CSE 176 Introduction to Machine Learning

Lecture 3: Supervised Learning: Classification and Regression

Some materials from Miguel Carreira-Perpiñán and Olga Veksler

Recap: 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
- Eigen decomposition
- Singular value decomposition

Slide from S. Srihari

Recap: Matrix times matrix

- If $\boldsymbol{A}$ is of shape mxn and B is of shape nxp then *matrix product* $\boldsymbol{C}$ is of shape mxp

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

- 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

Recap: Matrix times vector: Linear transformation

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

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

n equations in
n unknowns

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

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

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

Slide from S. Srihari

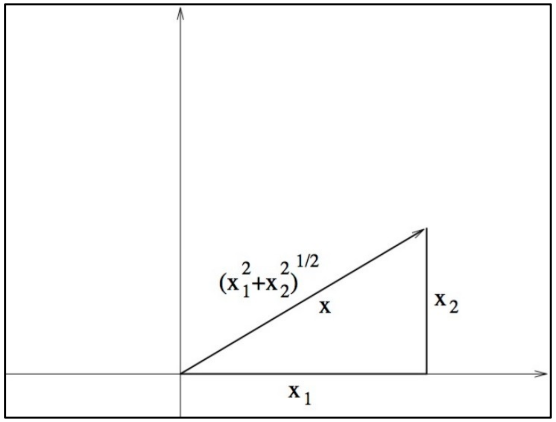
Recap: L^p Norm

- 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 ***x***
    - written simply as $||\boldsymbol{x}||$
    - Squared Euclidean norm is same as $\boldsymbol{x}^T\boldsymbol{x}$



– L^1 Norm

  - Sum of absolute value for each x_i

– L^∞ Norm

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

  - Called max norm

Slide from S. Srihari

Recap: Eigen decomposition

- 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

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

Slide from S. Srihari

Recap: 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

Recap: Conditional probability

- We are often interested in the probability of an event given that some other event has happened

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

Slide from S. Srihari

Recap: 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.

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

Slide from S. Srihari

Recap: 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

Today's topic

- ☐Supervised Learning
  - ☐Binary Classification
  - ☐Multi-class classification
  - ☐Regression
- ☐Problem definition and formulation

Example of Binary Classification

□Fish classification - *salmon* or *sea bass*?

□extract two features, *fish length* and *fish brightness*

feature vector

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

individual features $\mathbf{x}_k^i$

e.g. fish length and brightness

salmon

$\mathbf{y}^1=0$

sea bass

$\mathbf{y}^2=1$

salmon

$\mathbf{y}^3 = 0$

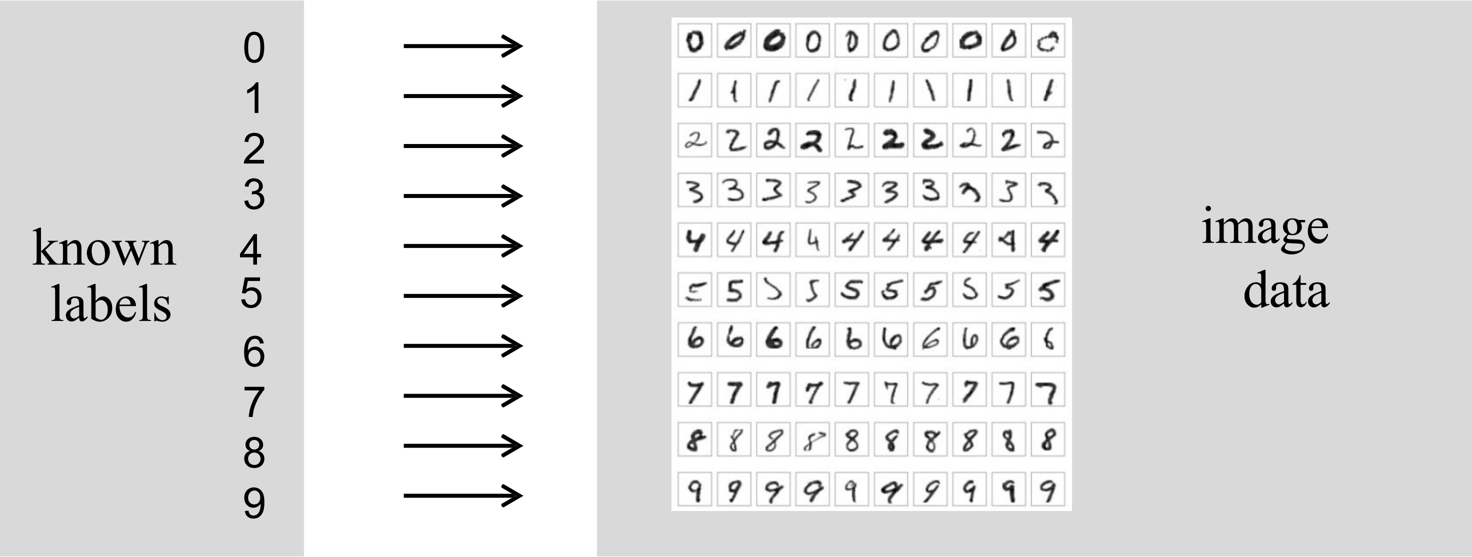
sea bass

$\mathbf{y}^4=1$

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

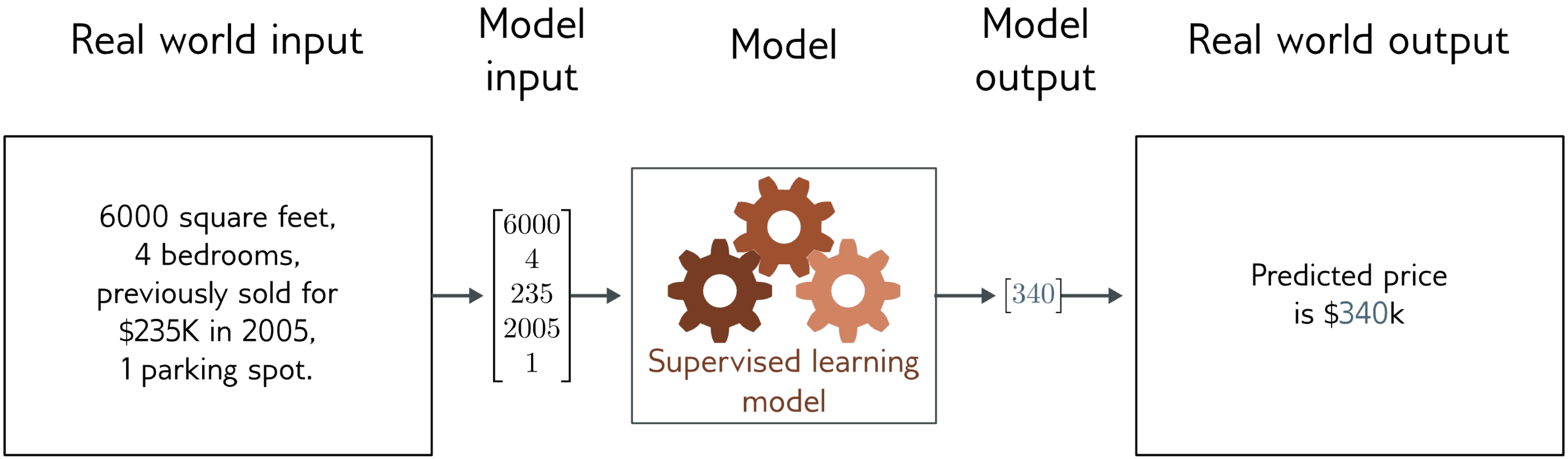
Example of Multi-class classification

- 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 of 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 $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$

$\Big\}$ *labeled data*

☐ **Training phase**

- estimate function $\mathbf{y} = \mathbf{h}(\mathbf{x})$ from labeled data

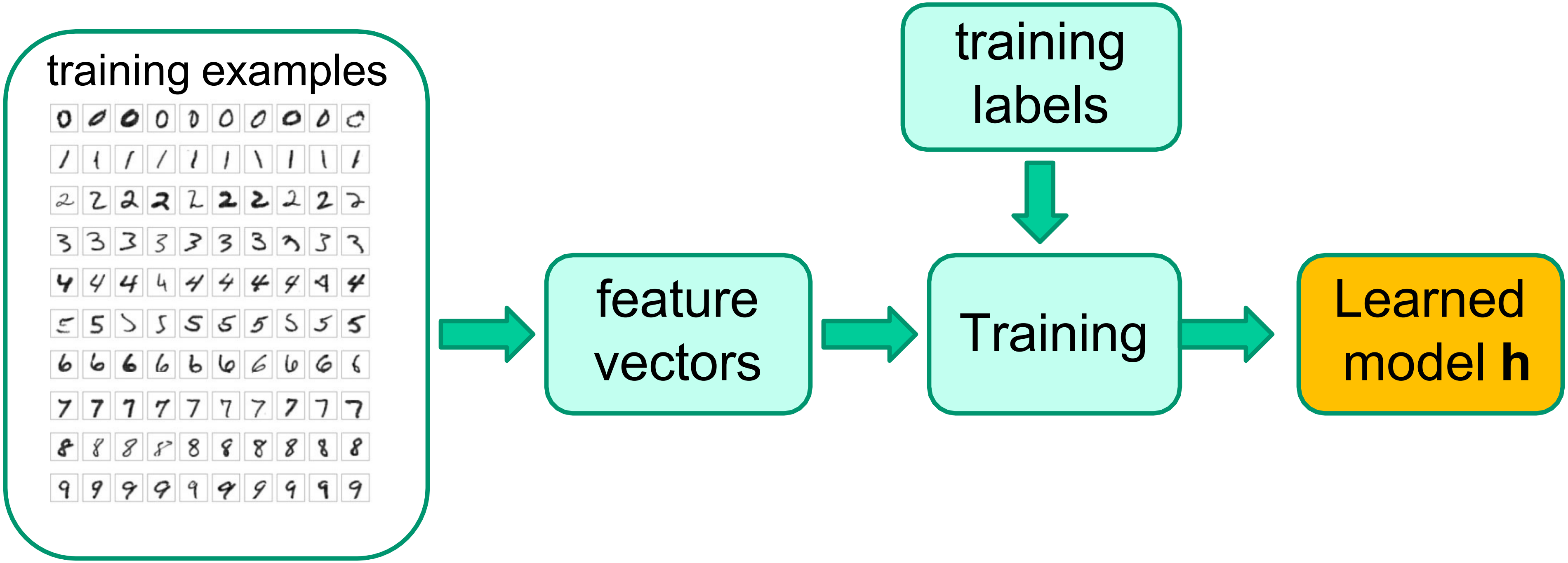
where $\mathbf{h}(\mathbf{x})$ is called *classifier*, *learning machine*, *prediction function*, etc.

☐ **Testing phase** (deployment)

☐ predict output $\mathbf{h}(\mathbf{x})$ for a new (unseen) sample $\mathbf{x}$

Training/Testing Phases Illustrated

Training



training examples

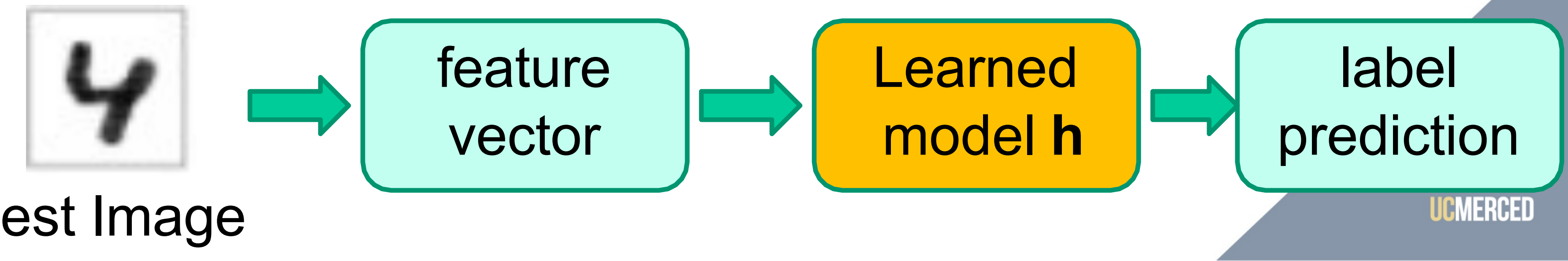
training labels

feature vectors

Training

Learned model **h**

Testing



test Image

feature vector

Learned model **h**

label prediction

Training phase as parameter estimation

☐Estimate prediction function $y = \mathrm{h}(x)$ from labeled data

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

$$\mathrm{h}_{\mathrm{w}}(x) \quad \text{or} \quad \mathrm{h}(w, x) \qquad\qquad \boxed{w = ?}$$

Goal: find classifier parameters (weights) w so that $\mathrm{h}(w, x^i) = y^i$ "as much as possible" for all training examples,

Loss function

- Training dataset of I pairs of input/output examples

$$\{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}$$

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

$$\boxed{\mathbf{w}^* = \mathbf{arg\,min_w} \; \Sigma_\mathbf{n} L(\mathbf{y_n}, \mathbf{h}(\mathbf{w},\mathbf{x_n}))}$$

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

Supervised ML algorithm

1. A model $h(\mathbf{x};\mathbf{\Theta})$ (hypothesis class) with parameters $\mathbf{\Theta}$. A particular value of $\mathbf{\Theta}$ determines a particular hypothesis in the class.
   Ex: for linear models, $\mathbf{\Theta}$ = slope w_1 and intercept w_0.
2. A *loss function* $L(\cdot,\cdot)$ to compute the difference between the desired output (label) y_n and our prediction to it $h(\mathbf{x}_n;\mathbf{\Theta})$. *Approximation error (loss)*:

$$E(\mathbf{\Theta};\mathcal{X}) = \sum_{n=1}^{N} L(y_n, h(\mathbf{x}_n;\mathbf{\Theta})) = \text{sum of errors over instances}$$

   Ex: 0/1 loss for classification, squared error for regression.

3. An optimization procedure (learning algorithm) to find parameters $\mathbf{\Theta}^*$ that minimize the error:

$$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta}} E(\mathbf{\Theta};\mathcal{X})$$

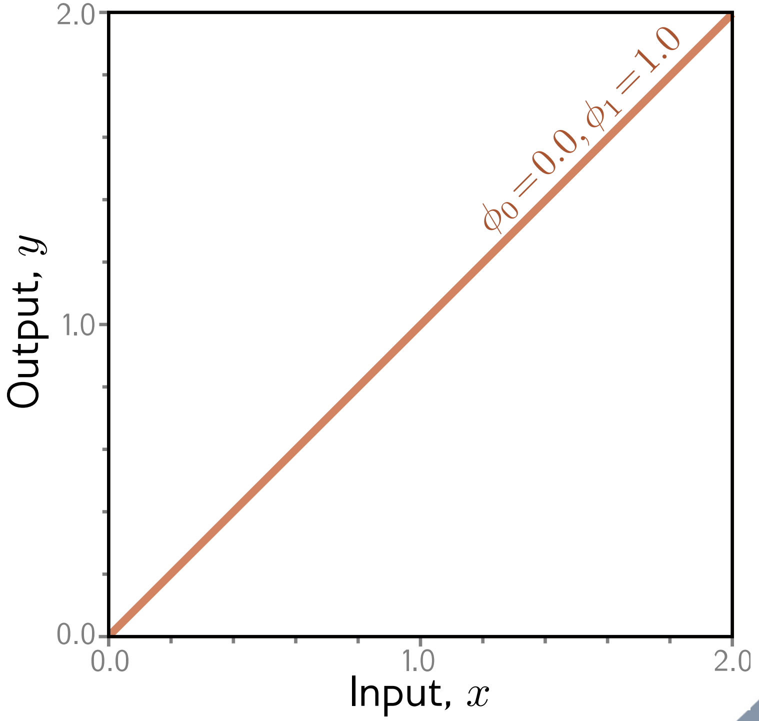
Example: 1D Linear regression

- ❑Model:

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

- ❑Parameters

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix} \begin{array}{l} \leftarrow \text{y-offset} \\ \leftarrow \text{slope} \end{array}$$

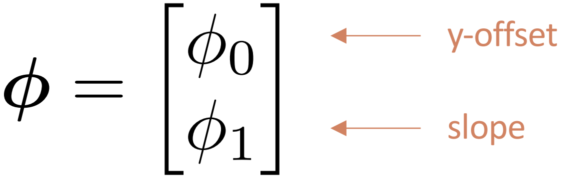


Example: 1D Linear regression

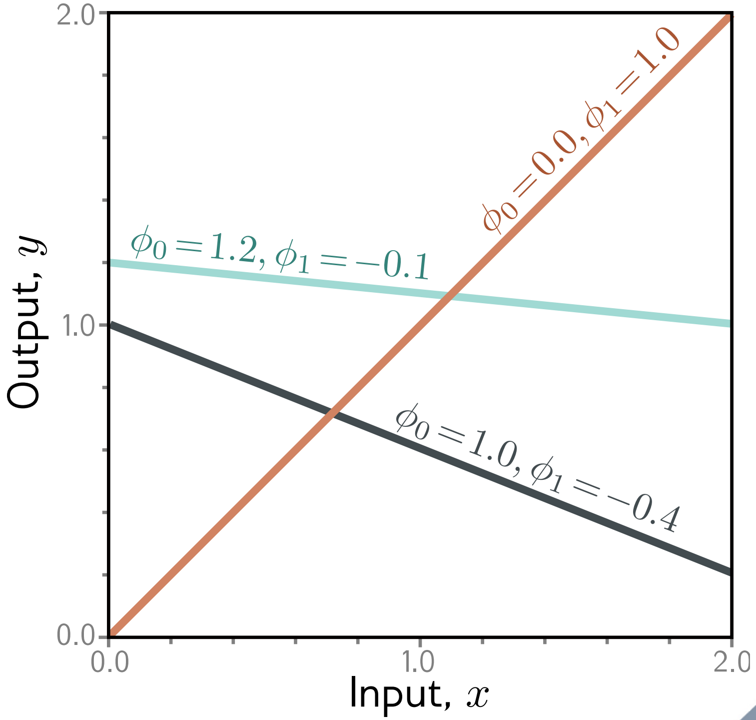
- ❑ Model:

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

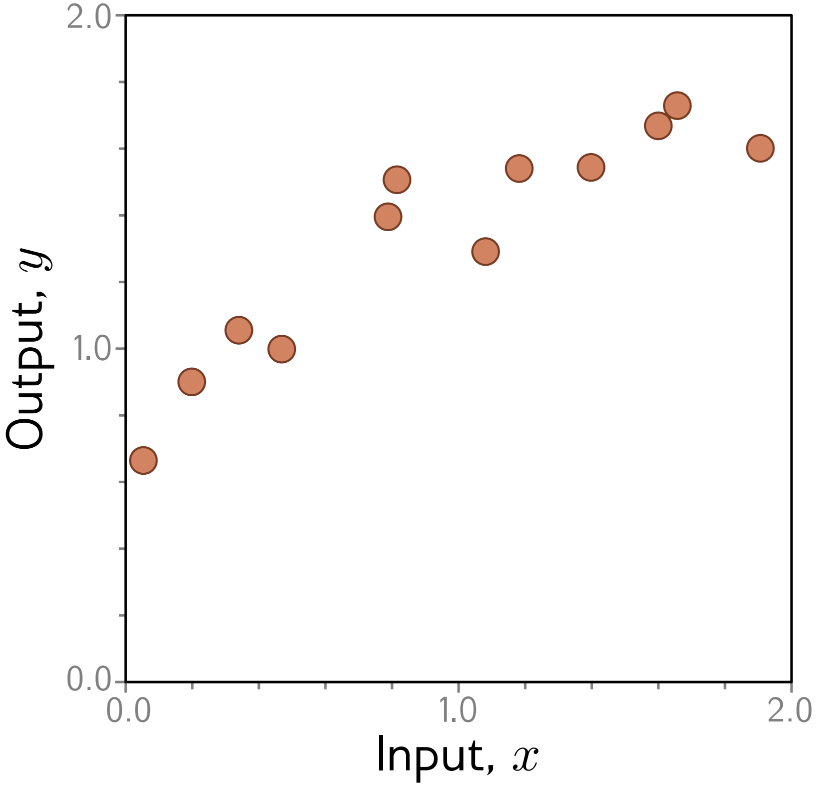
- ❑ Parameters



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



Example: 1D Linear regression training data

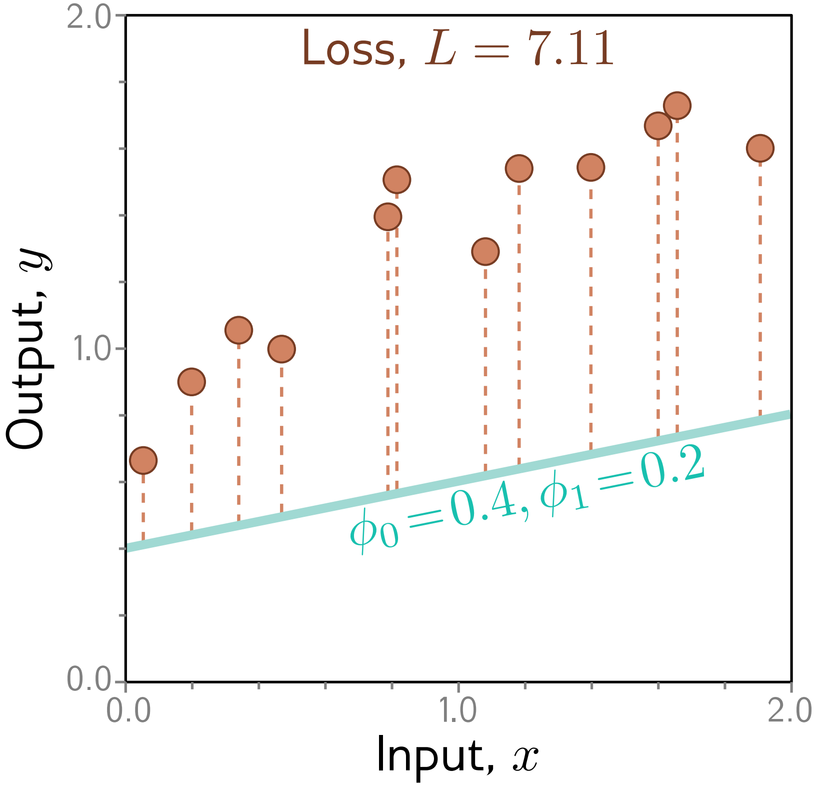


Loss function:

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

"Least squares loss function"

Example: 1D Linear regression training data



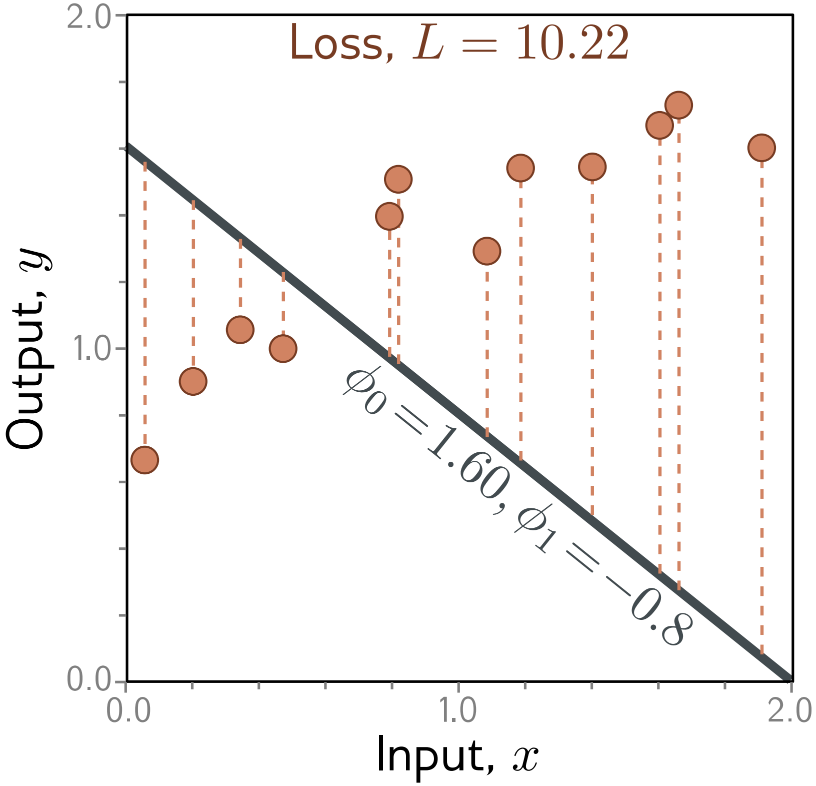
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression training data



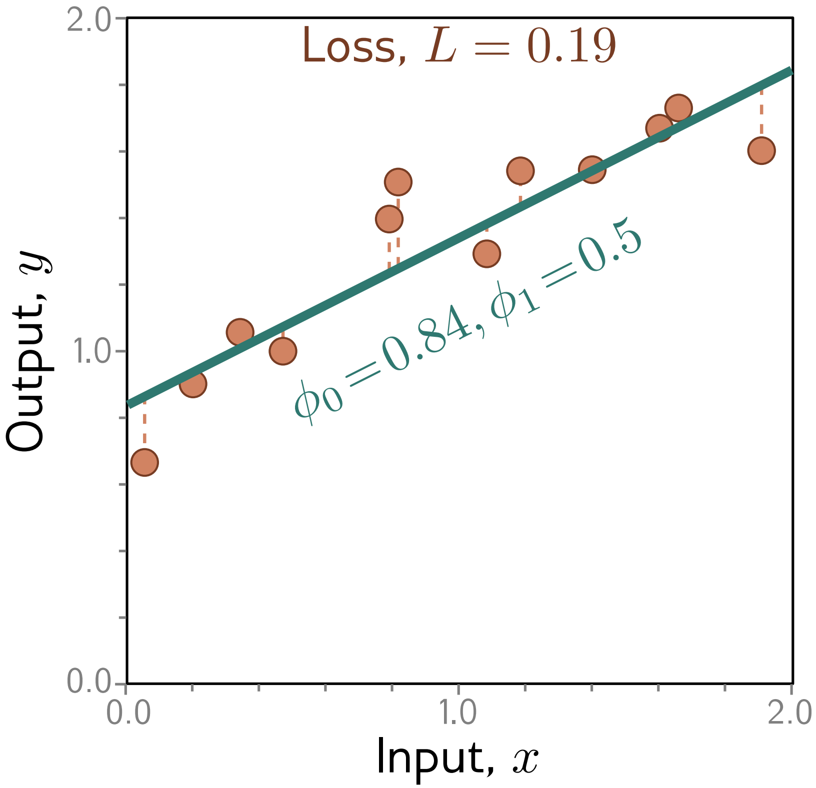
Loss function:

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

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

“Least squares loss function”

Example: 1D Linear regression training data



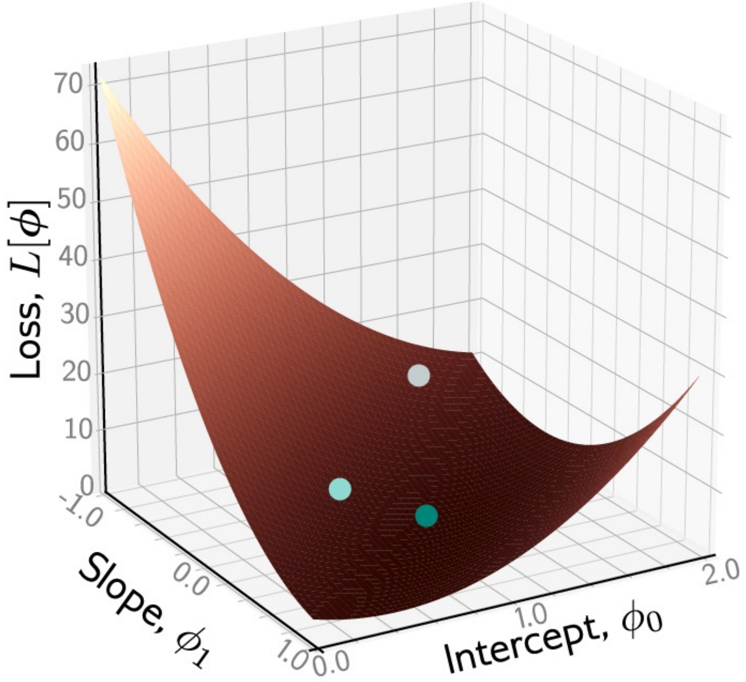
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression loss function

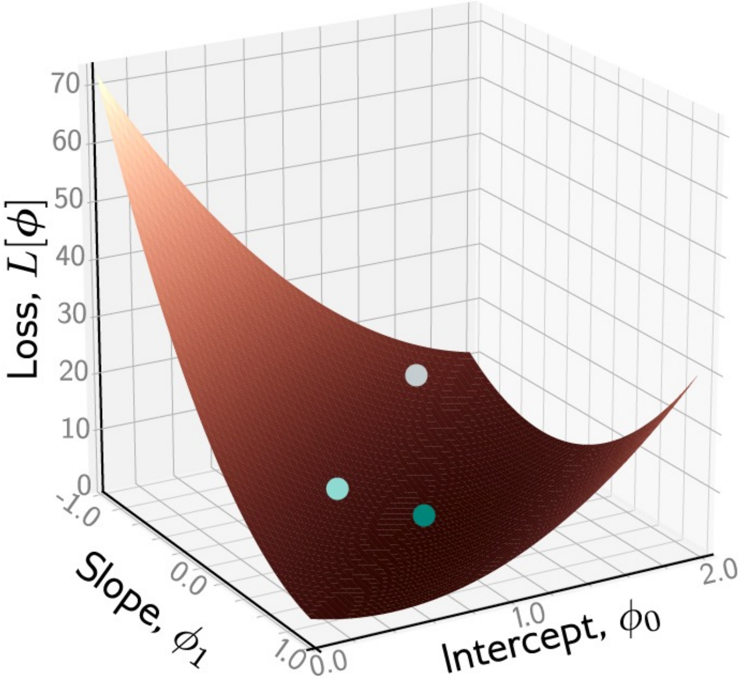


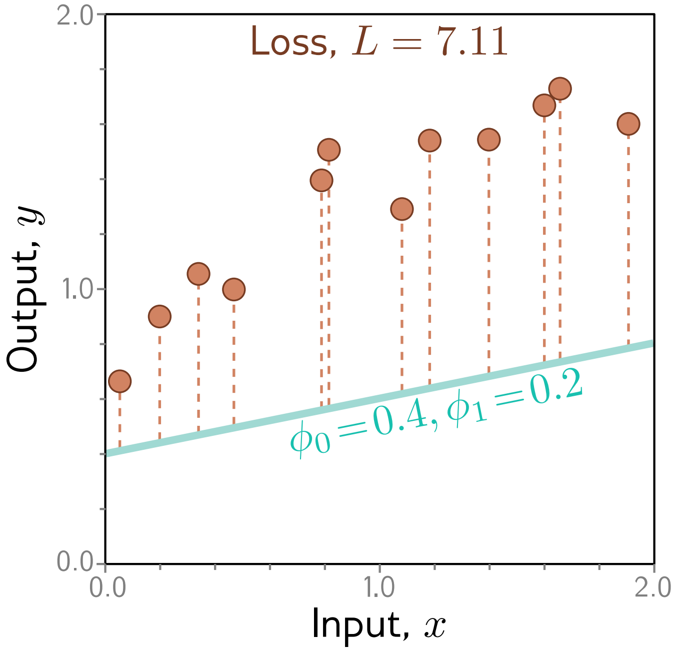
Loss function:

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

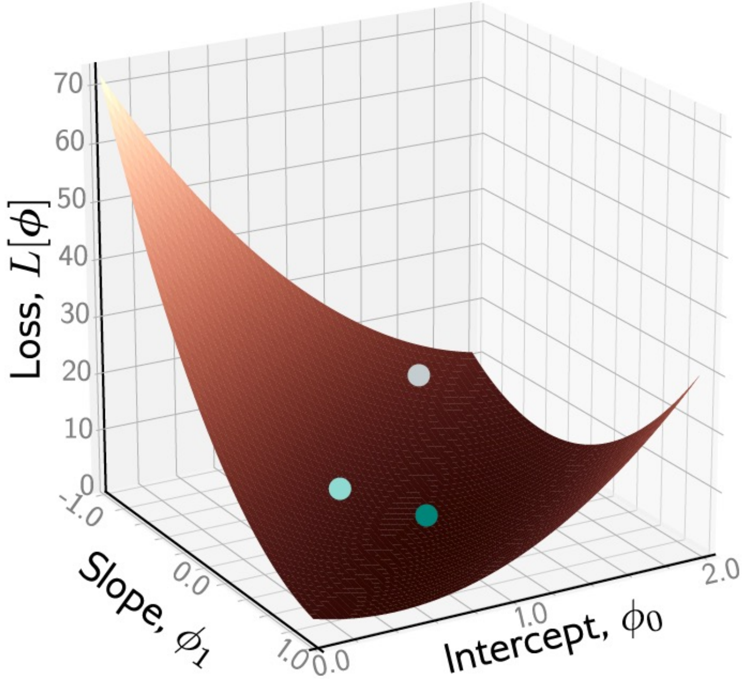
“Least squares loss function”

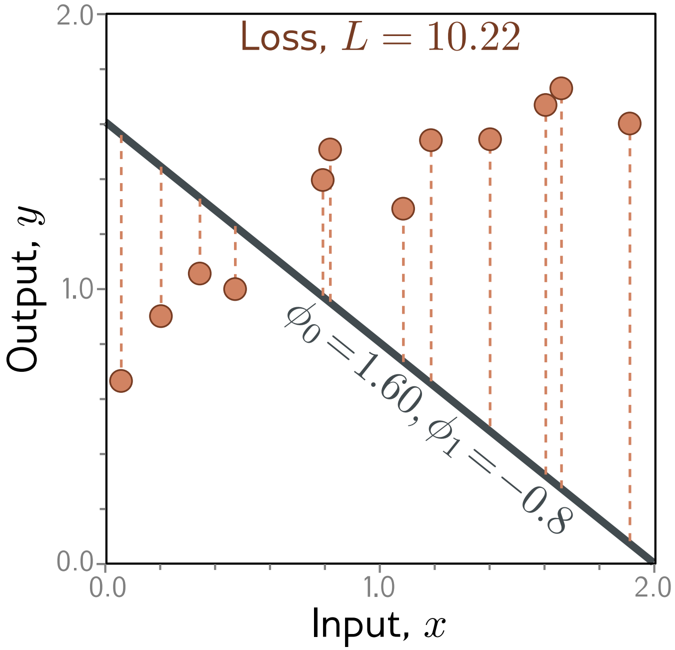
Example: 1D Linear regression loss function



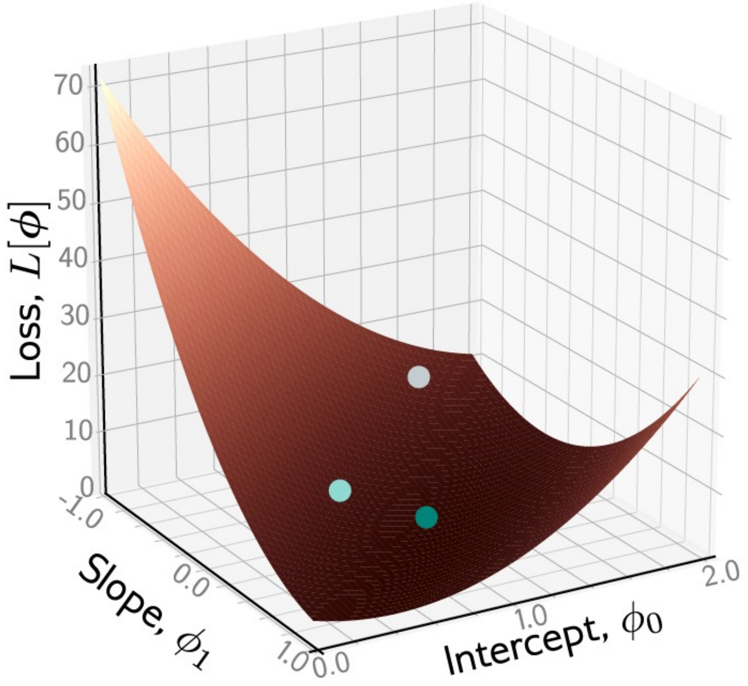


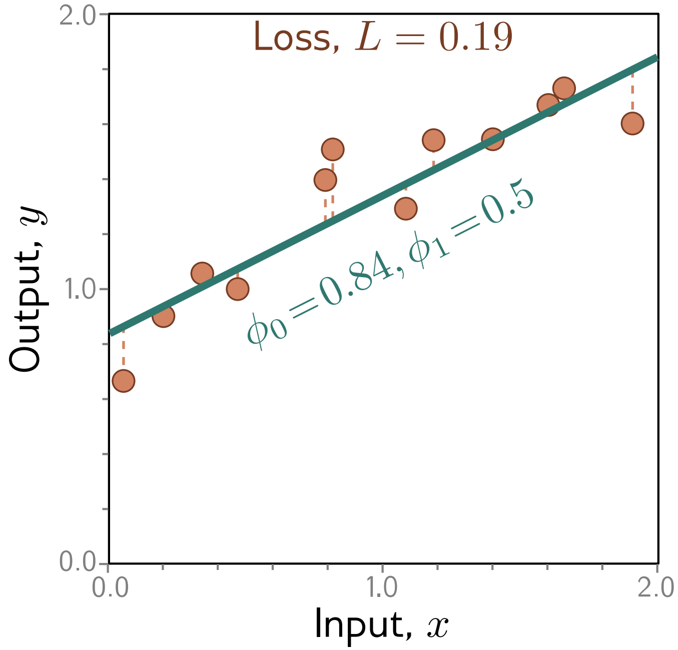
Example: 1D Linear regression loss function



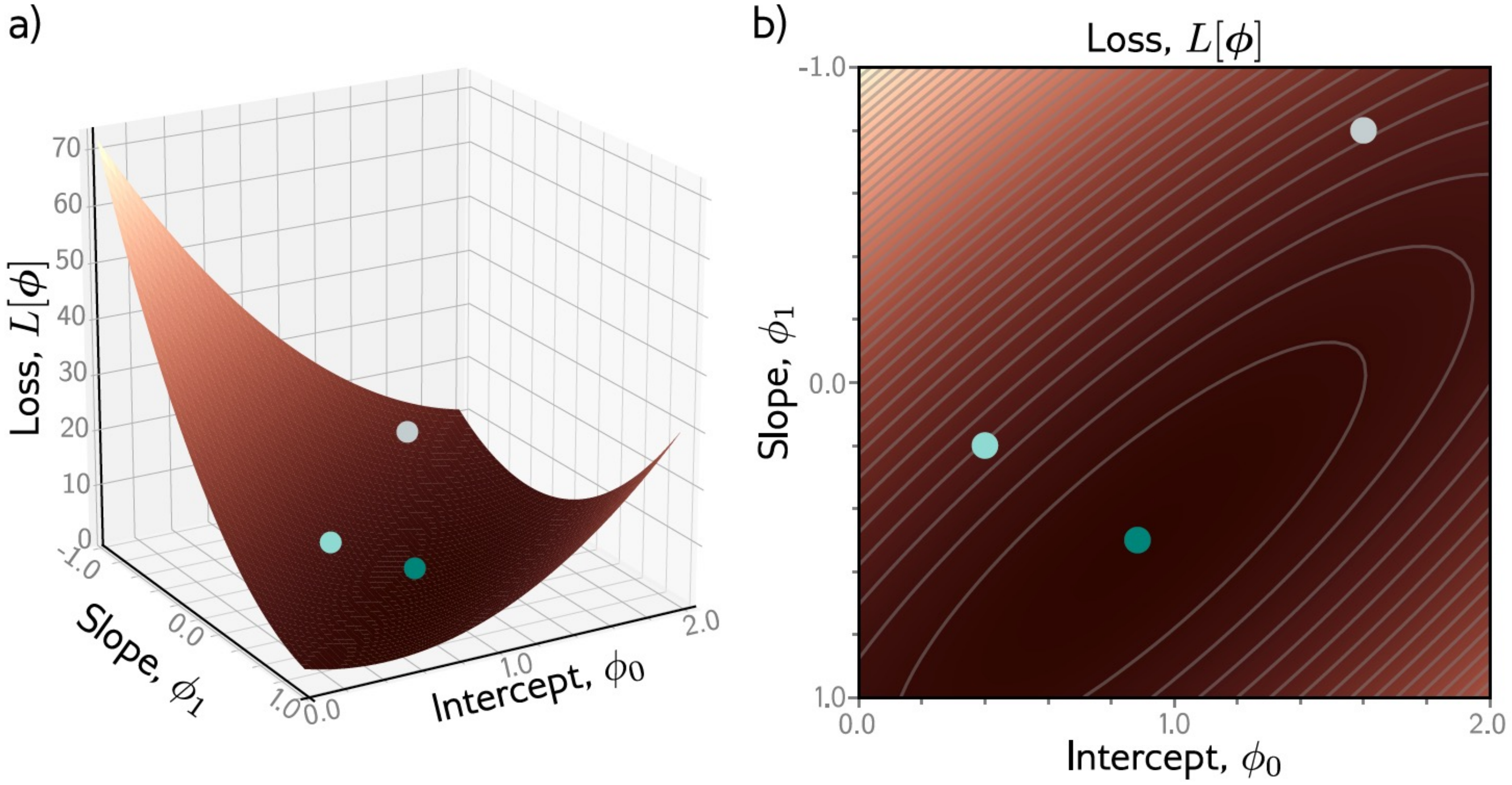


Example: 1D Linear regression loss function

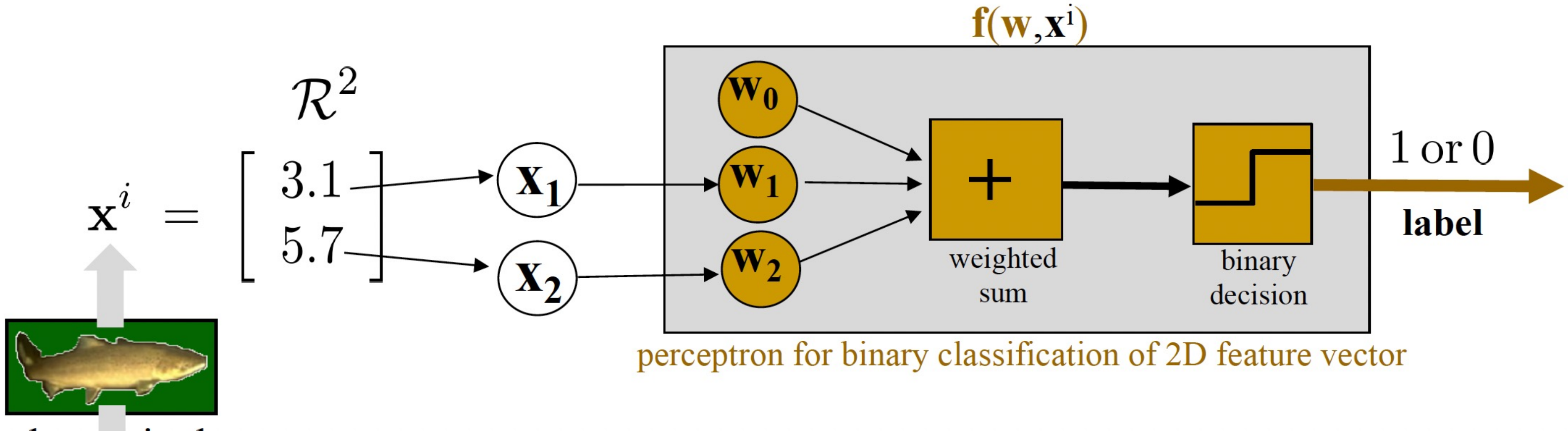




Example: 1D Linear regression loss function



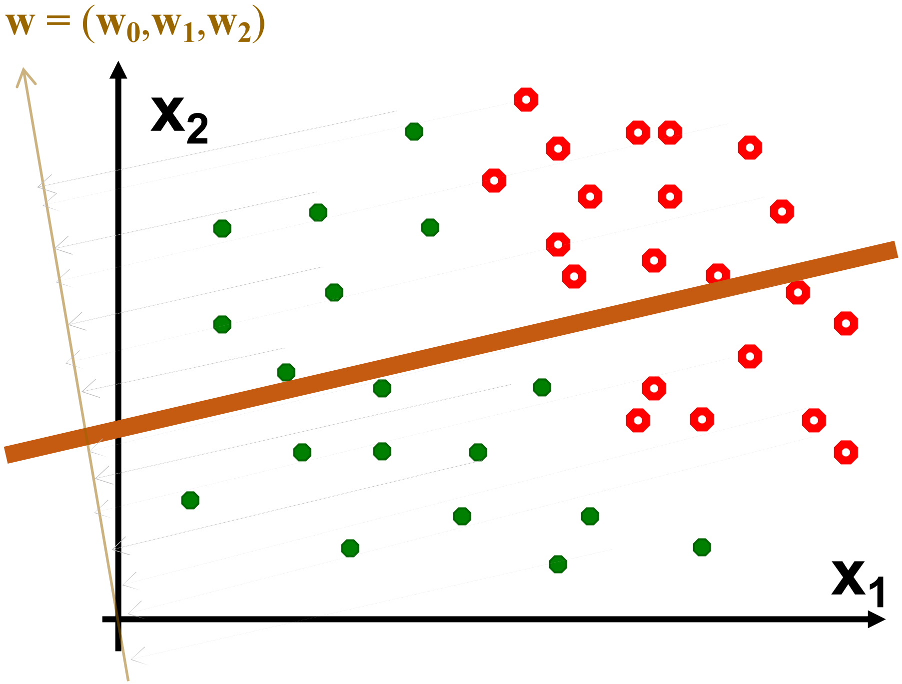
Linear classifier example: *perceptron*



perceptron for binary classification of 2D feature vector

Linear Classifiers

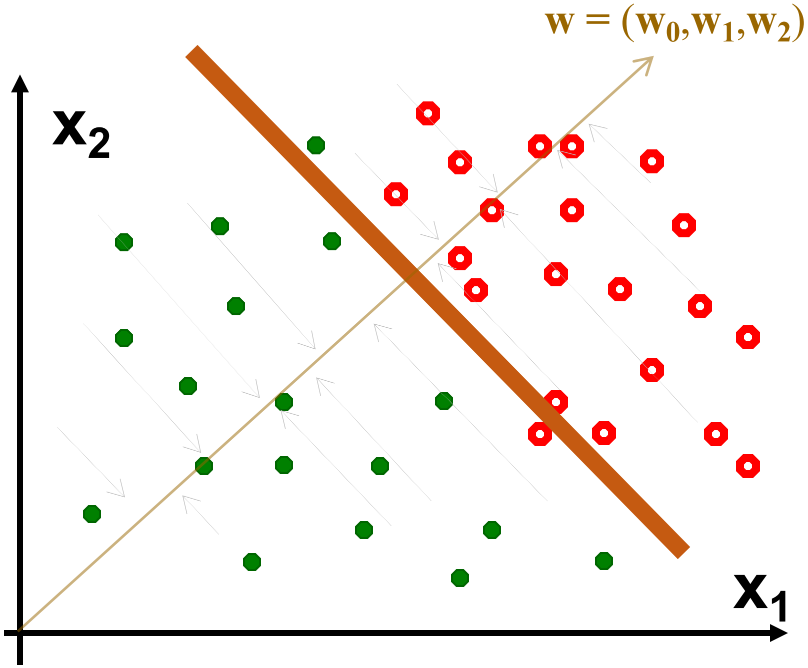
bad **w**



classification error 38%

projected points onto
normal line are all mixed-up

better **w**

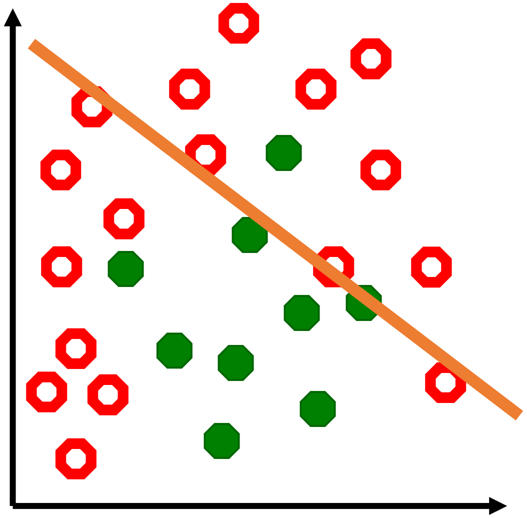


classification error 4%

projected points onto
normal line are well separated

Underfitting

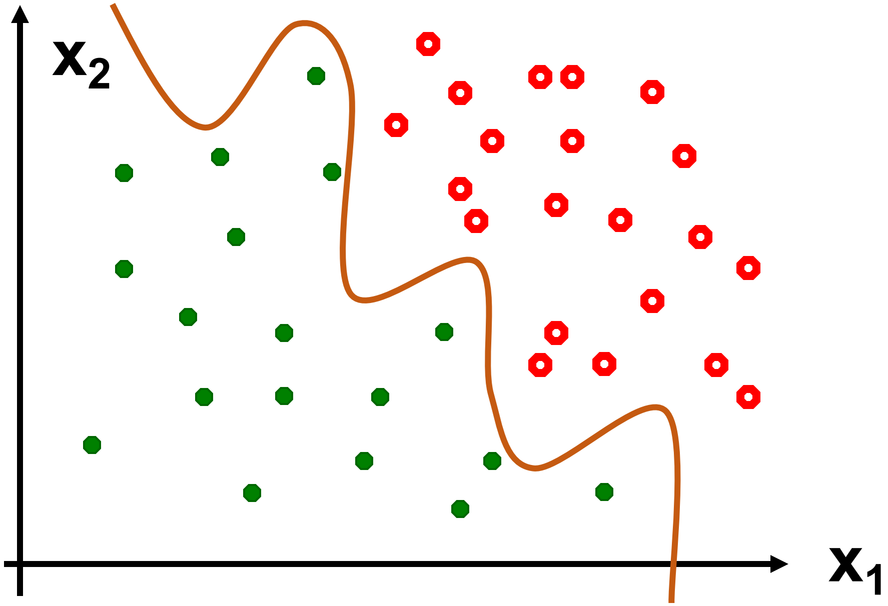
For some types of data
no linear decision boundary
can separate the samples well



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

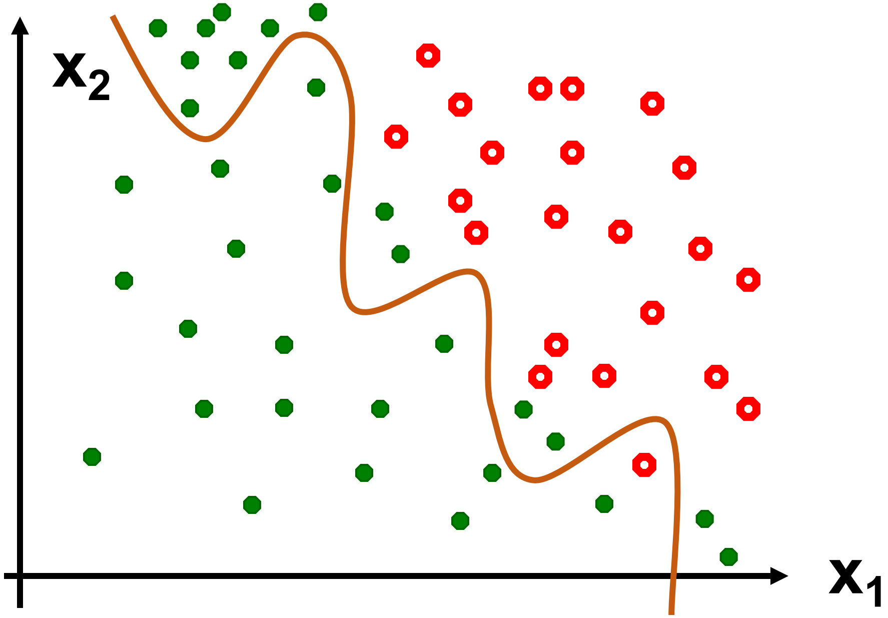
- chosen classifier type (hypothesis space) is not expressive enough

More complex (non-linear) classifiers



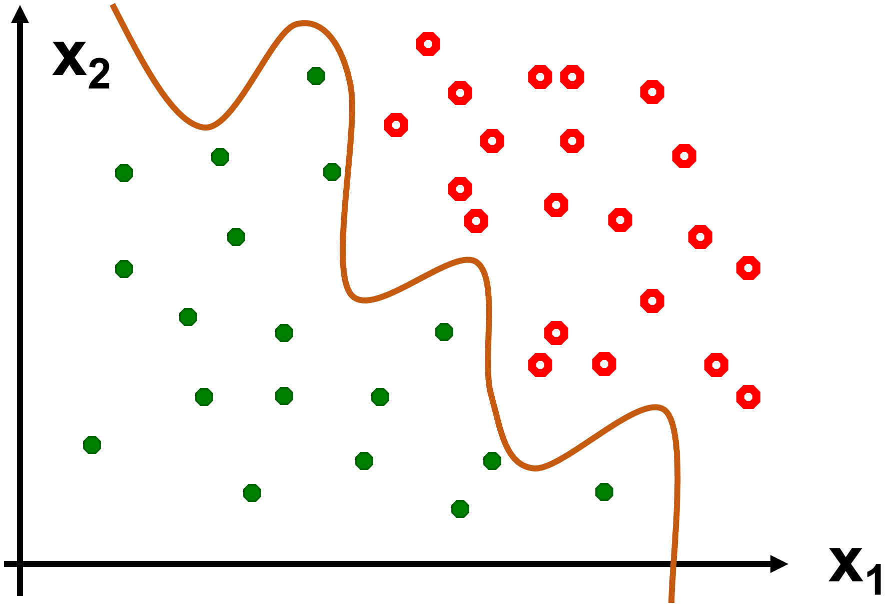
- ☐ for example, if **f**(**w**,**x**) is a polynomial of high degree
- ☐ can achieve 0% classification error

More complex (non-linear) classifiers



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

Overfitting



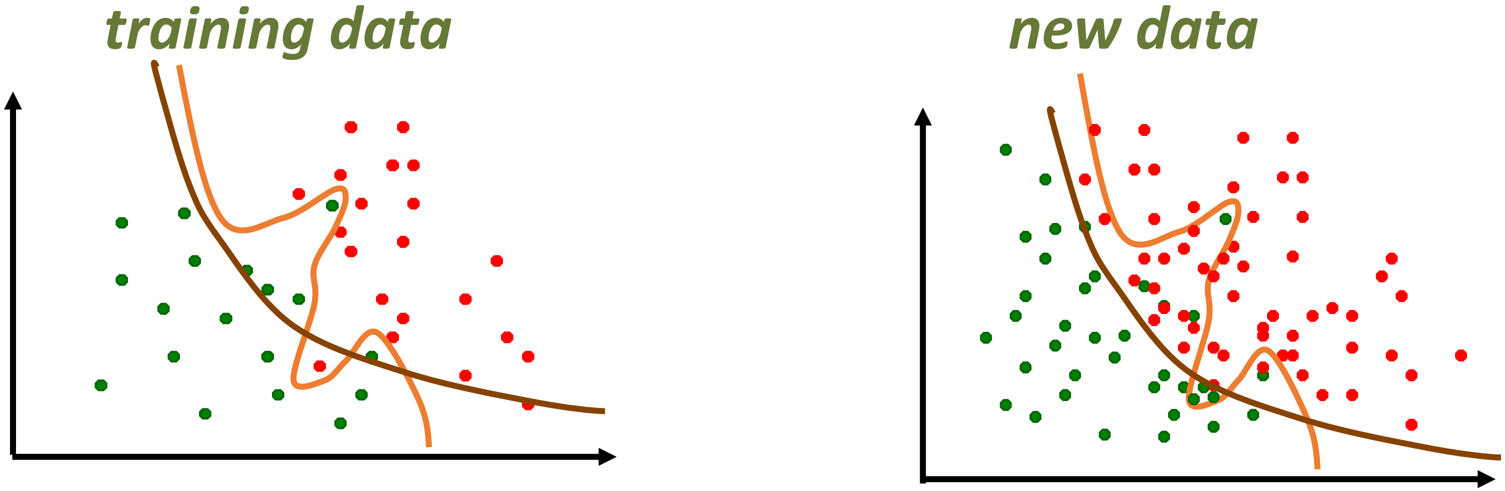
- ☐ Amount of data for training is always limited
- ☐ Complex model often has too many parameters to fit reliably to limited data
- ☐ Complex model may adapt too closely to “random noise” in training data, rather than look at a “big picture”

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

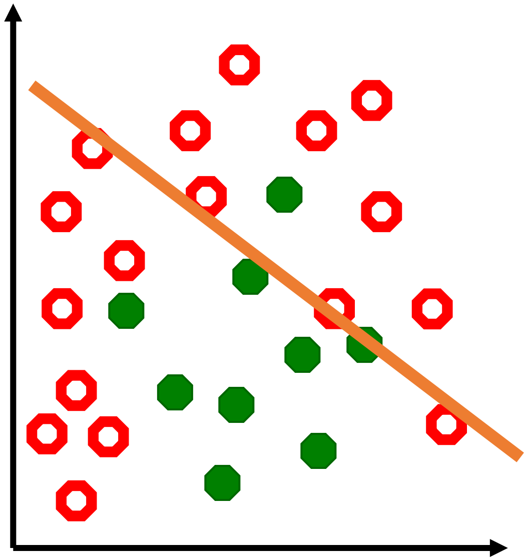
Generalization



- 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

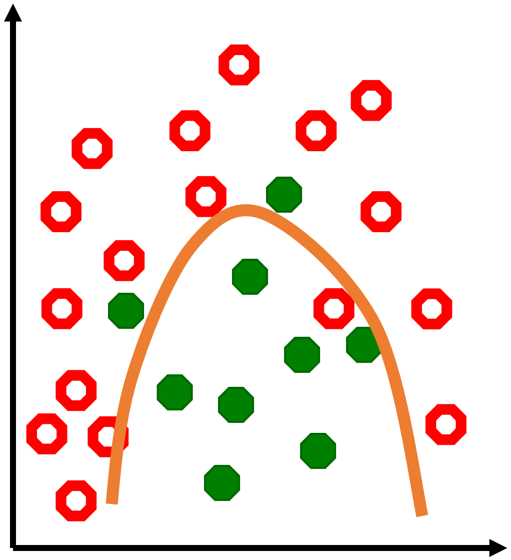
Underfitting → Overfitting

underfitting



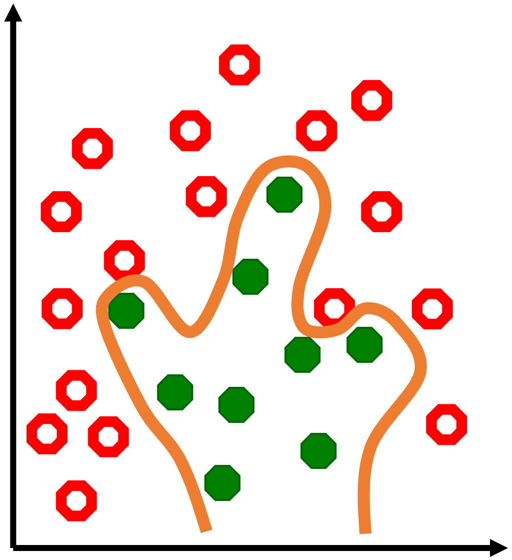
- ☐ high training error
- ☐ high test error

“just right”



- ☐ low training error
- ☐ low test error

overfitting



- ☐ low training error
- ☐ high test error

Model selection and generalization

- Machine learning problems (classification, regression and others) are typically ill-posed : the observed data is finite and does not uniquely determine the classification or regression function.
- How to choose the right inductive bias, in particular the right hypothesis class? This is the *model selection* problem.

Cross Validation

- Training set:
  - Used to train, i.e., to fit a hypothesis $h \in H_i$.
  - Optimize parameters of h given the model structure and hyperparameters.
  - Usually done with an optimization algorithm (the learning algorithm).
- Validation set:
  - Used to minimize the generalization error.
  - Optimize hyperparameters or model structure.
  - Usually done with a “grid search”. Ex: try all values of $H \in \{10, 50, 100\}$ and $\lambda \in \{10-5, 10-3, 10-1\}$.
- Test set:
  - Used to report the generalization error.
  - We optimize nothing on it, we just evaluate the final model on it

Cross Validation

1. For each class H_i, fit its optimal hypothesis h_i using the training set.

2. Of all the optimal hypotheses, pick the one that is most accurate in the validation set.

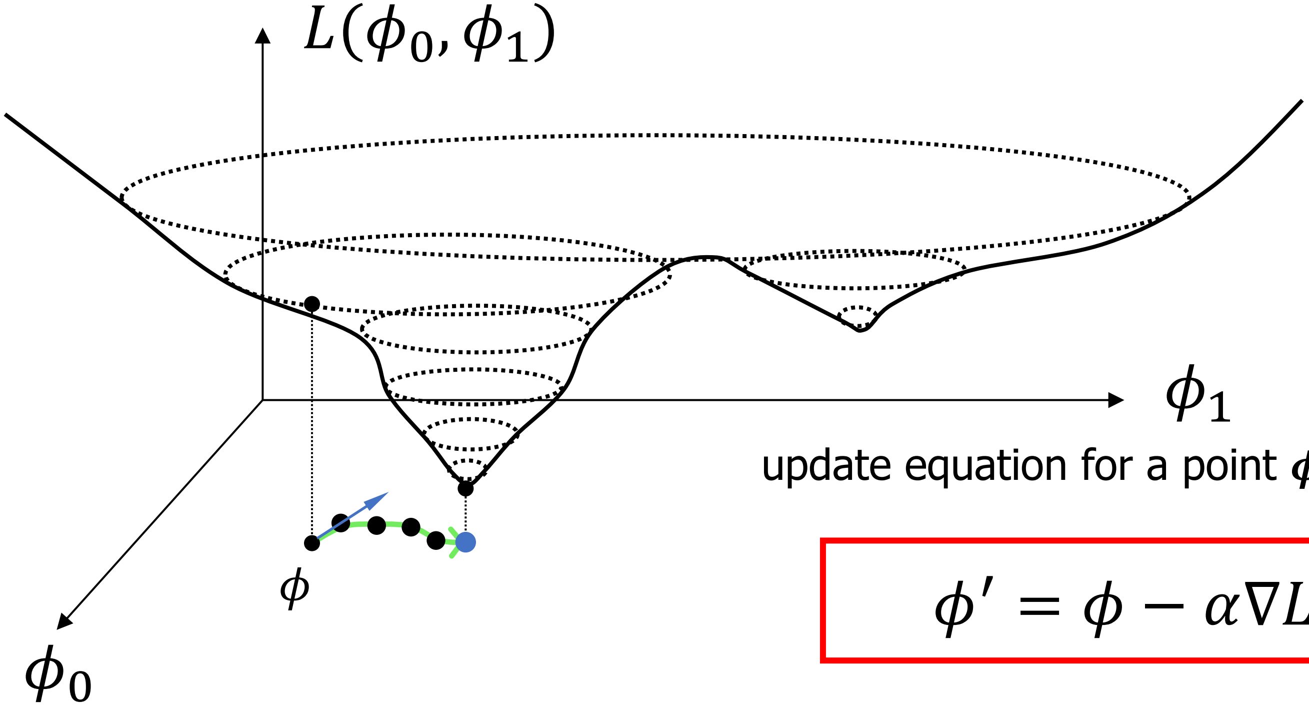
3. Report its error in the test set.

How to design ML algorithm?

- The model class is large enough to contain a good approximation to the underlying function that generated the data in X.
- The learning algorithm is efficient and accurate.
- We must have sufficient training data to pinpoint the right model

Gradient Descent

- ☐ Example: for a function of two variables

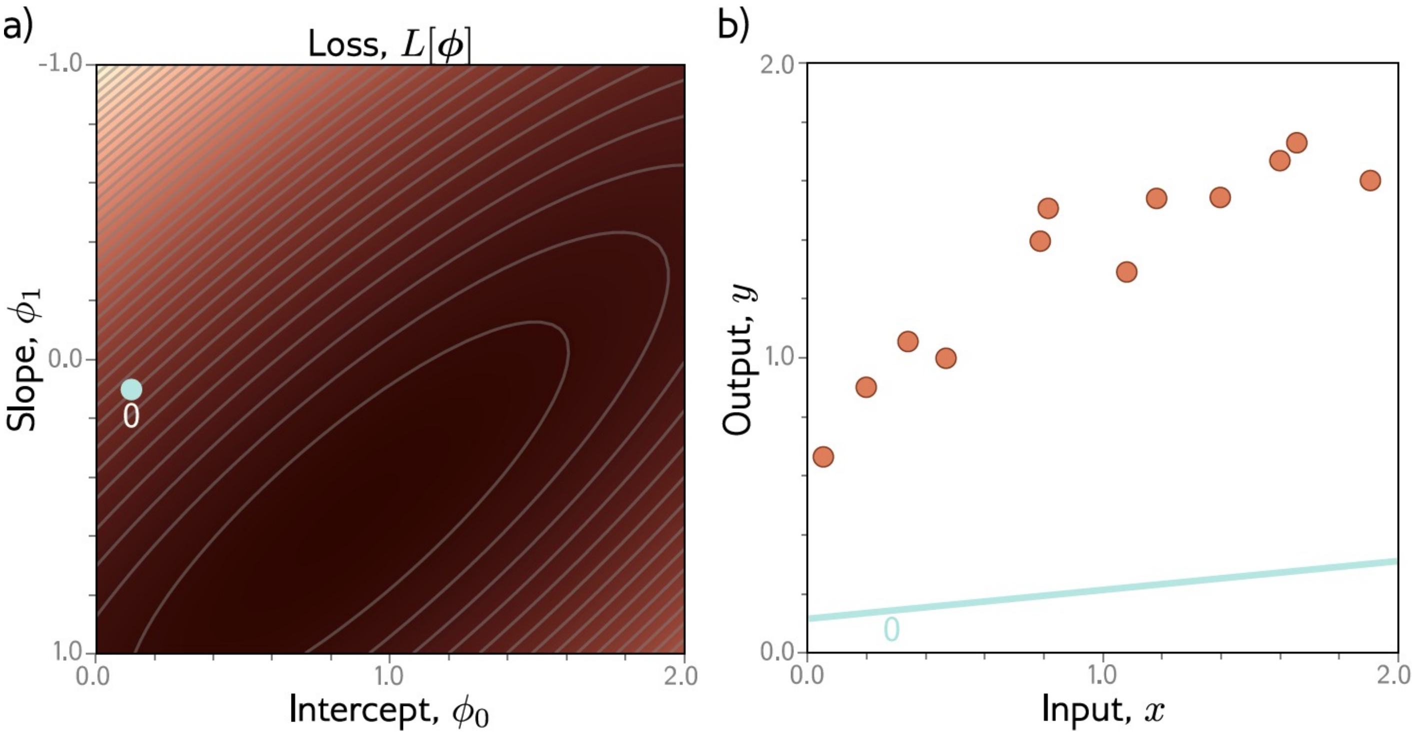


update equation for a point $\boldsymbol{\phi} = [\boldsymbol{\phi}_0, \boldsymbol{\phi}_1]$

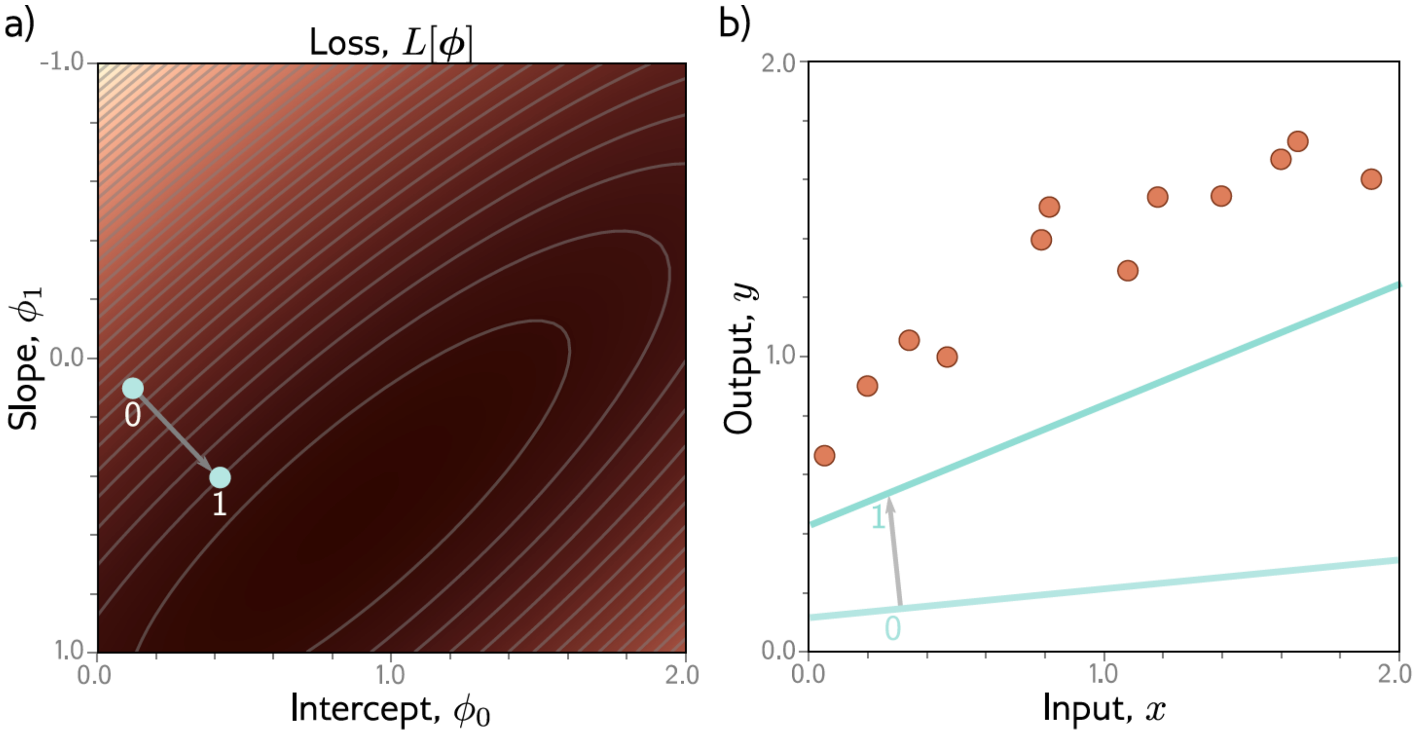
$$\boxed{\phi' = \phi - \alpha \nabla L}$$

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

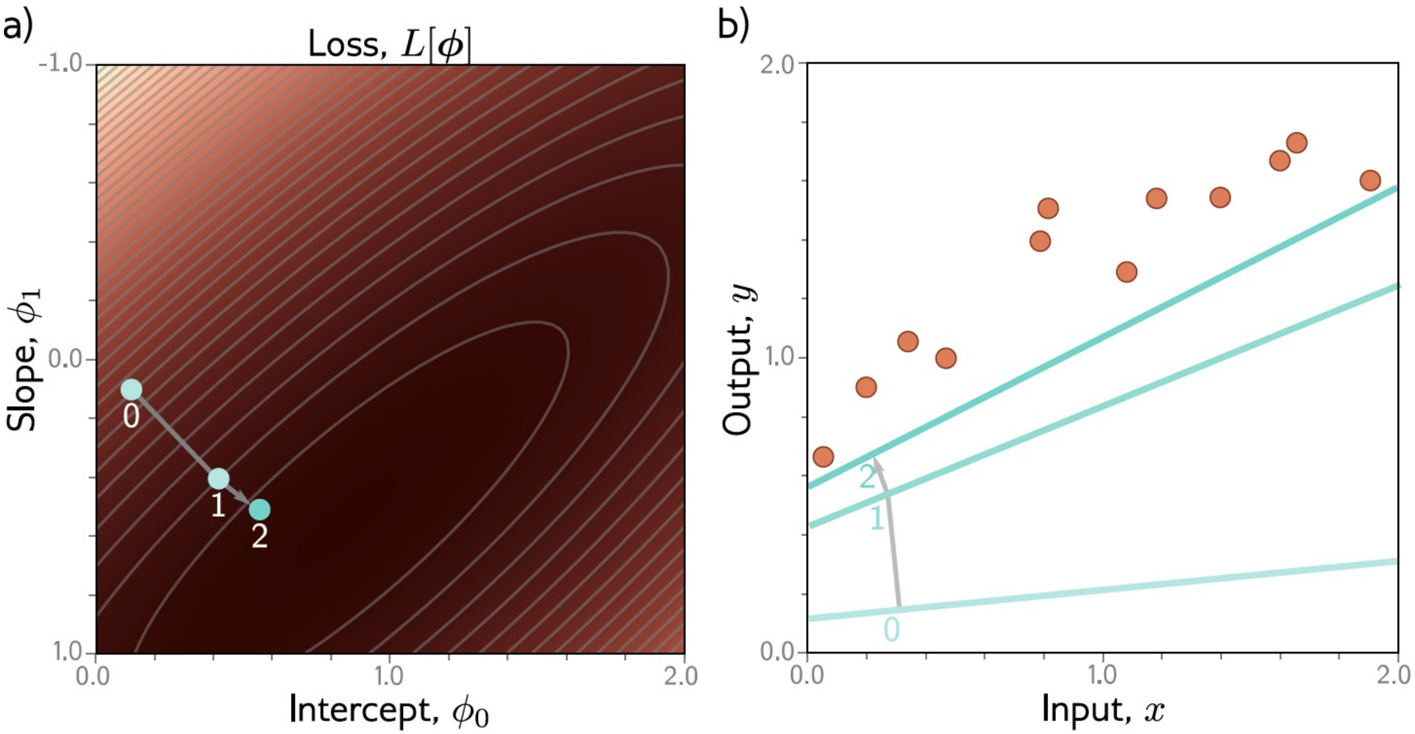
Example: 1D Linear regression training



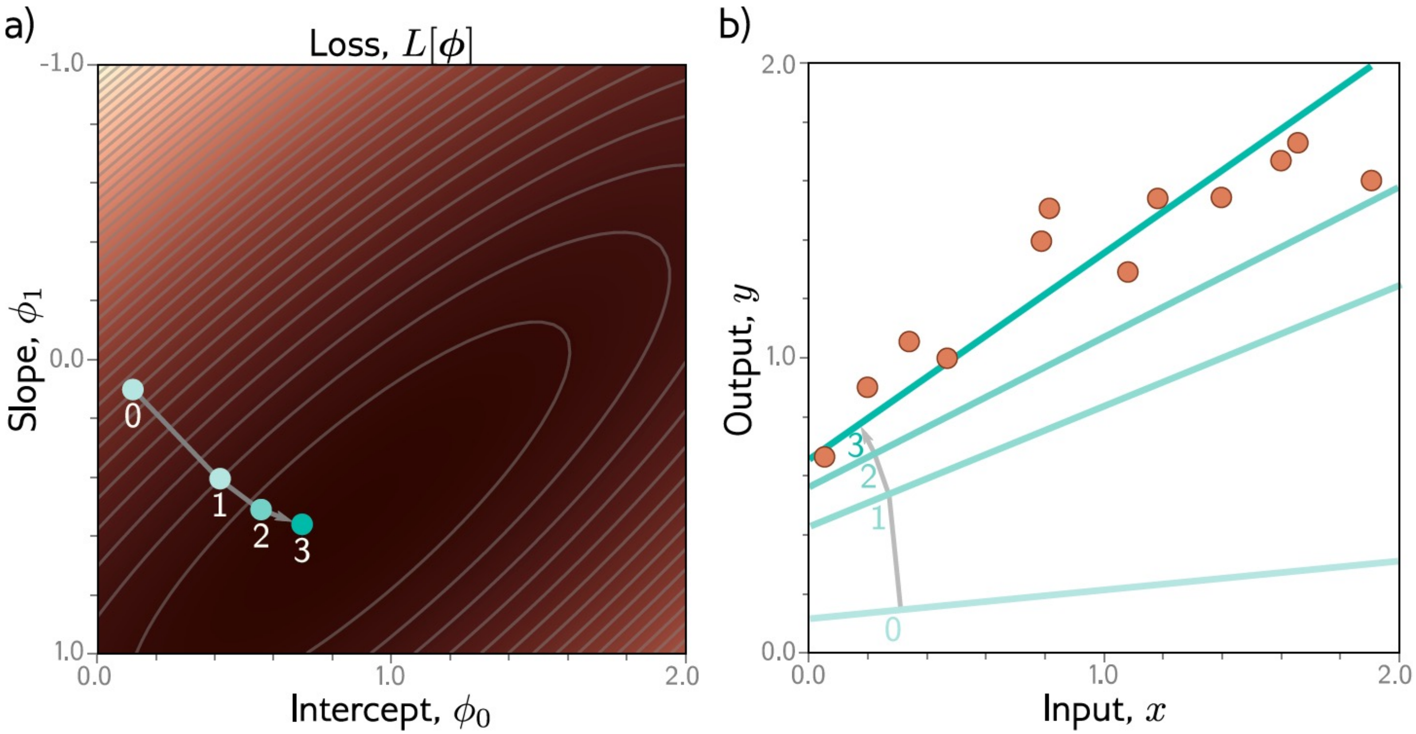
Example: 1D Linear regression training



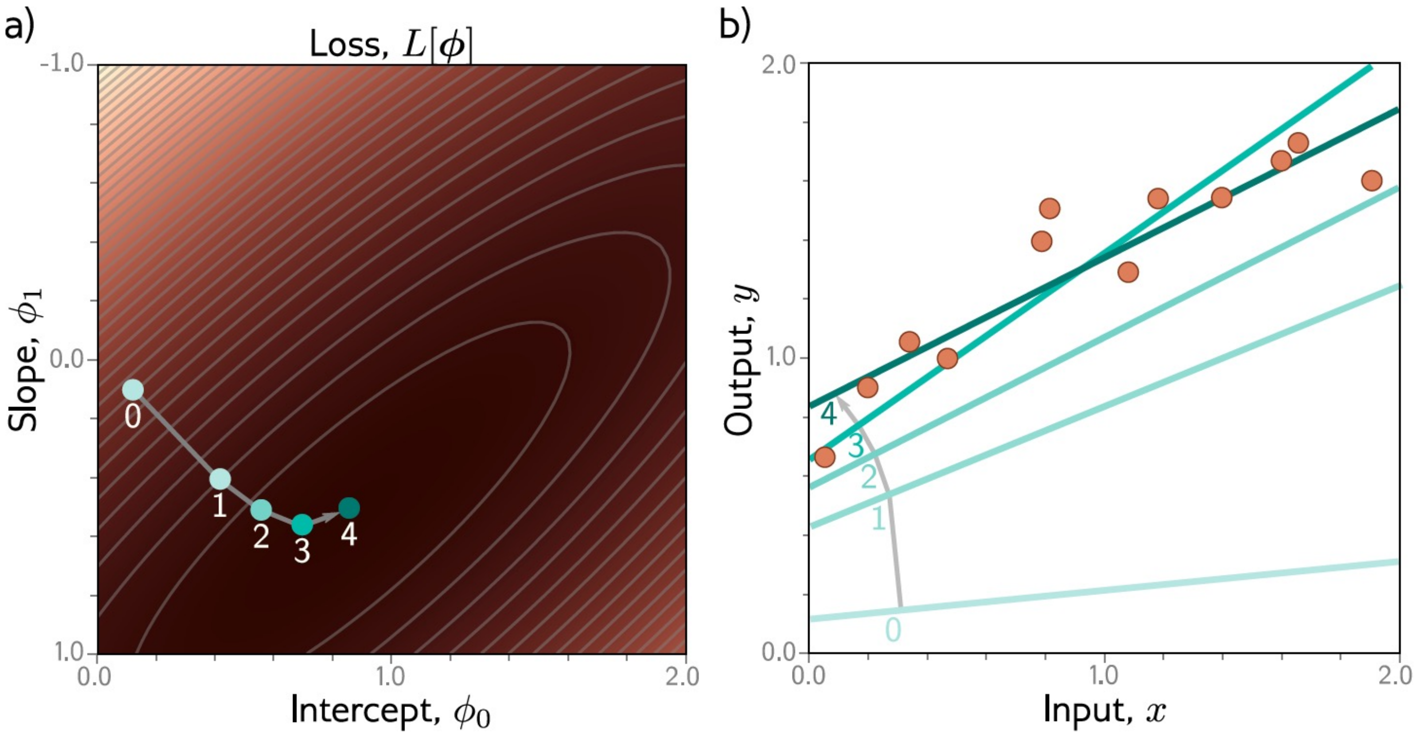
Example: 1D Linear regression training



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