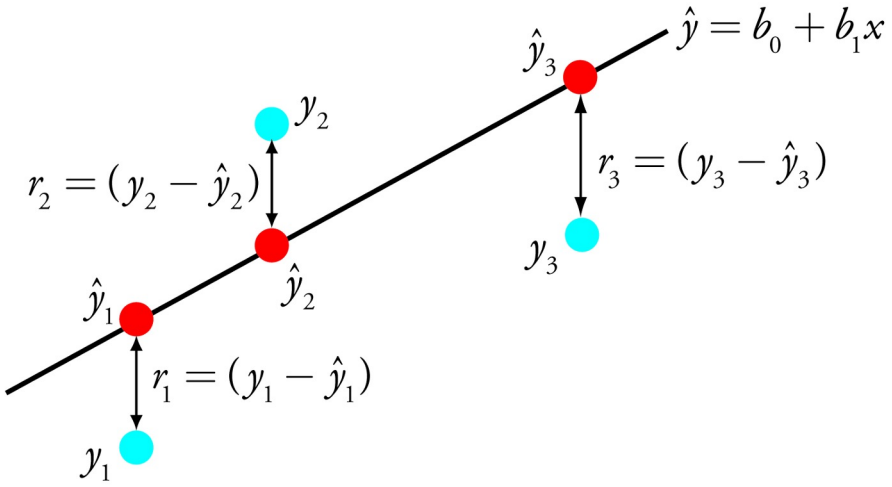
We want to pick **θ** that minimize the loss

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



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

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

Picking θ

- How to quantify best performance?

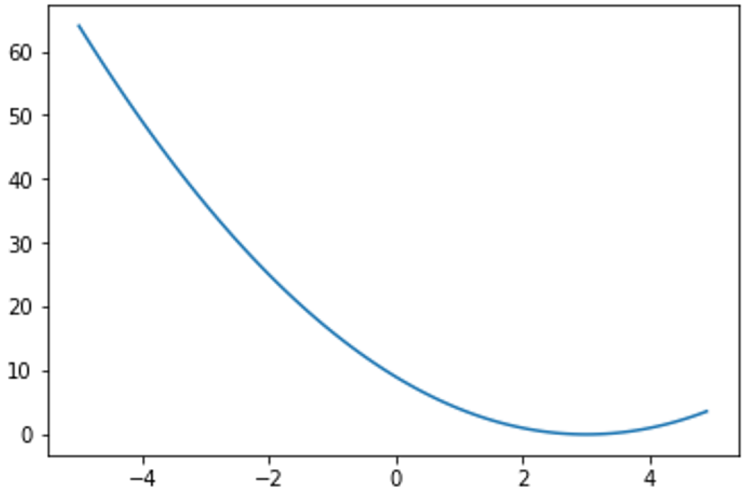
$$\frac{m}{2}J(\theta) = \frac{1}{2}\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x_i})^2$$

- Random until you get the best performance?
  - Can we do better than random chance?

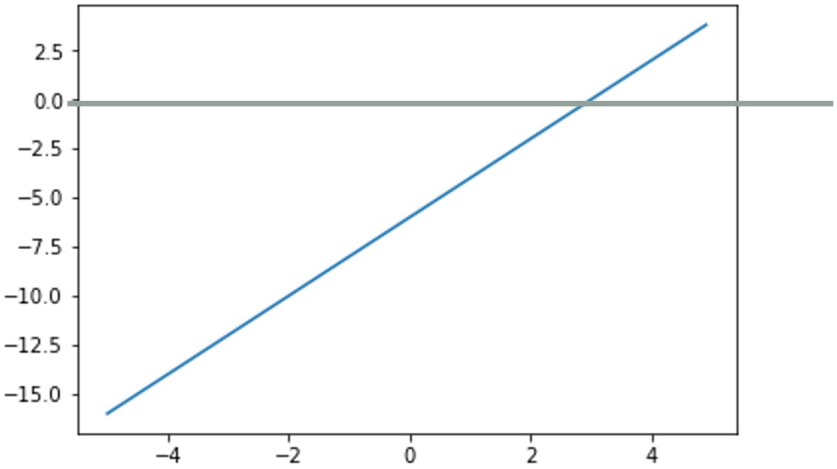
Minimizing a function

- 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



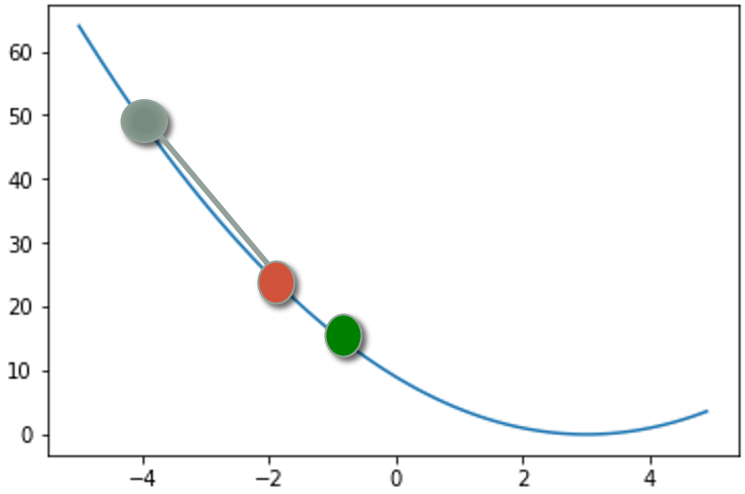
y



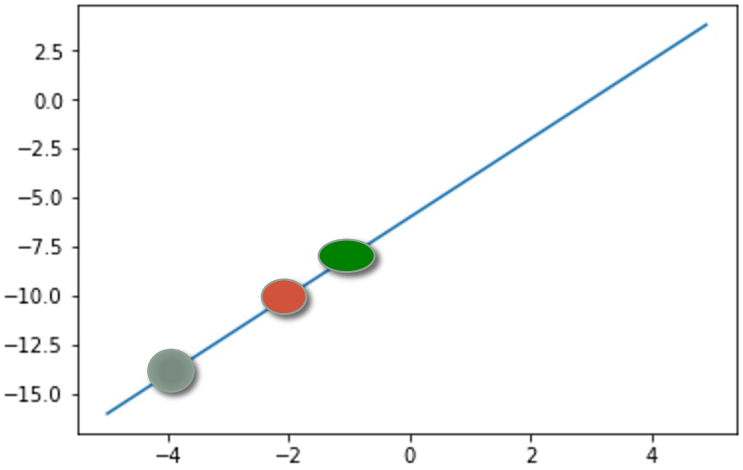
dy/dx

First what does dy/dx means?

Gradient descent



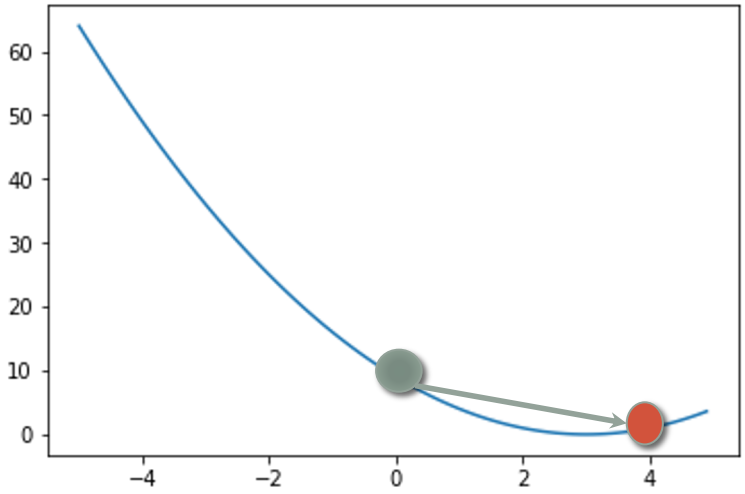
y



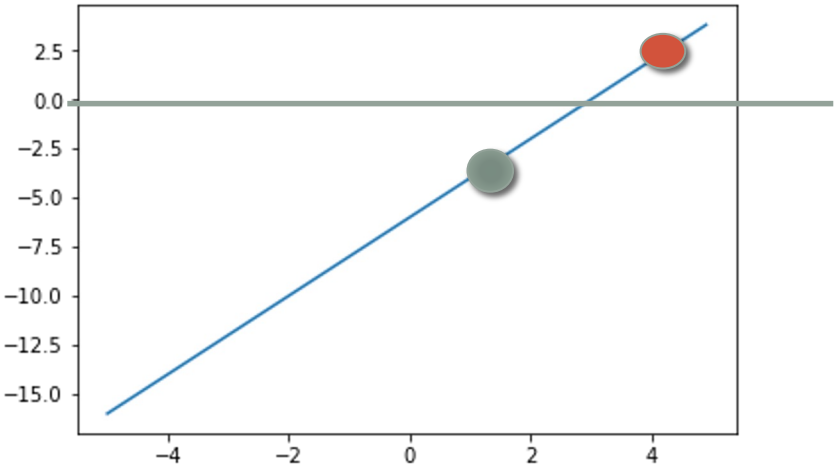
dy/dx

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

Gradient descent



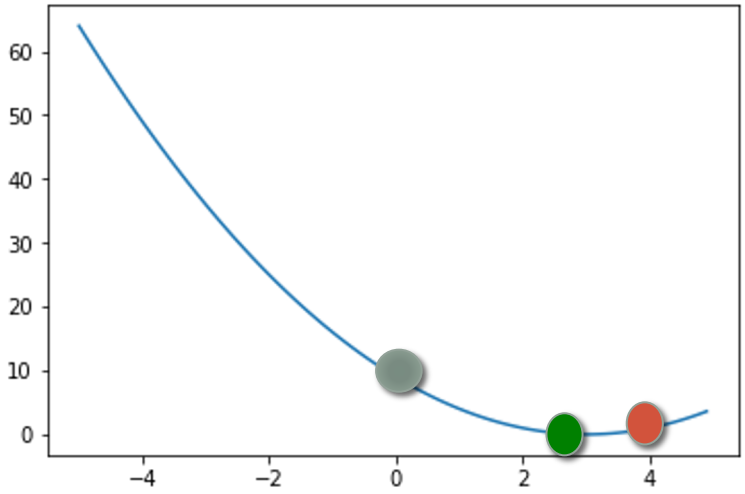
y



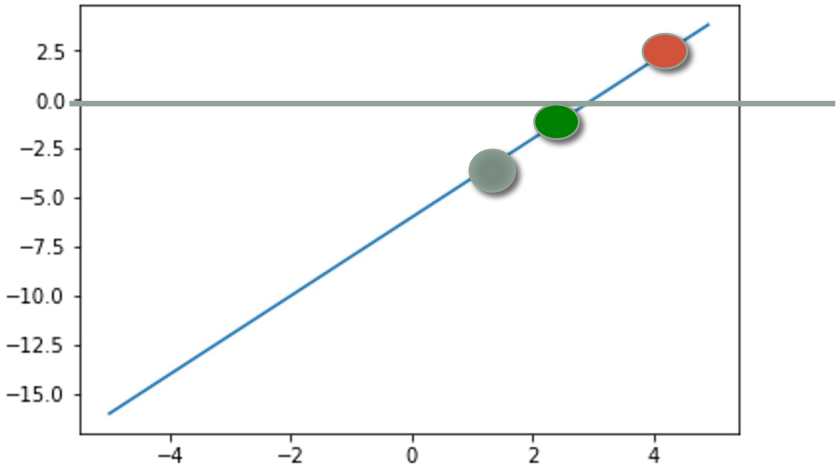
dy/dx

What happens when you overstep?

Gradient descent



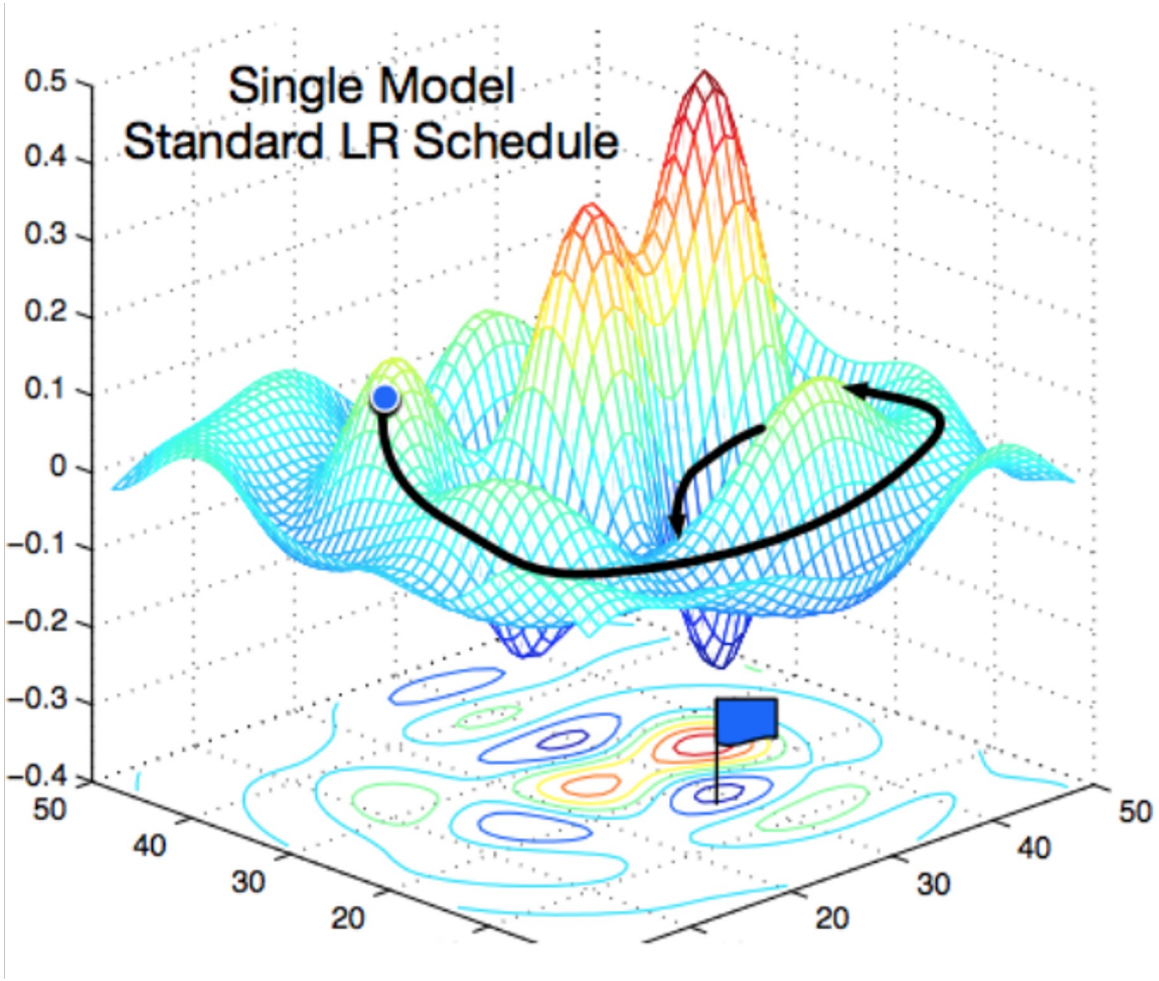
y



dy/dx

If you overstep you can move back

Gradient descent in 3d



https://openreview.net/pdf?id=BJYwwY9ll

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

Picking $\boldsymbol{\theta}$

- How to quantify best performance?

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

- Random until you get the best performance?
  - Can we do better than random chance?
    - Gradient descent (a slightly better search algorithm for $\boldsymbol{\theta}$)

LMS regression with gradient descent

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

$$\frac{\partial J}{\partial \theta_j} = -\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x}_i)x_i^{(j)}$$

LMS regression with gradient descent

$$\frac{\partial J}{\partial \theta_j} = -\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x}_i)x_i^{(j)}$$

$$\theta_j \Leftarrow \theta_j + r\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x}_i)x_i^{(j)}$$

Interpretation?

Batch updates vs mini-batch

$$\theta_j \Leftarrow \theta_j + r\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x}_i)x_i^{(j)}$$

- Batch updates (considering the whole training data) estimate the Loss function precisely
  - Can takes a long time if m is large
- Updates with a subset of m
  - We now have an estimate of the loss function
    - This can lead to a wrong direction, but we get faster updates
  - Called Stochastic Gradient Descent (SGD) or incremental gradient descent

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)

LMS regression with matrix derivatives

- First let's definite what's a derivative of a matrix
- For a function $f : \mathbb{R}^{n \times d} \mapsto \mathbb{R}$

- The derivative wrt to A is

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1d}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{n1}} & \cdots & \frac{\partial f}{\partial A_{nd}} \end{bmatrix}$$

https://cs229.stanford.edu/main_notes.pdf

Example

- Suppose

$$f : \mathbb{R}^{n \times d} \mapsto \mathbb{R}$$

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \qquad \nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1d}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{n1}} & \cdots & \frac{\partial f}{\partial A_{nd}} \end{bmatrix}$$

$$f(A) = \frac{3}{2} A_{11} + 5A_{12}^2 + A_{21} A_{22}$$

$$\nabla_A f(A) = \begin{bmatrix} \frac{3}{2} & 10A_{12} \\ A_{22} & A_{21} \end{bmatrix}$$

Trace of a matrix

- trA is the sum of the diagonals of matrix A (A must be a square matrix)

$$trA = \Sigma_i^N A_{ii}$$

- Trace of a real number? (1x1 matrix)

Trace properties

- tr (a) = a
- trA = trA^T
- tr(A+B) = trA+trB
- tr(aA) = atr(A)

$$\nabla_A trAB = B^T$$
$$\nabla_{A^T} f(A) = (\nabla_A f(A))^T$$
$$\nabla_A trABA^TC = CAB + C^TAB^T$$
$$\nabla_{A^T} trABA^TC = B^TA^TC^T + BA^TC$$

LMS regression with matrix derivatives

$$X = \begin{bmatrix} - & x_1^T & - \\ - & x_2^T & - \\ - & x_m^T & - \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ y_m \end{bmatrix}$$

$$X\theta - y = \begin{bmatrix} x_1^T\theta \\ | \\ x_m^T\theta \end{bmatrix} - \begin{bmatrix} y_1 \\ | \\ y_m \end{bmatrix}$$

LMS regression with matrix derivatives

$$X\theta - y = \begin{bmatrix} x_1^T\theta \\ | \\ x_m^T\theta \end{bmatrix} - \begin{bmatrix} y_1 \\ | \\ y_m \end{bmatrix}$$

$$\frac{1}{2}(X\theta - y)^T(X\theta - y) = \frac{1}{2}\sum_{i=1}^{m}(y_i - \theta^T x)^2$$

We want to minimize this term wrt to **θ**

LMS regression with matrix derivatives

$$\theta = (X^T X)^{-1} X^T y$$

LMS regression with matrix derivatives

$$\theta = (X^T X)^{-1} X^T y$$

Trace properties

1. tr (a) = a
2. trA = trA^T
3. tr(A+B) = trA+trB
4. tr(aA) = atr(A)

5. $$\nabla_A trAB = B^T$$
6. $$\nabla_{A^T} f(A) = (\nabla_A f(A))^T$$
7. $$\nabla_A trABA^TC = CAB + C^TAB^T$$
8. $$\nabla_{A^T} trABA^TC = B^TA^TC^T + BA^TC$$

LMS regression notes

$$\theta = (X^T X)^{-1} X^T y$$

Other loss functions

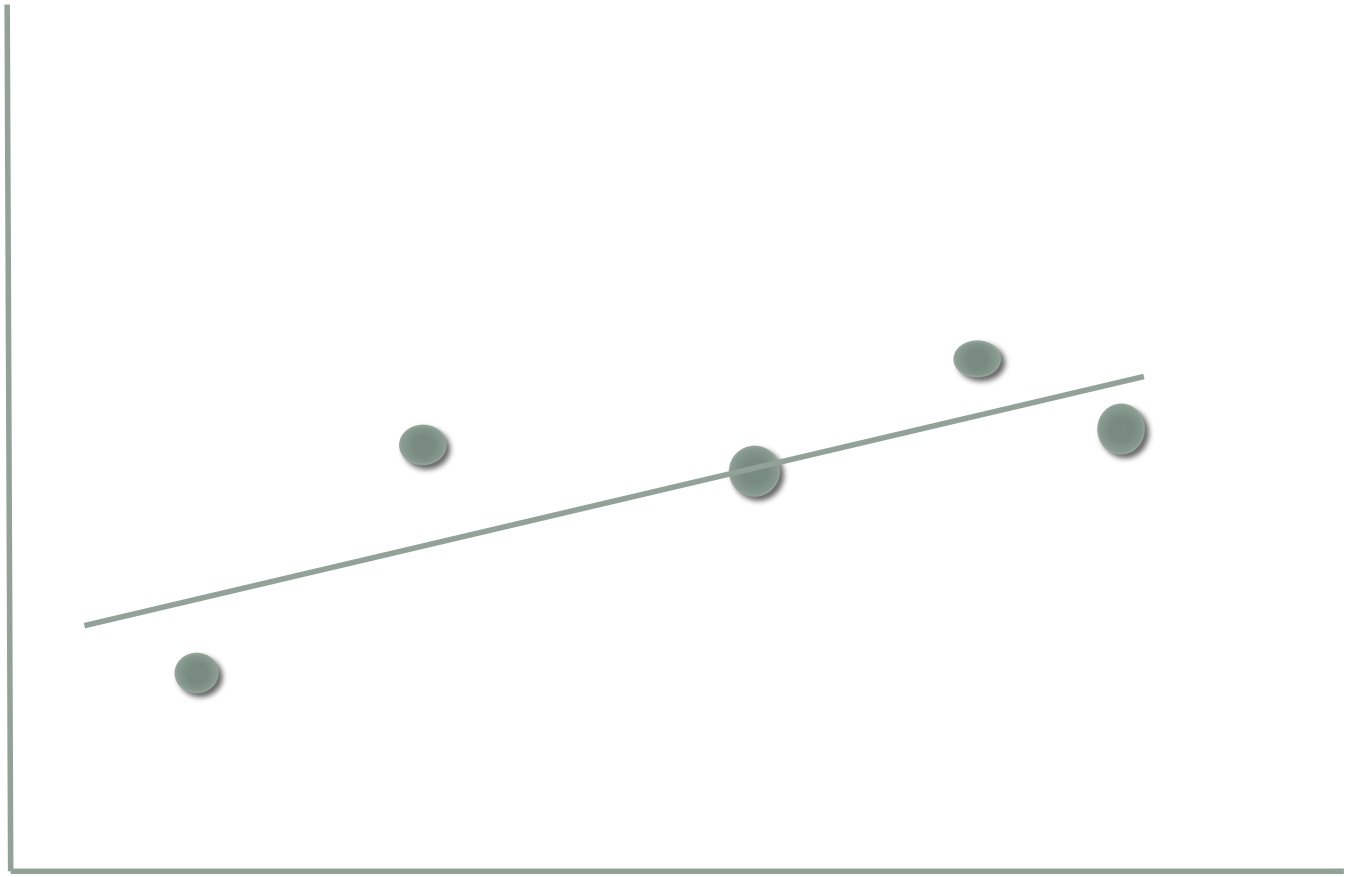
- MSE

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

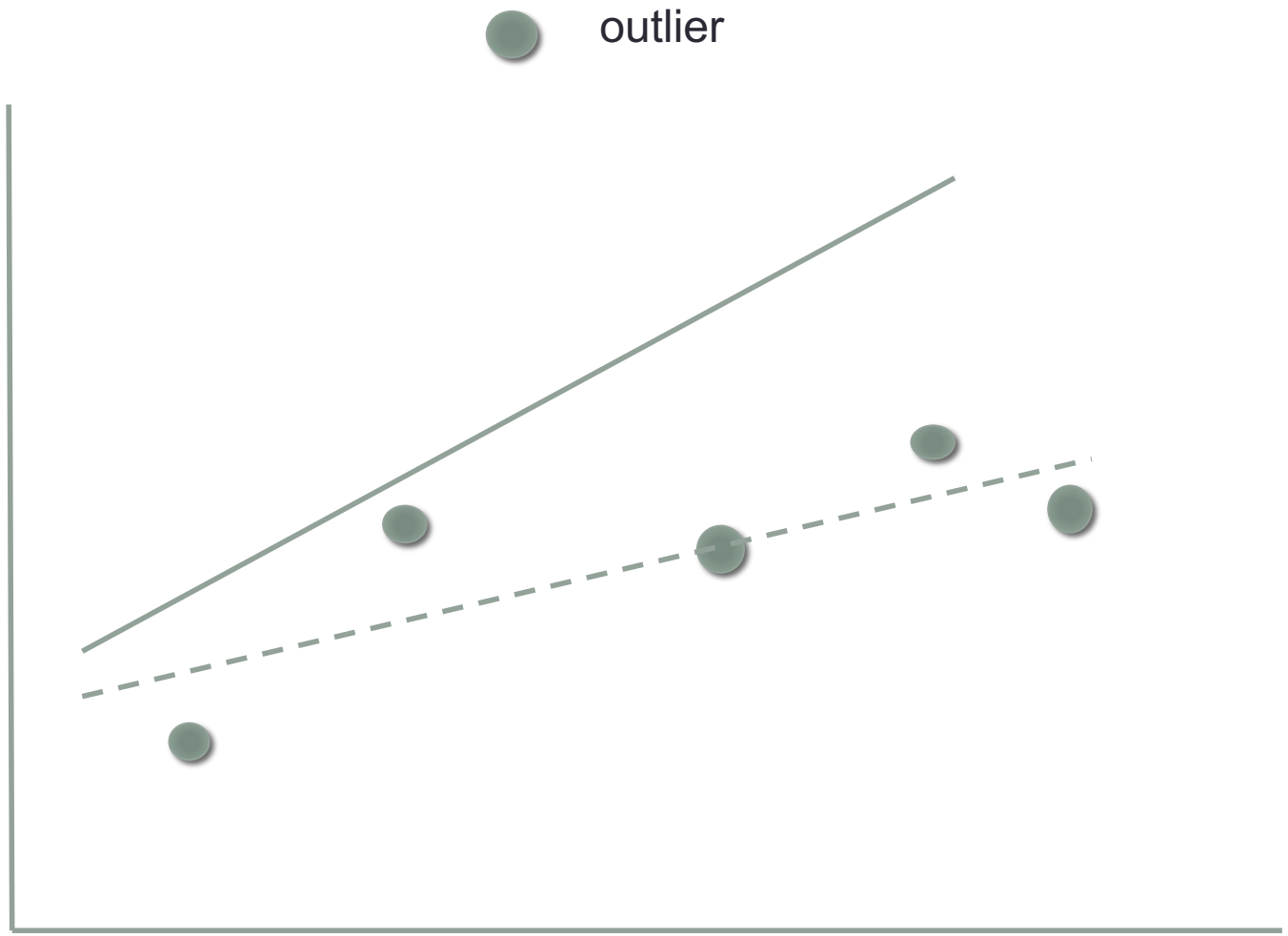
  - Also called L2 loss
- L1 loss

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

L2 vs L1 loss



L2 vs L1 loss



Outlier frequently happens in the real world

Norms (p-norm or Lp-norm)

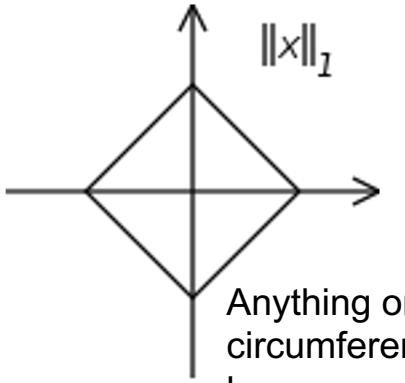
- For any real number p > 1

$$||\mathbf{x}||_p = (|x_1|^p + |x_2|^p + \ldots + |x_n|^p)^{\frac{1}{p}}$$

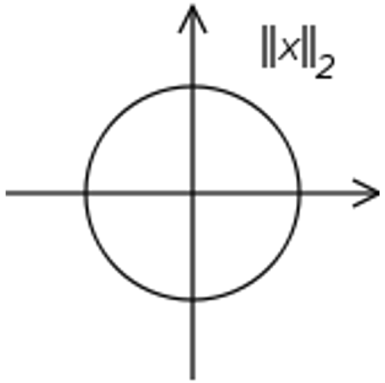
- For p = ∞

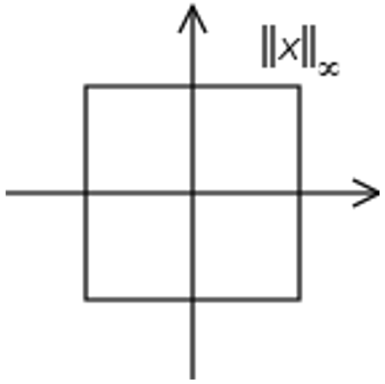
$$\|x\|_\infty = \max\{|x_1|, |x_2|, \ldots, |x_n|\}$$

- We'll see more of p-norms when we get to neural networks



Anything on the circumference has norm of 1





https://en.wikipedia.org/wiki/Lp_space

Predicting floods

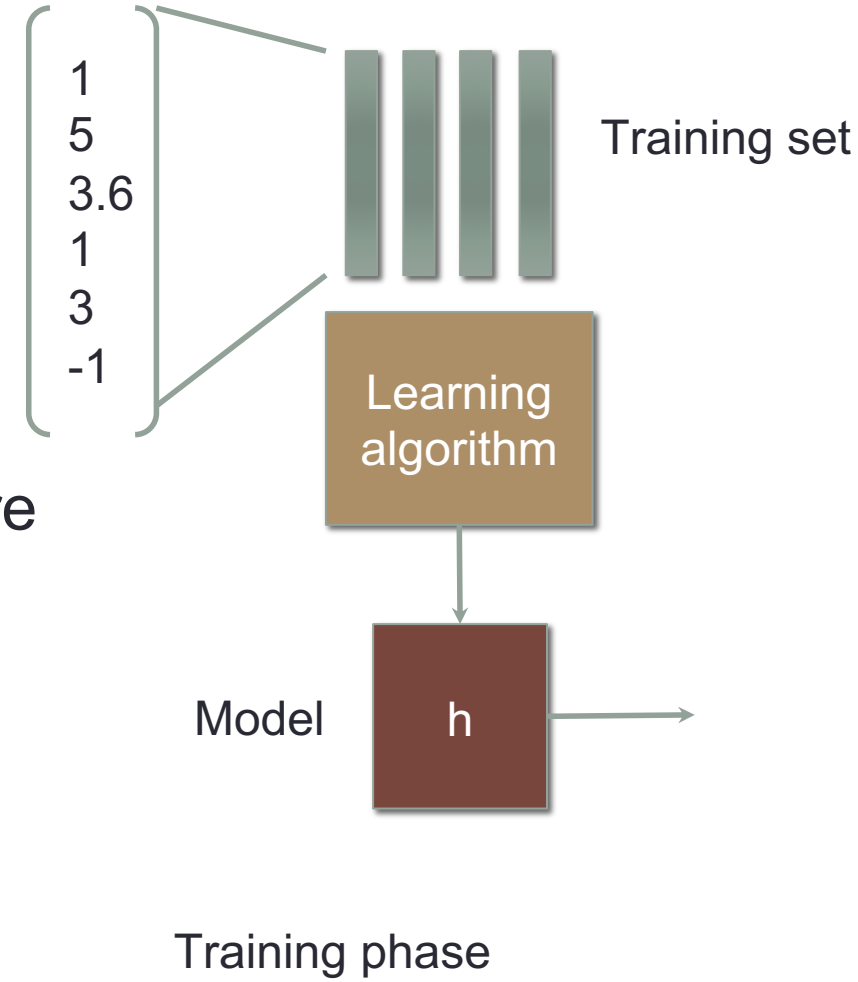
Cloth	Corn	Grass	Water	Beer	Flood?
4	6	3	10	0	**yes**
5	1	0	0	7	**yes**
6	0	3	5	7	**no**
5	0	3	12	0	**yes**
4	3	0	6	7	**?**

So far we talk about predicting an amount what if we want to do classification

Let's start with a binary choice. Flood or no flood

Flood or no flood

- What would be the output?
- y = 0 if not flooded
- y = 1 if flooded
- Anything in between is a score for how likely it is to flood



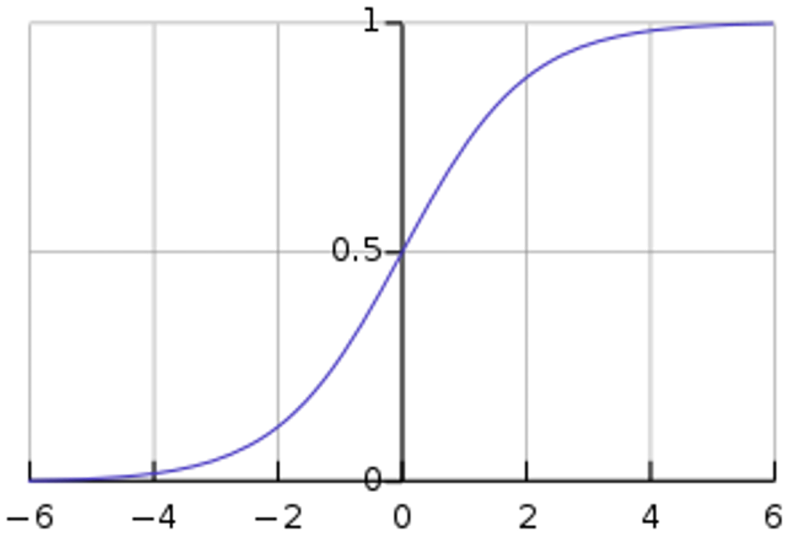
Can we use regression?

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

- But
- What does it mean when h is higher than 1?
- Can h be negative? What does it mean to have a negative flood value?

Logistic function

- Let's force h to be between 0 and 1 somehow
- Introducing the logistic function (sigmoid function)



$$f(x) = \frac{1}{1+e^{-x}}$$
$$= \frac{e^{x}}{1+e^{x}}$$

https://en.wikipedia.org/wiki/Logistic_function

Logistic Regression

- Pass $\theta^T \mathbf{x}$ through the logistic function

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Loss function?

- MSE error no longer a good candidate

Distribution parameter estimation

- P(head) = θ, θ = #heads/#tosses
- HHTTH

- L(θ) = P(X; θ) = P(HHTTH; θ)
- Maximum Likelihood Estimate (MLE)
  - Likelihood - Probability of encountering the data X given the parameters θ

Probabilistic Interpretation of linear regression

- Real world data is our model plus some error term
  - Noise in the data
  - Something that we do not model (features we are missing)

- Let's assume the error is normally distributed with mean zero and variance σ^2
  - Why Gaussian?
  - Why saying mean is zero is a valid assumption?

$$y_i = \theta^T \mathbf{x}_i + \epsilon_i$$

Probabilistic view of Linear regression

From our assumption we know that

$$y_i = \theta^T \mathbf{x}_i + \epsilon_i$$

error

$$p(y_i|\mathbf{x}_i;\theta) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2})$$

Error term is normally distributed with mean 0 and variance σ^2

We want to find θ

Maximize Likelihood of seeing x and y in training

or in other words maximize the probability of answering y from x in the training data $p(y_i|\mathbf{x}_i;\theta)$

Maximizing Likelihood

What is the assumption here?
Is it accurate?

- Max $L(\theta) = \Pi_{i=1}^{m} p(y_i | \mathbf{x}_i; \theta)$
  We use the log likelihood instead log(L(θ)) = l(θ)

From our previous lecture

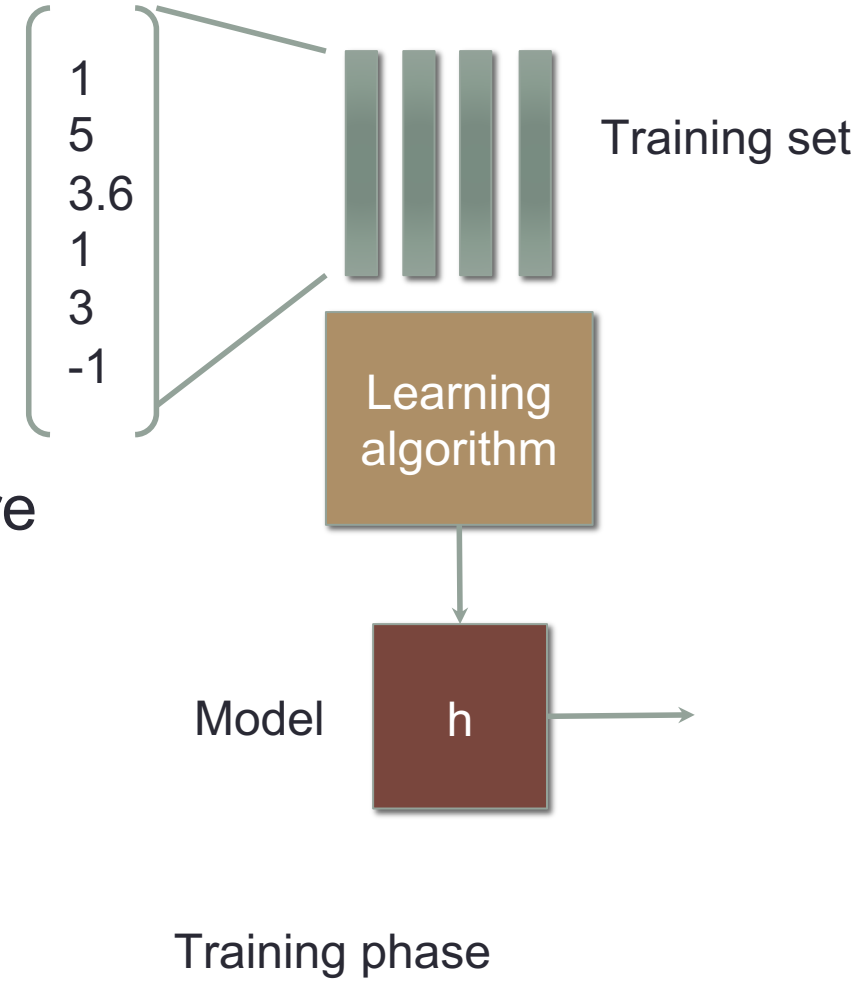
Min $J(\theta) = \frac{1}{m} \Sigma_{i=1}^{m} (y_i - \theta^T \mathbf{x_i})^2$

Mean square error solution and MLE solution

- Turns out MLE and MSE gets to the same solution
  - This justifies our choice of MSE as the Loss for linear regression
  - This does not mean MSE is the best Loss for regression, but you can at least justify it under this probabilistic reasoning and assumptions
- Note how our choice of variance σ^2 falls out of the maximization, so this derivation is true regardless of which size of the variance.
- Note that MLE derivation assumes that the error is normally distributed! This is a key assumption for linear regression.
  - Error is normally distributed is not that same as y is normally distributed.

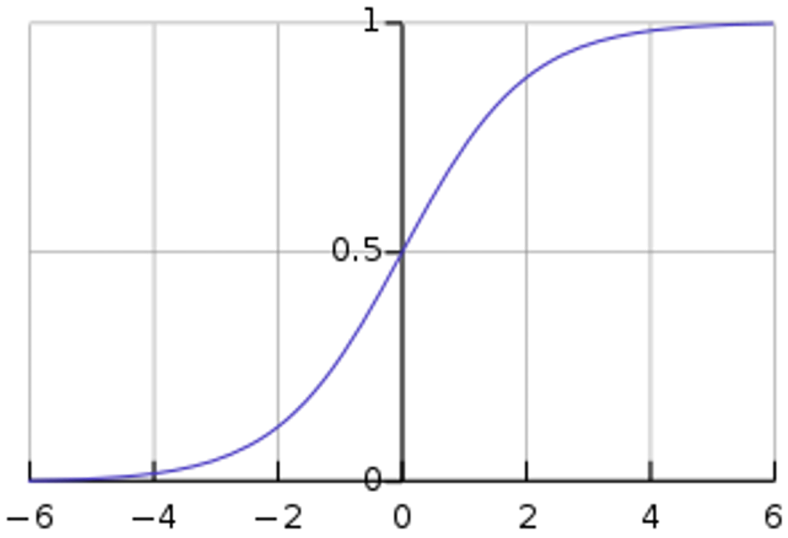
Flood or no flood

- What would be the output?
- y = 0 if not flooded
- y = 1 if flooded
- Anything in between is a score for how likely it is to flood



Logistic function

- Let's force h to be between 0 and 1 somehow
- Introducing the logistic function (sigmoid function)



$$f(x) = \frac{1}{1+e^{-x}}$$
$$= \frac{e^{x}}{1+e^{x}}$$

https://en.wikipedia.org/wiki/Logistic_function

Logistic Regression

- Pass $\theta^T \mathbf{x}$ through the logistic function

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Loss function?

- MSE error no longer a good candidate
- Let’s turn to use probabilistic argument for logistic regression

Logistic Function derivative

The derivative has a nice property by design.
This is also why many algorithm we'll learn later in class also uses the logistic function

$$
\begin{aligned}
g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\
&= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{(1+e^{-z})}\cdot\left(1-\frac{1}{(1+e^{-z})}\right) \\
&= g(z)(1-g(z)).
\end{aligned}
$$

Probabilistic view of Logistic Regression

- Let's assume, we'll classify as 1 with probability according to the output of

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

$$\begin{aligned} P(y = 1 \mid x; \theta) &= h_\theta(x) \\ P(y = 0 \mid x; \theta) &= 1 - h_\theta(x) \end{aligned}$$

or

$$p(y \mid x; \theta) = (h_\theta(x))^y \, (1 - h_\theta(x))^{1-y}$$

Maximizing log likelihood

$$p(y \mid x; \theta) = (h_\theta(x))^y \, (1 - h_\theta(x))^{1-y}$$

$$\begin{aligned} g'(z) &= \frac{d}{dz} \frac{1}{1+e^{-z}} \\ &= \frac{1}{(1+e^{-z})^2} \left(e^{-z}\right) \\ &= \frac{1}{(1+e^{-z})} \cdot \left(1 - \frac{1}{(1+e^{-z})}\right) \\ &= g(z)(1-g(z)). \end{aligned}$$

Maximizing log likelihood

$$p(y \mid x; \theta) = (h_\theta(x))^y \, (1 - h_\theta(x))^{1-y}$$

$$\begin{aligned} g'(z) &= \frac{d}{dz} \frac{1}{1+e^{-z}} \\ &= \frac{1}{(1+e^{-z})^2} \left(e^{-z}\right) \\ &= \frac{1}{(1+e^{-z})} \cdot \left(1 - \frac{1}{(1+e^{-z})}\right) \\ &= g(z)(1-g(z)). \end{aligned}$$

$$\theta_j \Leftarrow \theta_j + r \Sigma_{i=1}^{m} (y_i - h_\theta(x_i)) x_i^{(j)}$$

Logistic Regression update rule

$$\theta_j \Leftarrow \theta_j + r\Sigma_{i=1}^{m}(y_i - h_\theta(x_i))x_i^{(j)}$$

Update rule for linear regression

$$\theta_j \Leftarrow \theta_j + r\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x}_i)x_i^{(j)}$$

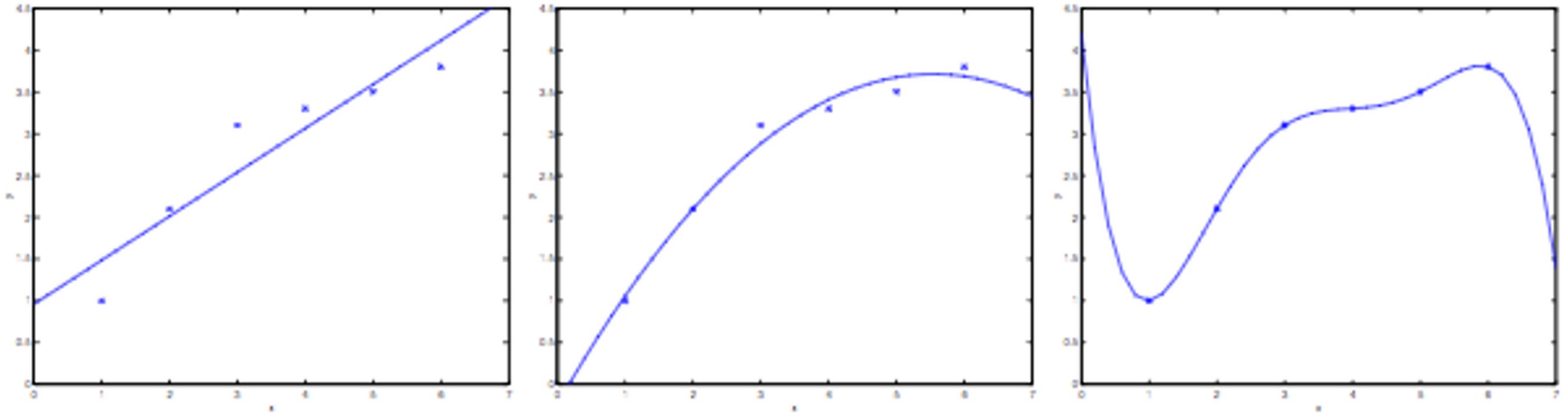
Regression with non-linear features

- 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 trade-off

- We will formulate overfitting and underfitting mathematically
- Using regression model

Regression with Gaussian noise

- $y = h(\mathbf{x}) + \varepsilon$
  - Where ε is normally distributed with mean zero and variance σ^2
  - The training data D = $\{(\mathbf{x_1}, y_1), (\mathbf{x_3}, y_3), (\mathbf{x_3}, y_3)\ldots\}$ is drawn from some distribution $P(\mathbf{x}, y)$ governing our universe!
  - Assume $(\mathbf{x_i}, y_i)$ is iid
- Given D we can train a regressor $h_D(x)$
- We calculate the expected error (squared error) on new $(\mathbf{x}, y)$ data with the regressor
  - $E_{(\mathbf{x},y)}[(h_D(\mathbf{x}) - y)^2] = \int_x \int_y (h_D(\mathbf{x}) - y)^2 \Pr(\mathbf{x}, y) \partial y \partial \mathbf{x}$
- But D is actually random too!

Regression with Gaussian noise

- We calculate the expected error (squared error) on new (**x**,y) data with the regressor
  - $E_{(\mathbf{x},y)}[(h_D(\mathbf{x}) - y)^2] = \int_x \int_y (h_D(\mathbf{x}) - y)^2 \Pr(\mathbf{x}, y) \partial y \partial \mathbf{x}$
- Consider parallel worlds, we can receive different training data D which yields different regression $h_D(x)$
- The expectation of error over all possible new test data point (**x**,y) and different possible training data D is

$$E_{\substack{(\mathbf{x},y)\sim P \\ D\sim P^n}} \left[(h_D(\mathbf{x}) - y)^2 \right] = \int_D \int_{\mathbf{x}} \int_y (h_D(\mathbf{x}) - y)^2 \mathrm{P}(\mathbf{x}, y) \mathrm{P}(D) \partial \mathbf{x} \partial y \partial D$$

http://www.cs.cornell.edu/courses/cs4780/2017sp/lectures/lecturenote11.html

Regression with Gaussian noise

- This expression tells the expected quality of our model with random training data and a random test data

$$E_{\substack{(\mathbf{x},y)\sim P \\ D\sim P^n}}\left[(h_D(\mathbf{x})-y)^2\right] = \int_D \int_{\mathbf{x}} \int_y (h_D(\mathbf{x})-y)^2 \mathrm{P}(\mathbf{x},y)\mathrm{P}(D)\partial\mathbf{x}\partial y\partial D$$

$\bar{h}(x) \triangleq E_D[h_D] = \int_D h_D p(D)\, dD$

Regression with Gaussian noise

- This expression tells the expected quality of our model with random training data and a random test data

$$E_{\substack{(\mathbf{x},y)\sim P \\ D\sim P^n}}\left[(h_D(\mathbf{x})-y)^2\right] = \int_D \int_{\mathbf{x}} \int_y (h_D(\mathbf{x})-y)^2 \mathrm{P}(\mathbf{x},y)\mathrm{P}(D)\partial\mathbf{x}\partial y\partial D$$

$= E_{xyD}\left[(h_D(x) - \bar{h}(x) + \bar{h}(x) - y)^2\right]$

$= E_{xyD}\left[\underbrace{(h_D(x)-\bar{h}(x))^2}_{1} + \underbrace{2(h_D(x)-\bar{h}(x))(\bar{h}(x)-y)}_{2} + \underbrace{(\bar{h}(x)-y)^2}_{3}\right]$

① $E_{xyD}\left[(h_D(x)-h(x))^2\right]$

$= E_{xD}\left[(h_D(x)-h(x))^2\right]$

$E_y[f(x)] = \int f(x)p(y)dy$

$= f(x)\int p(y)dy$

$= f(x)$

② $E_{xyD}\left[2(h_D(x)-h(x))(h(x)-y)\right]$

$= E_x\left[(h(x)-y)\,E_D[h_D(x)-h(x)]\right]$

$= 0$

$E_D[h_D(x)] = \bar{h}_D(x)$

③ $E_{xyD}\left[(h(x)-y)^2\right]$

$\bar{y}(x) = E_{y|x}[y] = \int_y y\,p(y|x)dy$

$= E_{xyD}\left[(h(x)-\bar{y}(x)+\bar{y}(x)-y)^2\right]$

$= E_{xyD}\left[(h(x)-\bar{y}(x))^2 + 2(h(x)-\bar{y}(x))(\bar{y}(x)-y) + (\bar{y}(x)-y)^2\right]$

④ ⑤ ⑥

(5) $E_{xy}\left[2(h(x)-\bar{y}(x))(\bar{y}(x)-y)\right]$

$= \int\int 2(h(x)-\bar{y}(x))(\bar{y}(x)-y)\,p(x,y)\,dx\,dy$

$p(x,y) = p(y|x)\,p(x)$

$= \int 2(h(x)-\bar{y}(x))\left[\int (\bar{y}(x)-y)\,p(y|x)\,dy\right]p(x)\,dx$

$E_{y|x}\left[\bar{y}(x)-y\right]$

$= 0$

Regression with Gaussian noise

- This expression tells the expected quality of our model with random training data and a random test data

$$E_{\substack{(\mathbf{x},y)\sim P \\ D\sim P^n}}\left[(h_D(\mathbf{x})-y)^2\right] = \int_D \int_{\mathbf{x}} \int_y (h_D(\mathbf{x})-y)^2 \mathrm{P}(\mathbf{x},y)\mathrm{P}(D)\partial\mathbf{x}\partial y\partial D$$

$$\underbrace{E_{\mathbf{x},y,D}\left[(h_D(\mathbf{x})-y)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[(h_D(\mathbf{x})-\bar{h}(\mathbf{x}))^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[(\bar{y}(\mathbf{x})-y)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x}))^2\right]}_{\text{Bias}^2}$$

① ③ ②

$$\bar{h}(x) \triangleq E_D[h_D] = \int_D h_D\, p(D)\, dD$$

Expected model

$$\bar{y}(x) \triangleq E_{y|x}[y] = \int_y y\, p(y|x)\, dy$$

Expected groundtruth

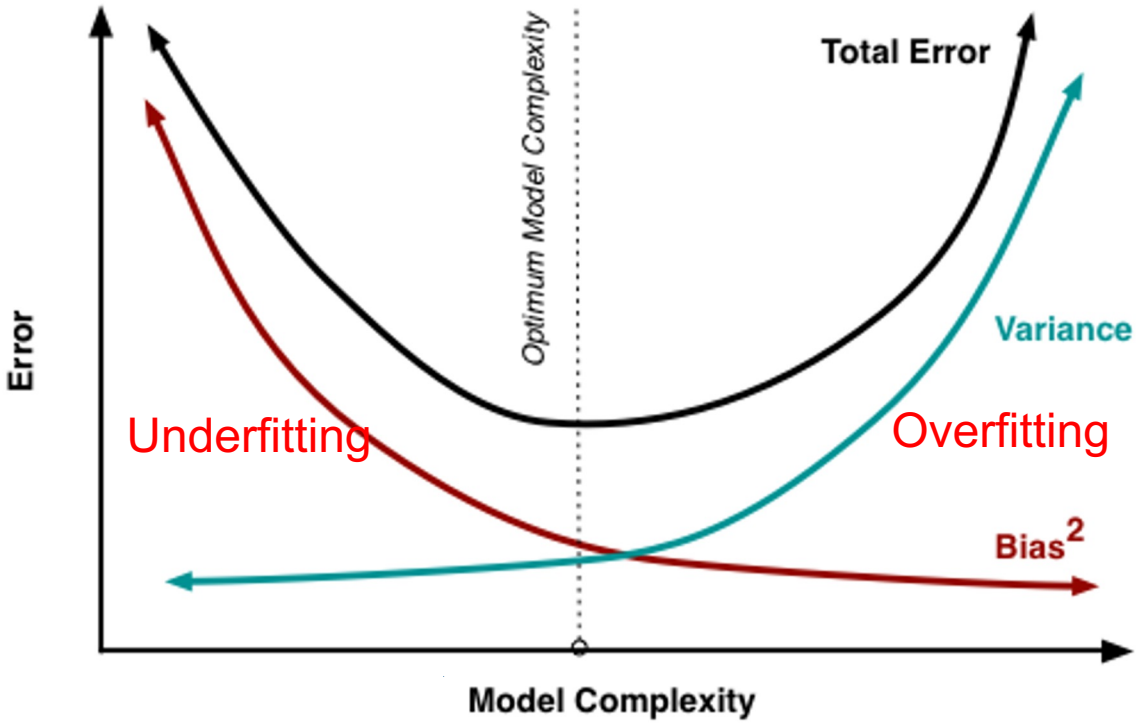
Variance, Bias, and noise

$$\underbrace{E_{\mathbf{x},y,D}\left[(h_D(\mathbf{x})-y)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[(h_D(\mathbf{x})-\bar{h}(\mathbf{x}))^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[(\bar{y}(\mathbf{x})-y)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x}))^2\right]}_{\text{Bias}^2}$$

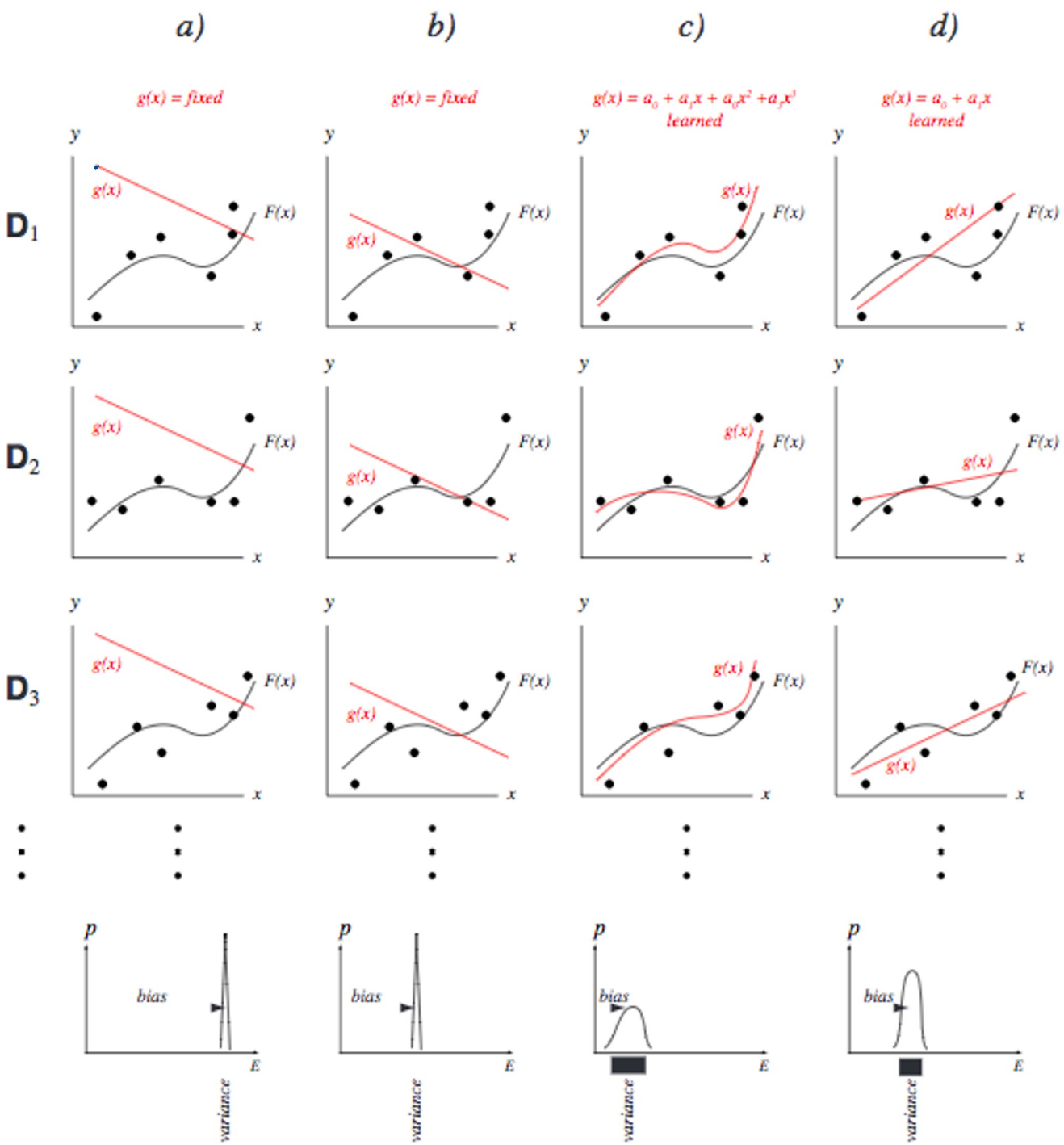
- Variance: how your classifier changes if the training data changes. Measures generalizability.
- Bias: The model's inherent error. If you have infinite training data, you will have the average classifier $\bar{h}$ and still left with this error.
  - For example, even with infinite training data, a linear classifier will still have errors if the distribution is non-linear.
- Noise: data-intrinsic noise. Noise from measurement, noise from feature extraction, etc. Regardless of your model this remains.

Bias-Variance Underfitting-Overfitting

- *Usually* if you try to reduce the bias of your model, the variance will increase, and vice versa.
- Called the “bias-variance trade-off”



http://www.cs.cornell.edu/courses/cs4780/2017sp/lectures/lecturenote11.html



Duda et al. Pattern Classification

Summary

- Linear Regression
  - Minimizing a loss function
    - Solve directly vs Iterative (gradient descent)
  - MSE view vs probabilisitic view
  - Linear regression vs Logistic regression

For linear regression, direct or iterative should yield similar answers as long as the learning rate is small enough
(No local minima to get stuck - Convex problem)

$$\theta = (X^T X)^{-1} X^T y$$

$$\theta_j \Leftarrow \theta_j + r\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x}_i)x_i^{(j)}$$

$$\theta_j \Leftarrow \theta_j + r\Sigma_{i=1}^{m}(y_i - h_\theta(x_i))x_i^{(j)}$$

Homework

Some some regression