Deep Learning for Vision & Language

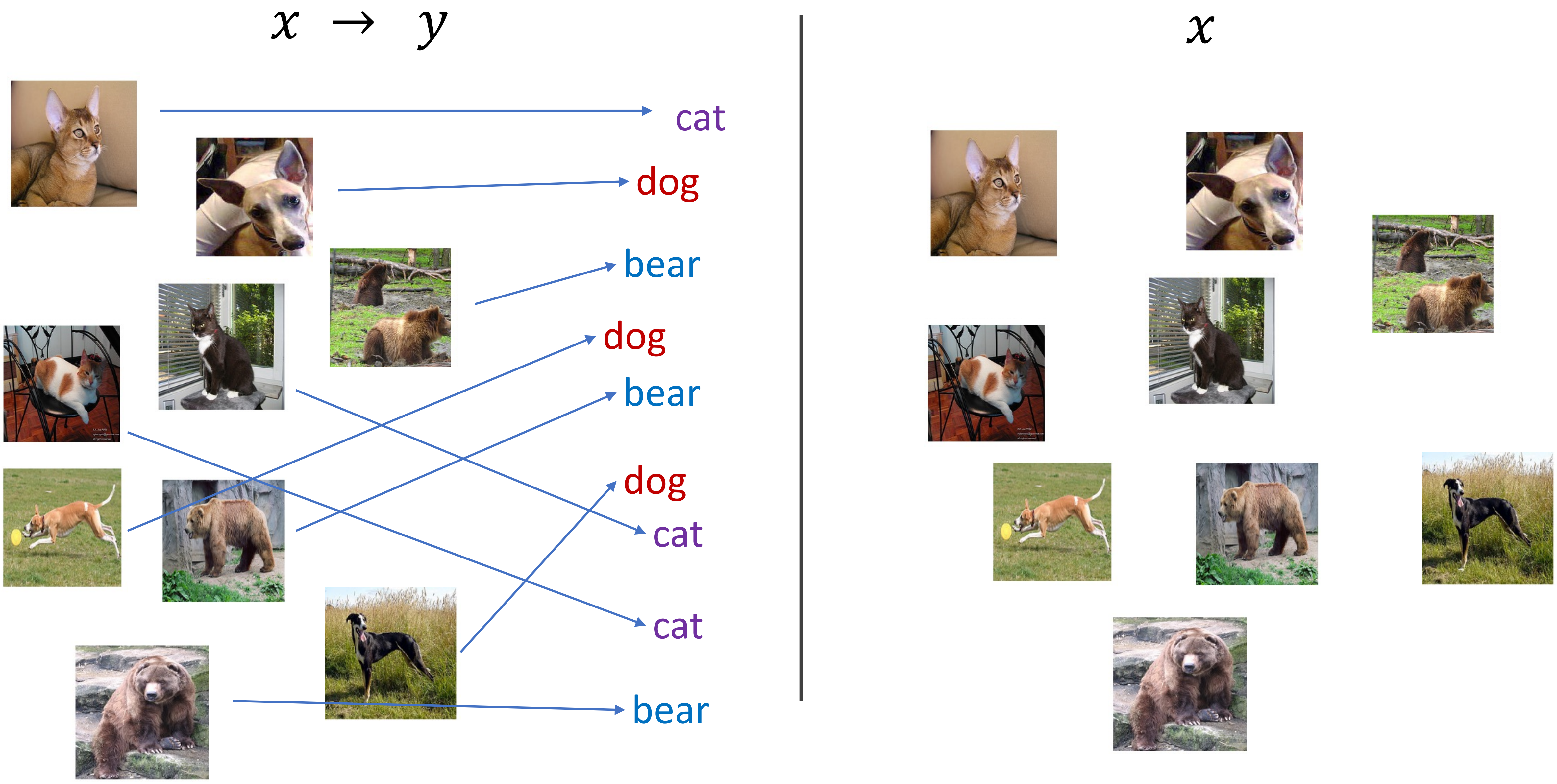
Machine Learning I: Supervised vs Unsupervised Learning
Linear Classifiers / Regressors

RICE UNIVERSITY

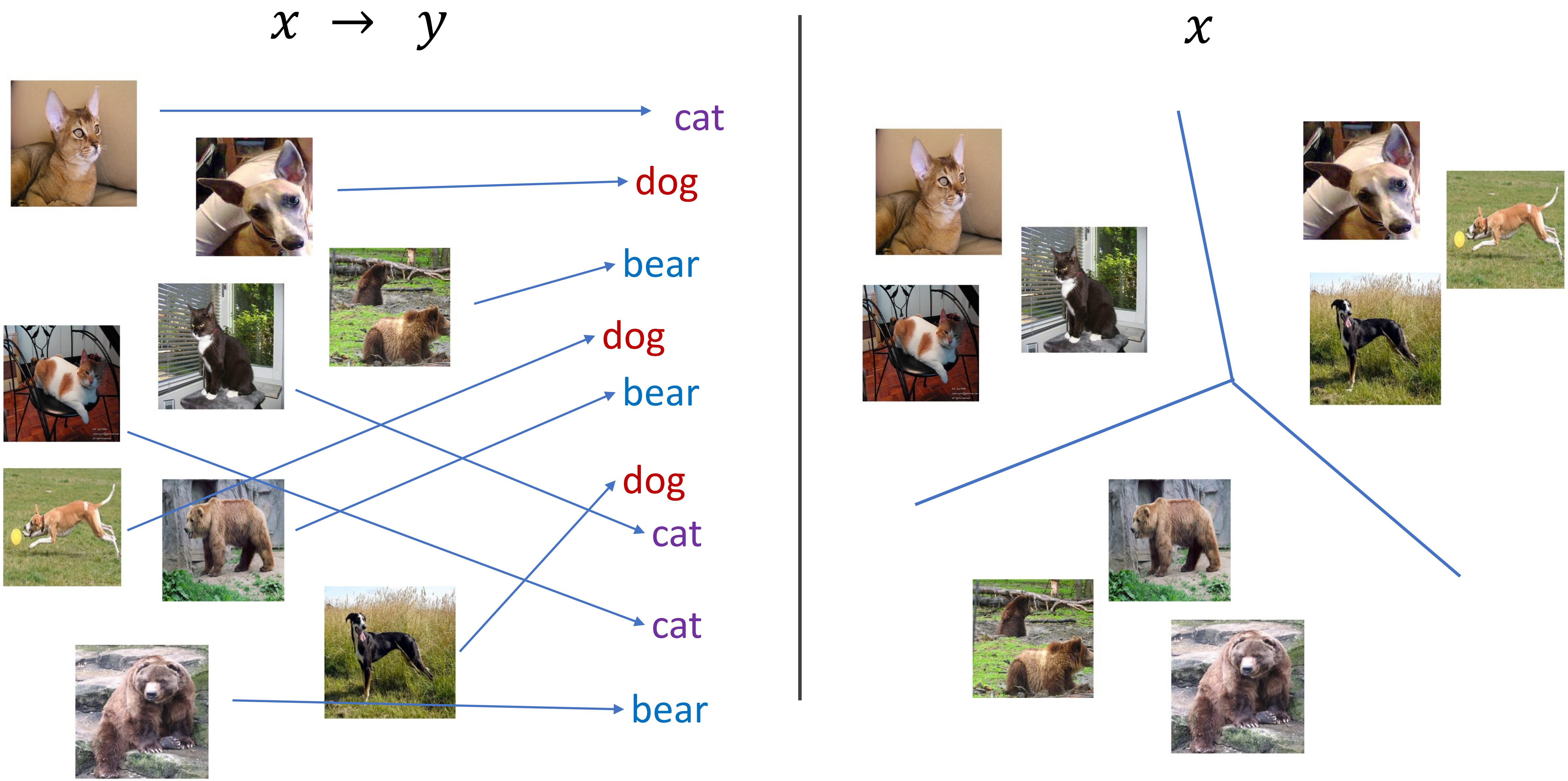
Machine Learning

The study of algorithms that learn from data.

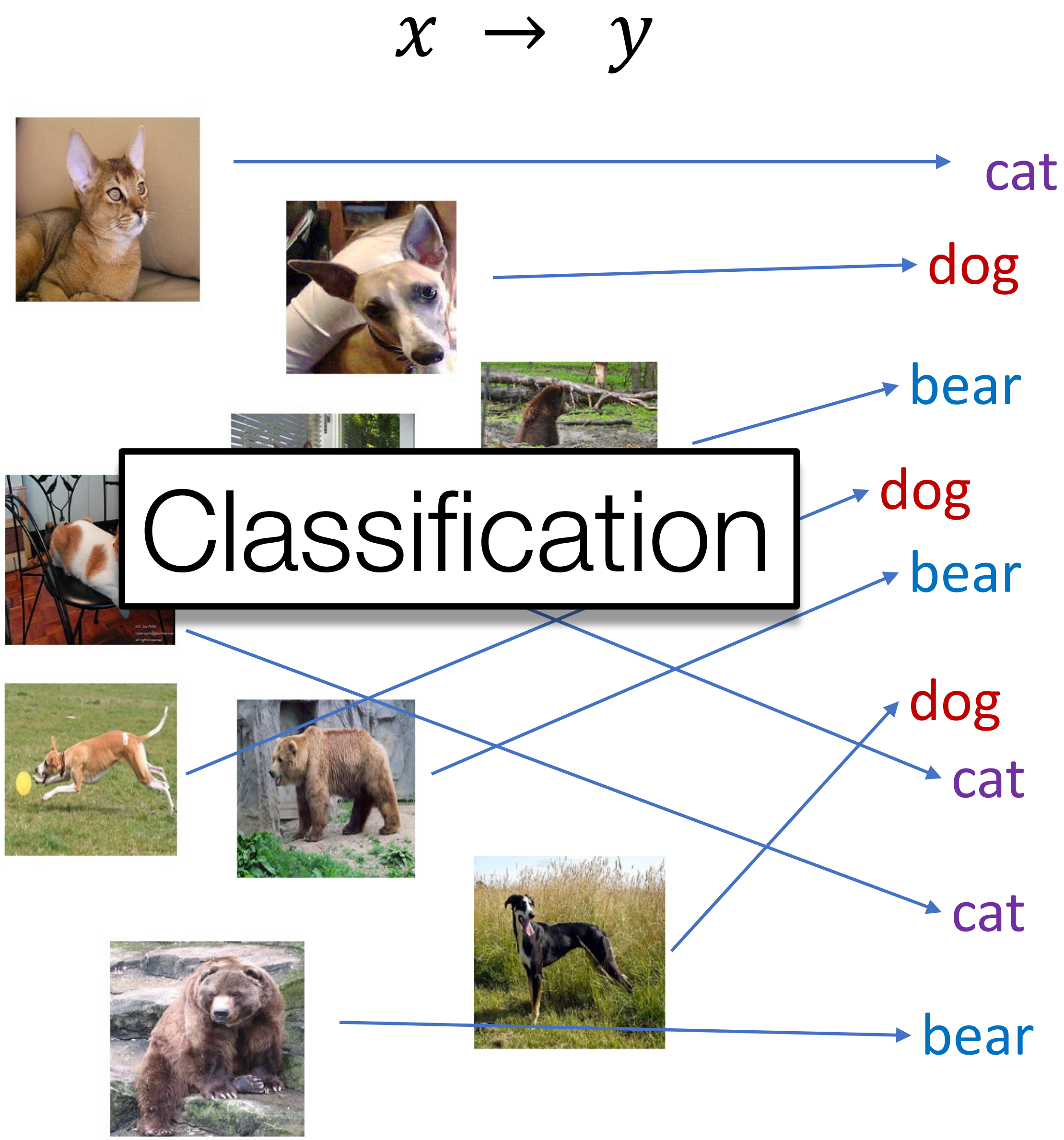
Supervised Learning vs Unsupervised Learning



Supervised Learning vs Unsupervised Learning



Supervised Learning vs Unsupervised Learning

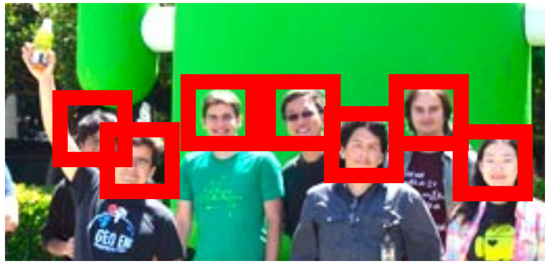


Supervised Learning...

Classification

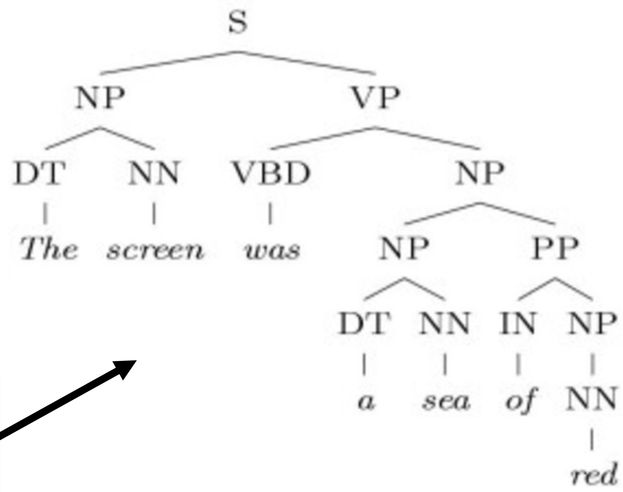
cat

Face Detection



The screen was
a sea of red

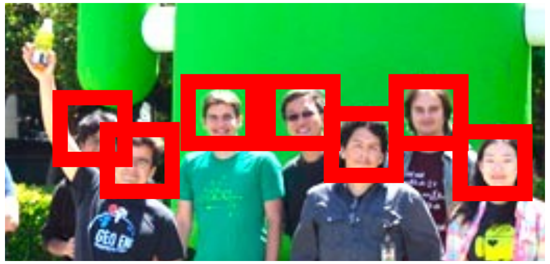
Language Parsing

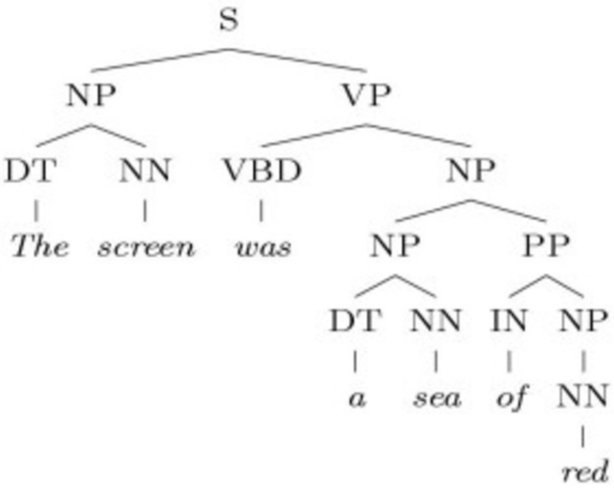


Structured Prediction

Supervised Learning…

cat $= f($  $)$

 $= f($  $)$



$= f($ The screen was a sea of red $)$

Machine Learning – Regression vs Classification

$$y = f(x)$$

- Regression:
  y is a continuous variable e.g. in some interval of values e.g. in (0, 10]

- Classification:
  y is a discrete variable e.g. could take a set of values {0, 1, 2, 3, 4}

Machine Learning – Regression vs Classification

$$y = f(x)$$

- Also notice that both y and x could be vectors – and they usually are for many problems we will study.

- Also notice that f can be any function from the simplest you can think of to the most complicated composition of functions.

For instance, Linear Regression and Classification

$$y = wx + b$$

- Note: (w, b) are the coefficients in the linear regression, and will also be referred as parameters.

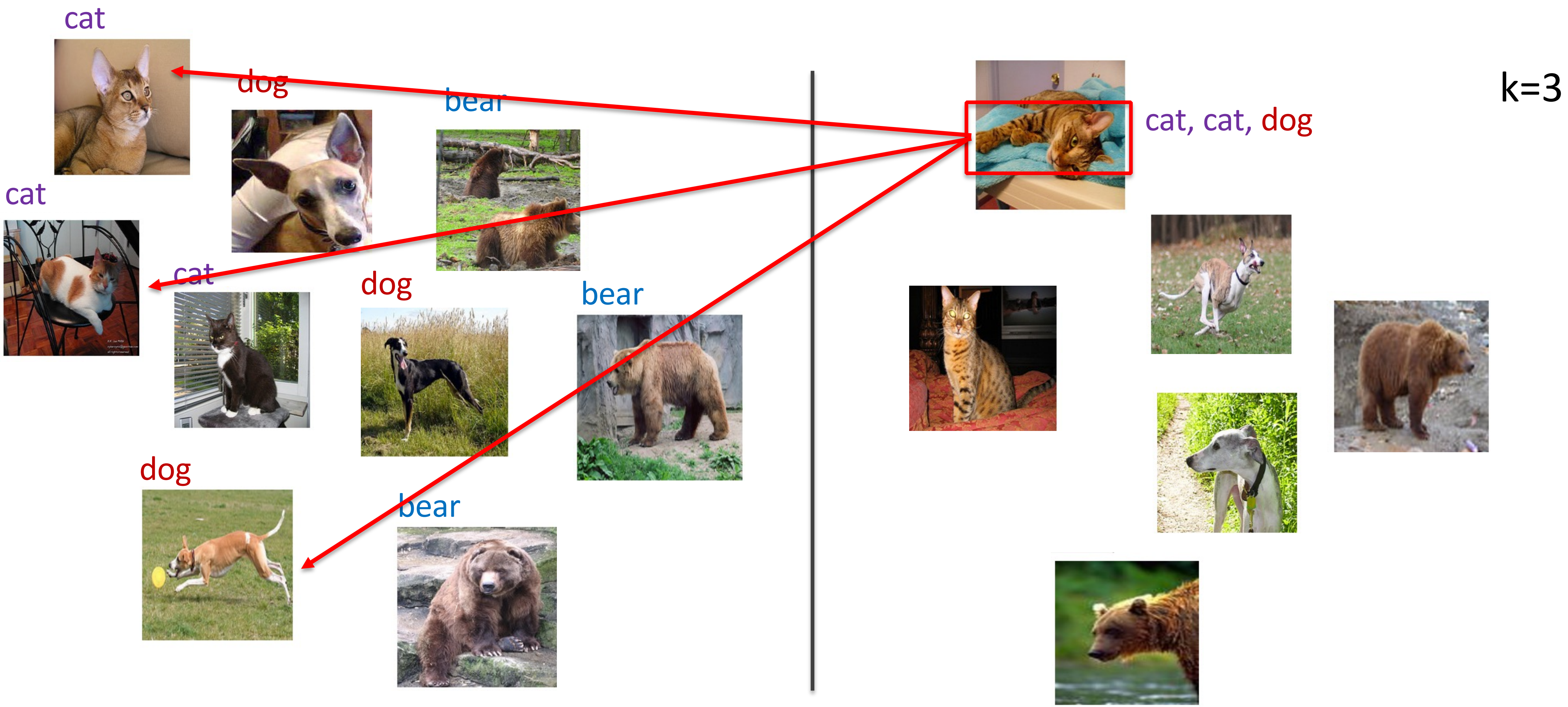
- Also notice if *x* is a vector then *w* must also be a vector of coefficients.

- A lot of work in Machine Learning and optimization is finding the right set of parameters (w, b) that can map any pairs of (x,y) values for a given problem.

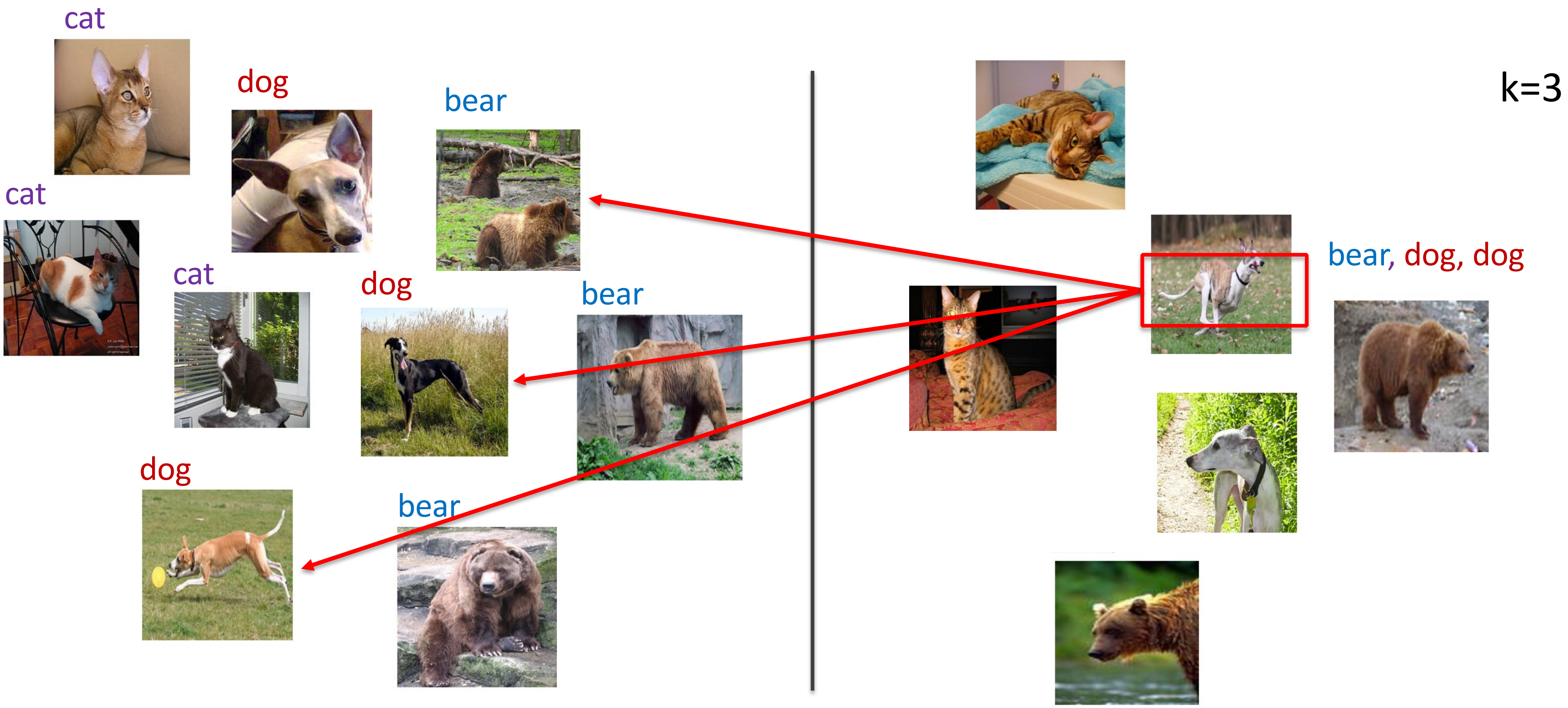
ML Classifier / Regression models

- K-nearest neighbors
- Linear classifier / Linear regression
- Naïve Bayes classifiers
- Decision Trees
- Random Forests
- Boosted Decision Trees
- Neural Networks

Supervised Learning – k-Nearest Neighbors



Supervised Learning – k-Nearest Neighbors



ML Classifier / Regression models

- K-nearest neighbors
- Linear classifier / Linear regression
- Naïve Bayes classifiers
- Decision Trees
- Random Forests
- Boosted Decision Trees
- Neural Networks

Linear Regression

Example: Hollywood movie data

input variables x — output variables y

production costs	promotional costs	genre of the movie	box office first week	total book sales	total revenue USA	total revenue international
$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$x_4^{(1)}$	$x_5^{(1)}$	$x_6^{(1)}$	$x_7^{(1)}$
$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	$x_4^{(2)}$	$x_5^{(2)}$	$x_6^{(2)}$	$x_7^{(2)}$
$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	$x_4^{(3)}$	$x_5^{(3)}$	$x_6^{(3)}$	$x_7^{(3)}$
$x_1^{(4)}$	$x_2^{(4)}$	$x_3^{(4)}$	$x_4^{(4)}$	$x_5^{(4)}$	$x_6^{(4)}$	$x_7^{(4)}$
$x_1^{(5)}$	$x_2^{(5)}$	$x_3^{(5)}$	$x_4^{(5)}$	$x_5^{(5)}$	$x_6^{(5)}$	$x_7^{(5)}$

Linear Regression

Example: Hollywood movie data

input variables x

production costs	promotional costs	genre of the movie	box office first week	total book sales
$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$x_4^{(1)}$	$x_5^{(1)}$
$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	$x_4^{(2)}$	$x_5^{(2)}$
$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	$x_4^{(3)}$	$x_5^{(3)}$
$x_1^{(4)}$	$x_2^{(4)}$	$x_3^{(4)}$	$x_4^{(4)}$	$x_5^{(4)}$
$x_1^{(5)}$	$x_2^{(5)}$	$x_3^{(5)}$	$x_4^{(5)}$	$x_5^{(5)}$

output variables y

total revenue USA	total revenue international
$y_1^{(1)}$	$y_2^{(1)}$
$y_1^{(2)}$	$y_2^{(2)}$
$y_1^{(3)}$	$y_2^{(3)}$
$y_1^{(4)}$	$y_2^{(4)}$
$y_1^{(5)}$	$y_2^{(5)}$

Linear Regression

Example: Hollywood movie data

	input variables x					output variables y	
	production costs	promotional costs	genre of the movie	box office first week	total book sales	total revenue USA	total revenue international
training data	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$x_4^{(1)}$	$x_5^{(1)}$	$y_1^{(1)}$	$y_2^{(1)}$
	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	$x_4^{(2)}$	$x_5^{(2)}$	$y_1^{(2)}$	$y_2^{(2)}$
	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	$x_4^{(3)}$	$x_5^{(3)}$	$y_1^{(3)}$	$y_2^{(3)}$
test data	$x_1^{(4)}$	$x_2^{(4)}$	$x_3^{(4)}$	$x_4^{(4)}$	$x_5^{(4)}$	$y_1^{(4)}$	$y_2^{(4)}$
	$x_1^{(5)}$	$x_2^{(5)}$	$x_3^{(5)}$	$x_4^{(5)}$	$x_5^{(5)}$	$y_1^{(5)}$	$y_2^{(5)}$

Linear Regression

$$\hat{y} = \sum_i w_i x_i$$

$$\hat{y} = W^T x$$

Prediction,
Inference,
Testing

$$D = \{(x^{(d)}, y^{(d)})\}$$

$$L(W) = \sum_{d=1}^{|D|} l(\hat{y}^{(d)}, y^{(d)})$$

$$W^* = \operatorname{argmin} L(W)$$

Training,
Learning,
Parameter estimation
Objective minimization

Linear Regression

$$\hat{y}_j = \sum_i w_{ji} x_i$$

$$\hat{y} = W^T x$$

Prediction,
Inference,
Testing

$$D = \{(x^{(d)}, y^{(d)})\}$$

$$L(W) = \sum_{d=1}^{|D|} \sum_j l(\hat{y}_j{}^{(d)}, y_j{}^{(d)})$$

$$W^* = \operatorname{argmin} L(W)$$

Training,
Learning,
Parameter
estimation
Objective
minimization

Linear Regression – Least Squares

$$\hat{y}_j = \sum_i w_{ji} x_i$$

$$\hat{y} = W^T x$$

$$D = \{(x^{(d)}, y^{(d)})\}$$

$$\boxed{L(W) = \sum_{d=1}^{|D|} \sum_j \left(\hat{y}_j{}^{(d)} - y_j{}^{(d)}\right)^2}$$

$$W^* = \operatorname{argmin} L(W)$$

Training,
Learning,
Parameter estimation
Objective minimization

Linear Regression – Least Squares

$$L(W) = \sum_{d=1}^{|D|} \sum_{j} \left(\hat{y}_j^{(d)} - y^{(d)} \right)^2$$

$$\hat{y}_j^{(d)} = \sum_i w_{ji} x_i^{(d)}$$

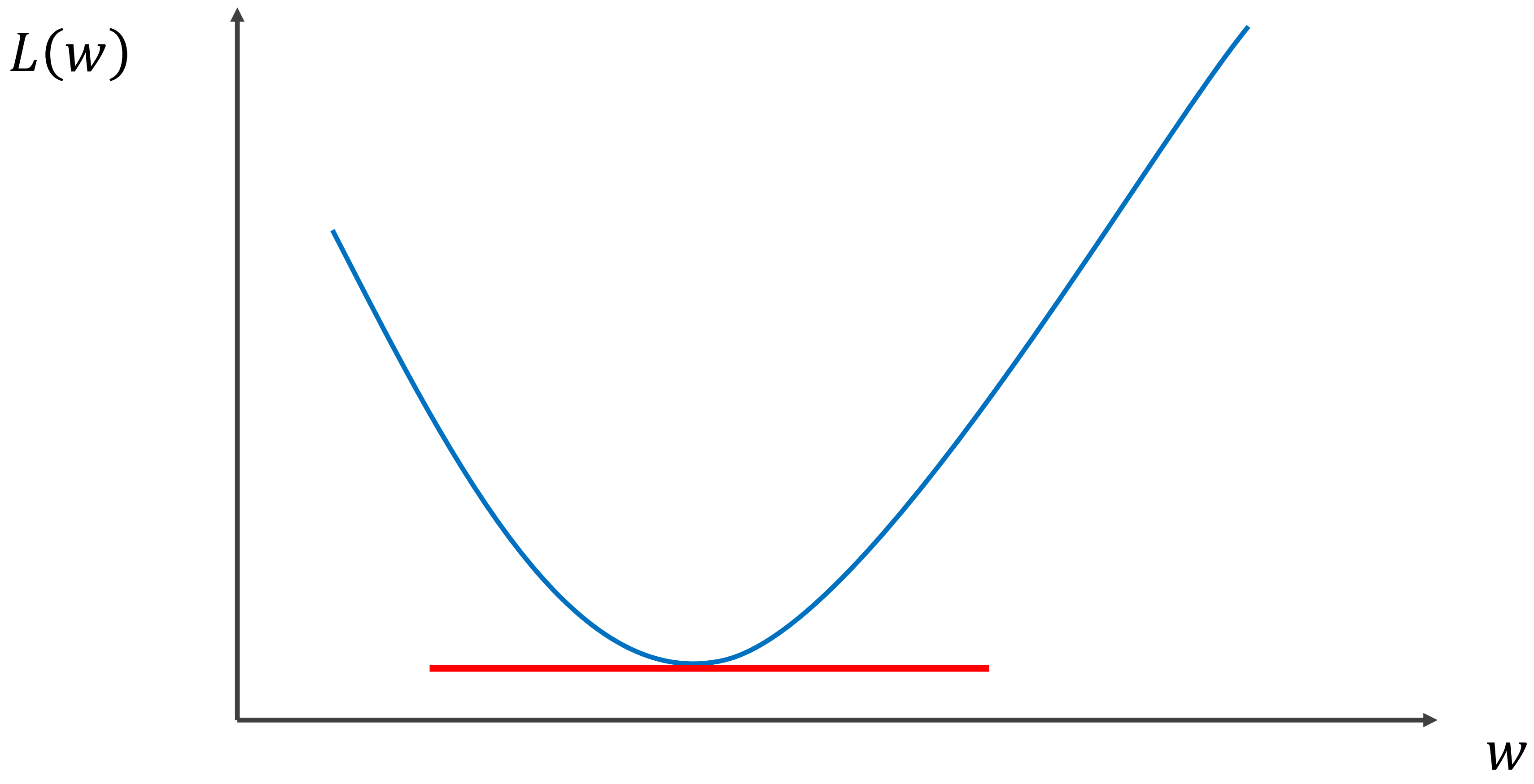
$$L(W) = \sum_{d=1}^{|D|} \sum_{j} \left(\sum_i w_{ji} x_i^{(d)} - y^{(d)} \right)^2$$

Linear Regression – Least Squares

$$L(W) = \sum_{d=1}^{|D|} \sum_{j} \left(\sum_{i} w_{ji} x_i^{(d)} - y^{(d)} \right)^2$$

$$W^* = \operatorname{argmin} L(W)$$

How to find the minimum of a function L(W)?



$$\frac{\partial L(w)}{\partial w} = 0$$

Linear Regression – Least Squares

$$\frac{\partial L(W)}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \left(\sum_{d=1}^{|D|} \sum_j \left(\sum_i w_{ji} x_i^{(d)} - y^{(d)}\right)^2\right)$$

$$\frac{\partial L(W)}{\partial w_{ji}} = 0$$

…

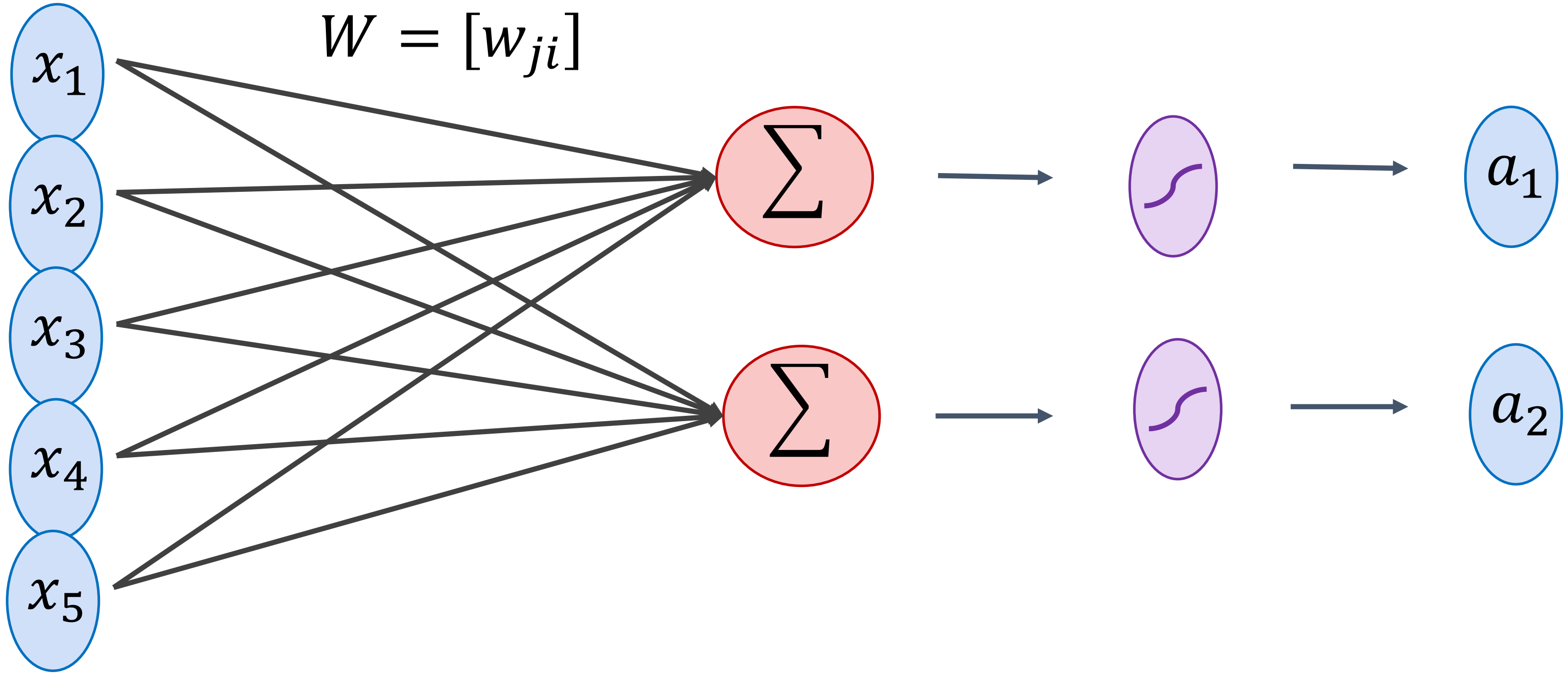
$$W = (X^T X)^{-1} X^T Y$$

ML Classifier / Regression models

- K-nearest neighbors
- Linear classifier / Linear regression
- Naïve Bayes classifiers
- Decision Trees
- Random Forests
- Boosted Decision Trees
- Neural Networks

Neural Network with One Layer

$$sigmoid(z) = \frac{1}{1+e^{-z}}$$



$$a_j = sigmoid(\textstyle\sum_i w_{ji} x_i + b_j)$$

Neural Network with One Layer

$$L(W, b) = \sum_{d=1}^{|D|} \left(a^{(d)} - y^{(d)}\right)^2$$

$$a_j^{(d)} = sigmoid(\sum_i w_{ji} x_i^{(d)} + \boxed{b_j})$$

Bias parameters

$$L(W, b) = \sum_{j,d} \left(sigmoid(\sum_i w_{ji} x_i^{(d)} + b_j) - y_j^{(d)} \right)^2$$

Neural Network with One Layer

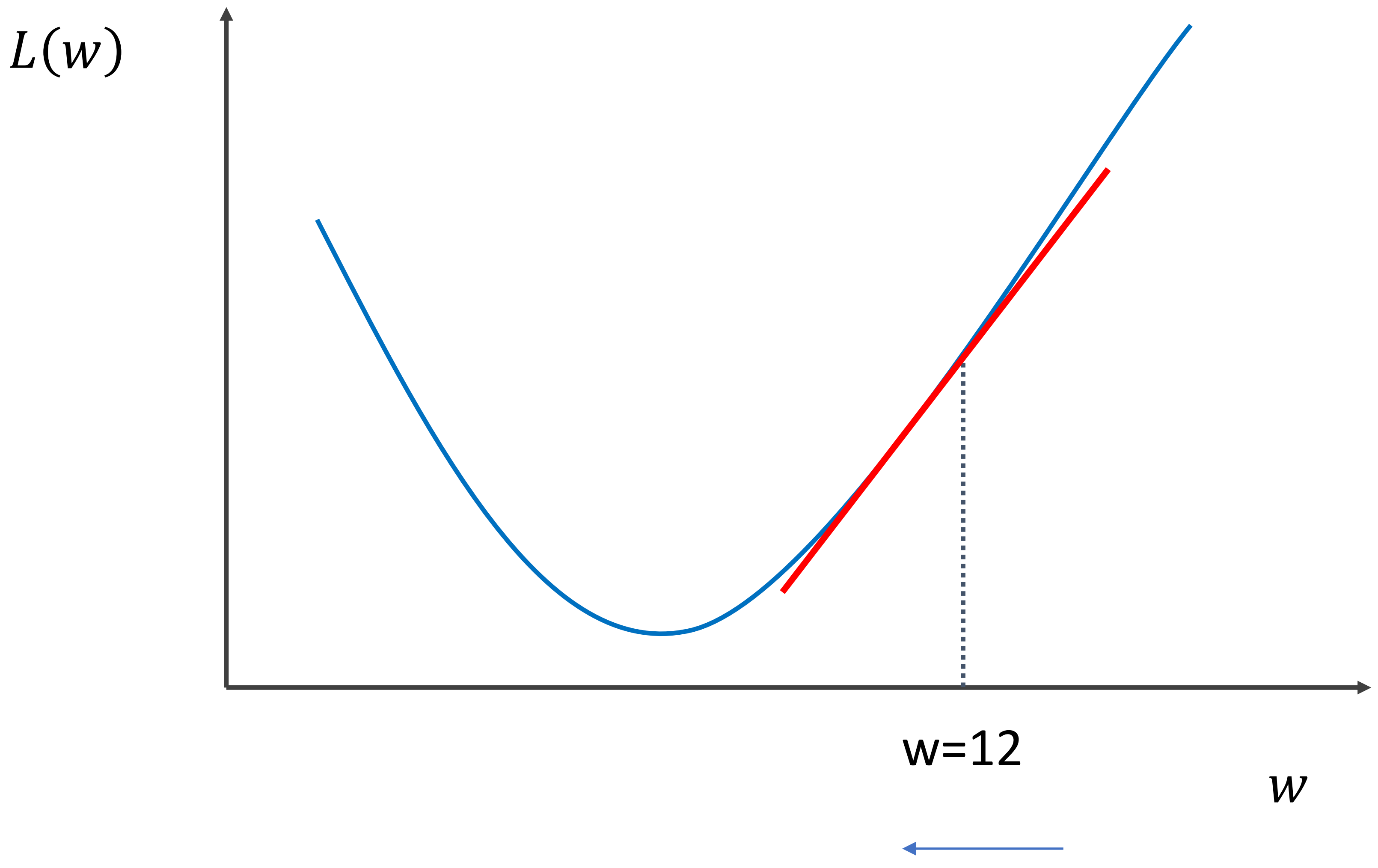
$$L(W,b) = \sum_{j,d}\left(sigmoid(\sum_{i} w_{ji}x_i^{(d)} + b_j) - \ y_j^{(d)}\right)^2$$

$$\frac{\partial L}{\partial w_{uv}} = 0$$

(1) We can compute this derivative but often there will be no closed-form solution for W when dL/dw = 0

(2) Also, even for linear regression where the solution was $W\ = (X^T X)^{-1} X^T Y$, computing this expression might be expensive or infeasible. e. g. think of computing $(X^T X)^{-1}$ for a very large dataset with a million x_i

Gradient Descent



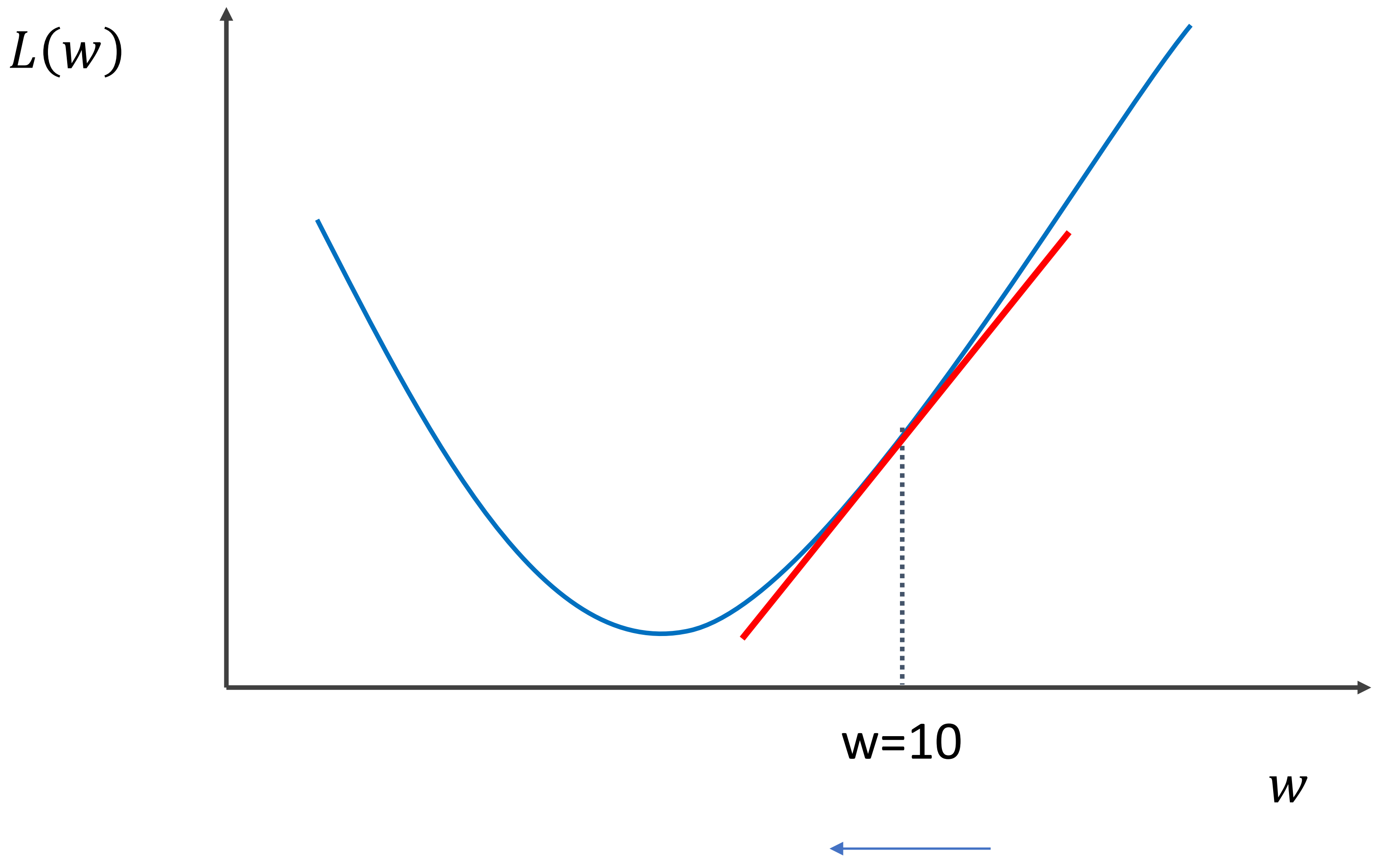
1. Start with a random value of w (e.g. w = 12)

2. Compute the gradient (derivative) of L(w) at point w = 12. (e.g. dL/dw = 6)

3. Recompute w as:

w = w – lambda * (dL / dw)

Gradient Descent

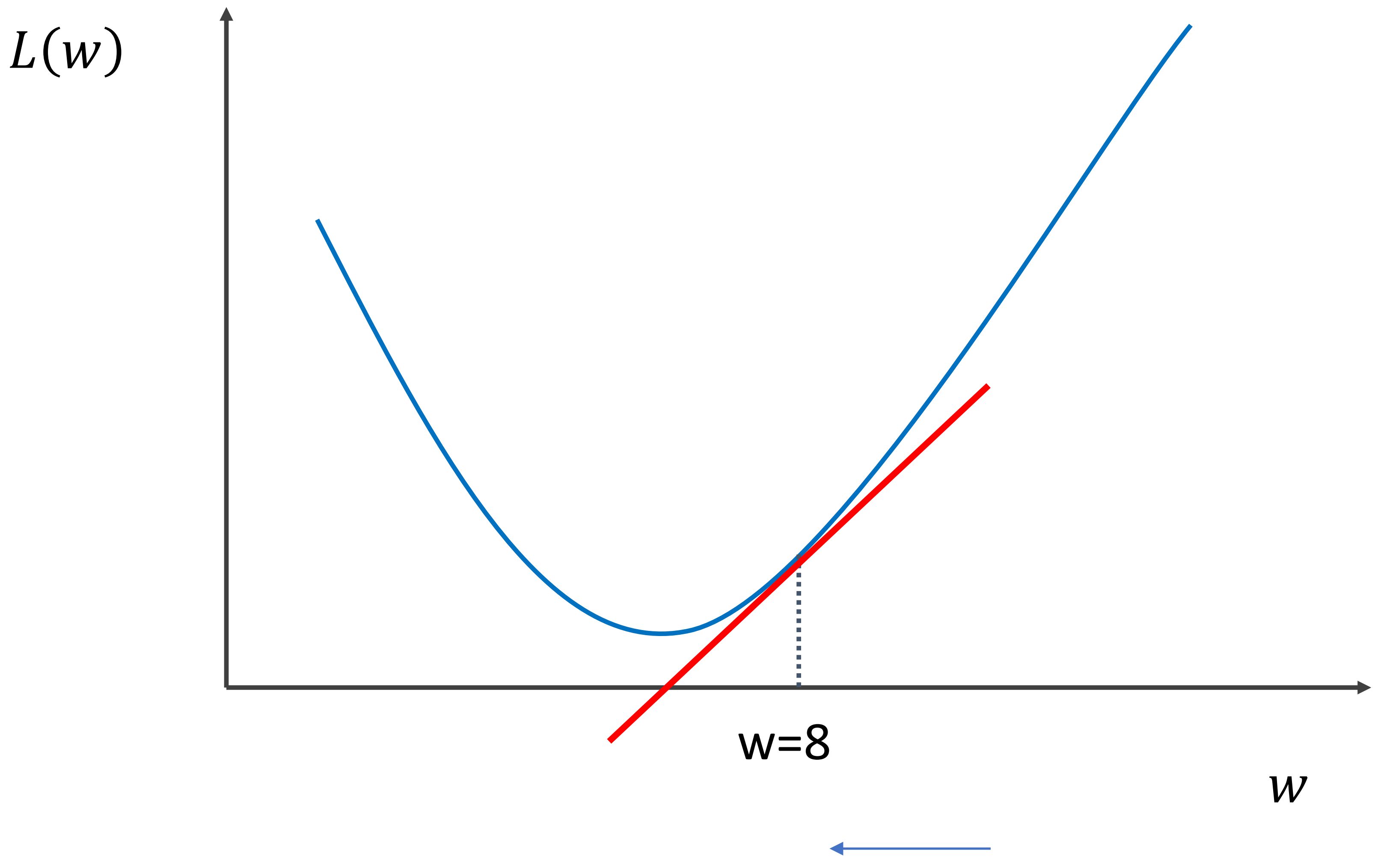


2. Compute the gradient (derivative) of L(w) at point w = 12. (e.g. dL/dw = 6)

3. Recompute w as:

w = w – lambda * (dL / dw)

Gradient Descent



2. Compute the gradient (derivative) of L(w) at point w = 12. (e.g. dL/dw = 6)

3. Recompute w as:

w = w – lambda * (dL / dw)

Gradient Descent

$\lambda = 0.01$

Initialize w and b randomly

for e = 0, num_epochs **do**

Compute: $dL(w,b)/dw$ and $dL(w,b)/db$

Update w: $w = w - \lambda\, dL(w,b)/dw$

Update b: $b = b - \lambda\, dL(w,b)/db$

Print: $L(w,b)$ // Useful to see if this is becoming smaller or not.

end

expensive

$$L(w,b) = \sum_{i=1}^{n} l(w,b)$$

Stochastic Gradient Descent (mini-batch)

$\lambda = 0.01$

$$L_B(w,b) = \sum_{i=1}^{B} l(w,b)$$

Initialize w and b randomly

for e = 0, num_epochs **do**

for b = 0, num_batches **do**

Compute: $dL_B(w,b)/dw$ and $dL_B(w,b)/db$

Update w: $w = w \ - \lambda\, dl(w,b)/dw$

Update b: $b = b \ - \lambda\, dl(w,b)/db$

Print: $L_B(w,b)$ // Useful to see if this is becoming smaller or not.

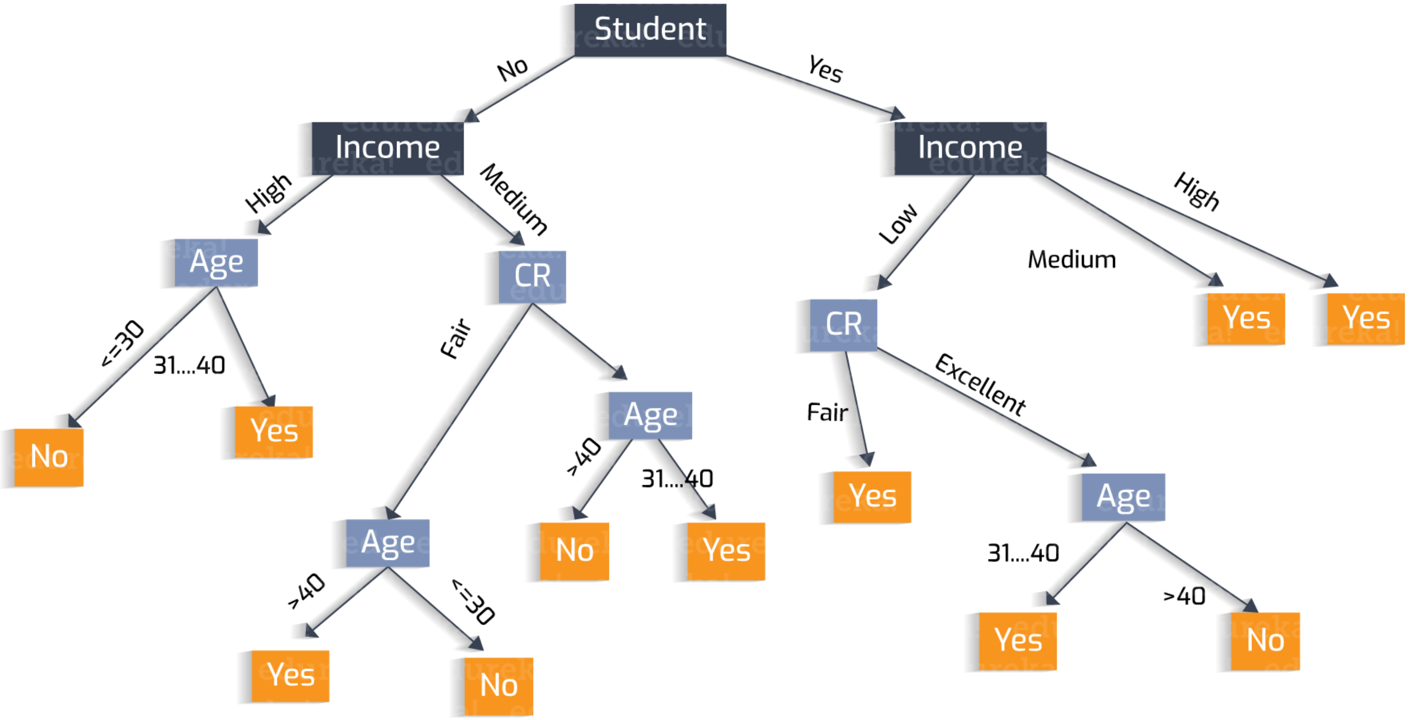
end
end

In this class we will mostly rely on…

- K-nearest neighbors
- Linear classifiers
- Naïve Bayes classifiers
- Decision Trees
- Random Forests
- Boosted Decision Trees
- Neural Networks

Why?

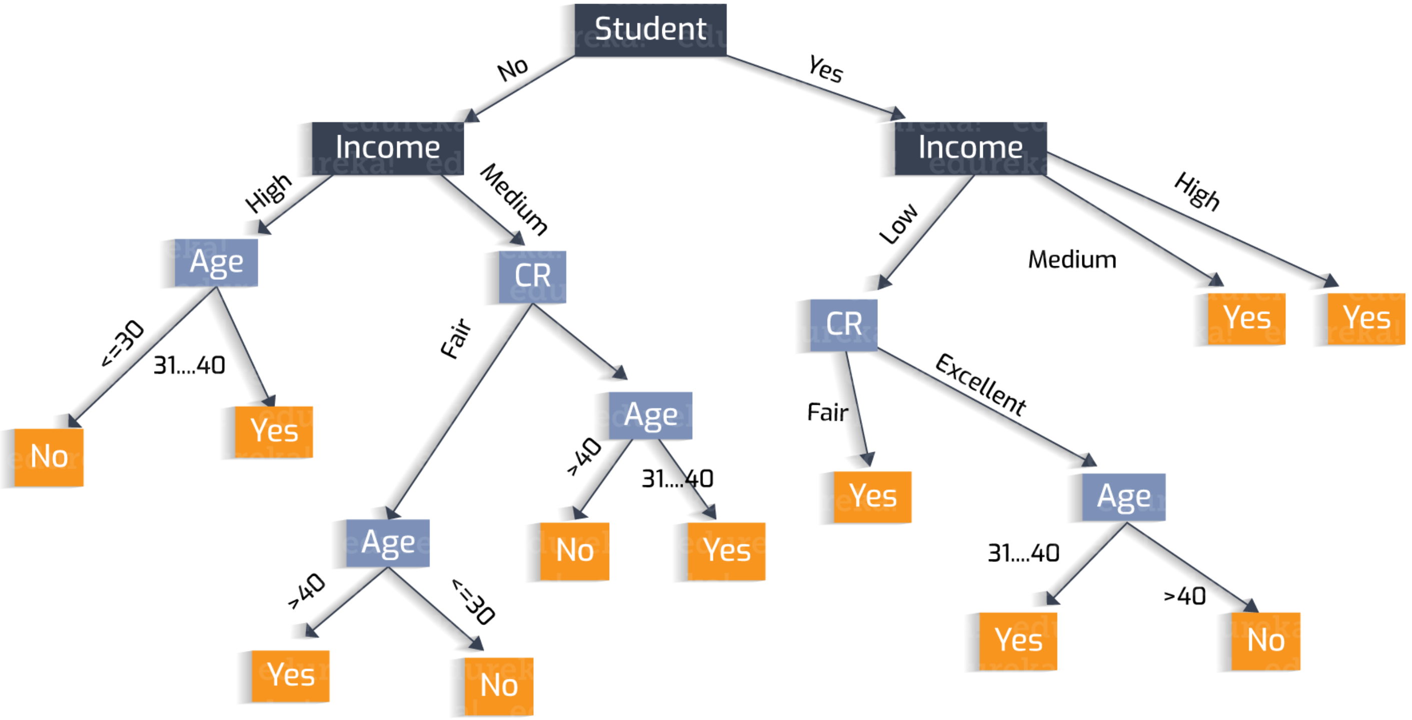
- Decisions Trees



https://heartbeat.fritz.ai/understanding-the-mathematics-behind-decision-trees-22d86d55906 by Nikita Sharma

Why?

- Decisions Trees are great because they are often interpretable.
- However, they usually deal better with categorical data – not input pixel data.



https://heartbeat.fritz.ai/understanding-the-mathematics-behind-decision-trees-22d86d55906 by Nikita Sharma

Regression vs Classification

Regression
- Labels are continuous variables – e.g. distance.
- Losses: Distance-based losses, e.g. sum of distances to true values.
- Evaluation: Mean distances, correlation coefficients, etc.

Classification
- Labels are discrete variables (1 out of K categories)
- Losses: Cross-entropy loss, margin losses, logistic regression (binary cross entropy)
- Evaluation: Classification accuracy, etc.

Supervised Learning - Classification

Training Data


cat

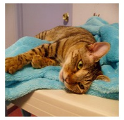

dog

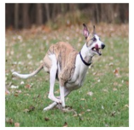

cat

⋮


bear

Test Data





⋮

Supervised Learning - Classification

Training Data

$x_1 =$ [] $y_1 =$ [cat]

$x_2 =$ [] $y_2 =$ [dog]

$x_3 =$ [] $y_3 =$ [cat]

$\vdots$

$x_n =$ [] $y_n =$ [bear]

Supervised Learning - Classification

Training Data

inputs	targets / labels / ground truth	predictions
$x_1 = [x_{11} \quad x_{12} \quad x_{13} \quad x_{14}]$	$y_1 = 1$	$\hat{y}_1 = 1$
$x_2 = [x_{21} \quad x_{22} \quad x_{23} \quad x_{24}]$	$y_2 = 2$	$\hat{y}_2 = 2$
$x_3 = [x_{31} \quad x_{32} \quad x_{33} \quad x_{34}]$	$y_3 = 1$	$\hat{y}_3 = 2$
. . .		
$x_n = [x_{n1} \quad x_{n2} \quad x_{n3} \quad x_{n4}]$	$y_n = 3$	$\hat{y}_n = 1$

We need to find a function that maps *x* and *y* for any of them.

$$\hat{y}_i = f(x_i; \theta)$$

How do we ”learn” the parameters of this function?

We choose ones that makes the following quantity small:

$$\sum_{i=1}^{n} Cost(\hat{y}_i, y_i)$$

Stochastic Gradient Descent

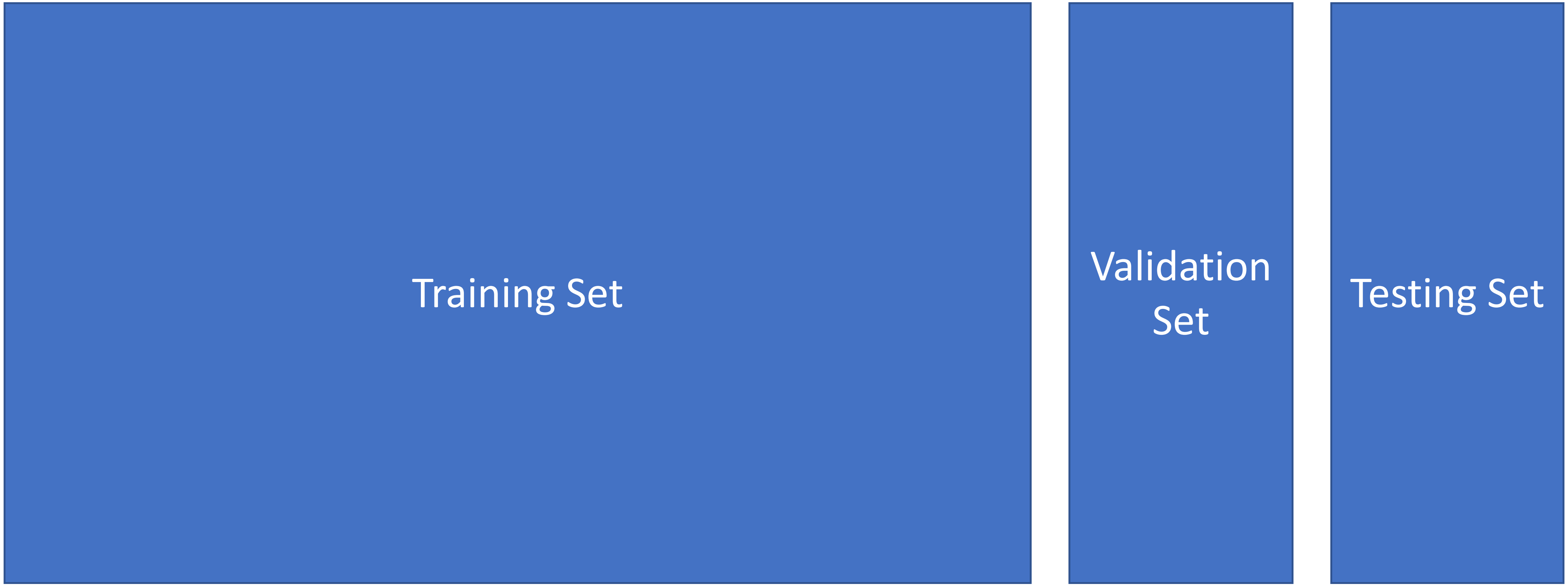
- How to choose the right batch size B?
- How to choose the right learning rate lambda?
- How to choose the right loss function, e.g. is least squares good enough?
- How to choose the right function/classifier, e.g. linear, quadratic, neural network with 1 layer, 2 layers, etc?

Linear Regression

Example: Hollywood movie data

	input variables x					output variables y	
	production costs	**promotional costs**	**genre of the movie**	**box office first week**	**total book sales**	**total revenue USA**	**total revenue international**
training data	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$x_4^{(1)}$	$x_5^{(1)}$	$y_1^{(1)}$	$y_2^{(1)}$
	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	$x_4^{(2)}$	$x_5^{(2)}$	$y_1^{(2)}$	$y_2^{(2)}$
	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	$x_4^{(3)}$	$x_5^{(3)}$	$y_1^{(3)}$	$y_2^{(3)}$
test data	$x_1^{(4)}$	$x_2^{(4)}$	$x_3^{(4)}$	$x_4^{(4)}$	$x_5^{(4)}$	$y_1^{(4)}$	$y_2^{(4)}$
	$x_1^{(5)}$	$x_2^{(5)}$	$x_3^{(5)}$	$x_4^{(5)}$	$x_5^{(5)}$	$y_1^{(5)}$	$y_2^{(5)}$

Training, Validation (Dev), Test Sets



Training, Validation (Dev), Test Sets

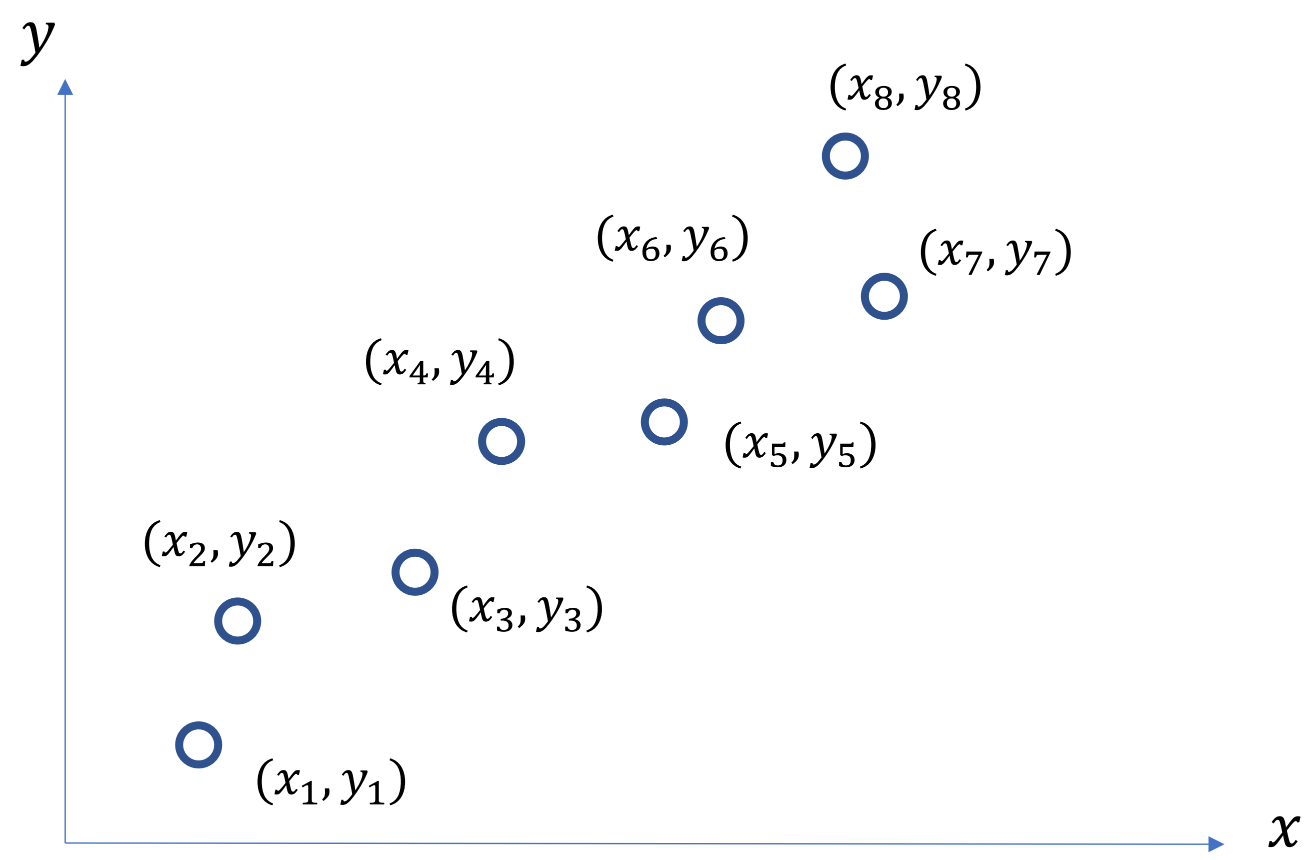
Used during development

Training, Validation (Dev), Test Sets

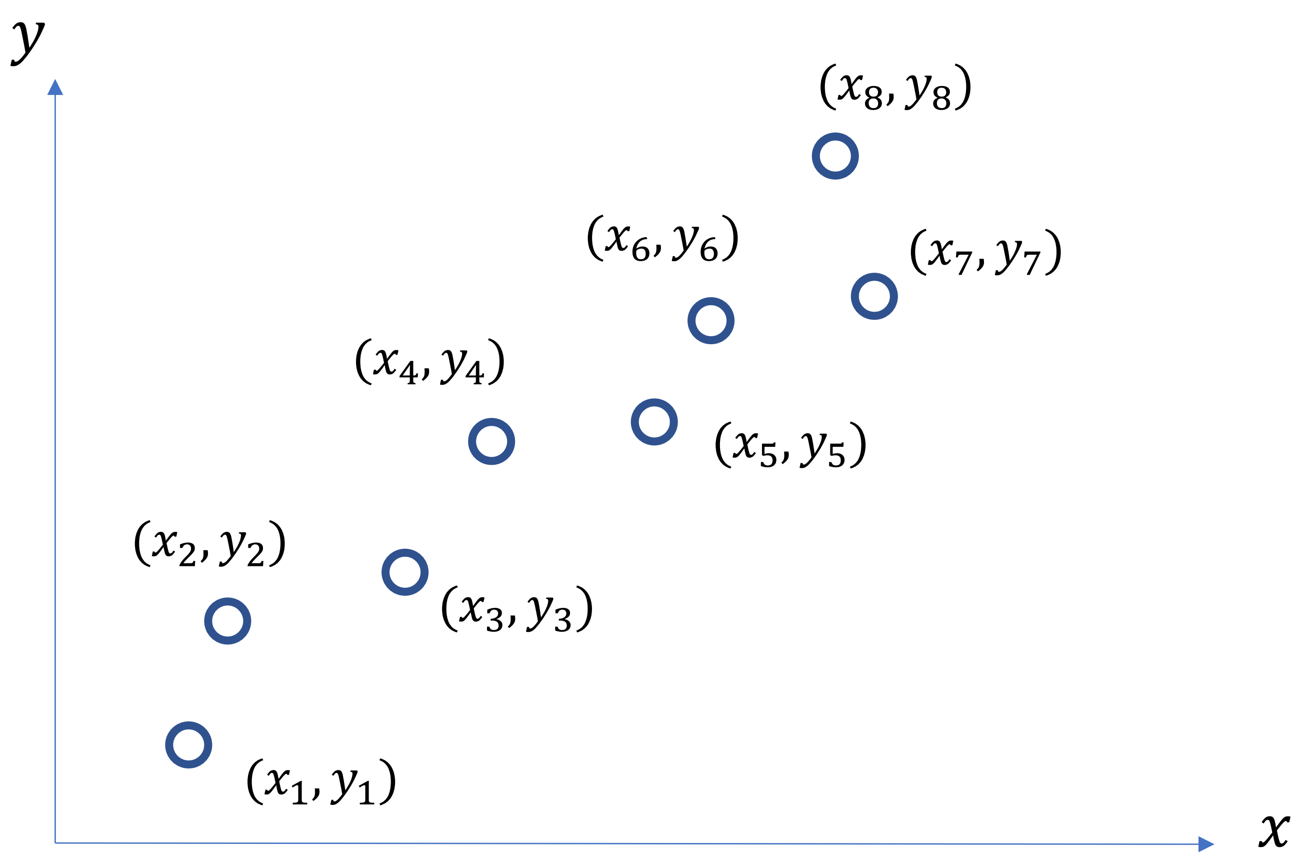
Only to be used for evaluating the model at the very end of development and any changes to the model after running it on the test set, could be influenced by what you saw happened on the test set, which would invalidate any future evaluation.

How to pick the right model?

Linear Regression – 1 output, 1 input

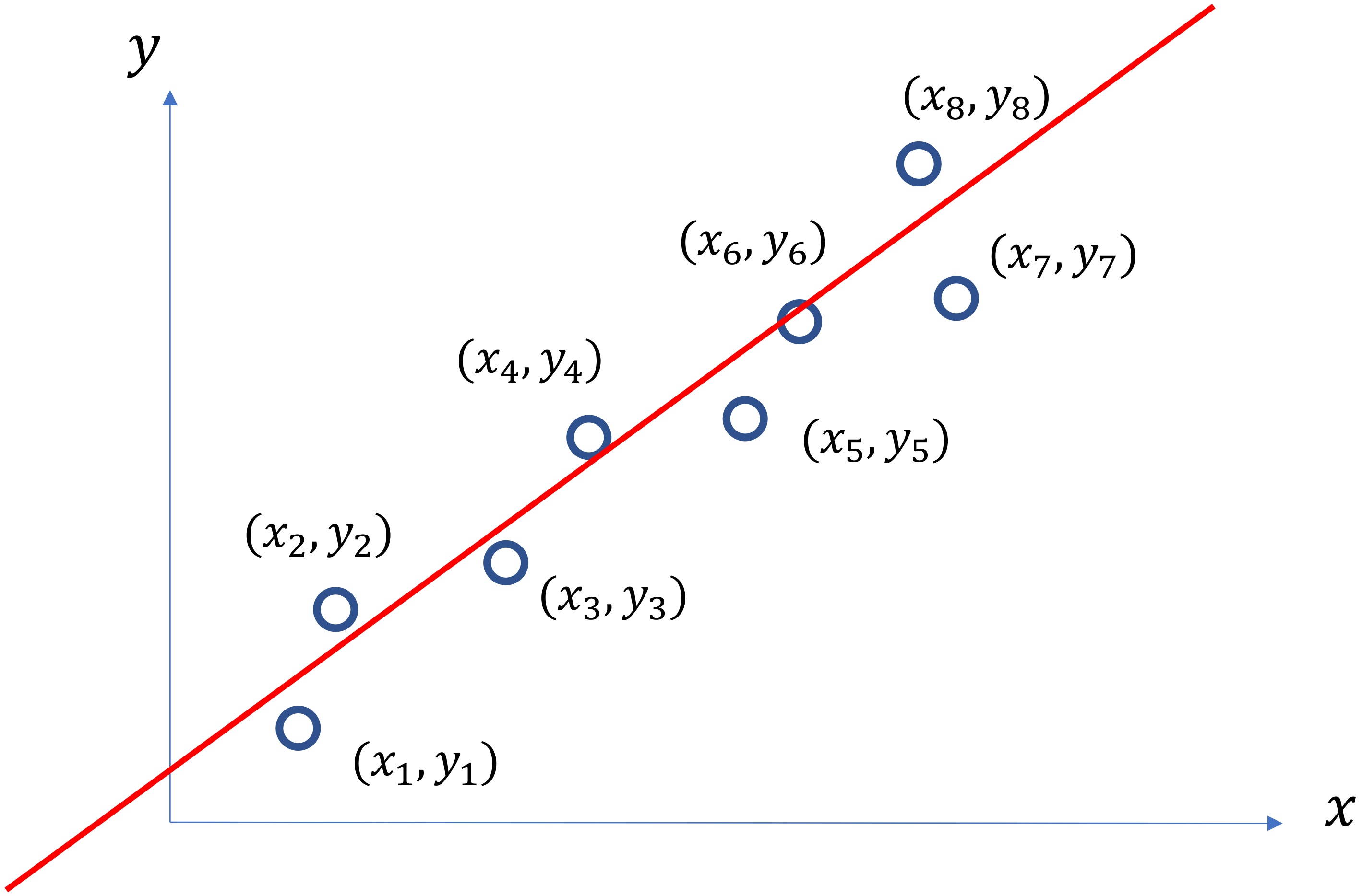


Linear Regression – 1 output, 1 input



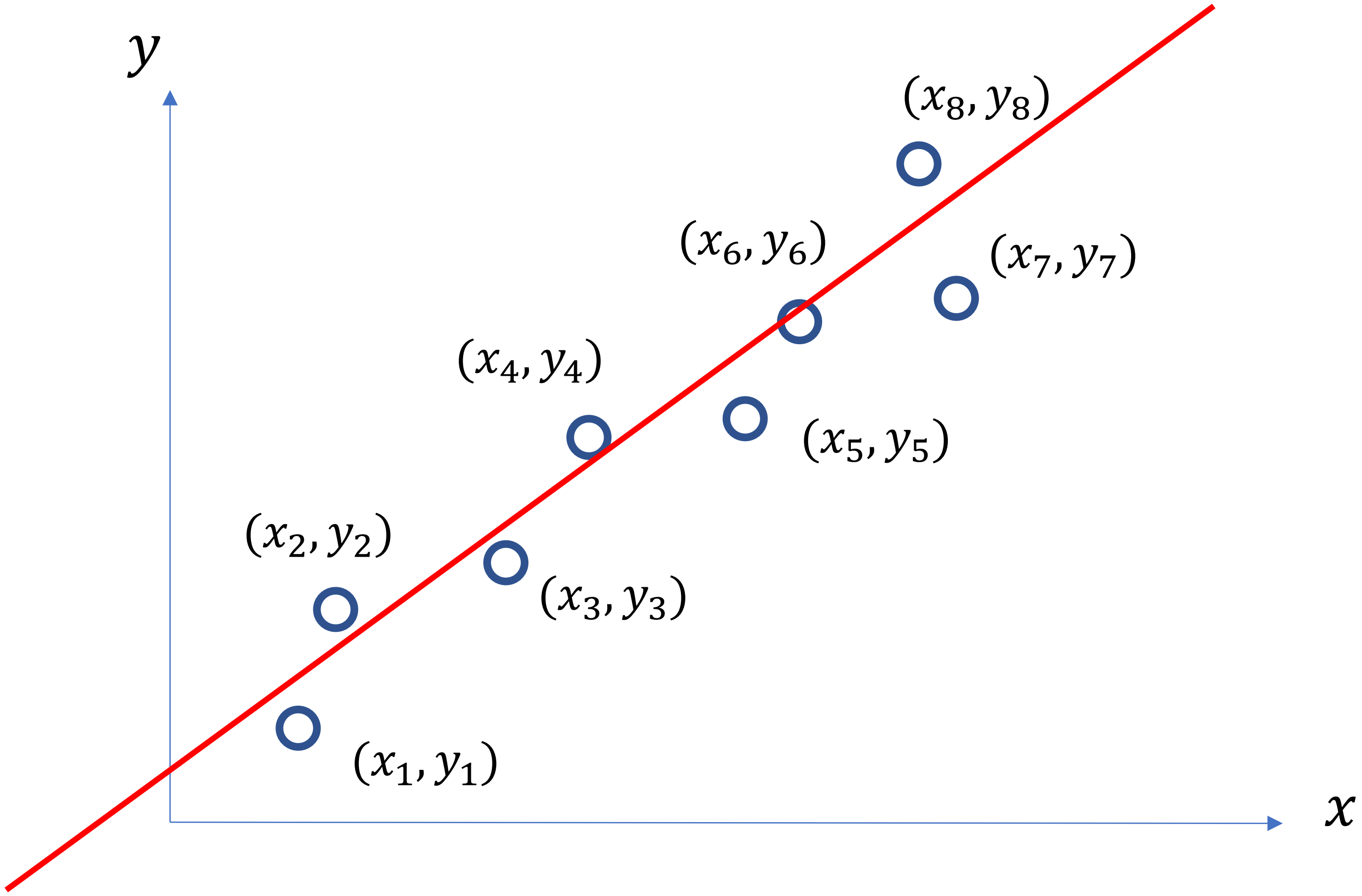
Model: $\hat{y} = wx + b$

Linear Regression – 1 output, 1 input



Model: $\hat{y} = wx + b$

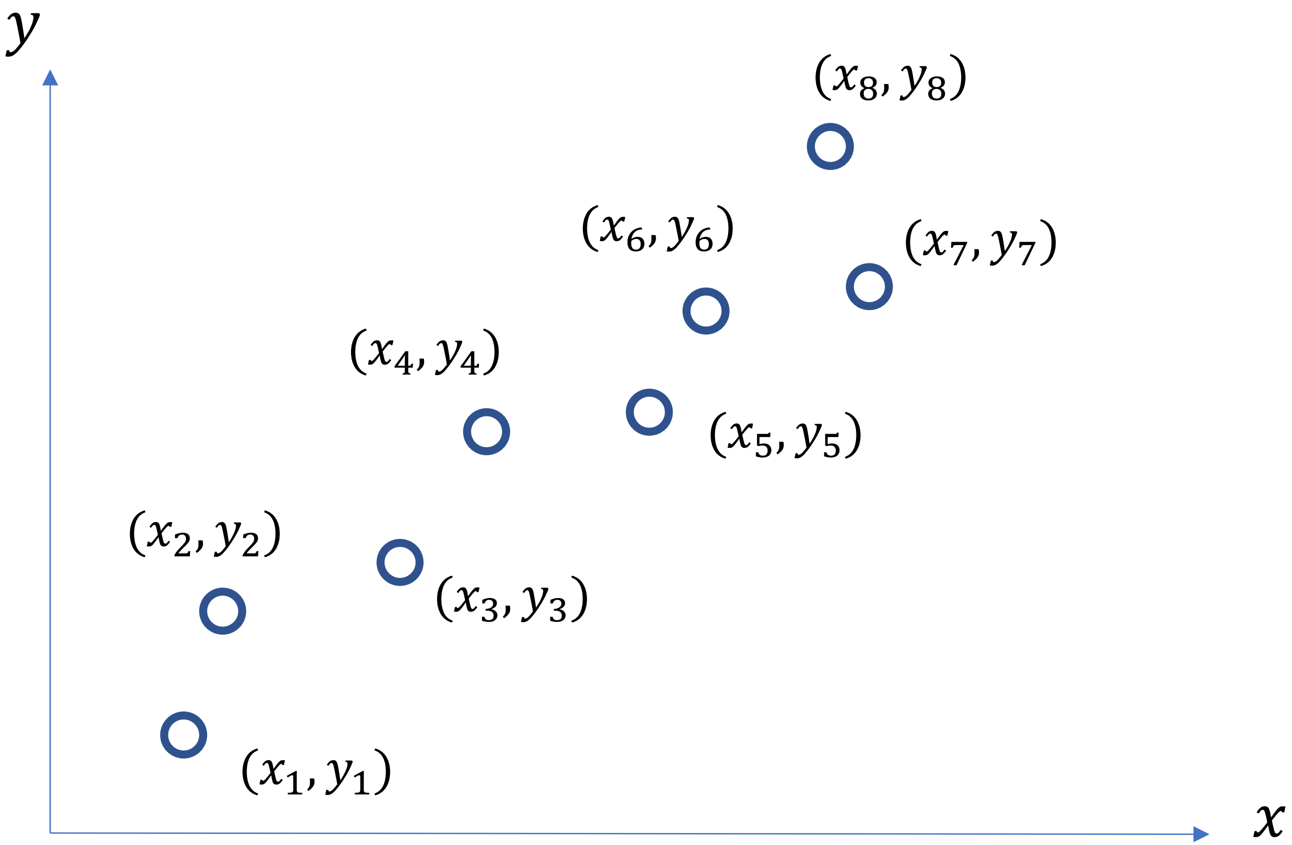
Linear Regression – 1 output, 1 input



Model: $\hat{y} = wx + b$

Loss: $L(w, b) = \sum_{i=1}^{i=8} (\hat{y}_i - y_i)^2$

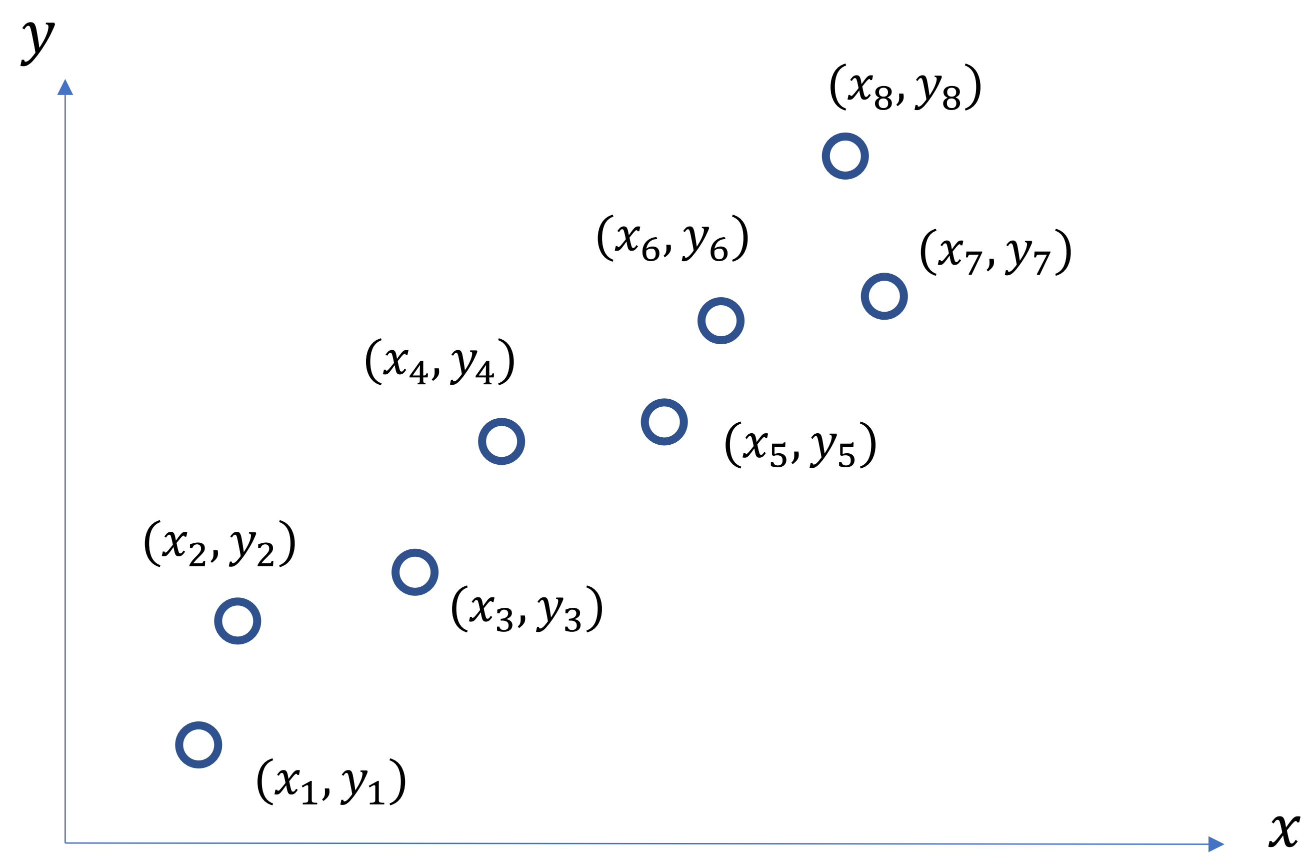
Quadratic Regression



Model: $\hat{y} = w_1 x^2 + w_2 x + b$

Loss: $L(w, b) = \sum_{i=1}^{i=8} (\hat{y}_i - y_i)^2$

n-polynomial Regression

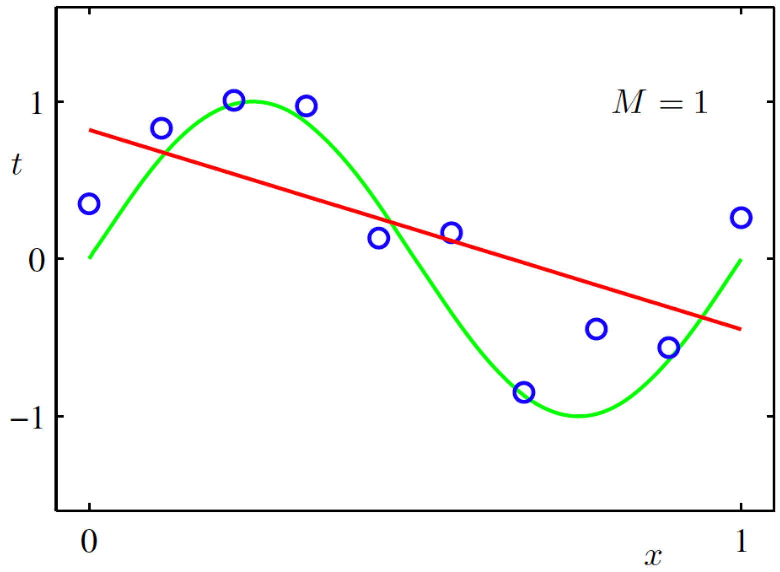


Model: $\hat{y} = w_n x^n + \cdots + w_1 x + b$

Loss: $L(w, b) = \sum_{i=1}^{i=8} (\hat{y}_i - y_i)^2$

Overfitting

f is linear

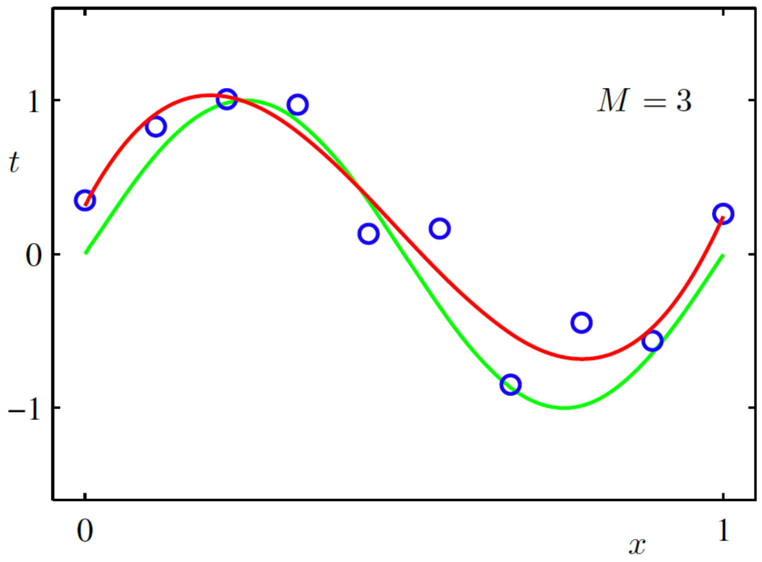


$Loss(w)$ is high

Underfitting

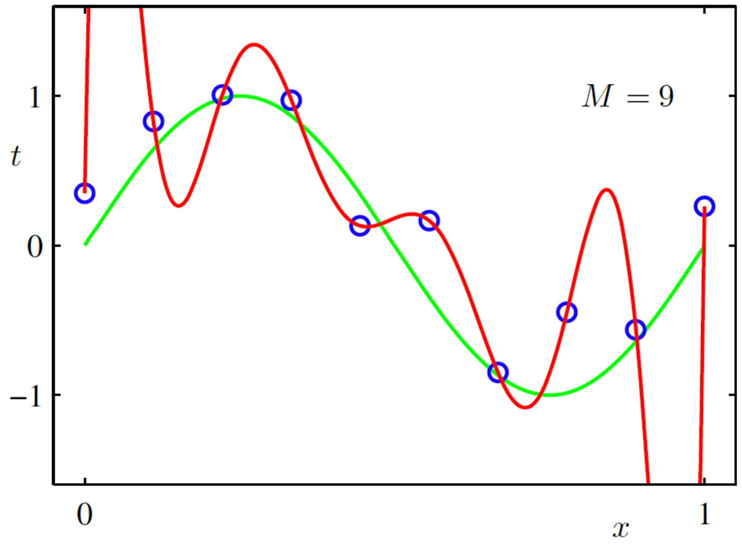
High Bias

f is cubic



$Loss(w)$ is low

f is a polynomial of degree 9



$Loss(w)$ is zero!

Overfitting

High Variance

Christopher M. Bishop – Pattern Recognition and Machine Learning

Questions?