EECS 230 Deep Learning
Lecture 2: Machine Learning

Some slides from Olga Veksler and Simon Prince

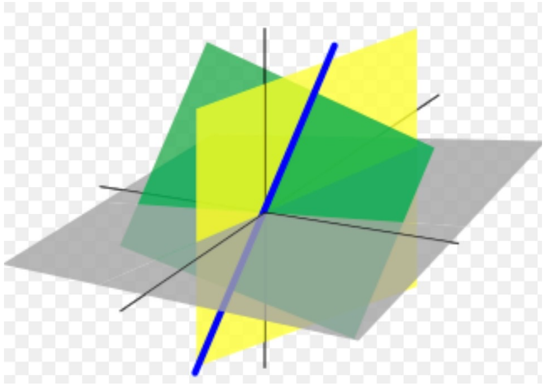
Linear Algebra

What is linear algebra?

- Linear algebra is the branch of mathematics concerning linear equations such as

$$a_1x_1+\ldots..+a_nx_n=b$$

  - In vector notation we say $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}=b$
  - Called a linear transformation of $\boldsymbol{x}$
- Linear algebra is fundamental to geometry, for defining objects such as lines, planes, rotations



Linear equation $a_1x_1+\ldots..+a_nx_n=b$ defines a plane in $(x_1,..,x_n)$ space
Straight lines define common solutions to equations

Linear Algebra Topics

- Scalars, Vectors, Matrices and Tensors
- Multiplying Matrices and Vectors
- Identity and Inverse Matrices
- Linear Dependence and Span
- Norms
- Special kinds of matrices and vectors
- Eigendecomposition
- Singular value decomposition
- The Moore Penrose pseudoinverse
- The trace operator
- The determinant
- Ex: principal components analysis

Slide from S. Srihari

Scalar

- Single number
  - In contrast to other objects in linear algebra, which are usually arrays of numbers
- Represented in lower-case italic x
  - They can be real-valued or be integers
    - E.g., let $x \in \mathbb{R}$ be the slope of the line
      - Defining a real-valued scalar
    - E.g., let $n \in \mathbb{N}$ be the number of units
      - Defining a natural number scalar

Vector

- An array of numbers arranged in order
- Each no. identified by an index
- Written in lower-case bold such as $\boldsymbol{x}$
  - its elements are in italics lower case, subscripted

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- If each element is in R then $\boldsymbol{x}$ is in R
- We can think of vectors as points in space

Slide from S. Srihari

Matrices

- 2-D array of numbers
  - So each element identified by two indices
- Denoted by bold typeface $\boldsymbol{A}$
  - Elements indicated by name in italic but not bold
    - $A_{1,1}$ is the top left entry and $A_{m,n}$ is the bottom right entry
    - We can identify nos in vertical column *j* by writing : for the horizontal coordinate
    - E.g., $\boldsymbol{A} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$
    - $A_{i:}$ is i^{th} row of A, $A_{:j}$ is j^{th} column of $\boldsymbol{A}$
- If A has shape of height m and width n with real-values then $A \in R^{m \times n}$

Slide from S. Srihari

Tensor

- Sometimes need an array with more than two axes
  - E.g., an RGB color image has three axes
- A tensor is an array of numbers arranged on a regular grid with variable number of axes
  - See figure next
- Denote a tensor with this bold typeface: $\mathbf{A}$
- Element (i,j,k) of tensor denoted by $\mathrm{A}_{i,j,k}$

Multiplying matrices

- For product $\boldsymbol{C}=\boldsymbol{AB}$ to be defined, $\boldsymbol{A}$ has to have the same no. of columns as the no. of rows of $\boldsymbol{B}$
  - If $\boldsymbol{A}$ is of shape mxn and B is of shape nxp then *matrix product* $\boldsymbol{C}$ is of shape mxp

$$C = AB \Rightarrow C_{i,j} = \sum_k A_{i,k} B_{k,j}$$

  - Note that the standard product of two matrices is not just the product of two individual elements
    - Such a product does exist and is called the element-wise product or the Hadamard product $\boldsymbol{A} \circ \boldsymbol{B}$

Slide from S. Srihari

Linear transformation

- $\boldsymbol{Ax=b}$
  - where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{b} \in \mathbb{R}^{n}$
  - More explicitly

$$\begin{aligned} A_{11}x_1 + A_{12}x_2 + \dots + A_{1n}x_n &= b_1 \\ A_{21}x_1 + A_{22}x_2 + \dots + A_{2n}x_n &= b_2 \\ \\ A_{n1}x_1 + A_{m2}x_2 + \dots + A_{n,n}x_n &= b_n \end{aligned}$$

n equations in
n unknowns

$$A = \begin{bmatrix} A_{1,1} & . & A_{1,n} \\ & & \\ A_{n,1} & . & A_{nn} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \\ x_n \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ \\ b_n \end{bmatrix}$$

n x n $\qquad$ n x 1 $\qquad$ n x 1

Can view A as a *linear transformation* of vector $\boldsymbol{x}$ to vector $\boldsymbol{b}$

- Sometimes we wish to solve for the unknowns $\boldsymbol{x} = \{x_1, \dots, x_n\}$ when A and $\boldsymbol{b}$ provide constraints

Slide from S. Srihari

Matrix inverse

- Inverse of square matrix A defined as $A^{-1}A = I_n$
- We can now solve $A\boldsymbol{x}=\boldsymbol{b}$ as follows:

$$
\begin{aligned}
&A\boldsymbol{x} = \boldsymbol{b} \\
&A^{-1}A\boldsymbol{x} = A^{-1}\boldsymbol{b} \\
&I_n\boldsymbol{x} = A^{-1}\boldsymbol{b} \\
&\boldsymbol{x} = A^{-1}\boldsymbol{b}
\end{aligned}
$$

- This depends on being able to find A^{-1}
- If A^{-1} exists there are several methods for finding it

Slide from S. Srihari

Norms

- Used for measuring the size of a vector
- Norms map vectors to non-negative values
- Norm of vector $\boldsymbol{x}=[x_1,..,x_n]^T$ is distance from origin to $\boldsymbol{x}$
  - It is any function f that satisfies:

$$f(\boldsymbol{x}) = 0 \Rightarrow \boldsymbol{x} = 0$$

$$f(\boldsymbol{x}+\boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y}) \quad \text{Triangle Inequality}$$

$$\forall \alpha \in R \quad f(\alpha \boldsymbol{x}) = |\alpha| f(\boldsymbol{x})$$

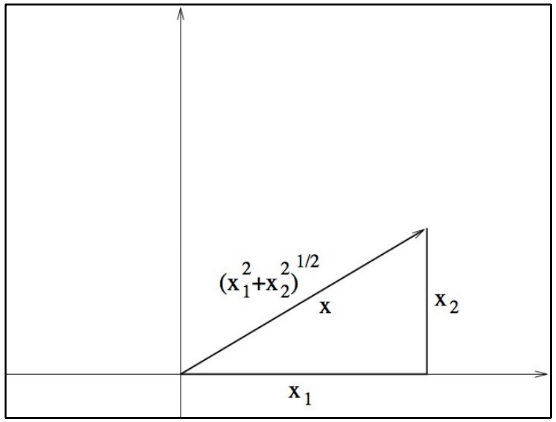
UC MERCED

Slide from S. Srihari

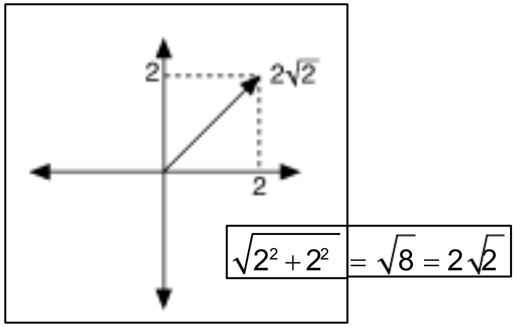
L^p Norm

- Definition:

$$\|\boldsymbol{x}\|_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$$



  - L^2 Norm
    - Called Euclidean norm
      - Simply the Euclidean distance between the origin and the point $\boldsymbol{x}$
      - written simply as $||\boldsymbol{x}||$
      - Squared Euclidean norm is same as $\boldsymbol{x}^T\boldsymbol{x}$



  - L^1 Norm
    - Useful when 0 and non-zero have to be distinguished
      - Note that L^2 increases slowly near origin, e.g., $0.1^2=0.01$)
  - L^∞ Norm

$$\|\boldsymbol{x}\|_\infty = \max_i |x_i|$$

- Called max norm

Slide from S. Srihari

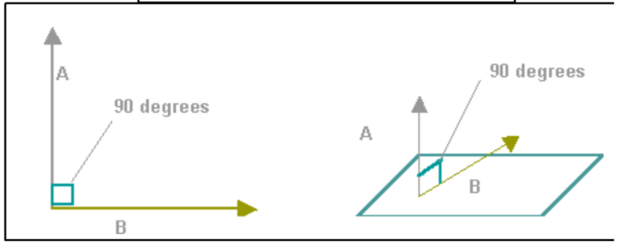
Special kind of vectors

- Unit Vector
  - A vector with unit norm

$$\|x\|_2 = 1$$

- Orthogonal Vectors

$$\begin{bmatrix} 2 \\ -3 \\ -2 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 4 \\ -2 \\ 7 \end{bmatrix}$$



  - A vector ***x*** and a vector ***y*** are orthogonal to each other if $\boldsymbol{x}^T\boldsymbol{y}=0$
    - If vectors have nonzero norm, vectors at 90 degrees to each other
  - Orthonormal Vectors
    - Vectors are orthogonal & have unit norm
    - Orthogonal Matrix
      - A square matrix whose rows are mutually
      - orthonormal: $A^TA=AA^T=I$

        $A^{-1}=A^T$

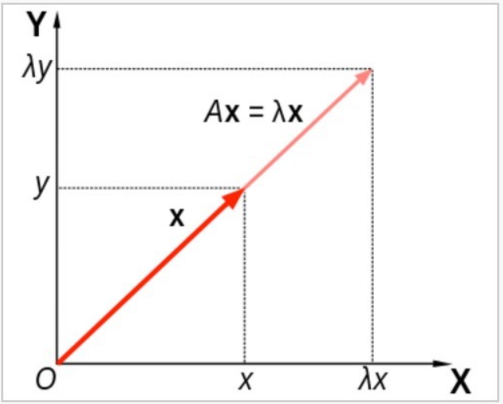
Slide from S. Srihari

Eigenvector

- An eigenvector of a square matrix $\boldsymbol{A}$ is a non-zero vector $\boldsymbol{v}$ such that multiplication by $\boldsymbol{A}$ only changes the scale of $\boldsymbol{v}$

$$\boldsymbol{Av} = \lambda \boldsymbol{v}$$

  - The scalar λ is known as eigenvalue
- If $\boldsymbol{v}$ is an eigenvector of $\boldsymbol{A}$, so is any rescaled vector $s\boldsymbol{v}$. Moreover $s\boldsymbol{v}$ still has the same eigen value. Thus look for a unit eigenvector



Matrix A acts by stretching the vector x, not changing its direction, so x is an eigenvector of A.

Wikipedia

Slide from S. Srihari

Eigendecomposition

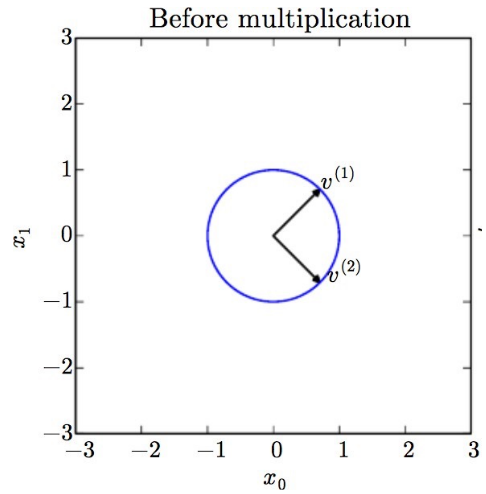
- Suppose that matrix A has n linearly independent eigenvectors $\{\boldsymbol{v}^{(1)},..,\boldsymbol{v}^{(n)}\}$ with eigenvalues $\{\lambda_1,..,\lambda_n\}$
- Concatenate eigenvectors to form matrix V
- Concatenate eigenvalues to form vector
  $\lambda=[\lambda_1,..,\lambda_n]$
- Eigendecomposition of A is given by

$$A=V\mathrm{diag}(\lambda)V^{-1}$$

Effect of eigenvalue and eigenvector

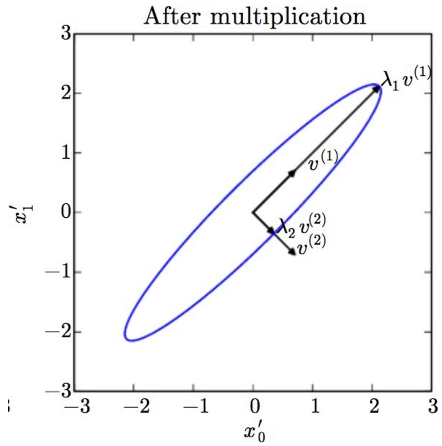
- Example of 2×2 matrix
- Matrix A with two orthonormal eigenvectors
  - – $v^{(1)}$ with eigenvalue λ_1, $v^{(2)}$ with eigenvalue λ_2

Plot of unit vectors $\boldsymbol{u} \in \mathbb{U}^2$ (circle)



with two variables x_1 and x_2

Plot of vectors $A\boldsymbol{u}$ (ellipse)



Slide from S. Srihari

Positive Semidefinite Matrix (PSD)

- A matrix whose eigenvalues are all positive is called *positive definite*
  - Positive or zero is called *positive semidefinite*
- If eigen values are all negative it is *negative definite*
  - Positive definite matrices guarantee that $\boldsymbol{x}^T A \boldsymbol{x} \geq 0$

Slide from S. Srihari

Singular Value Decomposition (SVD)

- Eigendecomposition has form: $A=V\text{diag}(\lambda)V^{-1}$
  - If A is not square, eigendecomposition is undefined
- SVD is a decomposition of the form $A=UDV^T$
- SVD is more general than eigendecomposition
  - Used with any matrix rather than symmetric ones
  - Every real matrix has a SVD
    - Same is not true of eigen decomposition

Slide from S. Srihari

Probability and Statistics

Probability and Statistics

- Probability Theory
  - A mathematical framework for representing uncertain statements
  - Provides a means of quantifying uncertainty and axioms for deriving new uncertain statements
- Use of probability theory in artificial intelligence
  - 1.Tells us how AI systems should reason
    - So we design algorithms to compute or approximate various expressions using probability theory

  2.Theoretically analyze behavior of AI systems

UCMERCED

Slide from S. Srihari

Random Variable

- Variable that can take different values randomly
- Scalar random variable denoted x
- Vector random variable is denoted in bold as
- Values of r.v.s denoted in italics x or $\boldsymbol{x}$
  - Values denoted as $\mathrm{Val(x)}=\{x_1,x_2\}$
- Random variable must has a probability distribution to specify how likely the states are
- Random variables can be discrete or continuous
  - Discrete values need not be integers, can be named states
  - Continuous random variable is associated with a real value

UCMERCED

Slide from S. Srihari

Probability Distribution

- A probability distribution is a description of how likely a random variable or a set of random variables is to take each of its possible states
- The way to describe the distribution depends on whether it is discrete or continuous

Continuous Variables and PDFs

- When working with continuous variables, we describe probability distributions using probability density functions
- To be a pdf p must satisfy:

- The domain of p must be the set of all possible states of x.
- $\forall x \in \mathrm{x}, p(x) \geq 0$. Note that we do not require $p(x) \leq 1$.
- $\int p(x)dx = 1$.

Slide from S. Srihari

Marginal distribution

- Sometimes we know the joint distribution of several variables
- And we want to know the distribution over some of them
- It can be computed using

$$\forall x \in \mathrm{x}, P(\mathrm{x} = x) = \sum_{y} P(\mathrm{x} = x, \mathrm{y} = y)$$

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

Slide from S. Srihari

Conditional probability

- We are often interested in the probability of an event given that some other event has happened
- This is called conditional probability
- It can be computed using

$$P(\mathrm{y} = y \mid \mathrm{x} = x) = \frac{P(\mathrm{y} = y, \mathrm{x} = x)}{P(\mathrm{x} = x)}.$$

Chain rule of conditional probability

- Any probability distribution over many variables can be decomposed into conditional distributions over only one variable

$$P(\mathrm{x}^{(1)}, \ldots, \mathrm{x}^{(n)}) = P(\mathrm{x}^{(1)}) \Pi_{i=2}^{n} P(\mathrm{x}^{(i)} \mid \mathrm{x}^{(1)}, \ldots, \mathrm{x}^{(i-1)})$$

- An example with three variables

$$\begin{aligned} P(\mathrm{a}, \mathrm{b}, \mathrm{c}) &= P(\mathrm{a} \mid \mathrm{b}, \mathrm{c}) P(\mathrm{b}, \mathrm{c}) \\ P(\mathrm{b}, \mathrm{c}) &= P(\mathrm{b} \mid \mathrm{c}) P(\mathrm{c}) \\ P(\mathrm{a}, \mathrm{b}, \mathrm{c}) &= P(\mathrm{a} \mid \mathrm{b}, \mathrm{c}) P(\mathrm{b} \mid \mathrm{c}) P(\mathrm{c}) \end{aligned}$$

Slide from S. Srihari

Independence and conditional independence

- Independence: $\boxed{\mathrm{x} \perp \mathrm{y}}$
  - Two variables x and y are independent if their probability distribution can be expressed as a product of two factors, one involving only x and the other involving only y

$$\forall x \in \mathrm{x}, y \in \mathrm{y},\ p(\mathrm{x}=x, \mathrm{y}=y) = p(\mathrm{x}=x)p(\mathrm{y}=y)$$

- Conditional Independence: $\boxed{\mathrm{x} \perp \mathrm{y} \mid \mathrm{z}}$
  - Two variables x and y are independent given variable z, if the conditional probability distribution over x. and y factorizes in this way for every z

$$\forall x \in \mathrm{x}, y \in \mathrm{y}, z \in \mathrm{z},\ p(\mathrm{x}=x, \mathrm{y}=y \mid \mathrm{z}=z) = p(\mathrm{x}=x \mid \mathrm{z}=z)p(\mathrm{y}=y \mid \mathrm{z}=z)$$

Slide from S. Srihari

Common probability distribution

- Several simple probability distributions are useful in may contexts in machine learning
  - Bernoulli over a single binary random variable
  - Multinoulli distribution over a variable with k states
  - Gaussian distribution
  - Mixture distribution

Mixture of Distribution

- A mixture distribution is made up of several component distributions
- On each trial, the choice of which component distribution generates the sample is determined by sampling a component identity from a multinoulli distribution:

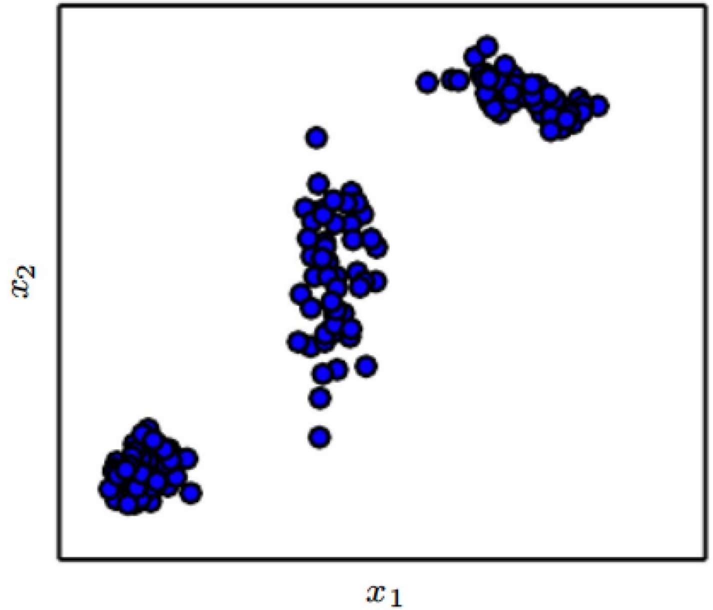
$$P(\mathrm{x}) = \sum_i P(\mathrm{c} = i) P(\mathrm{x} \mid \mathrm{c} = i)$$

  – where $P(\mathrm{c})$ is a multinoulli distribution

Slide from S. Srihari

Gaussian mixture model

- Components $p(\mathrm{x}|\mathrm{c}=i)$ are Gaussian
- Each component has a separately parameterized mean $\mu^{(i)}$ and covariance $\Sigma^{(i)}$
- Any smooth density can be approximated with enough components
- Samples from a GMM:
  - 3 components



Slide from S. Srihari

Bayes's rule

- **Bayes' theorem** (alternatively **Bayes' law** or **Bayes' rule**), named after Thomas Bayes, describes the probability of an event, based on prior knowledge of conditions that might be related to the event.
- For example, if the risk of health problems is known to increase with age, Bayes' theorem allows the risk to an individual of a known age to be assessed more accurately by conditioning it relative to their age.

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A) * P(B|A)}{P(B)}$$

Machine Learning

Major Types of machine learning

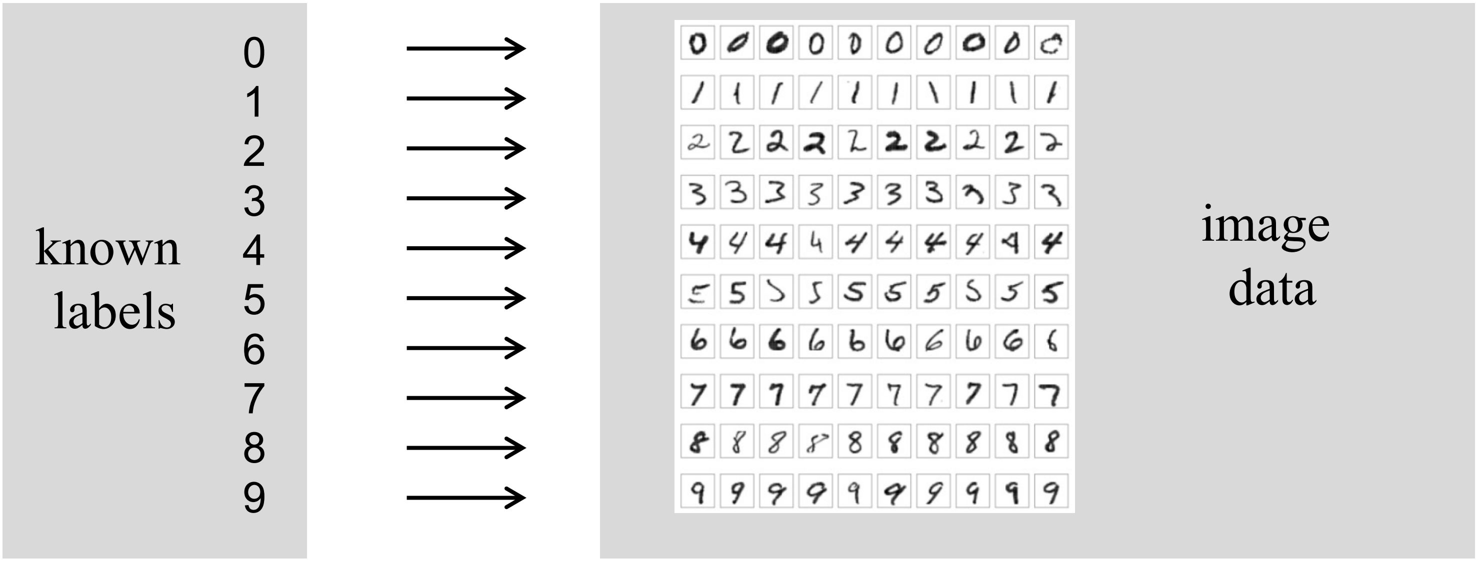
- Supervised learning: Given pairs of input-output, learn to map the input to output
  - Image classification
  - Speech recognition
  - Regression (continuous output)
- Unsupervised learning: Given unlabeled data, uncover the underlying structure or distribution of the data
  - Clustering
  - Dimensionality reduction
- Reinforcement learning: training an agent to make decisions within an environment to maximize a cumulative reward
  - Game playing (e.g., AlphaGo)
  - Robot control

Subtypes of supervised ML

- Classification
  - output belongs to a finite set
  - example: age $\in$ {baby, child, adult, elder}
  - output is also called class or label
- Regression
  - output is continuous
  - examples: age $\in [0,130]$
- Difference mostly in design of loss functions

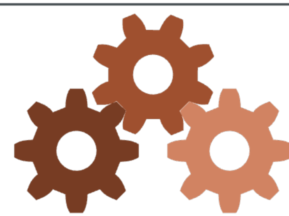
Example: supervised digit recognition

- ❑Easy to collect images of digits with their correct labels



- ❑ML algorithm can use collected data to produce a program for recognizing previously unseen images of digits

Example: Regression

Real world input	Model input	Model	Model output	Real world output
6000 square feet, 4 bedrooms, previously sold for $235K in 2005, 1 parking spot.	$\begin{bmatrix} 6000 \\ 4 \\ 235 \\ 2005 \\ 1 \end{bmatrix}$	 Supervised learning model	$[340]$	Predicted price is $340k

Supervised ML

- We are given
  1. Training examples $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$
  2. Target output for each sample $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^n$

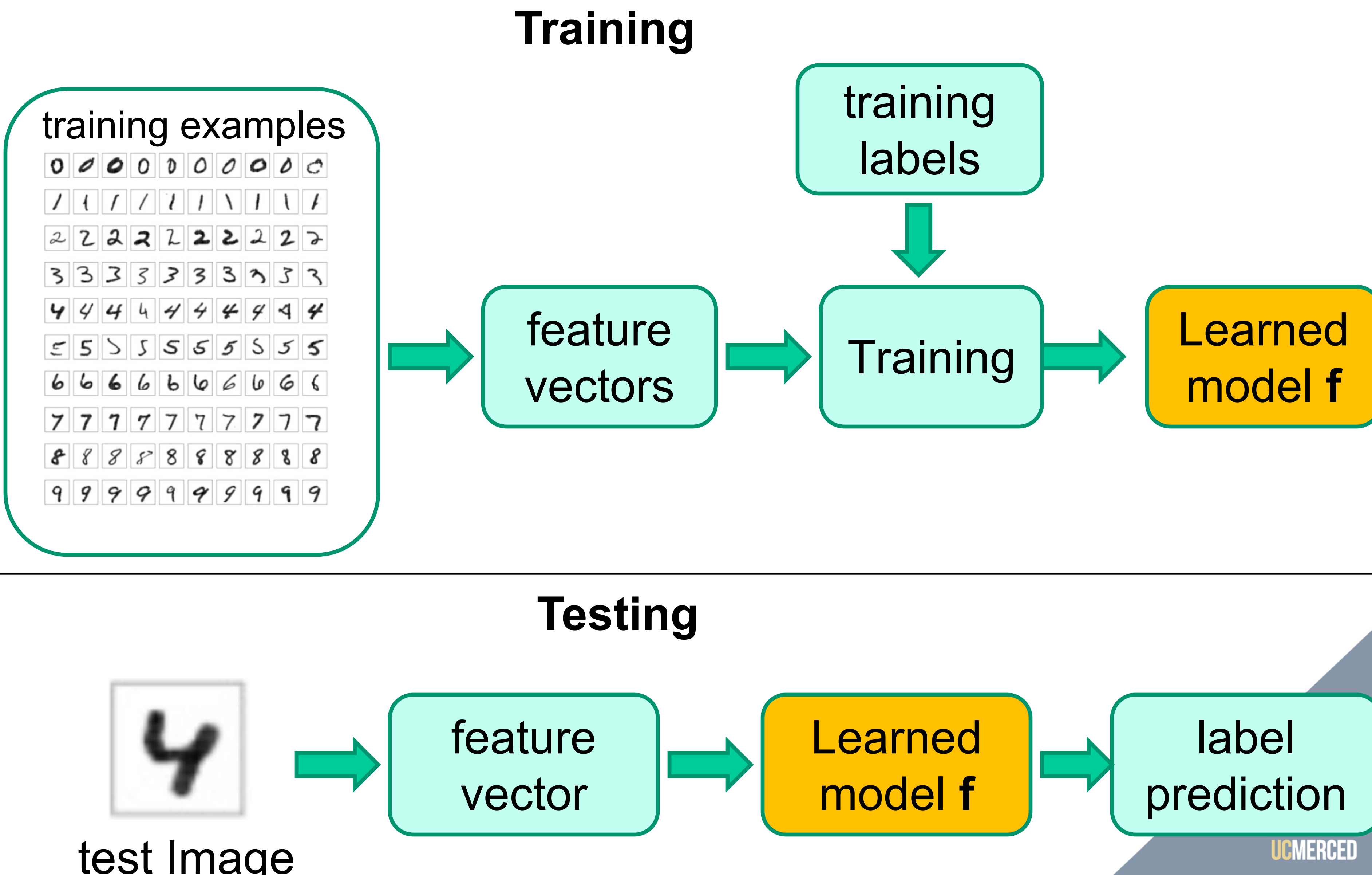
  labeled data

- **Training phase**
  - estimate function $\mathbf{y} = \mathbf{f}(\mathbf{x})$ from labeled data

    where $\mathbf{f}(\mathbf{x})$ is called *classifier*, *learning machine*, *prediction function*, etc.

- **Testing phase** (deployment)
  - predict output $\mathbf{f}(\mathbf{x})$ for a new (unseen) sample $\mathbf{x}$

Training/Testing Phases Illustrated



UCMERCED

Training phase as parameter estimation

- Estimate prediction function $y = f(x)$ from labeled data

Typically, search for f is limited to some type/group of functions ("*hypothesis space*") parameterized by *weights* $\mathbf{w}$ that must be estimated

$$f_w(x) \quad \text{or} \quad f(w, x) \qquad\qquad w = ?$$

Goal: find classifier parameters (weights) w so that $f(w, xi) = y^i$ "as much as possible" for all training examples,

Loss function

- Training dataset of I pairs of input/output examples

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}$$

- Loss function or cost function measures how bad model is:

$$\boxed{\mathbf{w}^* = \arg\min_{\mathbf{w}} \; \Sigma_{\mathbf{i}} \; L(\mathbf{y}^{\mathbf{i}}, \mathbf{f}(\mathbf{w}, \mathbf{x}^{\mathbf{i}}))}$$

- $\boldsymbol{\phi}$ is also a common notation for weights

Example: 1D Linear regression

- Model:

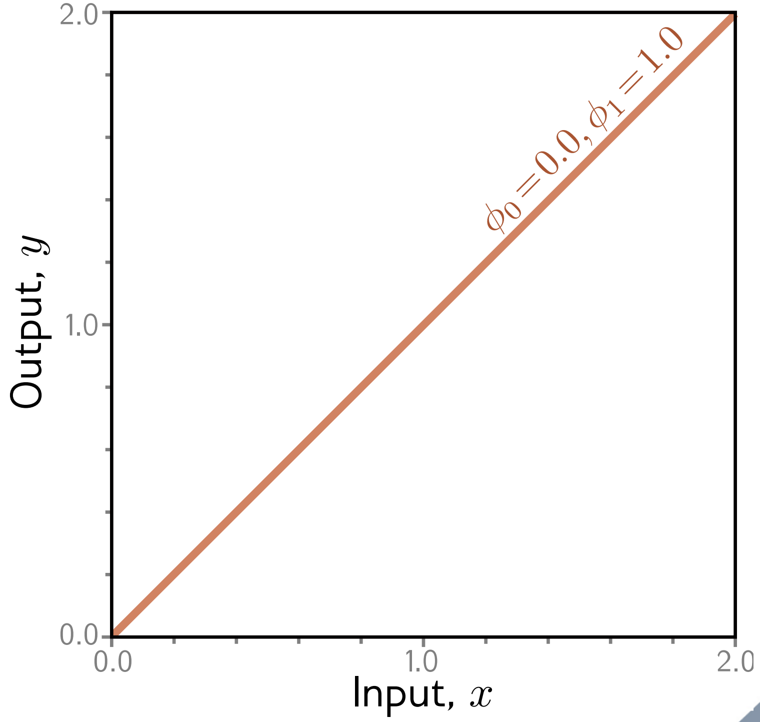
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}$$

y-offset

slope



Example: 1D Linear regression

- Model:

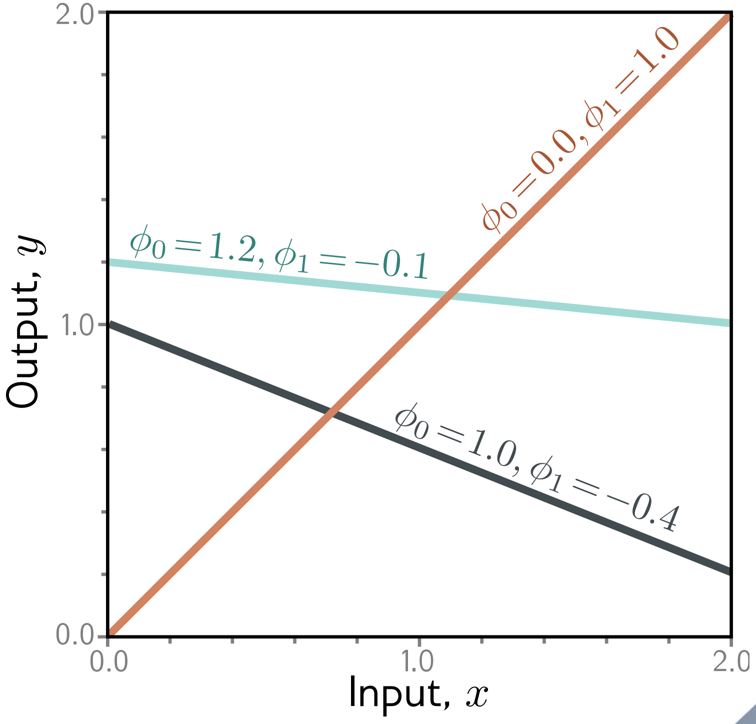
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

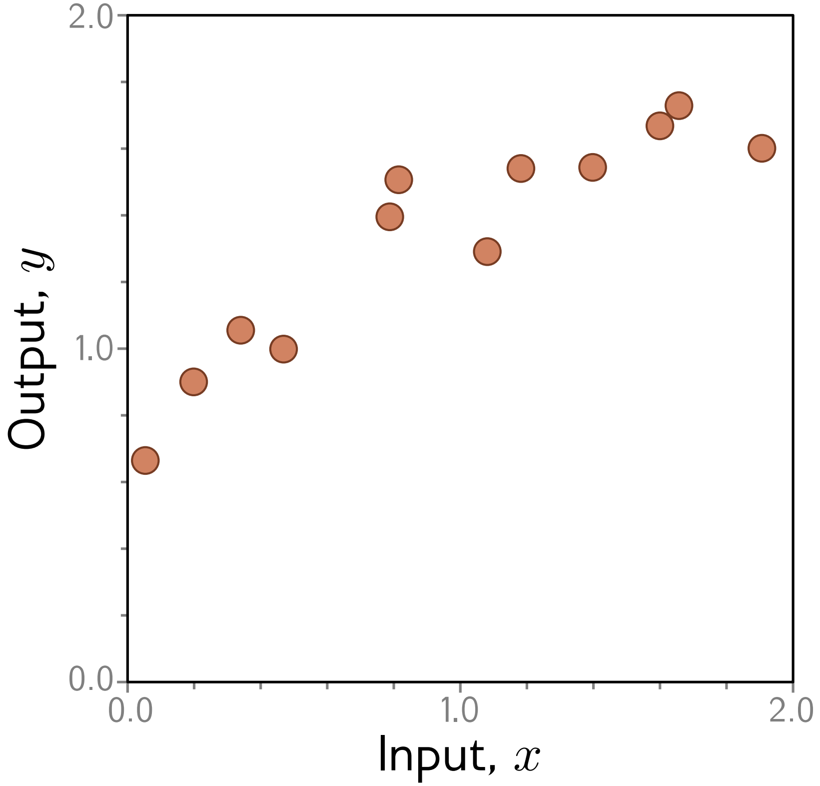
$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}$$

← y-offset

← slope



Example: 1D Linear regression training data

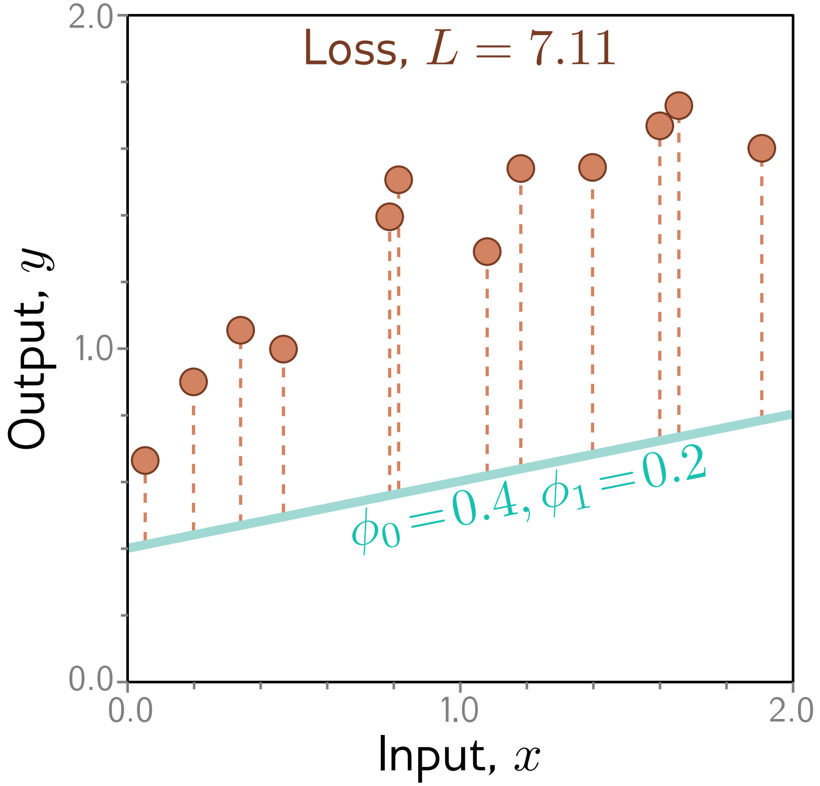


Loss function:

$$
\begin{aligned}
L[\boldsymbol{\phi}] &= \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2 \\
&= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2
\end{aligned}
$$

"Least squares loss function"

Example: 1D Linear regression training data



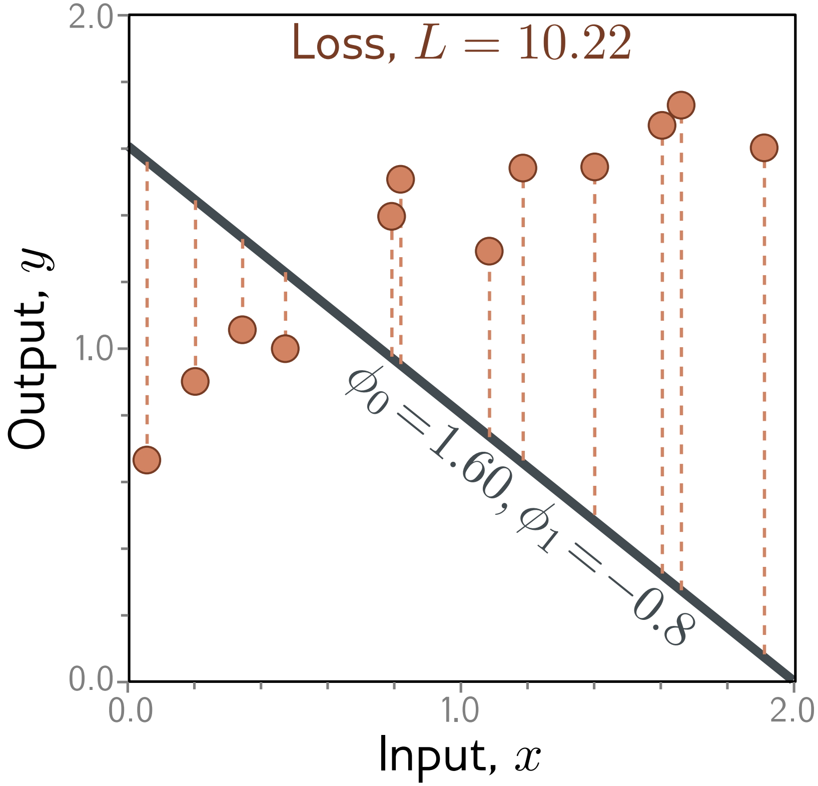
Loss function:

$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

"Least squares loss function"

Example: 1D Linear regression training data



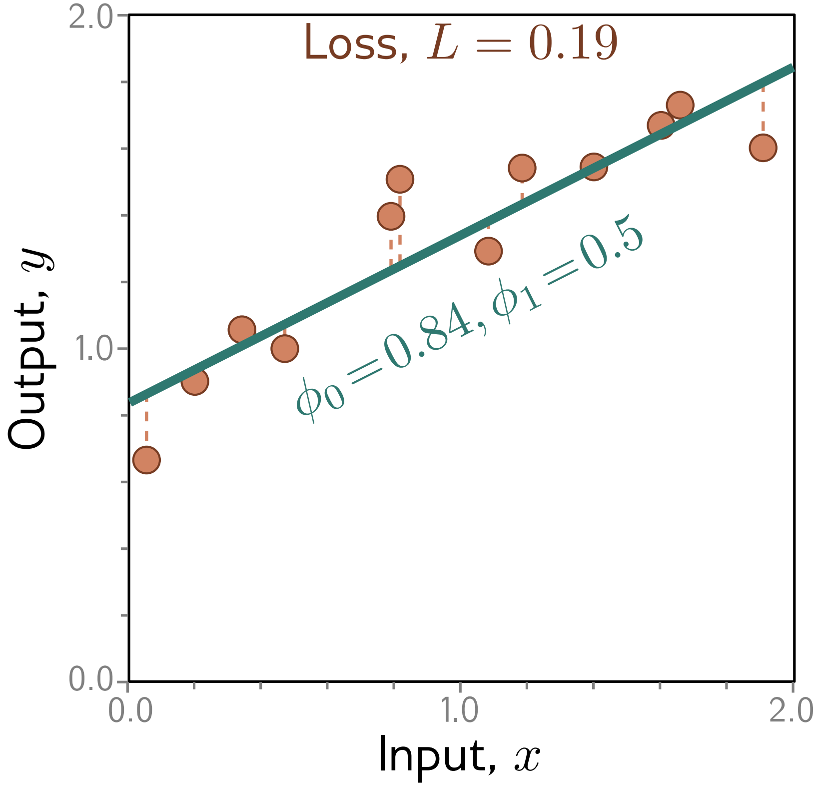
Loss function:

$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

"Least squares loss function"

Example: 1D Linear regression training data

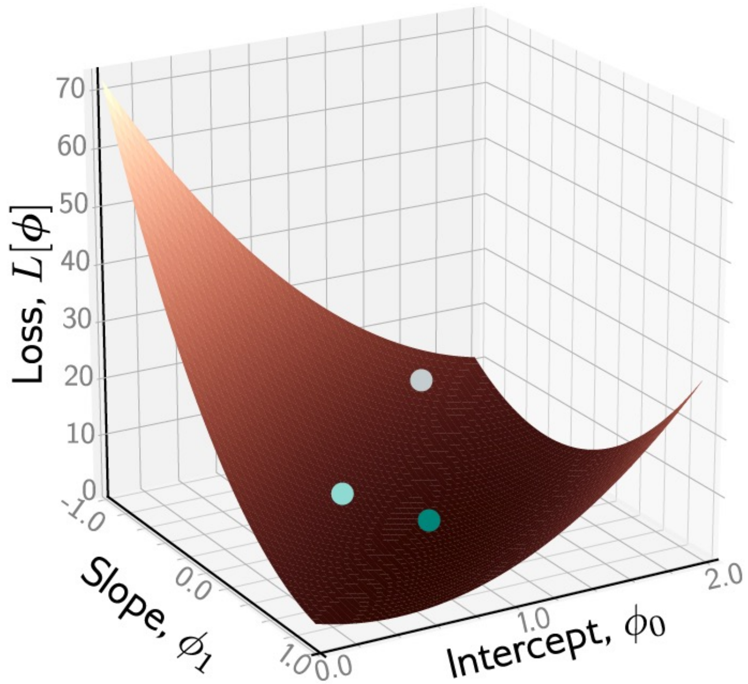


Loss function:

$$
\begin{aligned}
L[\boldsymbol{\phi}] &= \sum_{i=1}^{I} \left(\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i\right)^2 \\
&= \sum_{i=1}^{I} \left(\phi_0 + \phi_1 x_i - y_i\right)^2
\end{aligned}
$$

"Least squares loss function"

Example: 1D Linear regression loss function

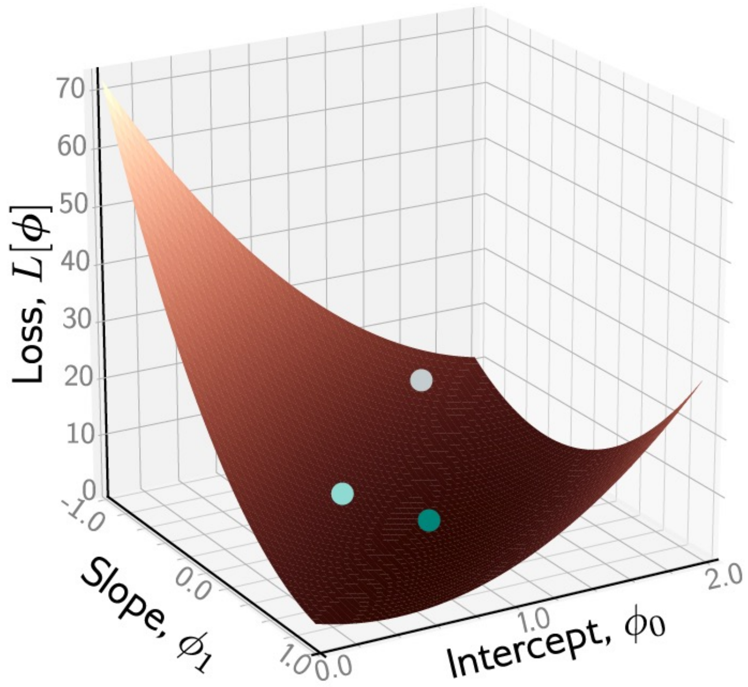


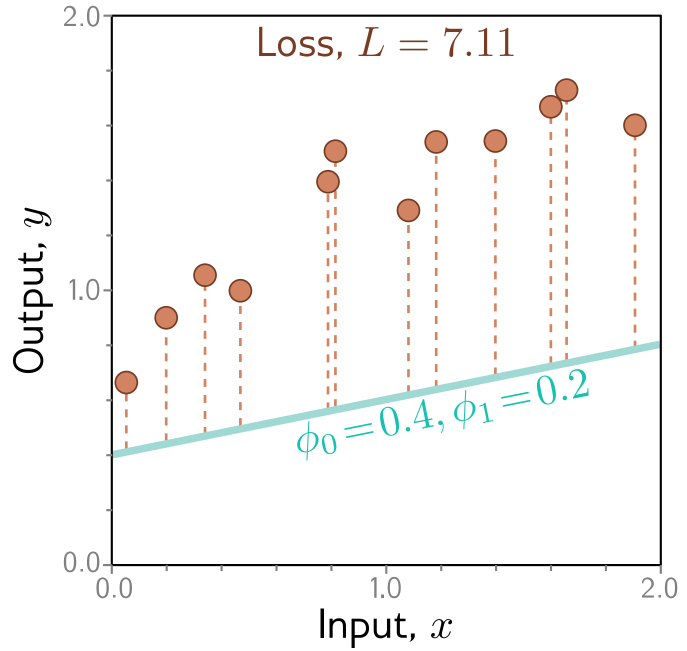
Loss function:

$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$
$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

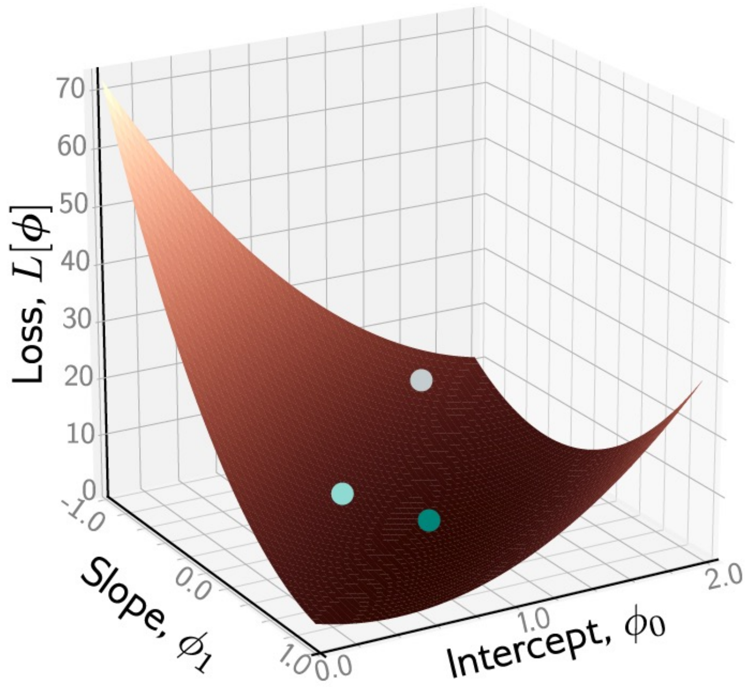
"Least squares loss function"

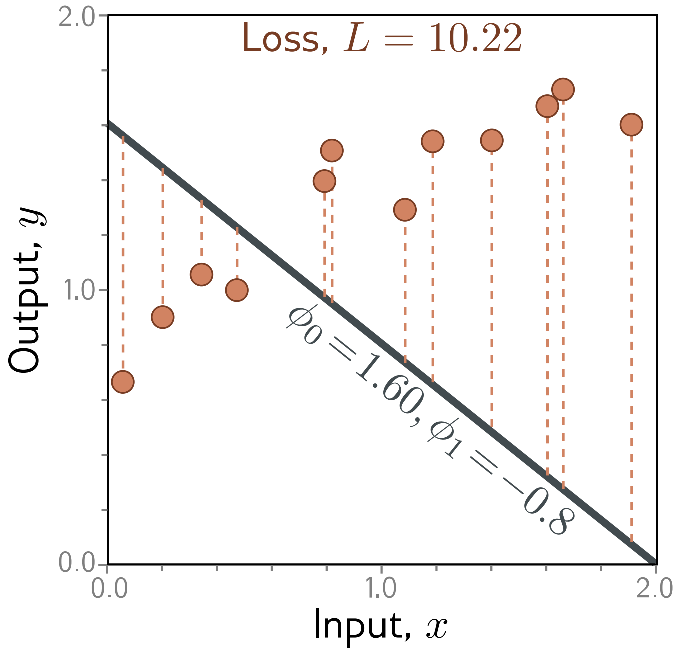
Example: 1D Linear regression loss function



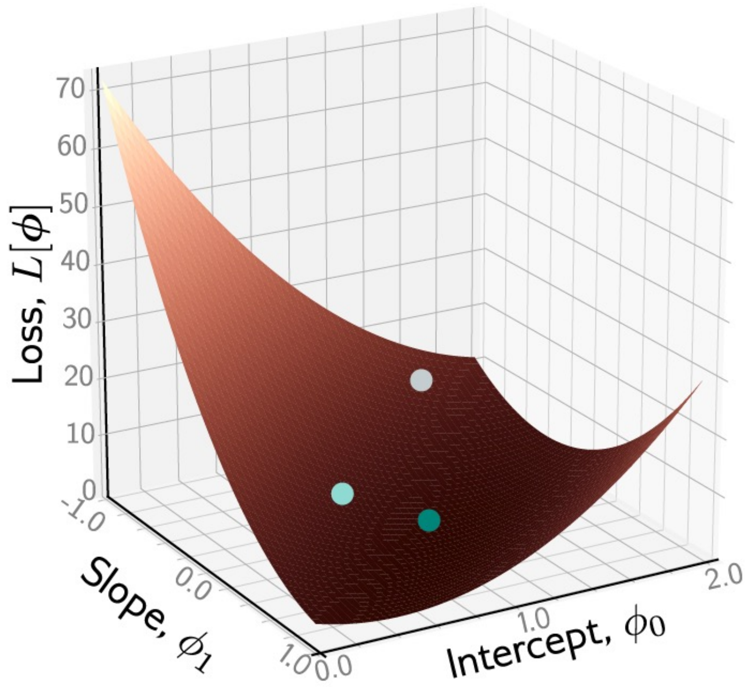


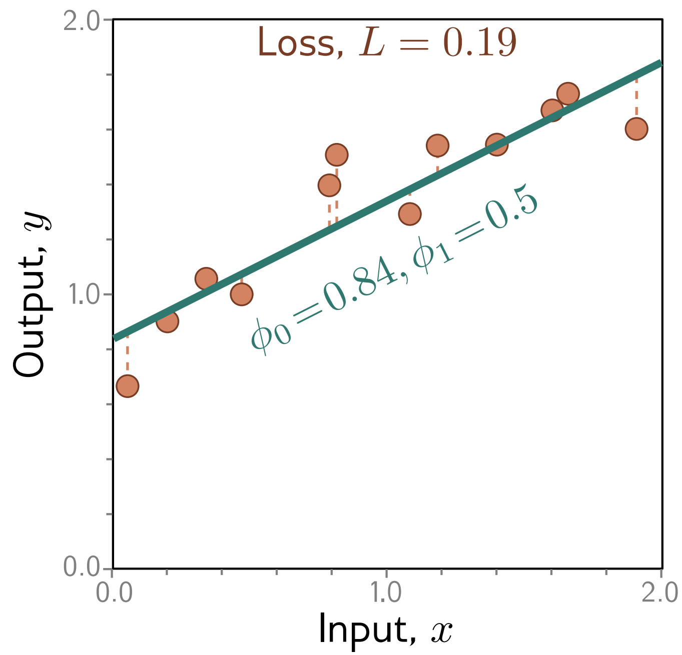
Example: 1D Linear regression loss function



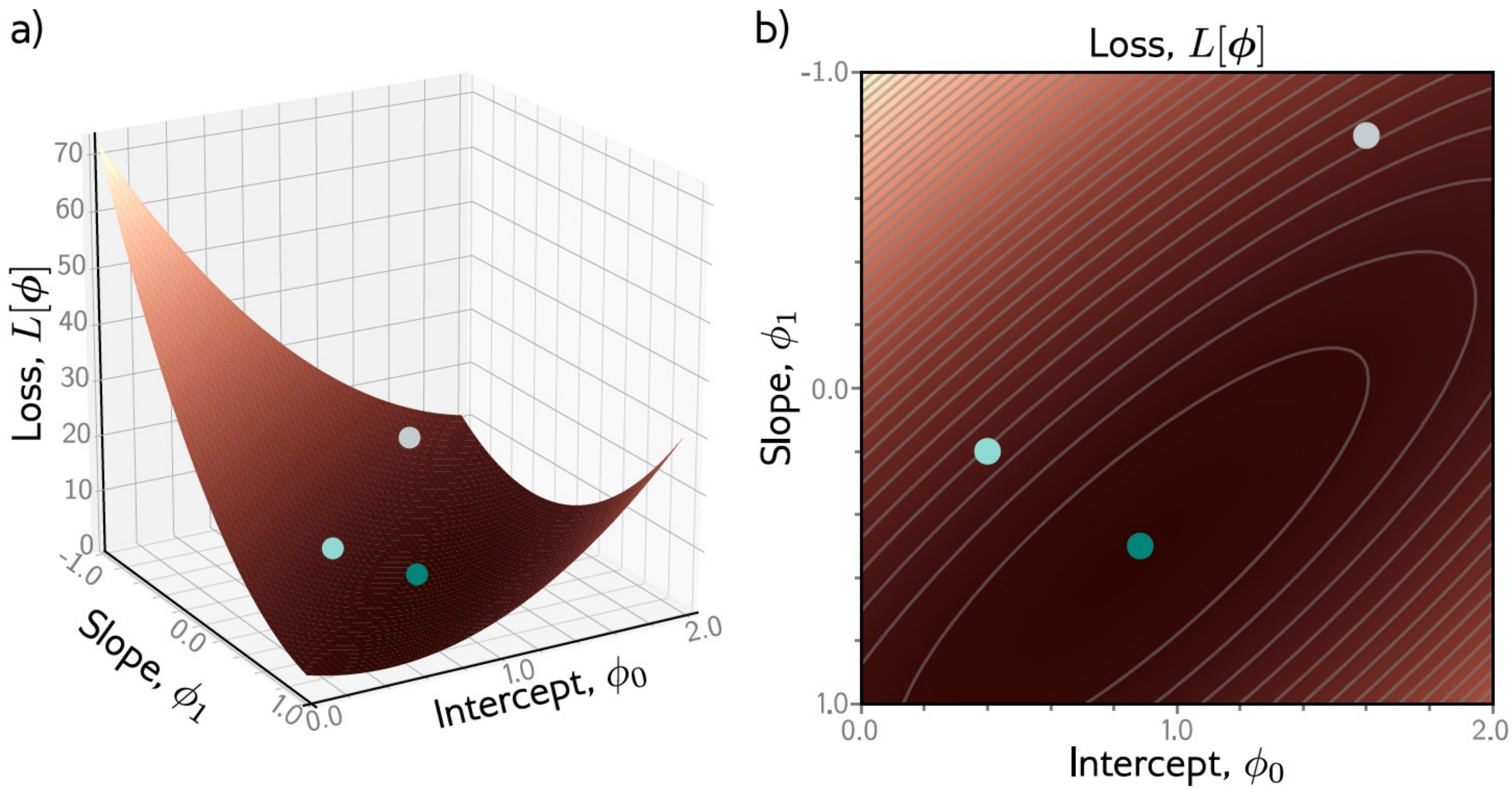


Example: 1D Linear regression loss function



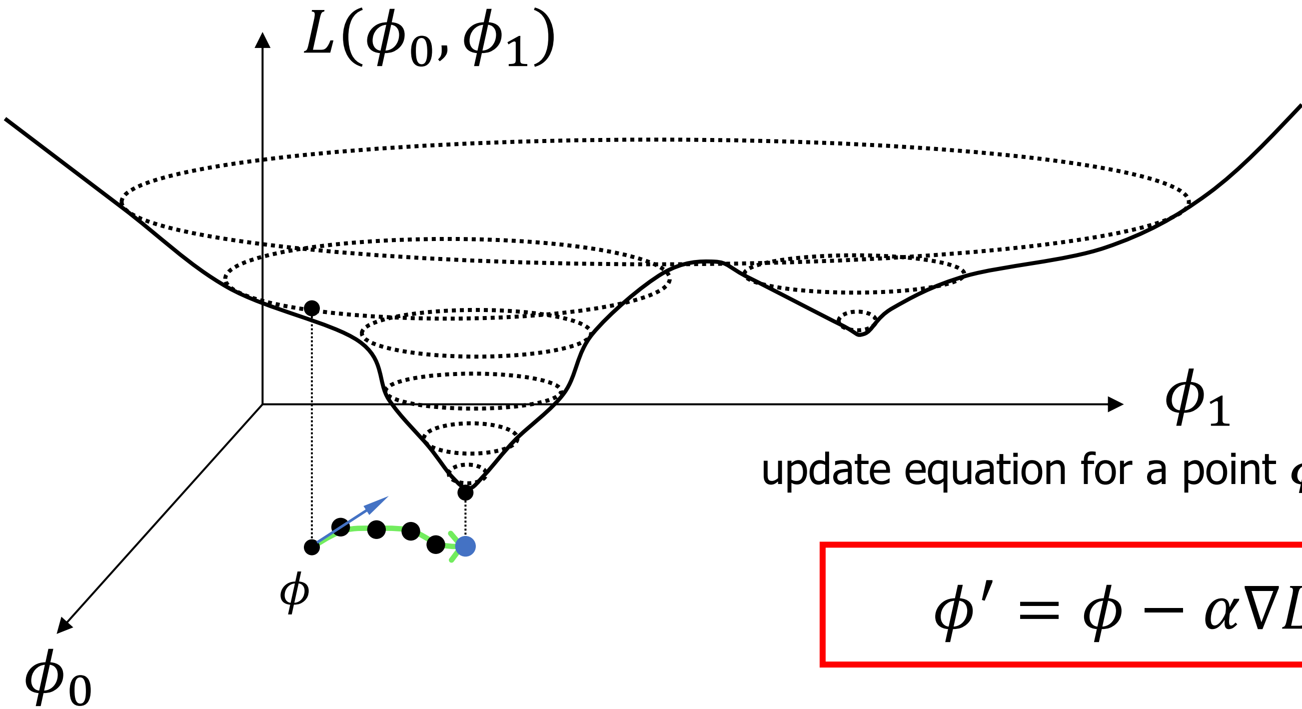


Example: 1D Linear regression loss function



Gradient Descent

- Example: for a function of two variables

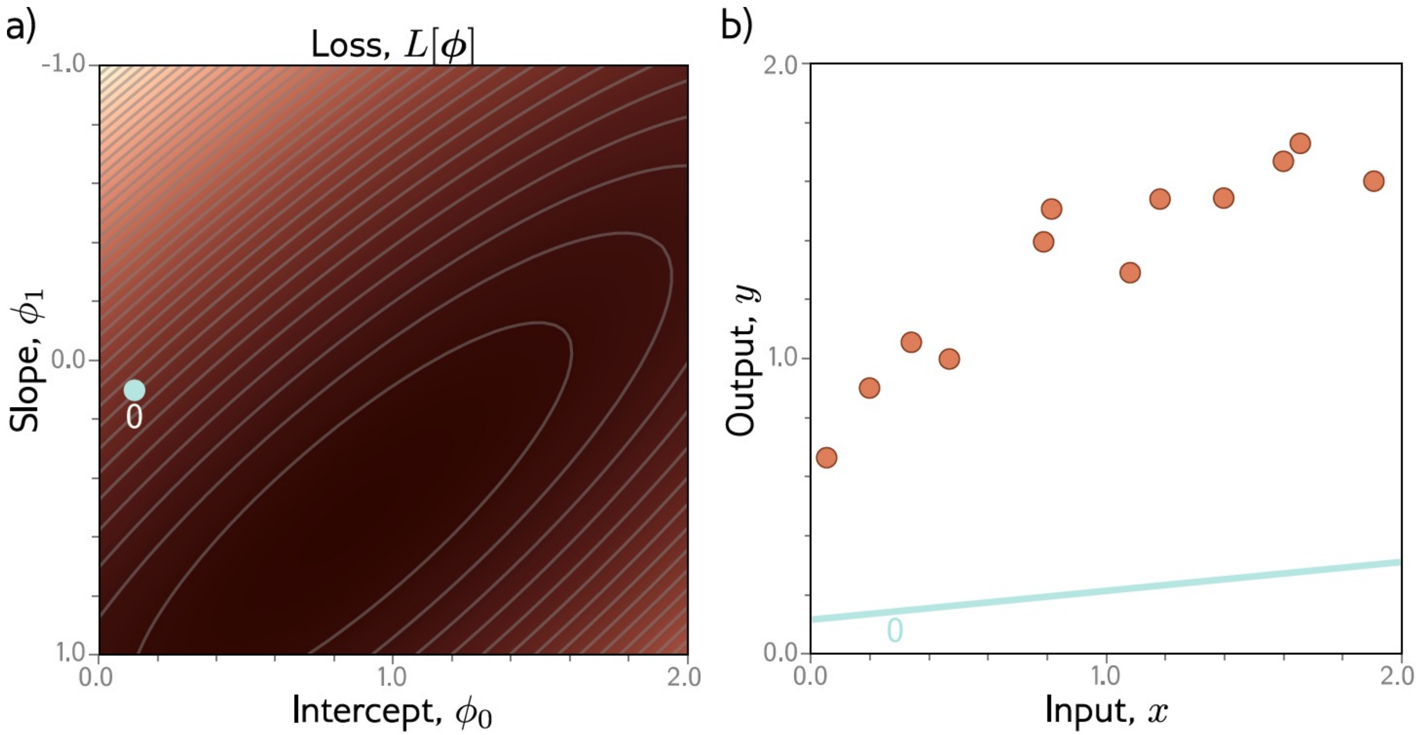


update equation for a point $\boldsymbol{\phi} = [\boldsymbol{\phi_0}, \boldsymbol{\phi_1}]$

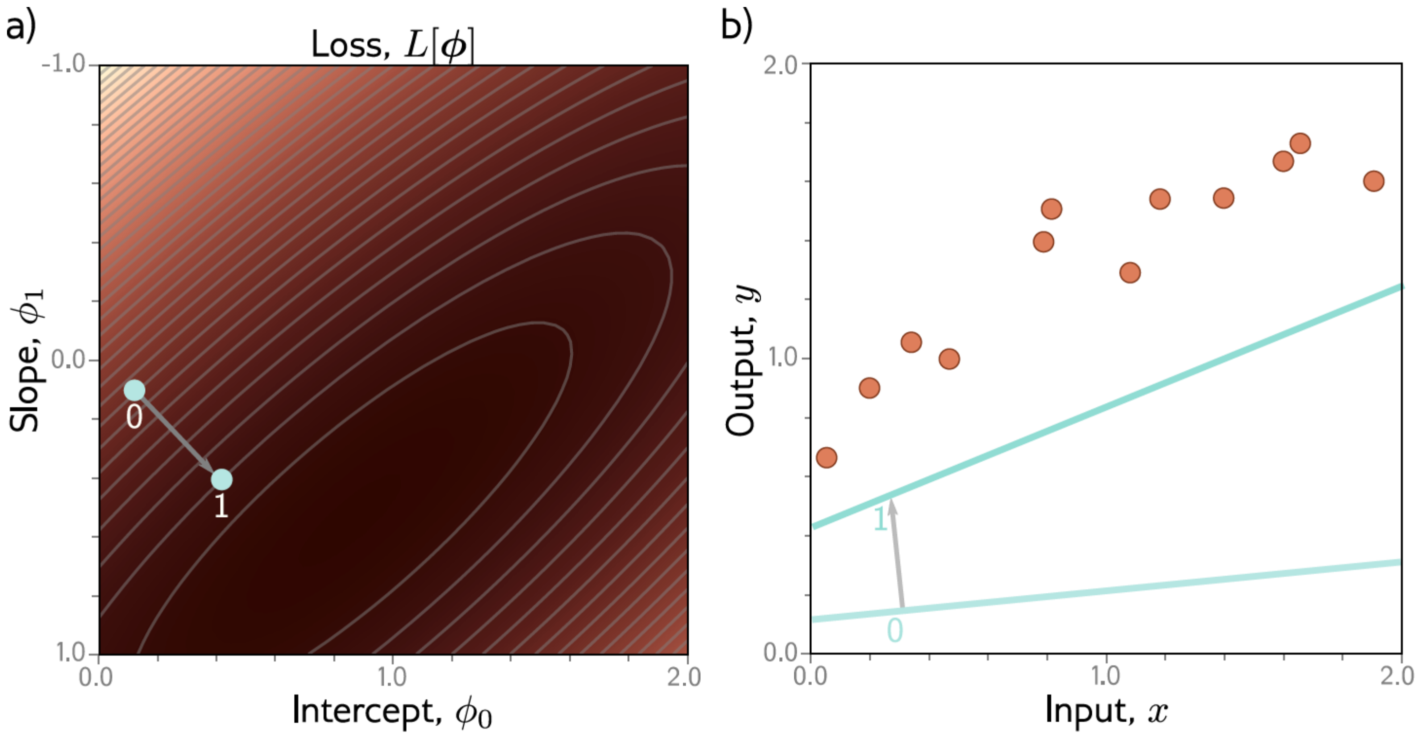
$$\boxed{\phi' = \phi - \alpha \nabla L}$$

Stop at a local minima where $\nabla L = \vec{0}$

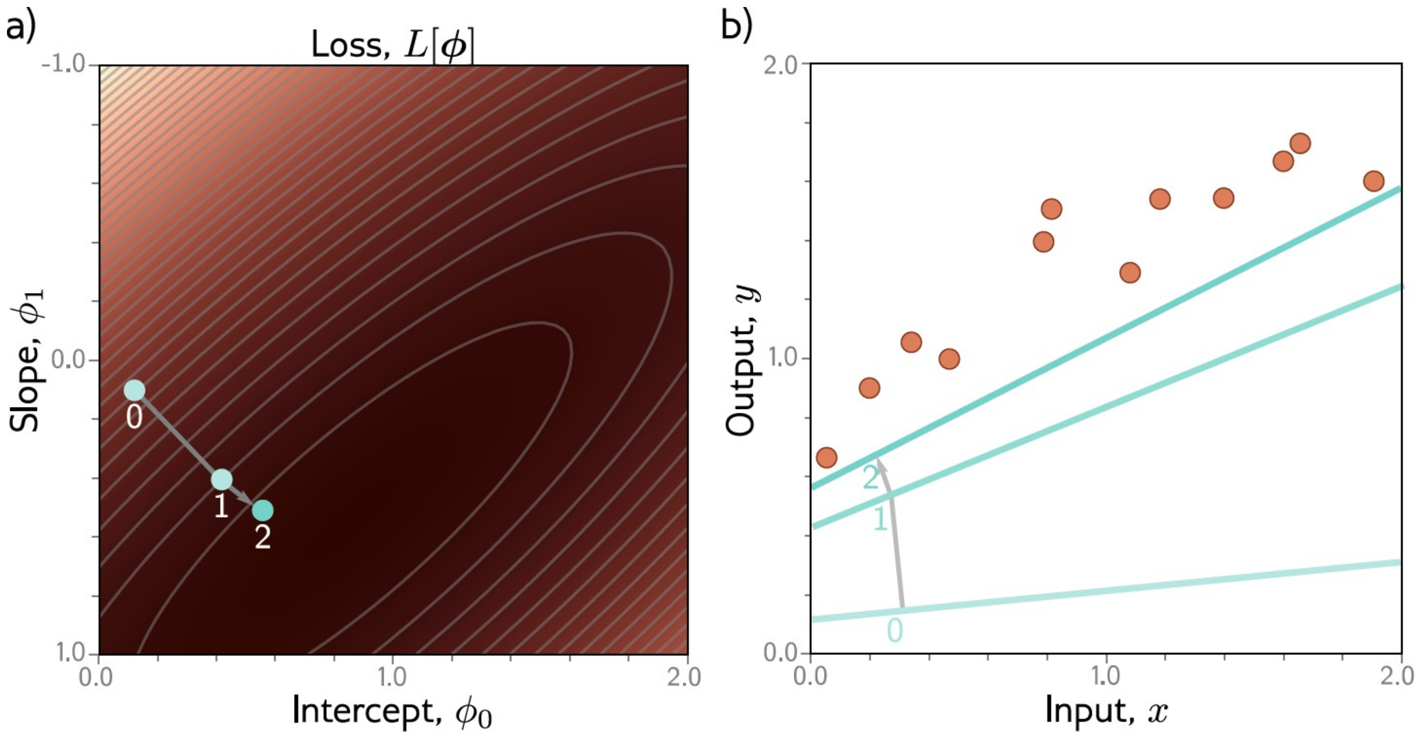
Example: 1D Linear regression training



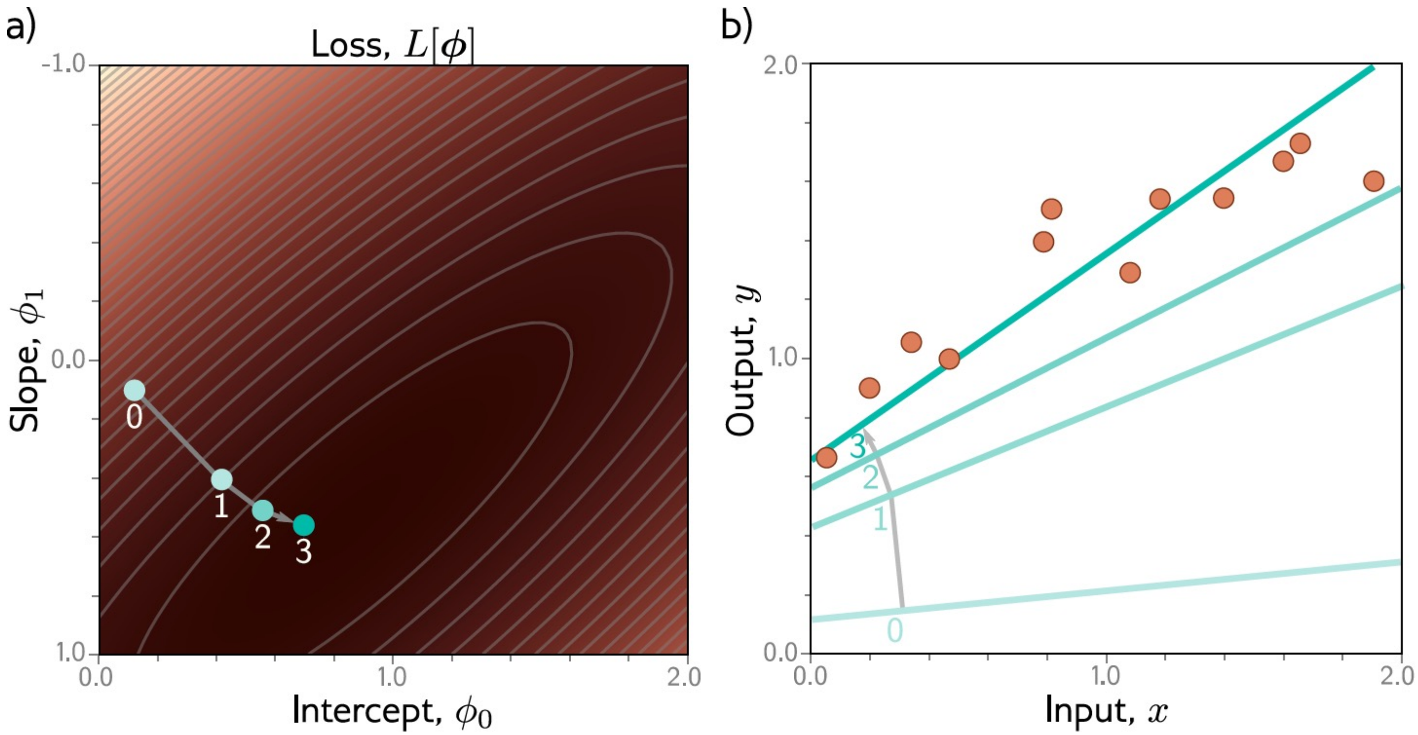
Example: 1D Linear regression training



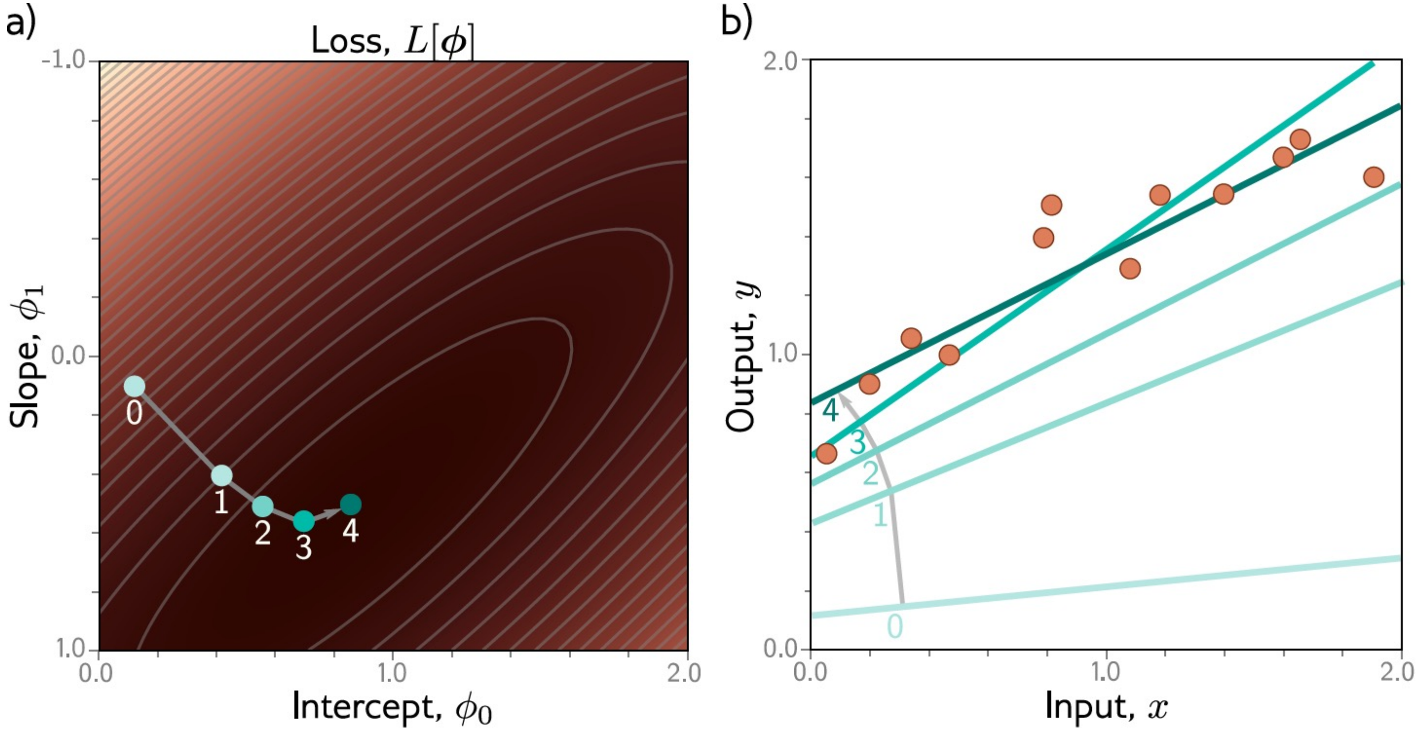
Example: 1D Linear regression training



Example: 1D Linear regression training



Example: 1D Linear regression training



Possible objections

- But you can fit the line model in closed form!
  - Yes – but we won't be able to do this for more complex models
- But we could exhaustively try every slope and intercept combo!
  - Yes – but we won't be able to do this when there are a million parameters

Example: Linear Classification

- For example: fish classification - *salmon* or *sea bass*?
- extract two features, *fish length* and *fish brightness*

feature vector

$$\mathbf{x}^1 = \begin{bmatrix} 3.3 \\ 5.7 \end{bmatrix} \quad \mathbf{x}^2 = \begin{bmatrix} 6.3 \\ 8.7 \end{bmatrix} \quad \mathbf{x}^3 = \begin{bmatrix} 2.3 \\ 1.7 \end{bmatrix} \quad \mathbf{x}^4 = \begin{bmatrix} 6.4 \\ 7.0 \end{bmatrix}$$

individual features $\mathbf{x}_k^i$
e.g. fish length and brightness

salmon

$\mathbf{y}^1=0$

sea bass

$\mathbf{y}^2=1$

salmon

$\mathbf{y}^3 = 0$

sea bass

$\mathbf{y}^4=1$

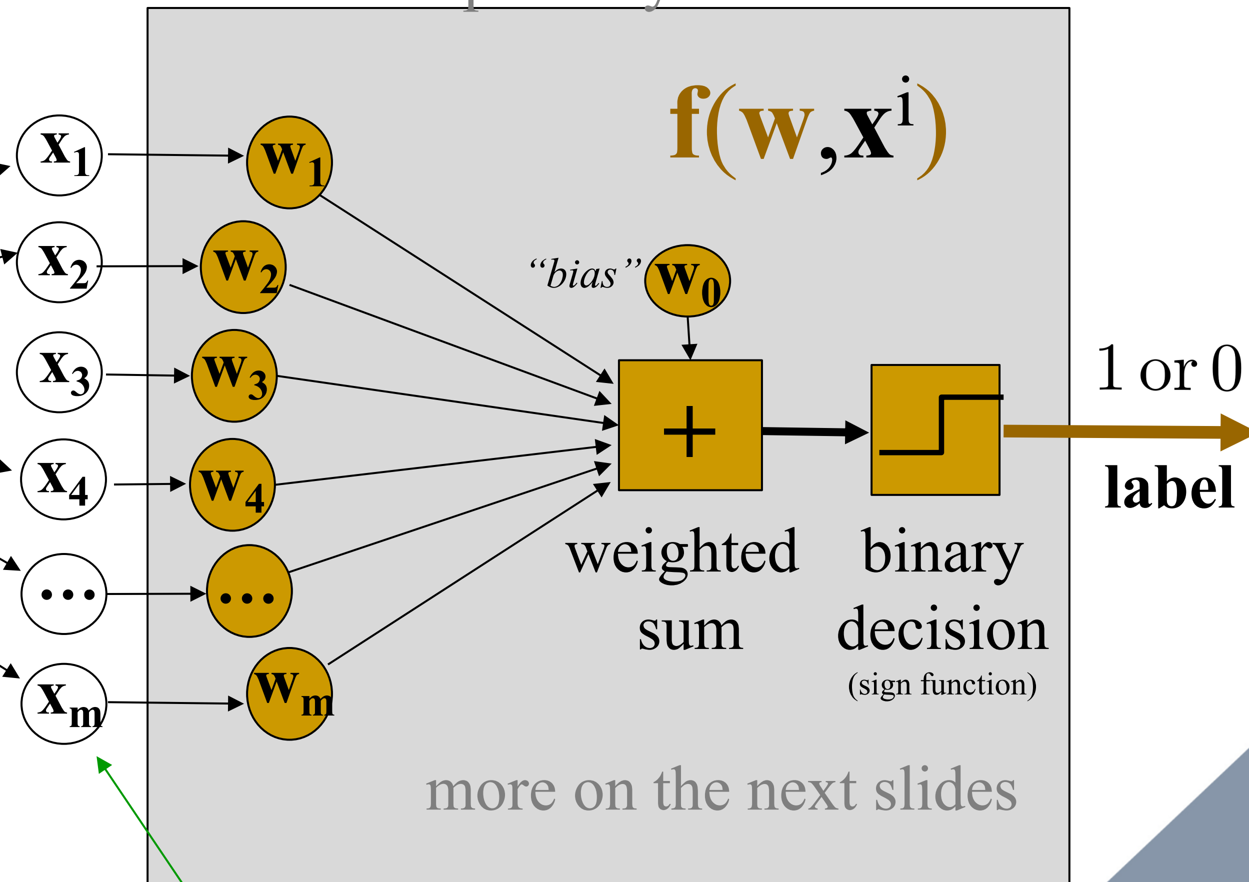
- $\mathbf{y}^i$ is the output (label or target) for example $\mathbf{x}^i$

Linear classifier example: *perceptron*

m-dimensional
feature vector $\mathbf{x^i} \in \mathcal{R}^m$
with *m* components

$$\mathbf{x}^{i} = \begin{bmatrix} \mathbf{x}_1^i \\ \mathbf{x}_2^i \\ \dots \\ \mathbf{x}_m^i \end{bmatrix}$$

Frank Rosenblatt, 1958
inspired by neurons

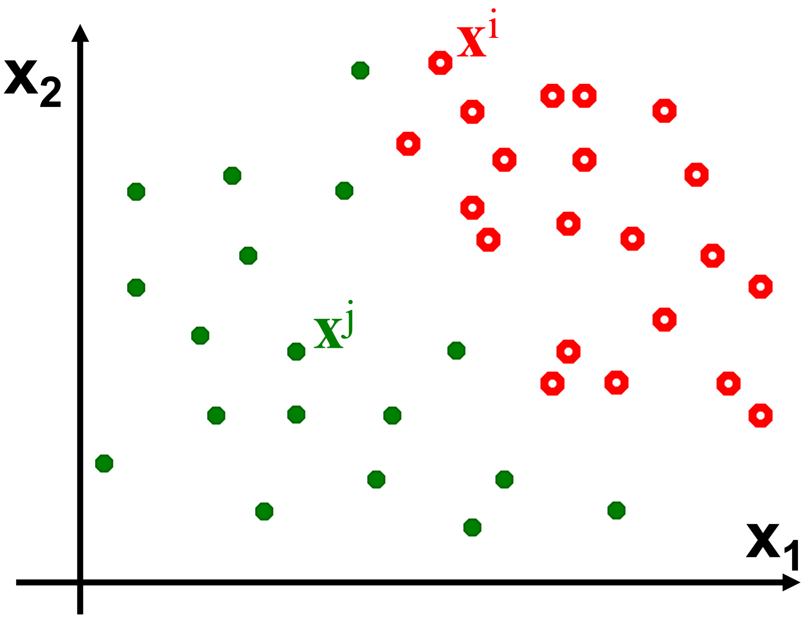


sub-indices are for feature components while **super-indices are for data points (feature vectors)**

NOTE: for simplicity, we omit **super-indices** (or **sub-indices**) assuming the context is "clear"

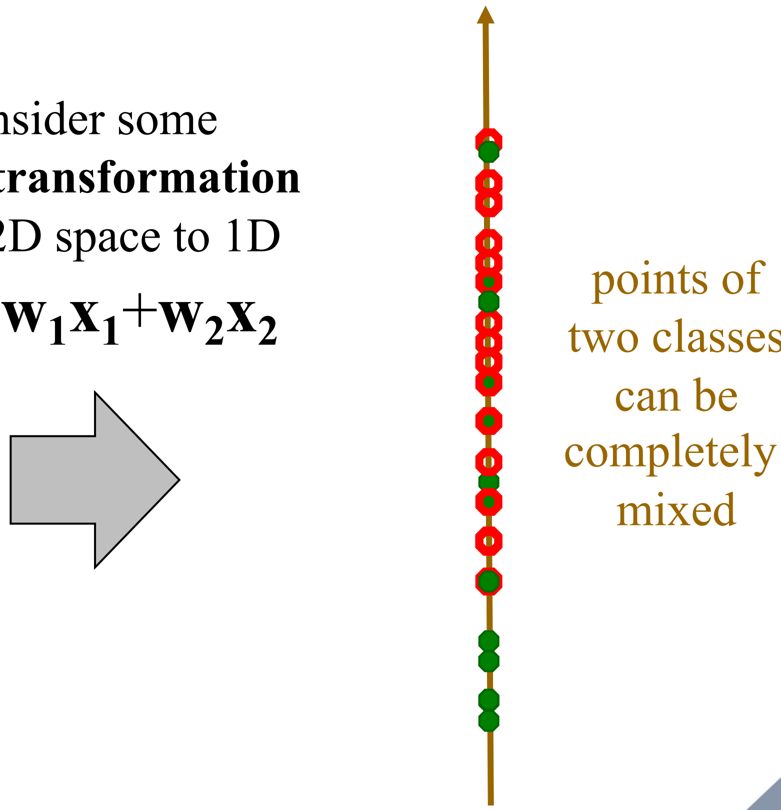
Linear classifier example: *perceptron*

For two class problem and 2-dimensional data (feature vectors)



consider some
linear transformation
from 2D space to 1D

$$\mathbf{w_0+w_1x_1+w_2x_2}$$

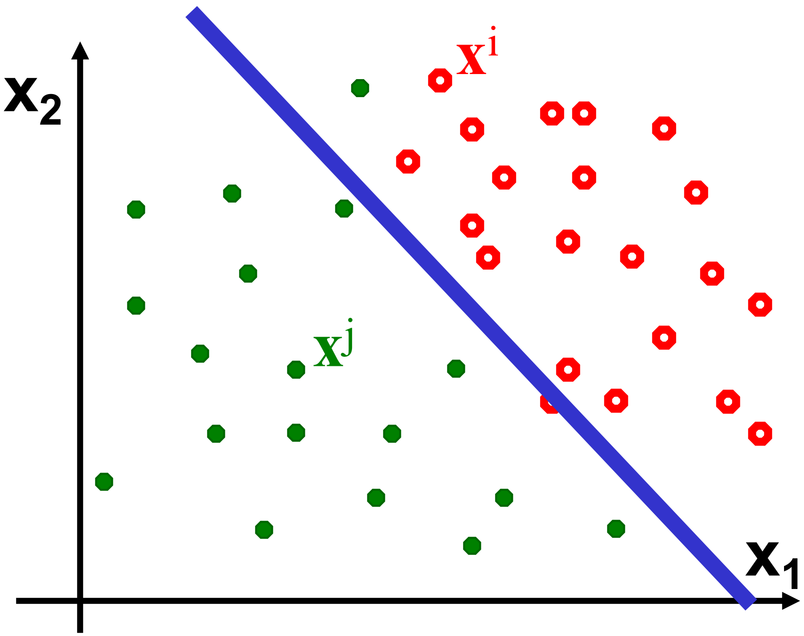


Question:

Is it possible to find a linear transformation onto 1D so that transformed 1D points can be separated (by a ***threshold***)?

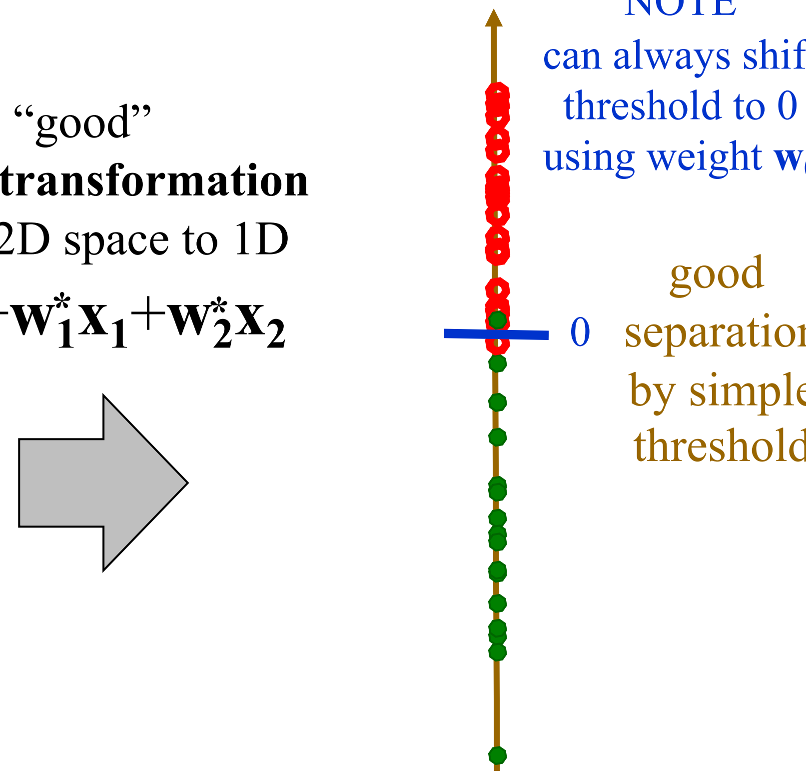
Linear classifier example: *perceptron*

For two class problem and 2-dimensional data (feature vectors)



"good"
linear transformation
from 2D space to 1D

$$w_0^* + w_1^* x_1 + w_2^* x_2$$

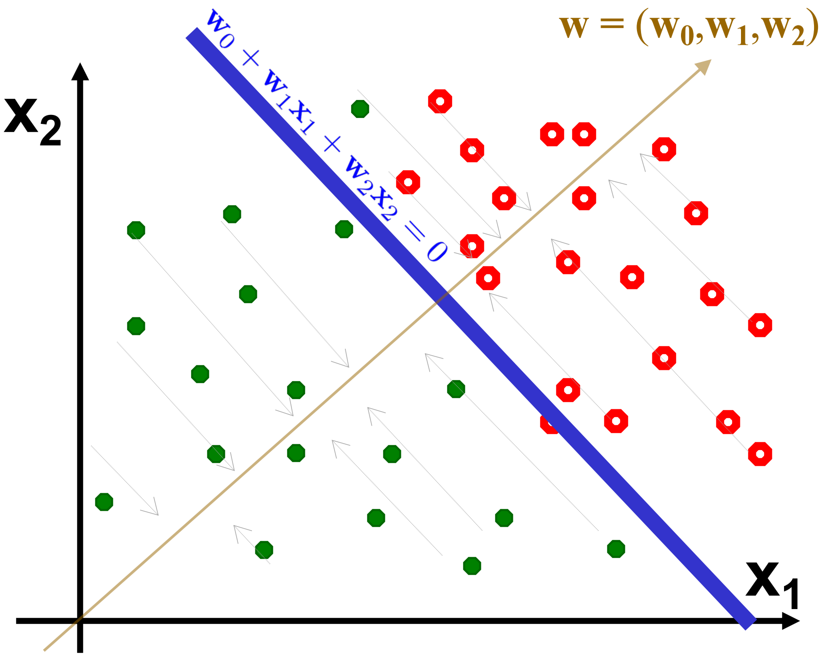


Answer:

In this case, YES, because the data is linearly separable in the original feature space. So, what is the transformation?

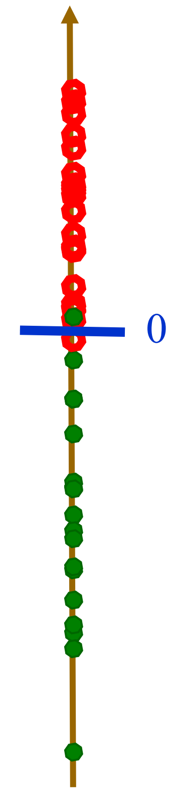
Linear classifier example: *perceptron*

For two class problem and 2-dimensional data (feature vectors)



"good"
linear transformation
from 2D space to 1D

$$\mathbf{w^*_0 + w^*_1 x_1 + w^*_2 x_2}$$



good separation by simple threshold

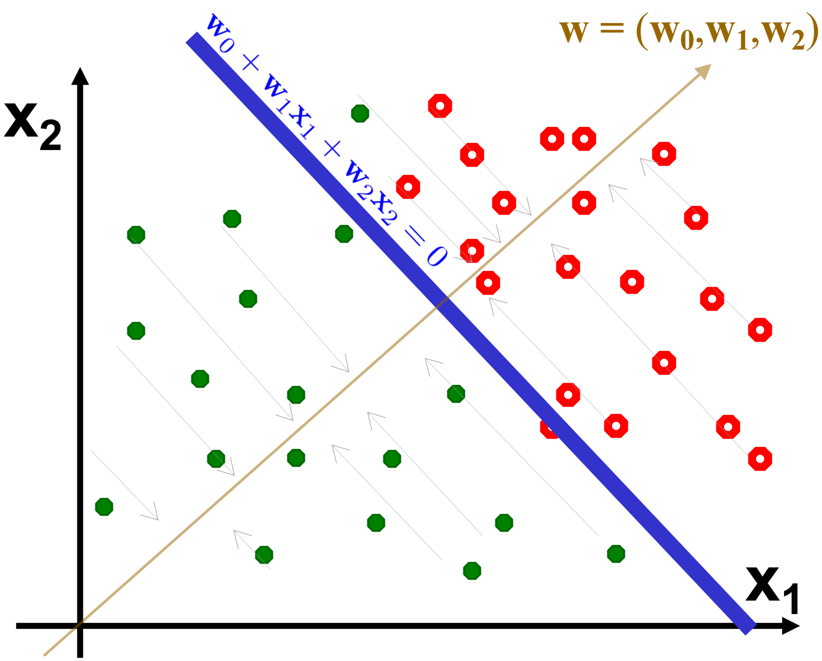
$(\mathbf{w_1},\mathbf{w_2})$ – are the coordinates of the **normal**
$\mathbf{w_0}$ – is "bias" (shifting threshold to 0)

Answer:

This 2D →1D linear transformation is a projection onto the **normal** of the separating **hyper-plane**.

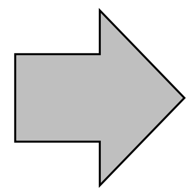
Linear classifier example: *perceptron*

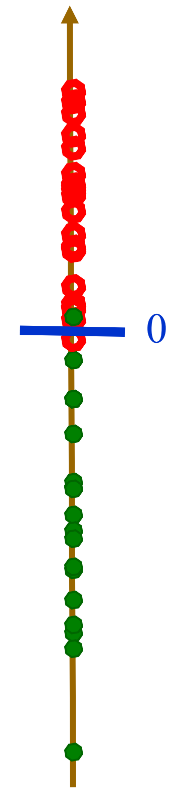
For two class problem and 2-dimensional data (feature vectors)



"good"
linear transformation
from 2D space to 1D

$$\mathbf{w_0^* + w_1^* x_1 + w_2^* x_2}$$





good separation by simple threshold

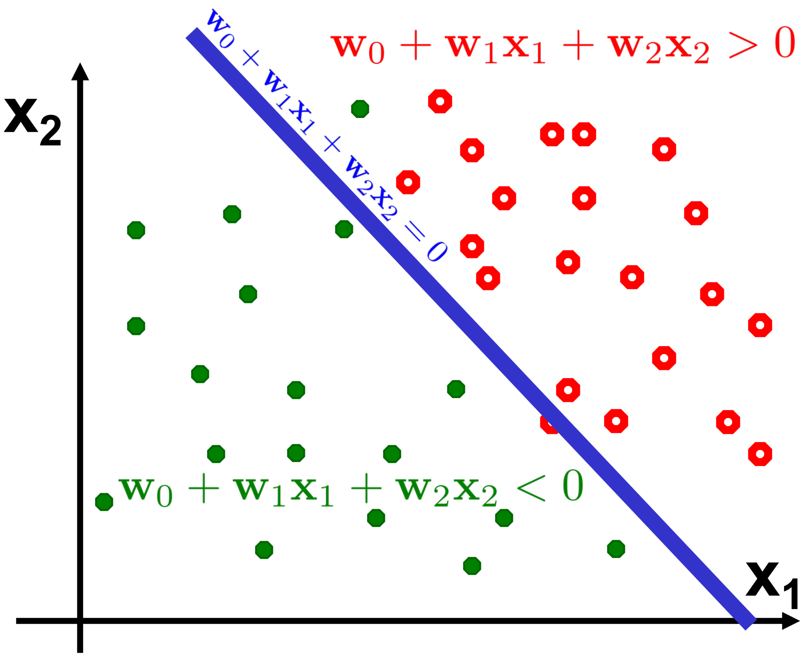
(w_1, w_2) – are the coordinates of the **normal**

w_0 – is "bias" (shifting threshold to 0)

In fact, **any** 2D →1D linear transformation $\mathbf{w = (w_0, w_1, w_2)}$ is a **projection onto normal of some hyper-plane**. So, original question really asks if there is a hyper-plane separating data.

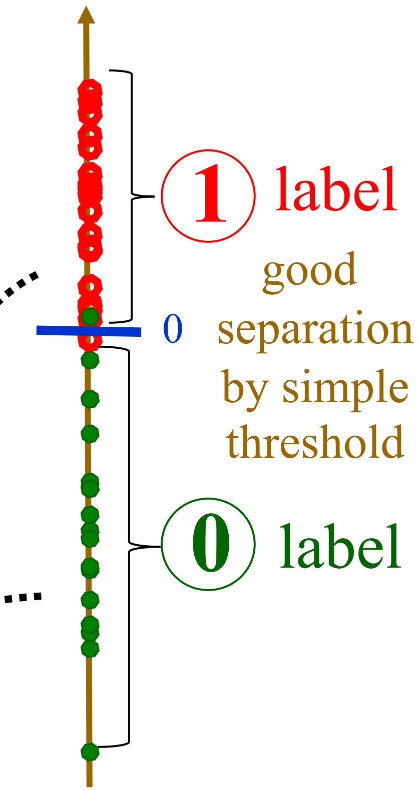
Linear classifier example: *perceptron*

For two class problem and 2-dimensional data (feature vectors)



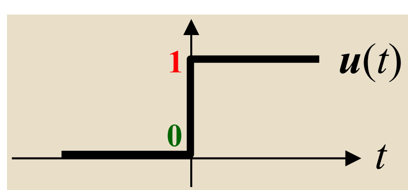
"good"
linear transformation
from 2D space to 1D

$$\mathbf{w}_0^* + \mathbf{w}_1^*\mathbf{x}_1 + \mathbf{w}_2^*\mathbf{x}_2$$



thresholding can be formally represented by this prediction function

$$\mathbf{f}(\mathbf{w},\mathbf{x}) = u\,(\mathbf{w}_0 + \mathbf{w}_1\mathbf{x}_1 + \mathbf{w}_2\mathbf{x}_2) \qquad \mathbf{f}(\mathbf{w},\mathbf{x}) \in \{0, 1\}$$

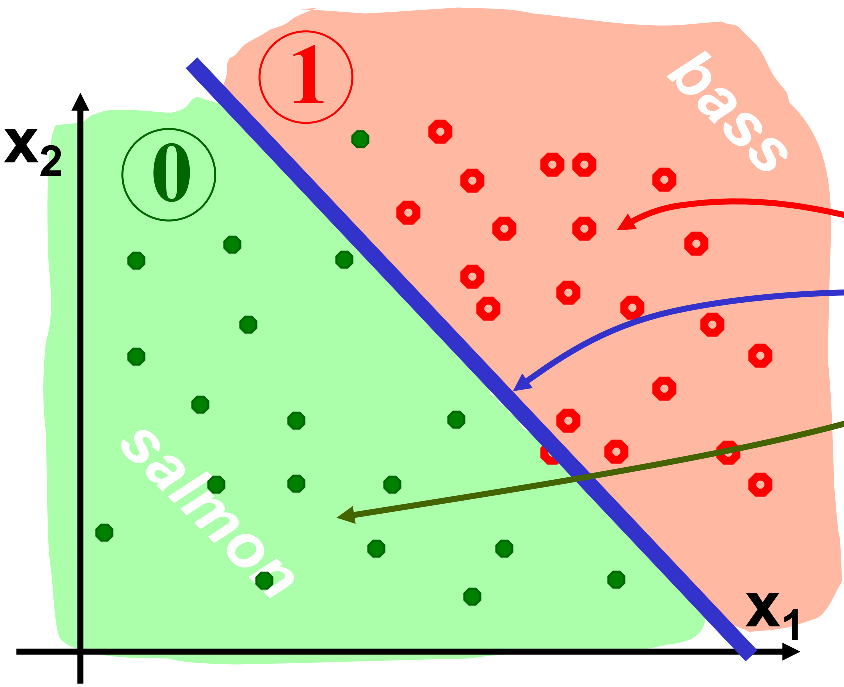


unit step function
(a.k.a. *Heaviside* function)

$$u(t) := \begin{cases} 1 & \text{if } t > 0 \\ 0 & O.W. \end{cases}$$

Linear classifier example: *perceptron*

For two class problem and 2-dimensional data (feature vectors)



$$\mathbf{f}(\mathbf{w},\mathbf{x}) = u\,(\mathbf{w}_0+\mathbf{w}_1\mathbf{x}_1+\mathbf{w}_2\mathbf{x}_2)$$

Can use this function to classify any (<u>new</u>) point.

- Can be generalized to feature vectors $\mathbf{x}$ of any dimension m :

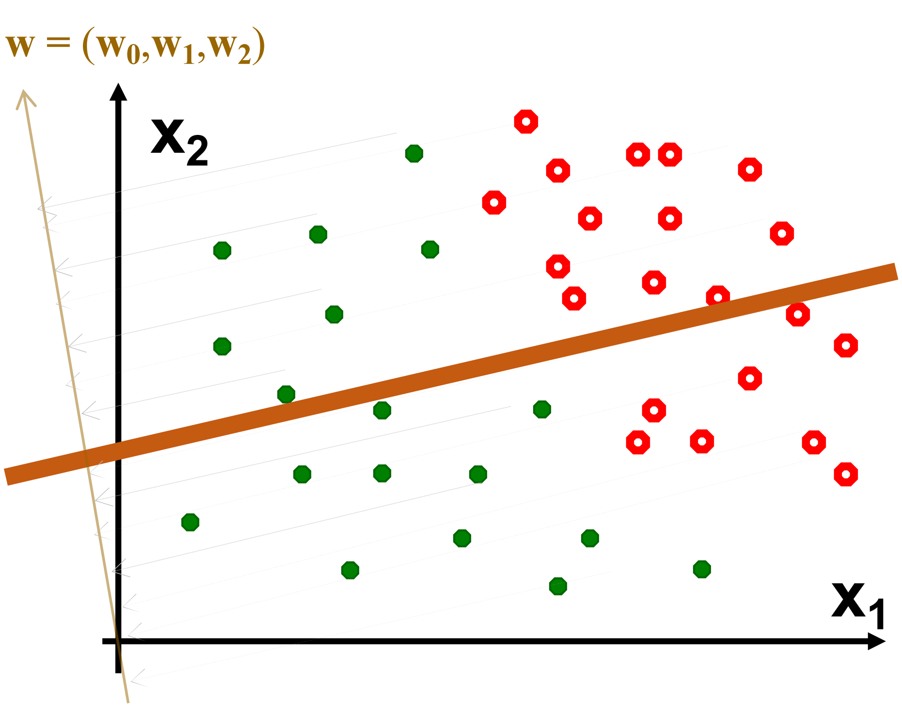
$\boxed{\mathbf{f}(W,X) = \mathbf{u}\,(W^T X)}$ for $W^T = [\underbrace{\mathbf{w}_0}_{\text{“bias”}}, \mathbf{w}_1, \dots, \mathbf{w}_m]$ and $X^T = [1, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m]$

homogeneous **representation of feature vector x**

- Classifier that makes decisions based on linear combination of features is called a **linear classifier**

Linear Classifiers

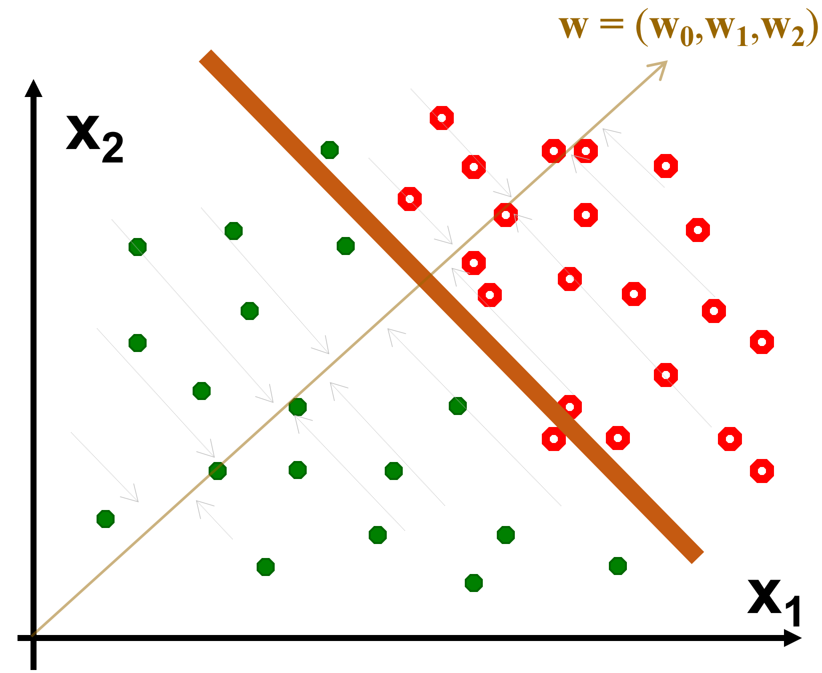
bad **w**



classification error 38%

projected points onto
normal line are all mixed-up

better **w**

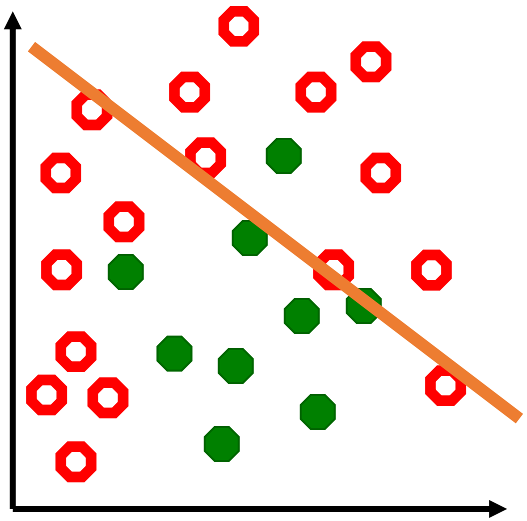


classification error 4%

projected points onto
normal line are well separated

Underfitting

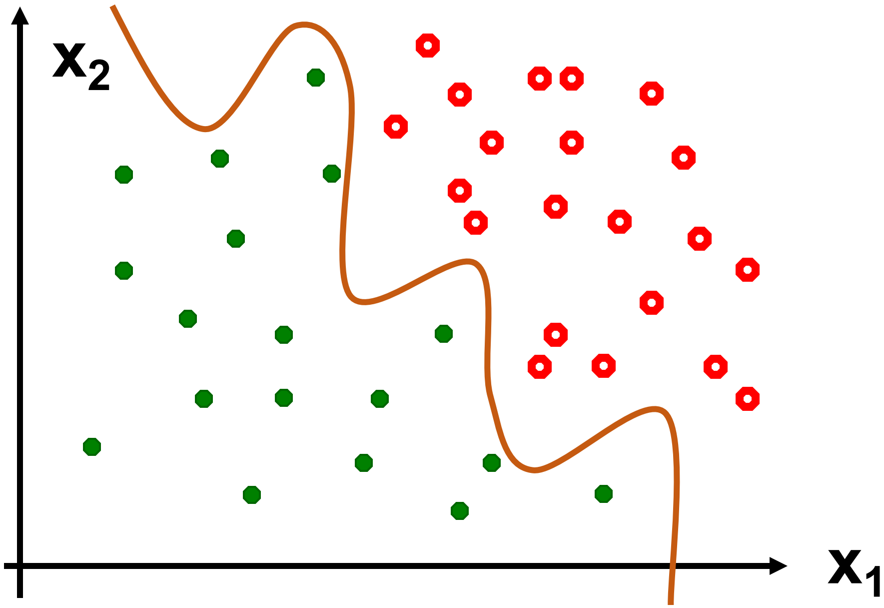
For some types of data
no linear decision boundary
can separate the samples well



- Classifier underfits the data if it can produce decision boundaries that are too simple for this type of data

  - chosen classifier type (hypothesis space) is not expressive enough

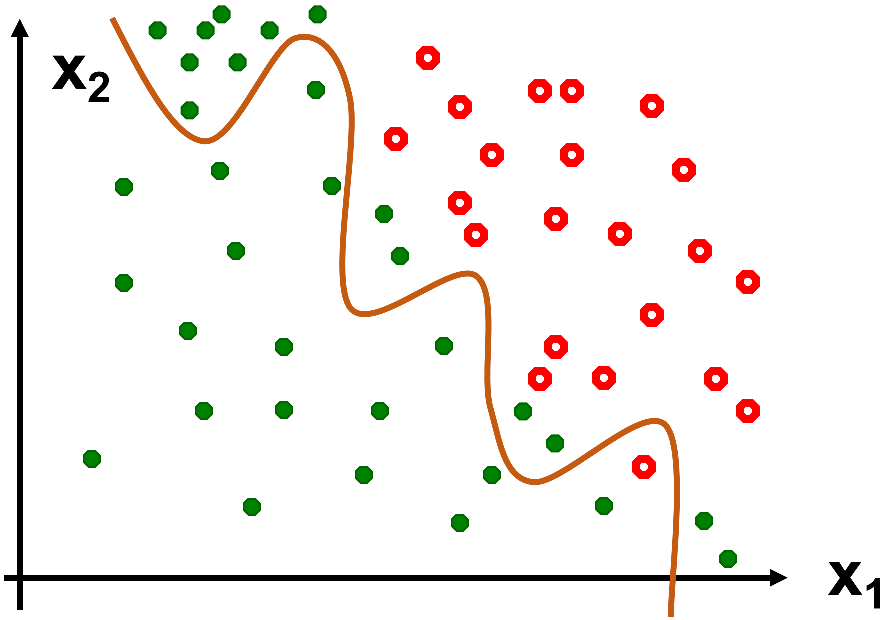
More complex (non-linear) classifiers



- for example, if $\mathbf{f}(\mathbf{w},\mathbf{x})$ is a polynomial of high degree
- can achieve 0% classification error

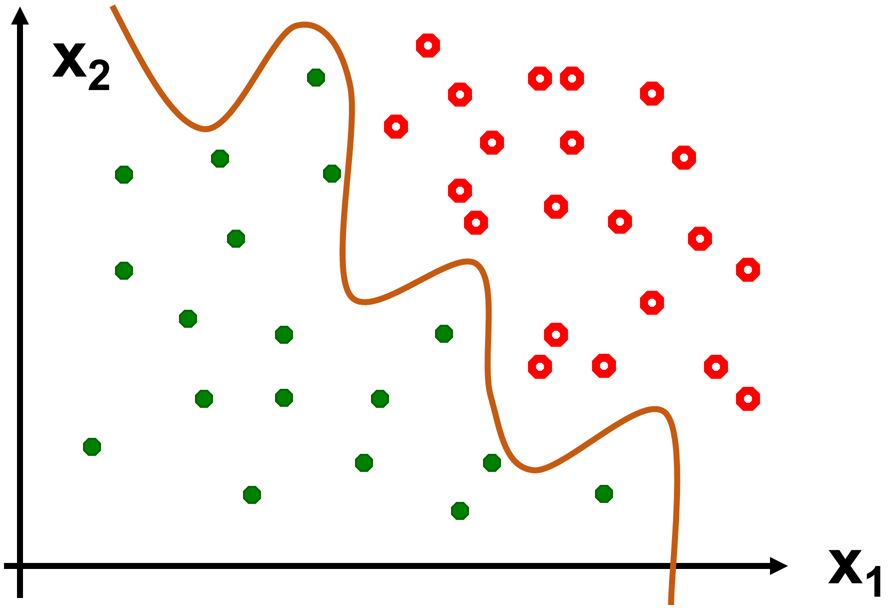
UCMERCED

More complex (non-linear) classifiers



- The goal is to classify well on **new data**
- Test "wiggly" classifier on new data: 25% error

Overfitting



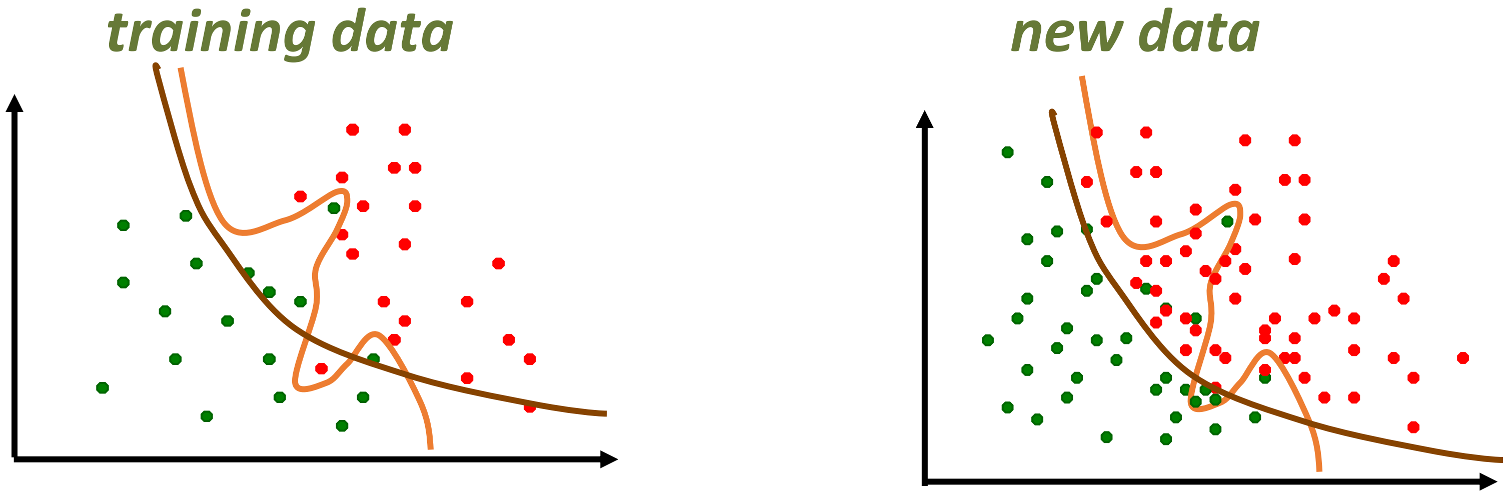
- ❑ Amount of data for training is always limited
- ❑ Complex model often has too many parameters to fit reliably to limited data
- ❑ Complex model may adapt too closely to “random noise” in training data, rather than look at a “big picture”

Overfitting: Extreme Example

- Two class problem: *face* and *non-face* images
- Memorize (i.e. store) all the "face" images
- For a new image, see if it is one of the stored faces
  - if yes, output "face" as the classification result
  - If no, output "non-face"
- **problem**:
  - zero error on stored data, 50% error on test (new) data
  - decision boundary is very irregular
- Such learning is **memorization without generalization**

slide is modified from Y. LeCun

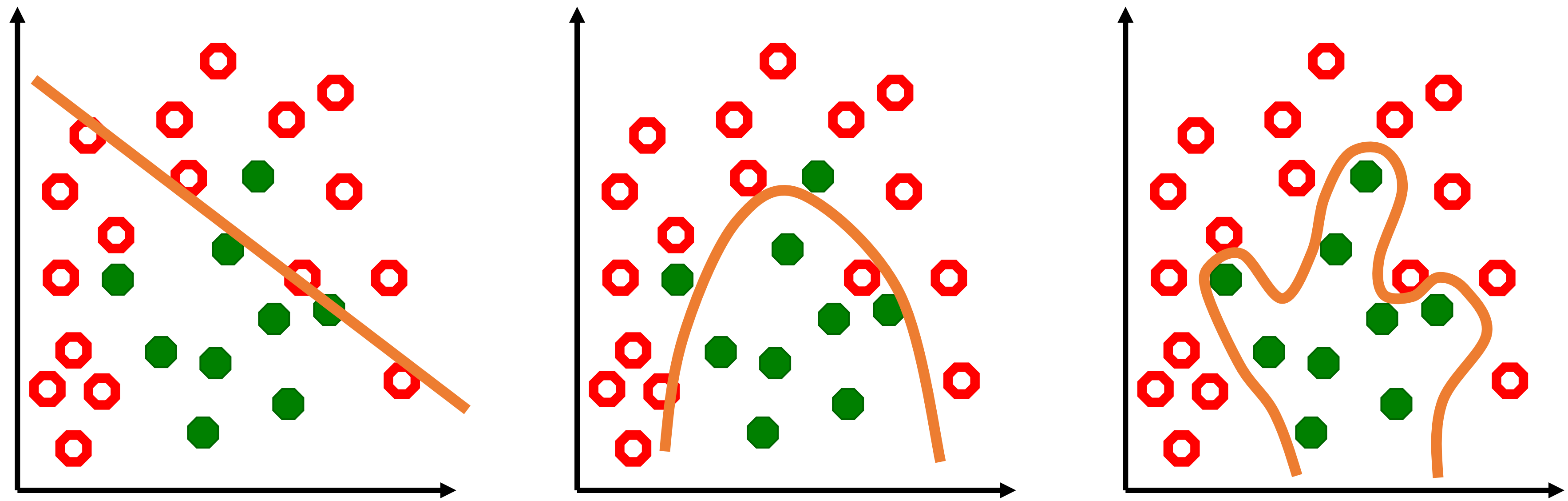
Generalization



- ❑ Ability to produce correct outputs on previously unseen examples is called **generalization**
- ❑ Big question of learning theory: how to get good generalization with a limited number of examples
- ❑ Intuitive idea: **favor simpler classifiers**
- ❑ Simpler decision boundary may not fit ideally to training data but tends to generalize better to new data

Underfitting → Overfitting

underfitting | ***"just right"*** | ***overfitting***



underfitting
- ❑ high training error
- ❑ high test error

"just right"
- ❑ low training error
- ❑ low test error

overfitting
- ❑ low training error
- ❑ high test error