BOSTON UNIVERSITY

Deep Learning for Data Science
DS 542

Lecture 02
Supervised Learning

Slides originally by Thomas Gardos.
Images from Understanding Deep Learning unless otherwise cited.

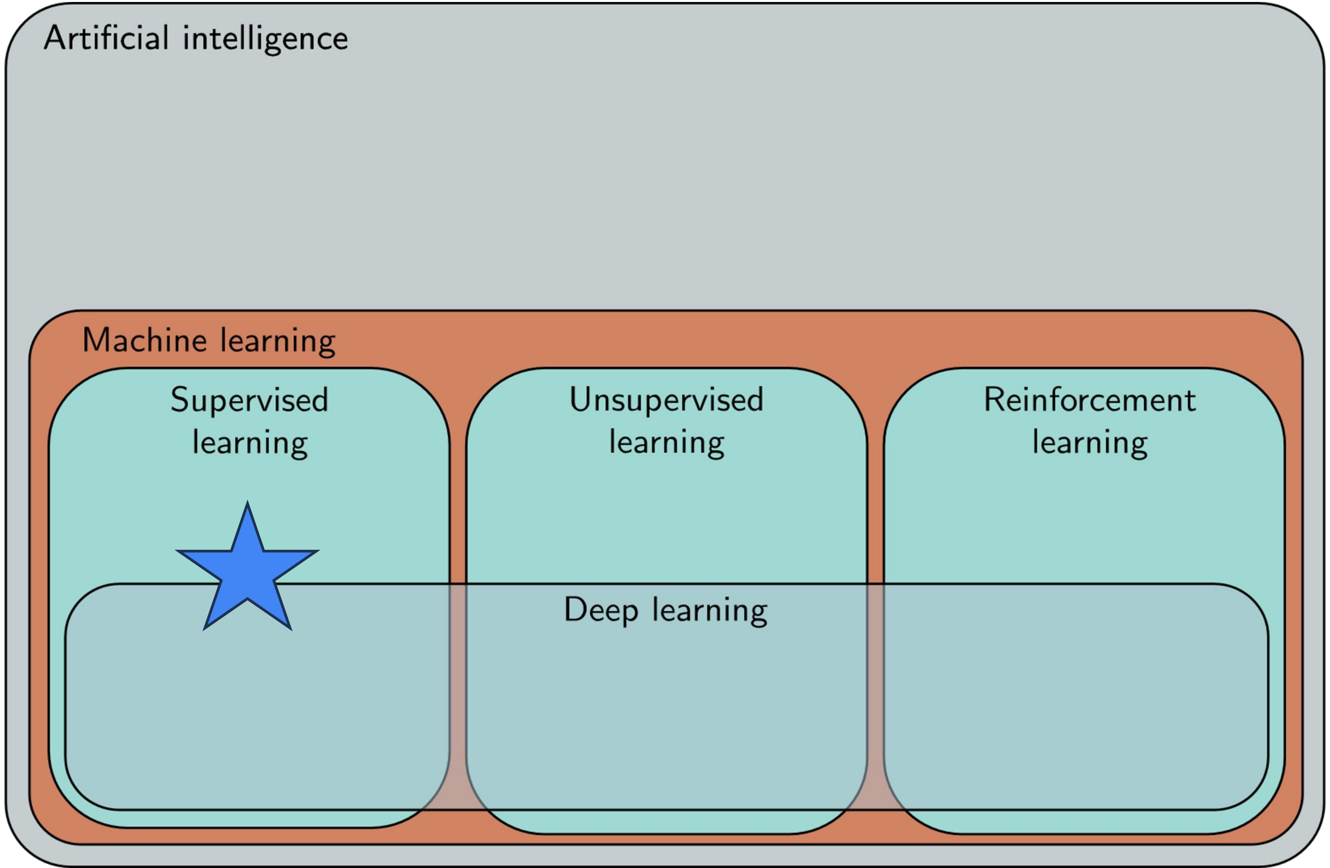
Administrivia

- Slides linked by QR code
- Wednesday office hours
  - Moved to 11-12
- Shared Computing Cluster
  - You should have gotten an email about access last Friday.
  - Discussion section will start covering how to use it this afternoon.
- Homework
  - Notebook 01 posted last week, due Wednesday.
  - Problem Set 02 posted today, due next Monday.
- Links to everything at https://dl4ds.github.io/fa2024/

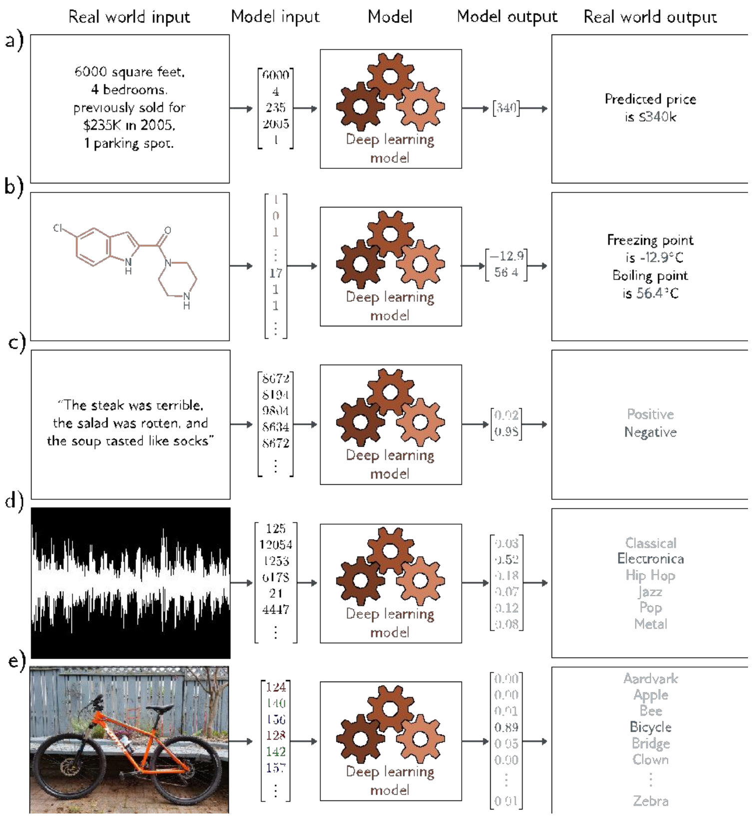
Lecture Outline

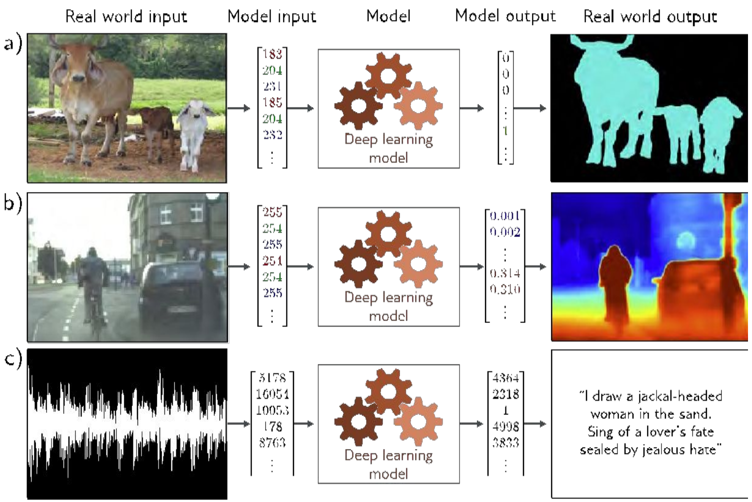
- Supervised Learning
- Preparing Data for Learning
- Where are We Going Next?

Supervised Learning Recap

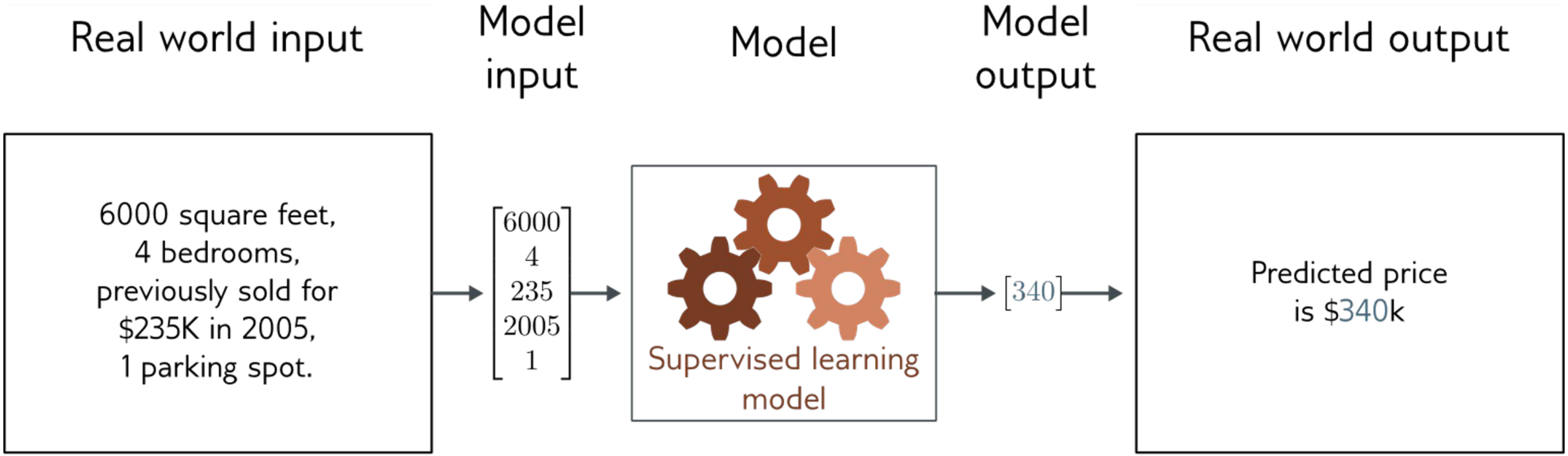


Supervised Learning Applications





Regression



- Univariate regression problem (one output, real value)

Supervised learning

- Overview
- Notation
  - Model
  - Loss function
  - Training
  - Testing
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing
- Where are we going?

Supervised learning

- Overview
- Notation
    - Model
    - Loss function
    - Training
    - Testing
- 1D Linear regression example
    - Model
    - Loss function
    - Training
    - Testing
- Where are we going?

Supervised learning overview

- Supervised learning models
  - Mapping from one or more inputs to one or more outputs. ← functionality
  - Based on example input/output pairs. ← supervision
- What is a model?
  - A family of equations → “inductive bias” (what we chose expecting a good match)
  - Or a specific member of that family
  - Or a code artifact implementing either…

Models and Parameters

- Within a family of models,
  - Individual models are distinguished by parameters.
  - Model outputs are a function of their parameters and the current inputs.
- Model operations
  - Prediction / Inference = computing the outputs from inputs using parameters
  - Training = updating parameters based on a given set of training inputs and outputs
    - Real goal: updated parameters should help predict non-training outputs “well”
    - Proxy goal: updated parameters do help predict training outputs “well”
    - “Empirical risk minimization” is general argument linking these goals.
    - “Well” to be defined…

Supervised learning

- Overview
- Notation
  - Model
  - Loss function
  - Training
  - Testing
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing

Also Appendix A of the book.

Notation:

- Input:

$$\mathbf{x}$$

Variables always Roman letters

Normal lowercase = scalar
Bold lowercase = vector
Capital Bold = matrix

- Output:

$$\mathbf{y}$$

- Model:

$$\mathbf{y} = \mathbf{f}[\mathbf{x}]$$

Functions always square brackets

Normal lower case = returns scalar
Bold lowercase = returns vector
Capital Bold = returns matrix

Notation example:

- Input:

$$\mathbf{x} = \begin{bmatrix} \text{age} \\ \text{mileage} \end{bmatrix}$$

Vector: Structured or tabular data

- Output:

$$y = \begin{bmatrix} \text{price} \end{bmatrix}$$

Scalar output

- Model:

$$y = \text{f}[\mathbf{x}]$$

Scalar output function (with vector input)

Model

- Parameters:

$$\phi$$

Parameters always Greek letters

- Model :

$$\mathbf{y} = \mathbf{f}[\mathbf{x}, \phi]$$

Data Set and Loss Function

- Training dataset of I pairs of input/output examples:

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

Data Set and 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:

$$L\Big[\underbrace{\phi, \mathrm{f}[\mathbf{x}, \phi]}_{\text{model}}, \underbrace{\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}}_{\text{train data}}\Big]$$

Data Set and 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 a model is:

$$L\Big[\underbrace{\phi, \mathrm{f}[\mathbf{x}, \phi]}_{\text{model}}, \underbrace{\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}}_{\text{train data}}\Big]$$

or for short:

$$L[\phi]$$

Returns a scalar that is smaller when model maps inputs to outputs better

Training

- Loss function:

$$L[\phi]$$

Returns a scalar that is smaller when model maps inputs to outputs better

- Find the parameters that minimize the loss:

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \Big[\mathrm{L}\left[\phi\right] \Big]$$

Supervised Learning with scikit-learn (we will use pytorch)

Easy to code up what we've seen so far -

```
model = sklearn.linear_model.LinearRegression(...)

model.fit(X, y)

model.predict(X)
```

Works for many off the shelf models, if

- there is existing code for the model family of interest, and
- the data is small enough to load at once, and
- the loss function is right for your application, and …

Testing (and evaluating)

- To test the model, run on a separate test dataset of input / output pairs
- See how well it generalizes to new data

Supervised learning

- Overview
- Notation
  - Model
  - Loss function
  - Training
  - Testing
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing

Example: 1D Linear Regression Model

- Model:

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

- Parameters

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

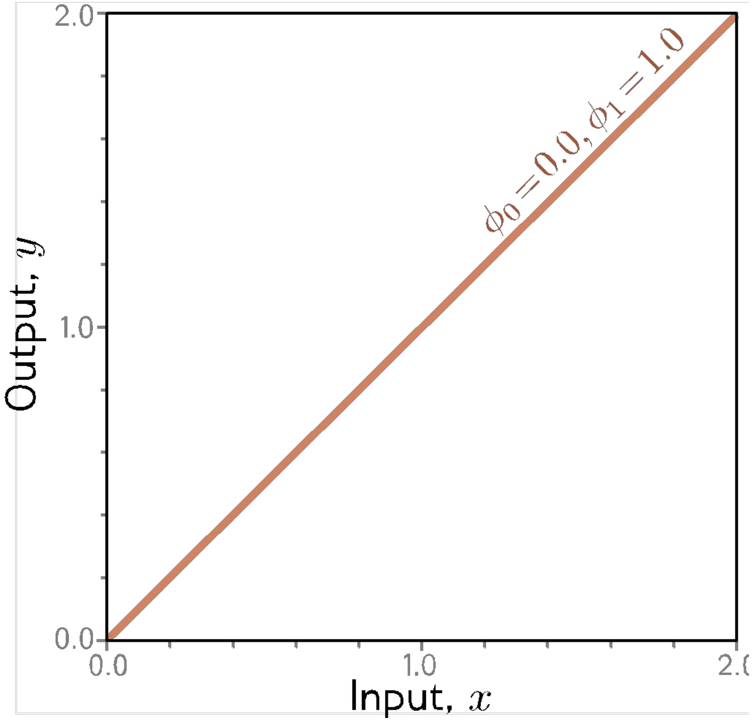
Example: 1D Linear Regression Model

- Model:

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

- Parameters

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



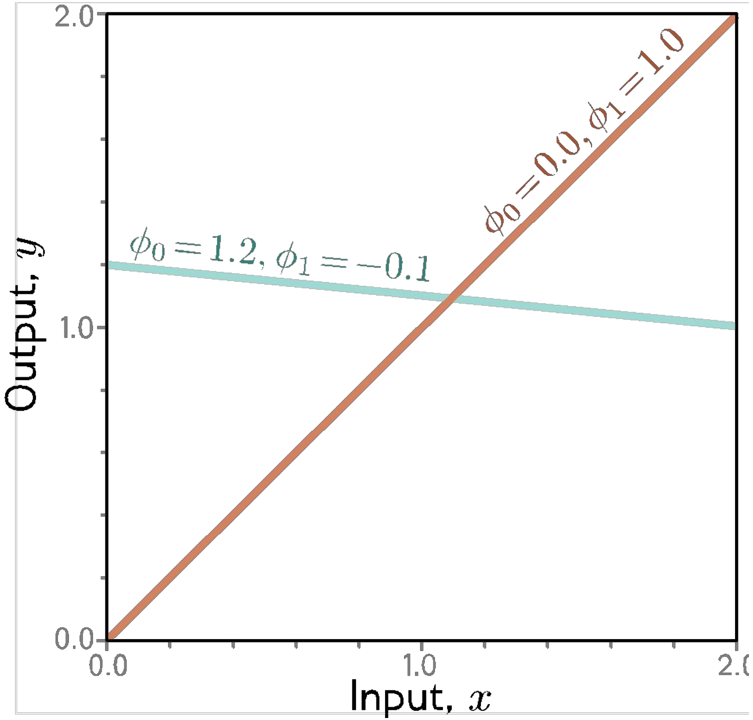
Example: 1D Linear Regression Model

- Model:

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

- Parameters

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



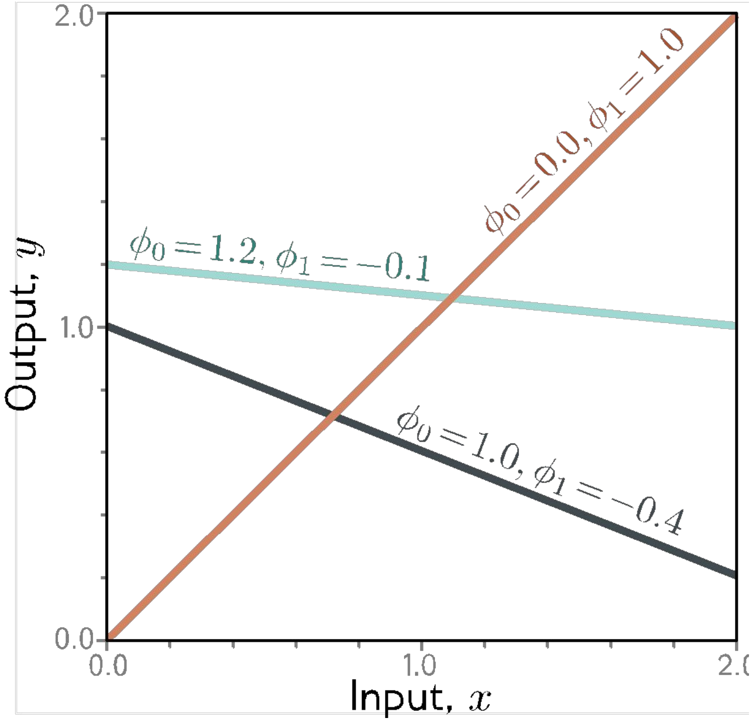
Example: 1D Linear Regression Model

- Model:

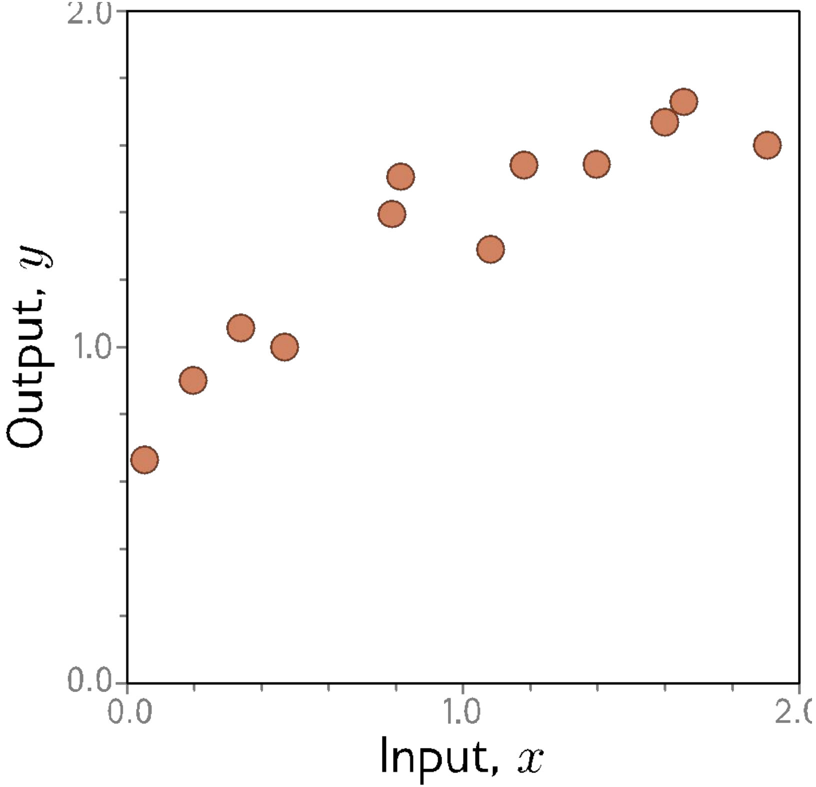
$$
\begin{aligned}
y &= \mathrm{f}[x, \boldsymbol{\phi}] \\
&= \phi_0 + \phi_1 x
\end{aligned}
$$

- Parameters

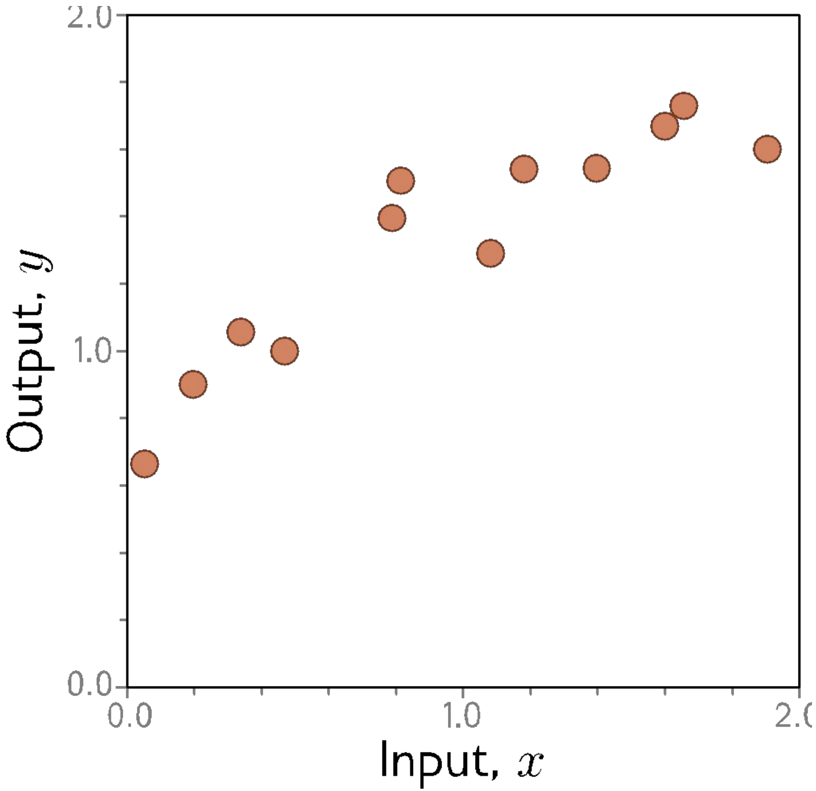
$$
\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}
\begin{array}{l} \longleftarrow \text{y-offset} \\ \longleftarrow \text{slope} \end{array}
$$



Example: 1D Linear Regression Training Data



Example: 1D Linear Regression Loss Function

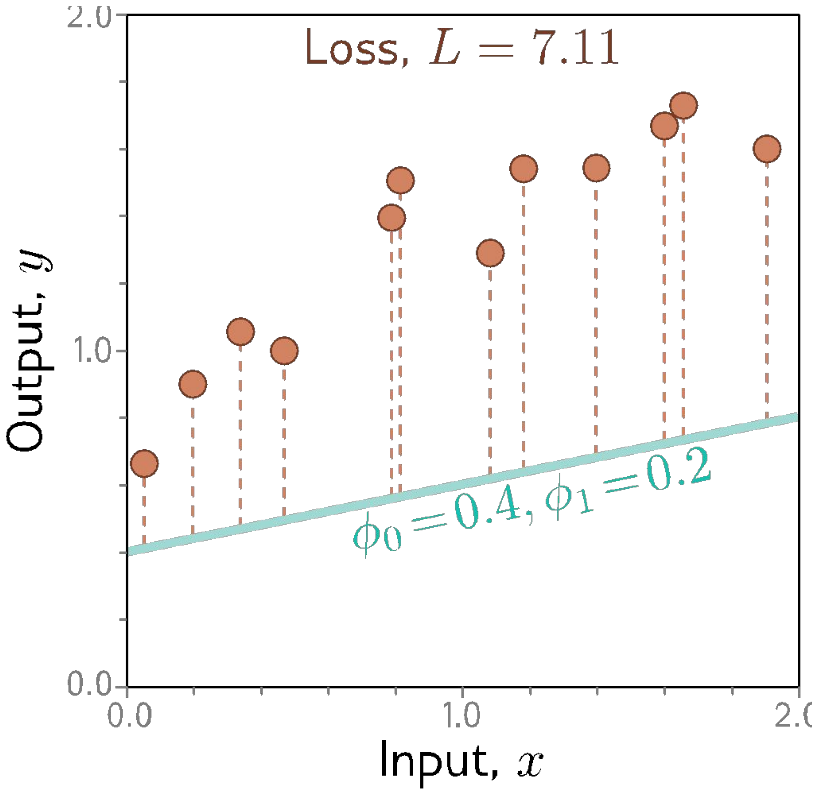


Loss function:

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

"Least squares loss function"

Example: 1D Linear Regression Loss Function



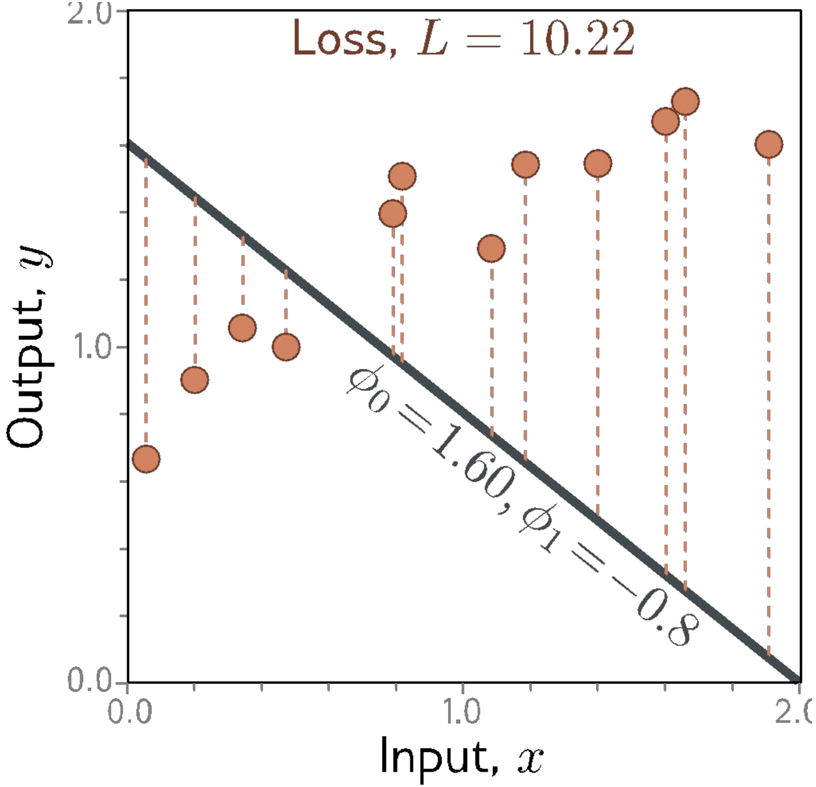
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear Regression Loss Function



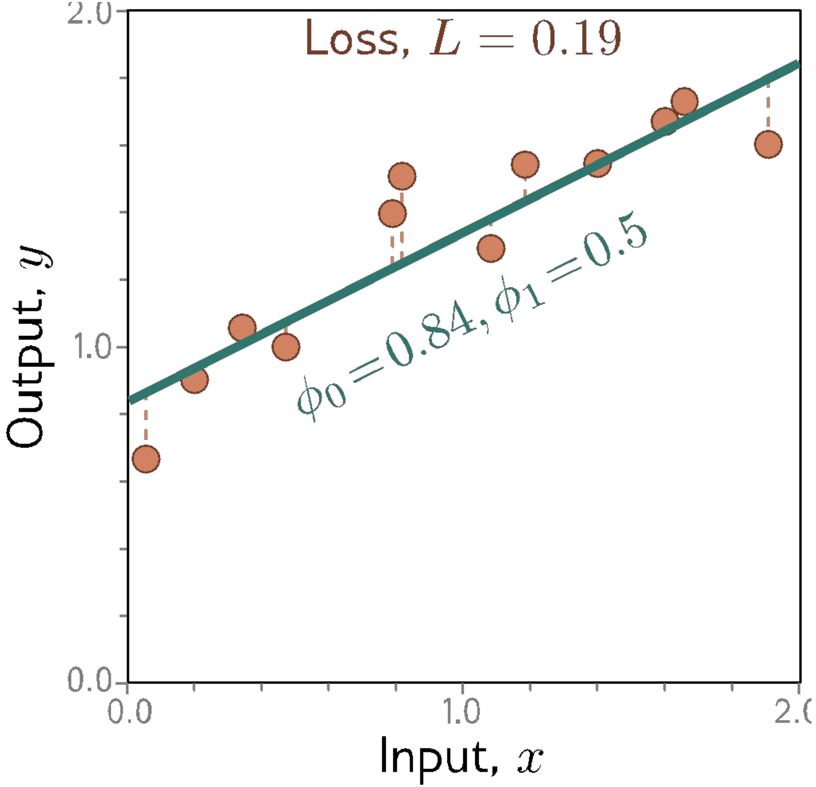
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear Regression Loss Function



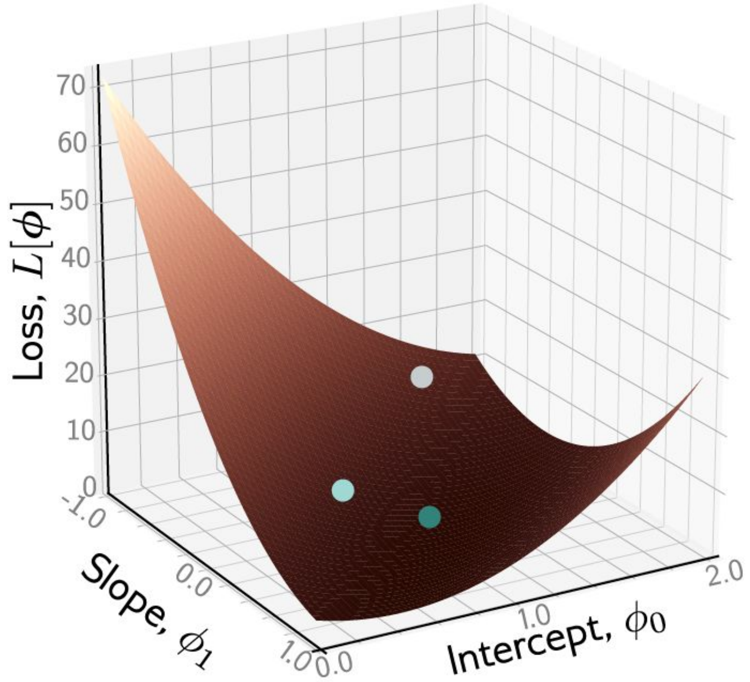
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear Regression Loss Function



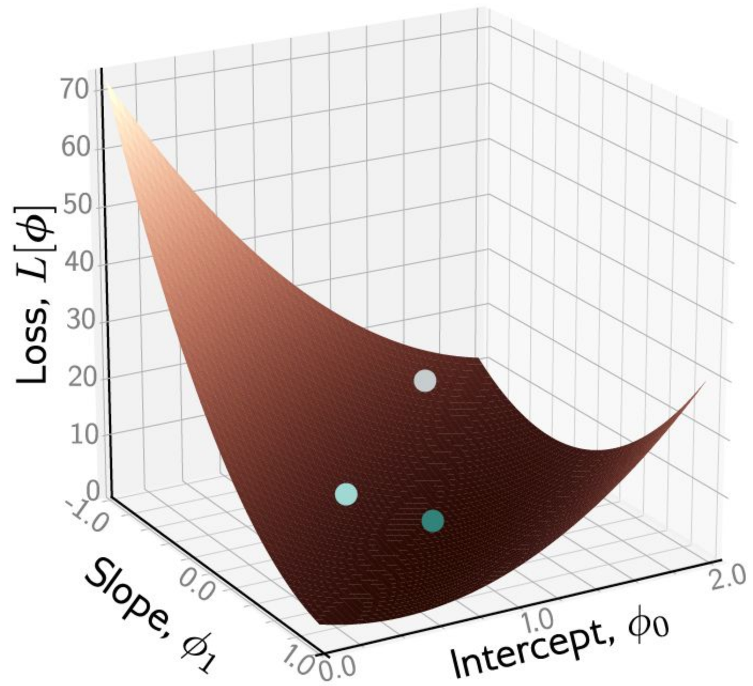
Loss function:

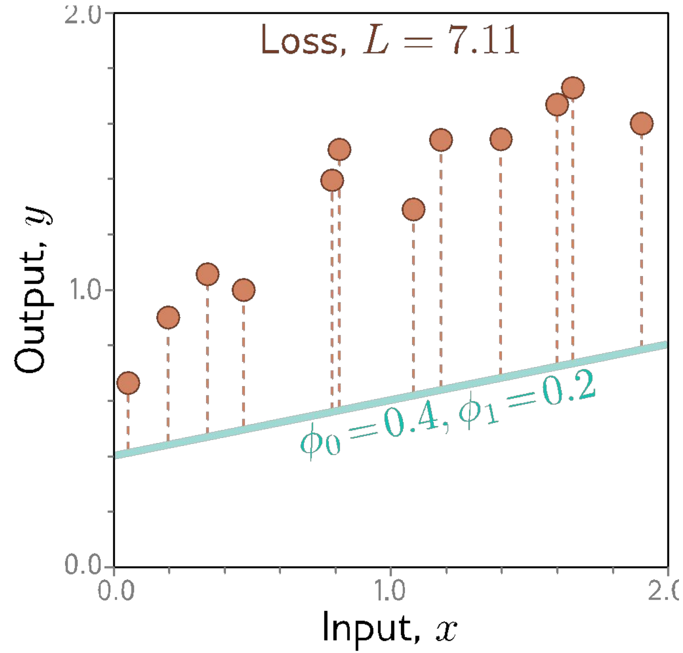
$$L[\phi] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \phi] - y_i)^2$$

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

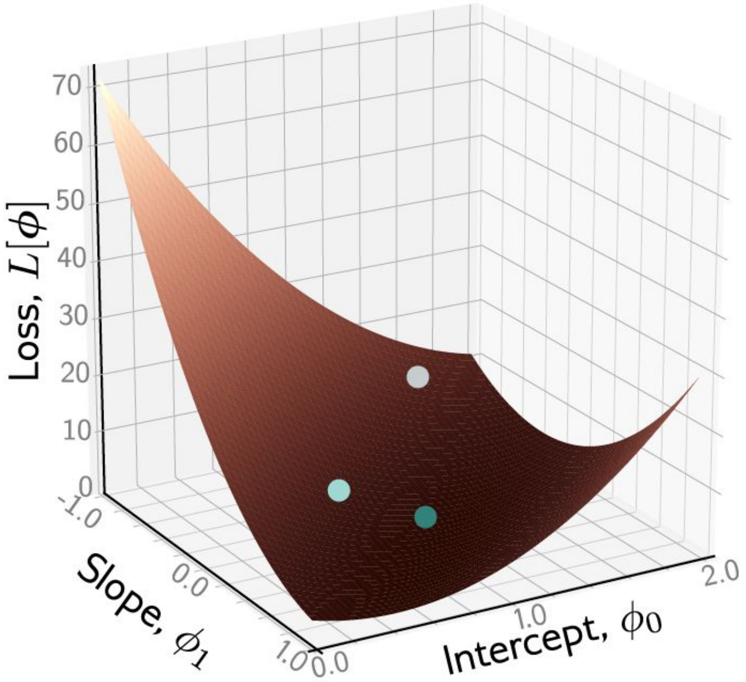
"Least squares loss function"

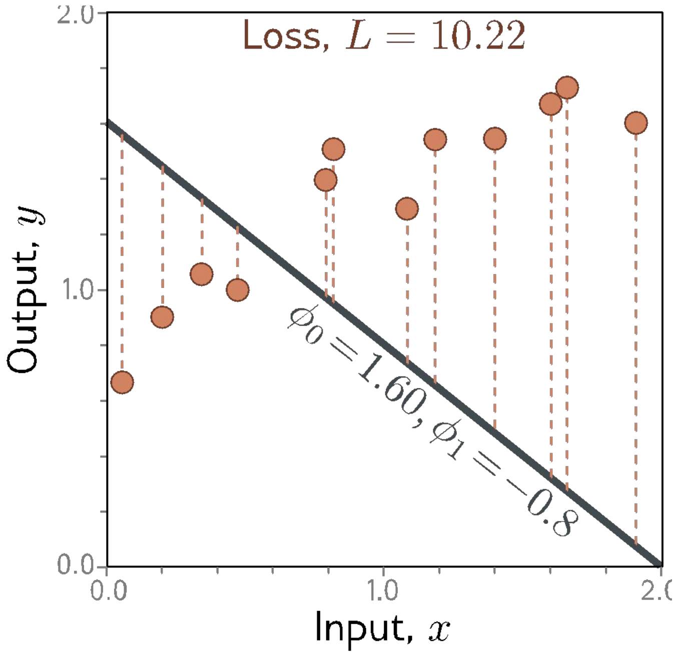
Example: 1D Linear Regression Loss Function



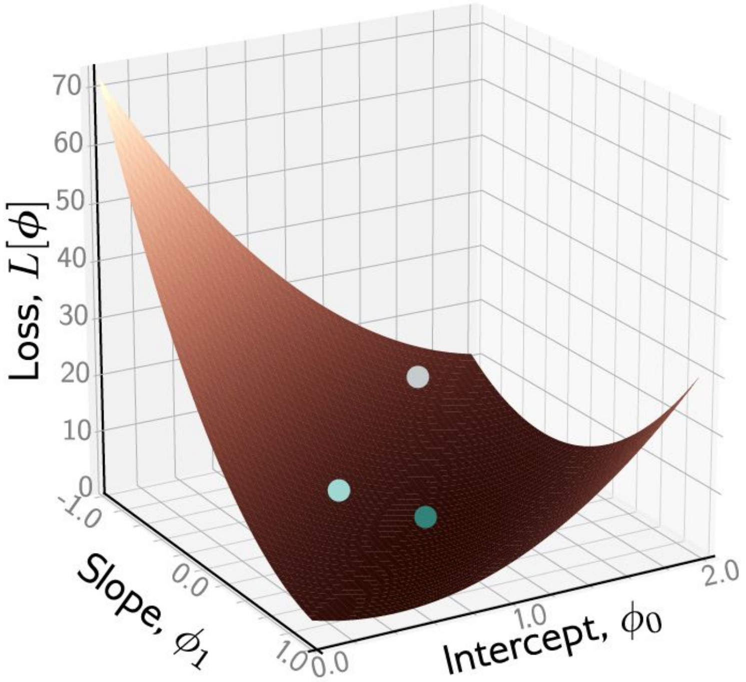


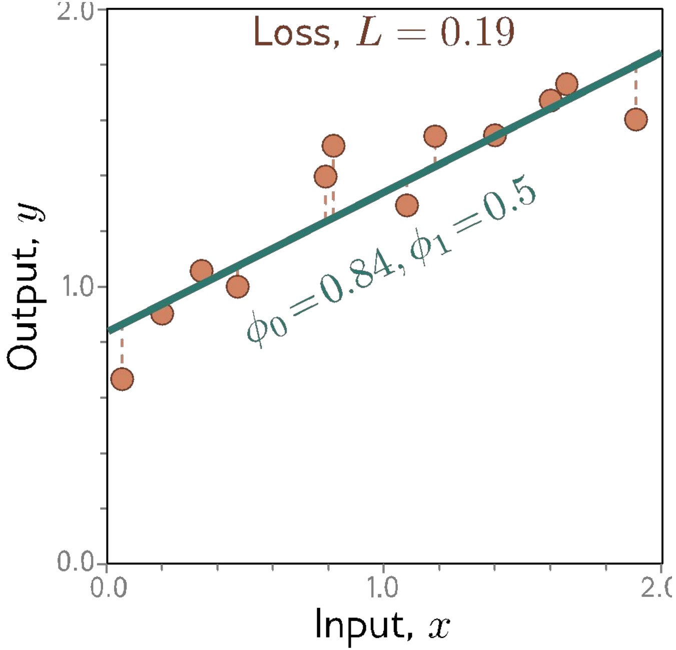
Example: 1D Linear Regression Loss Function



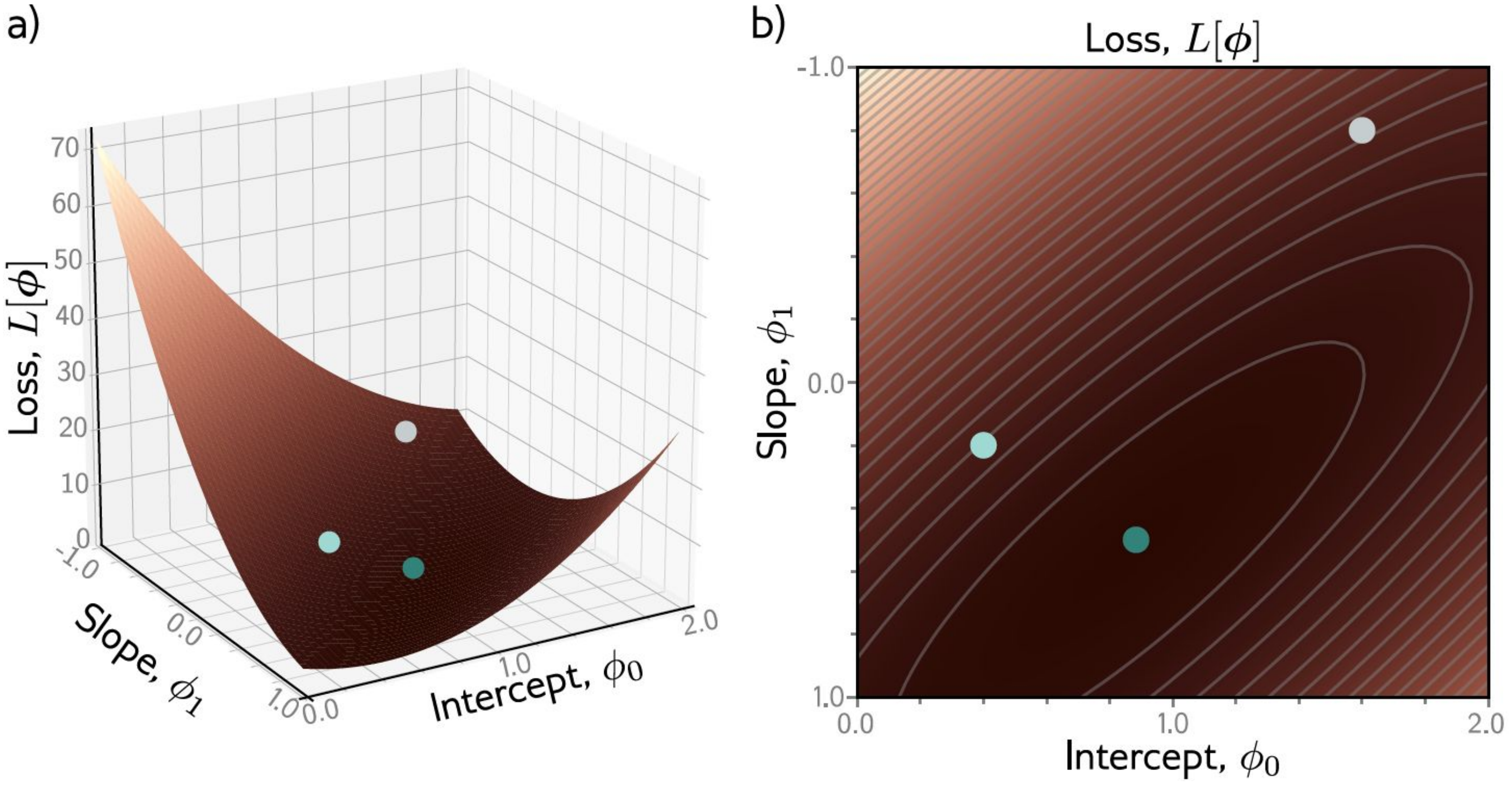


Example: 1D Linear Regression Loss Function

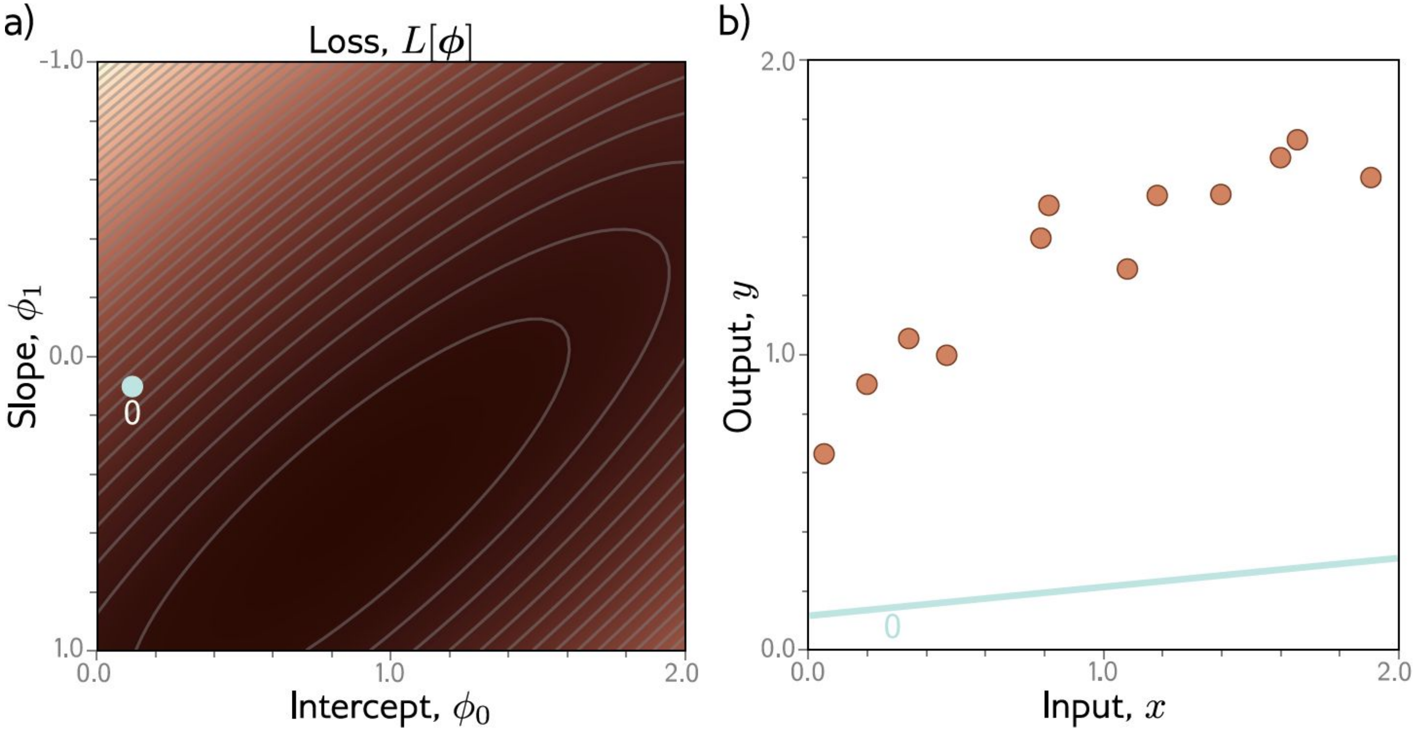




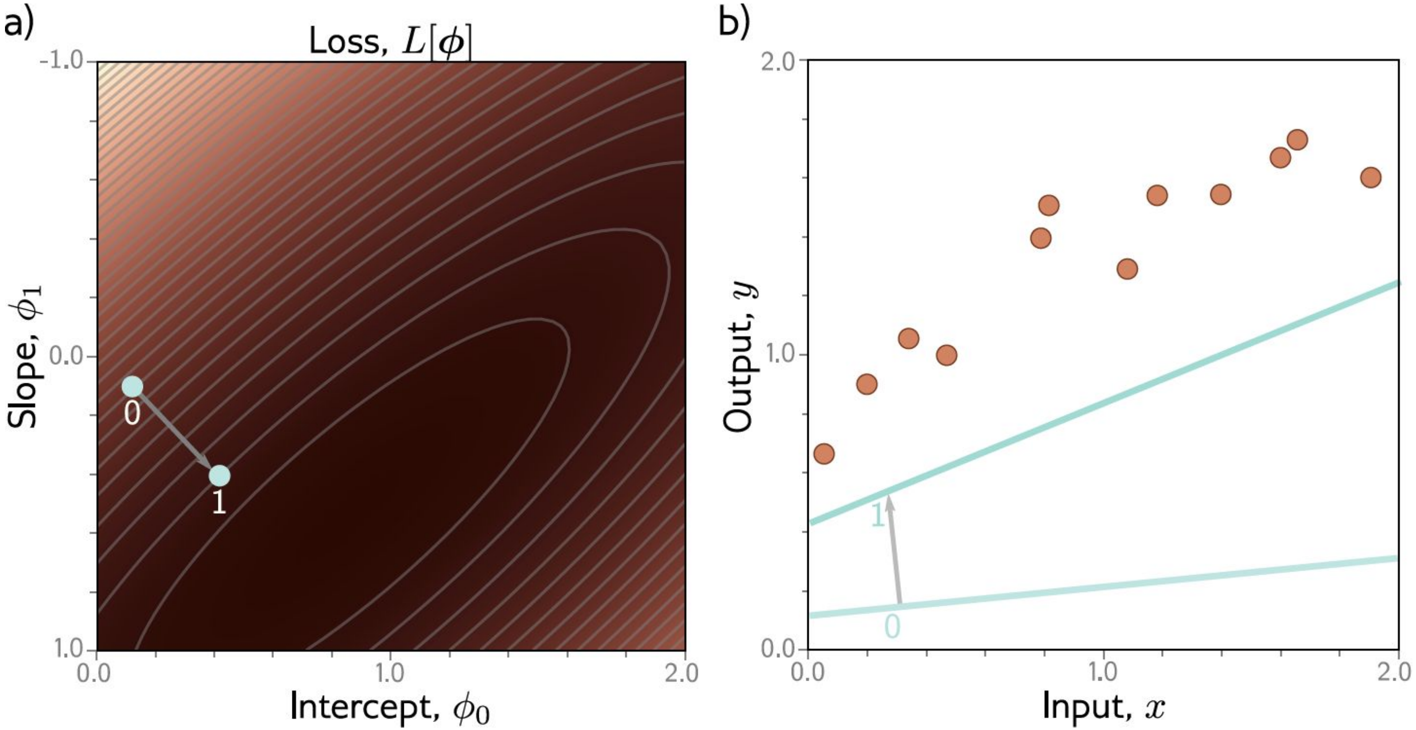
Example: 1D Linear Regression Loss Function



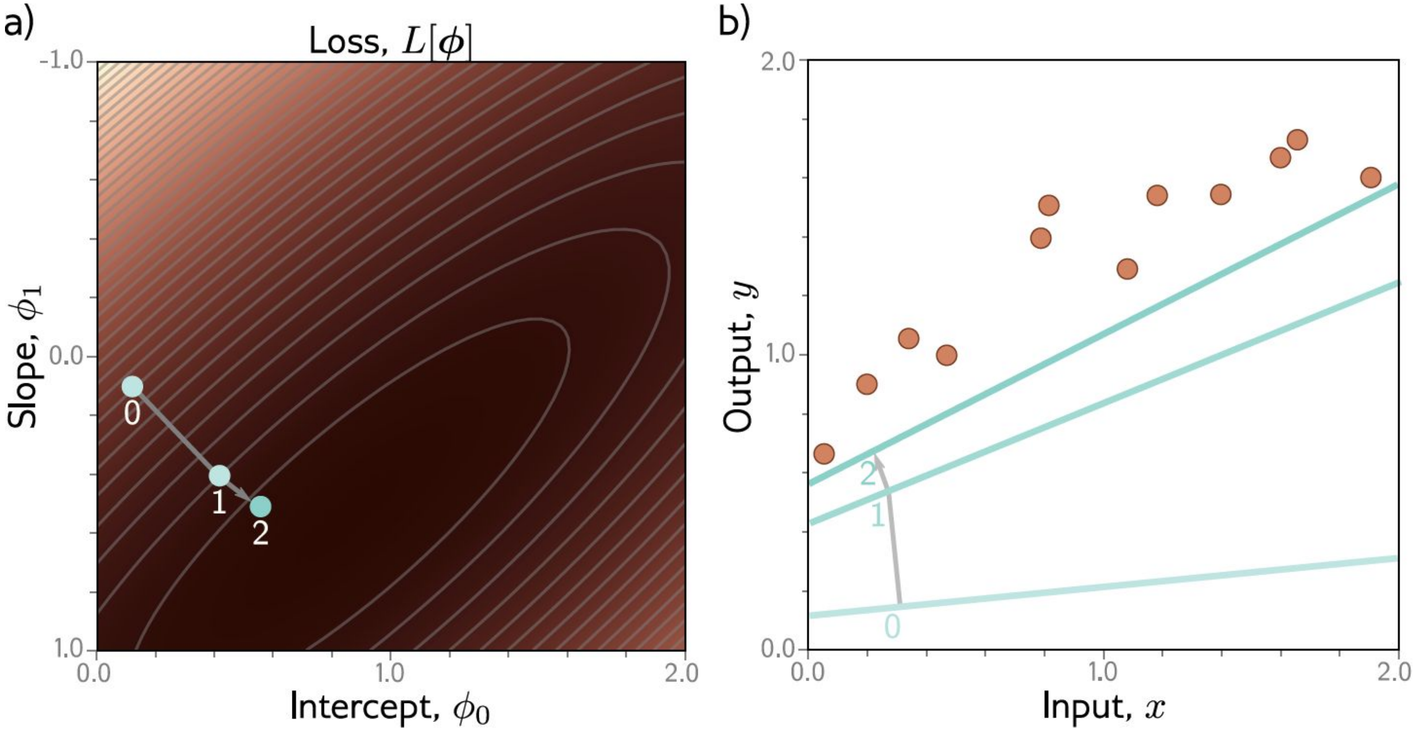
Example: 1D Linear Regression Training



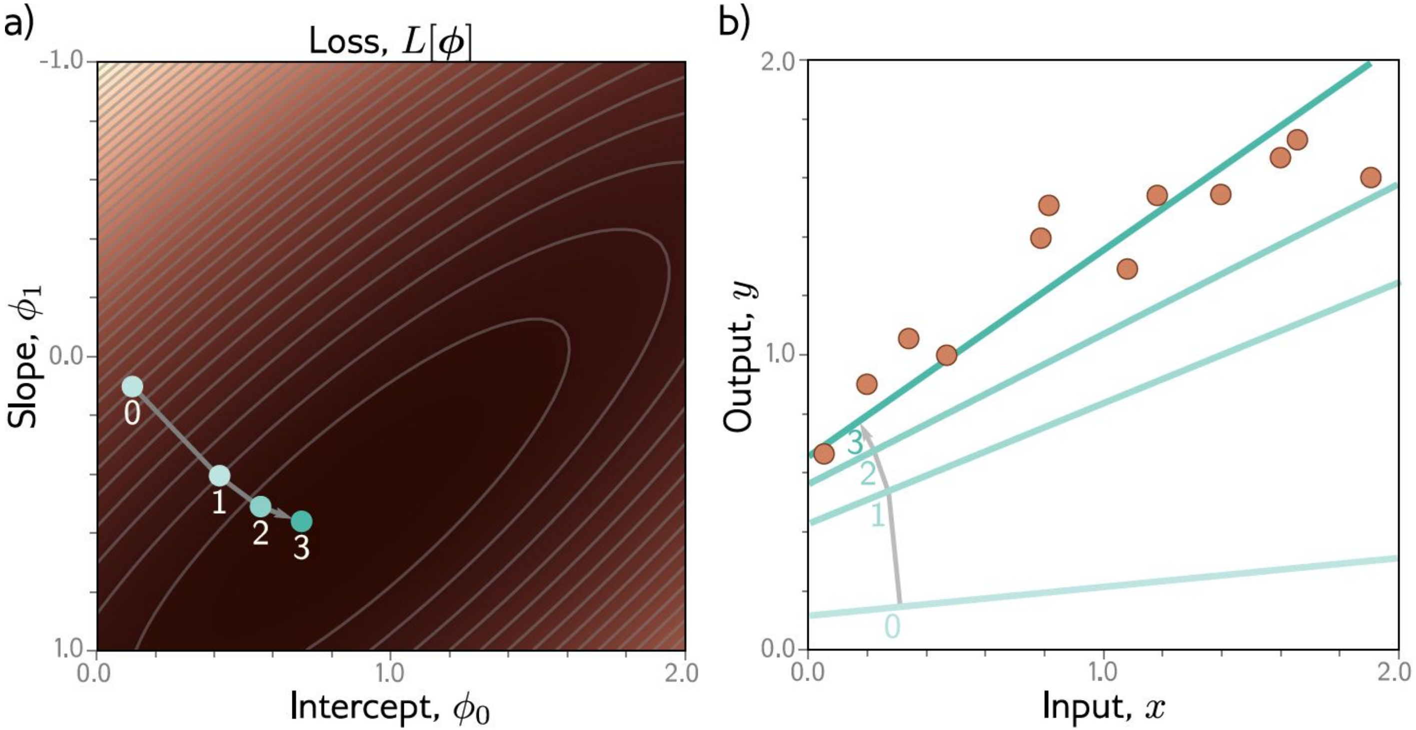
Example: 1D Linear Regression Training



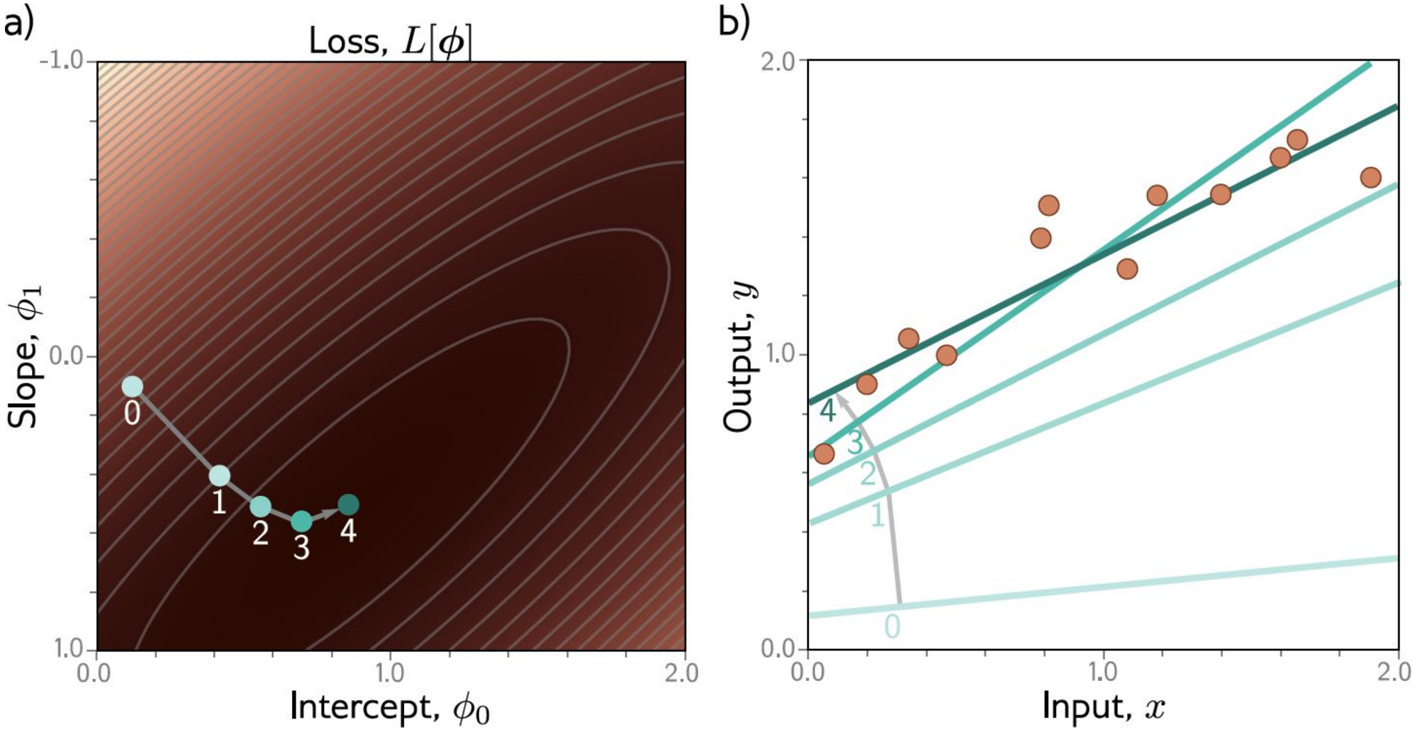
Example: 1D Linear Regression Training



Example: 1D Linear Regression Training



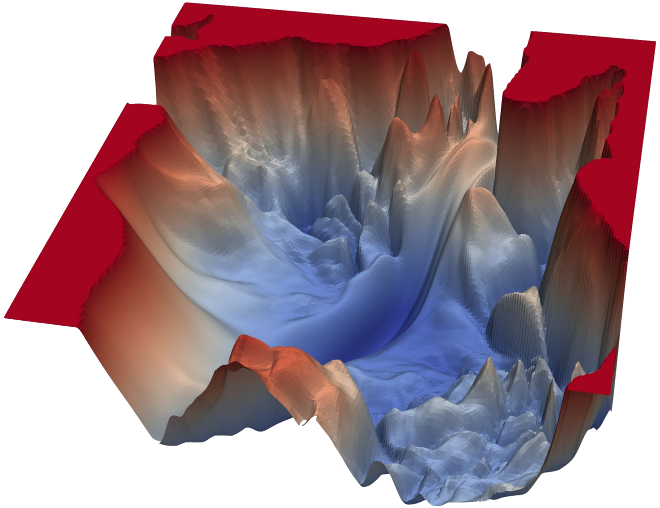
Example: 1D Linear Regression Training



This technique is known as gradient descent

Possible Objections to Gradient Descent

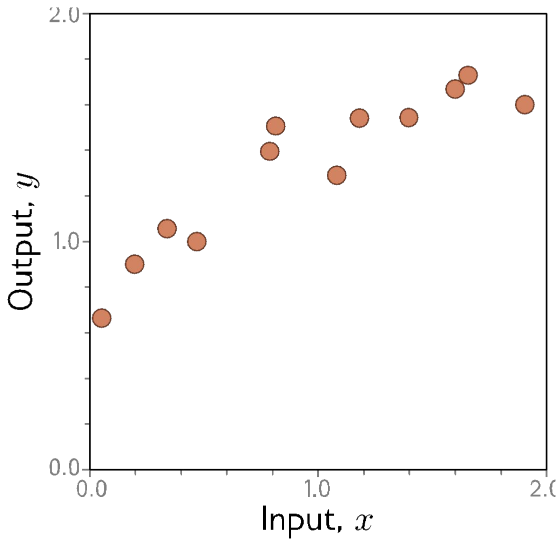
- But you can fit the line model in closed form!
  - Yes – but we won't be able to do this for more complex models
- But we could exhaustively try every slope and intercept combo!
  - Yes – but we won't be able to do this when there are a million parameters



Here's a visualization of the loss surface for the 56-layer neural network [VGG-56](http://arxiv.org/abs/1409.1556), from [Visualizing the Loss Landscape of Neural Networks](https://www.cs.umd.edu/~tomg/projects/landscapes/).

Example: 1D Linear Regression Testing

- Test with different set of paired input/output data (Test Set)
  - Measure performance
  - Degree to which Loss is same as training = generalization
- Might not generalize well because of
  - Underfitting - does not match real data trends
    - Model too simple?
    - Did not train enough?
  - Overfitting - fits to statistical peculiarities of data
    - Model too complex?
    - Trained too much?



Lecture Outline

- Supervised Learning
- Preparing Data for Learning
- Where are We Going Next?

Preparing Data for Learning

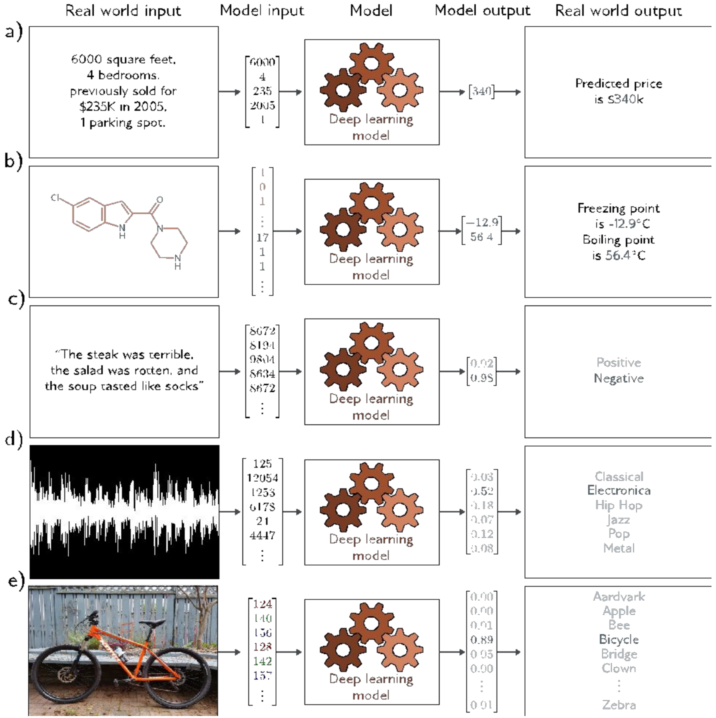
- Challenges
- Fixed Interface
- Sequence Interface

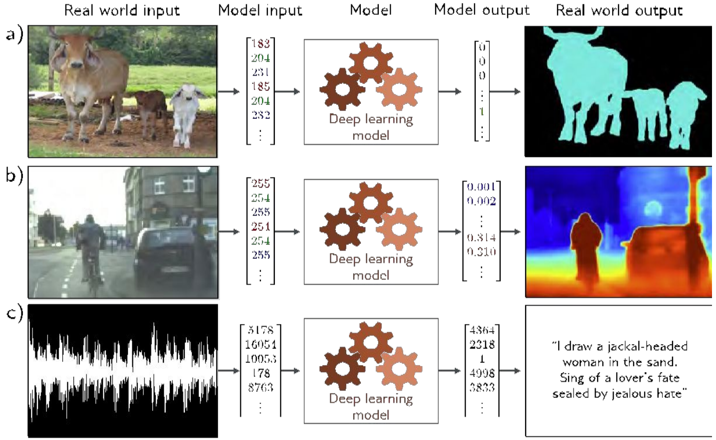
Preparing Data for Learning

- Challenges
- Fixed Interface
- Sequence Interface

Challenges - Wide Variety of Data to Model

Where do all these inputs and outputs come from?





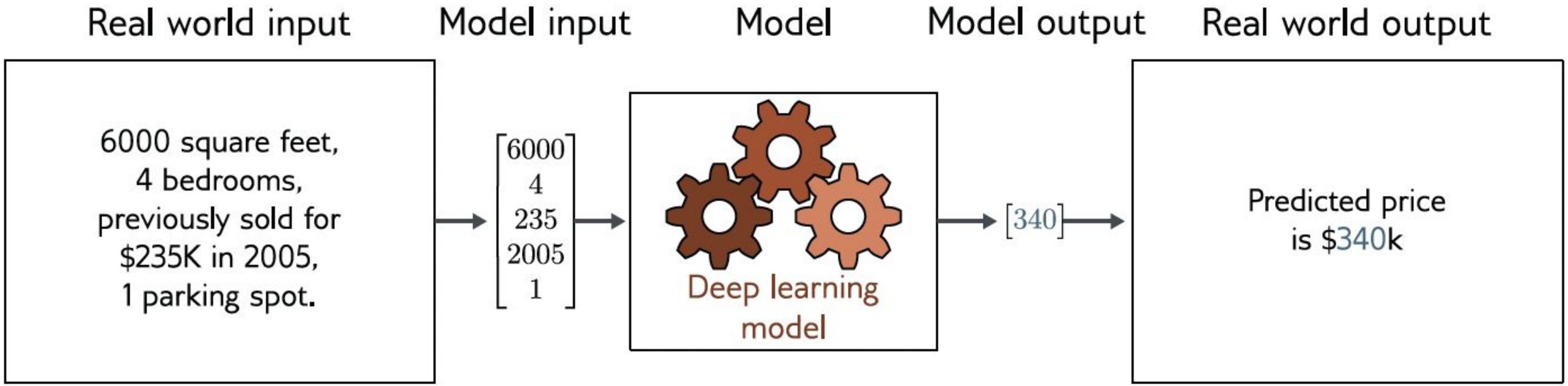
Preparing Data for Learning

- Challenges
- Fixed Interface
- Sequence Interface

Fixed Interface

- Encode real inputs and outputs as fixed size vectors of numbers.
- Model takes in fixed input vector and returns fixed size output vector.

Ad hoc Text Data Collection

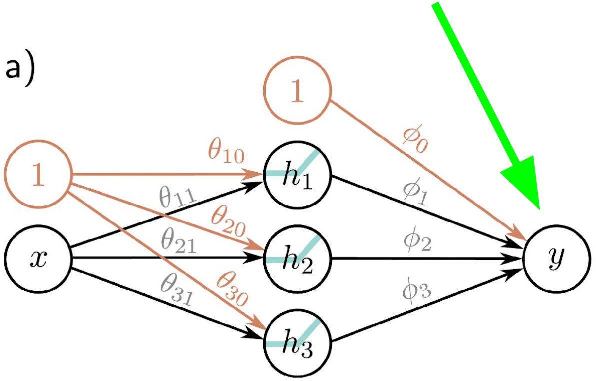


Pretty common for regression problems

- Text parsing… may have missing or weird values if parsing fails
- Database queries if you are lucky

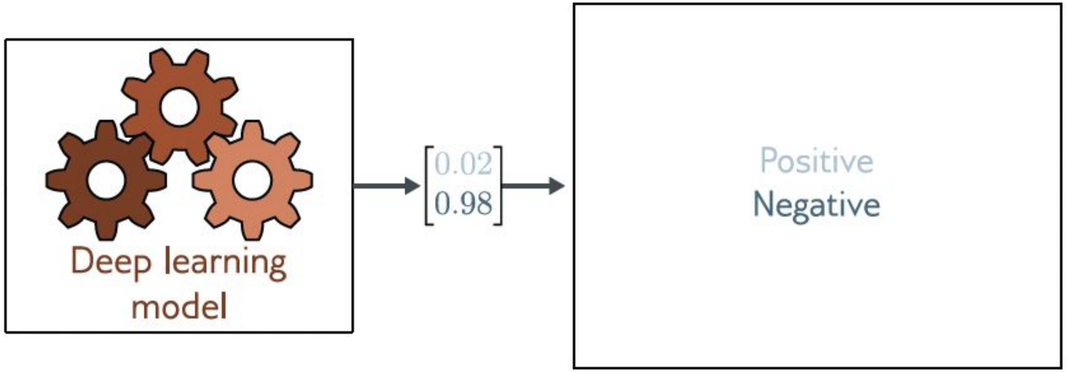
Regression Problems

- Model just outputs a number… should be close to the real one.
- No particular semantics?
- Any range constraints?
  - Non-negative?
  - Min/max value?
- May change structure of neural network based on these constraints…
  - Mostly in the activation function of the output node.



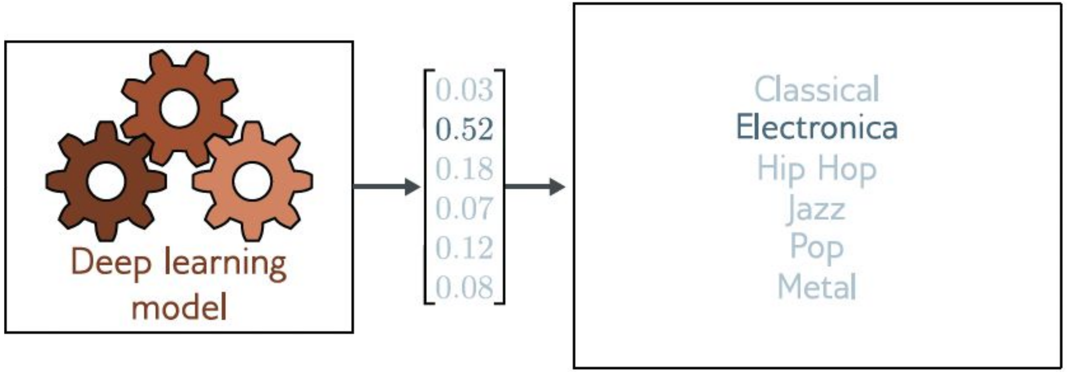
Binary Classification

- Training outputs:
  - Raw data says true/false, yes/no, 1/0, occasionally probabilities.
  - Usually map to 1/0 values, or keep probabilities.
  - One vs two output columns depends on model internals.
- Model outputs:
  - Should be constrained between zero and one.
  - Default interpretation as probabilities.



Multiclass Classification → One Hot Encoding

- Training outputs:
  - Raw data often is a class name as a string.
  - Map distinct class names to different columns.
  - Set column to 1 or 0 based on class match.
- Model outputs:
  - Should be constrained between zero and one.
  - Default interpretation as probabilities.
  - Need to make sure these add up to one.



Classical	Electronica	Hip Hop	Jazz	Pop	Metal
0	1	0	0	0	0
0	0	0	1	0	0
0	0	0	0	0	1

Multiclass Classification 💔 Binary Encoding

Why not use a binary encoding?

- Bits of binary encoding rarely have semantic information.
  - Partial column matches is not a sign of similarity.
  - Forces learning algorithm to learn decoding…
- Unclear interpretation of uncertain output
  - What does [0.6, 0.6, 0.7] mean?
  - Probability interpretations are nonsensical.
  - Rounding to 0/1 may not match a class.

Classical	0	0	0
Electronica	0	0	1
Hip Hop	0	1	0
Jazz	0	1	1
Pop	1	0	0
Metal	1	0	1

Preparing Data for Learning

- Challenges
- Fixed Interface
- Sequence Interface

Sequence View

- Encode more sophisticated inputs and outputs as sequences of numbers.
- Will apply some parts of our models repeatedly
  - Originally recurrent neural networks
  - Recently attention and transformers
- Brief look at strings now, much more later.

String Tokenization (then One Hot Encoding)

```
[3] import tiktoken
```

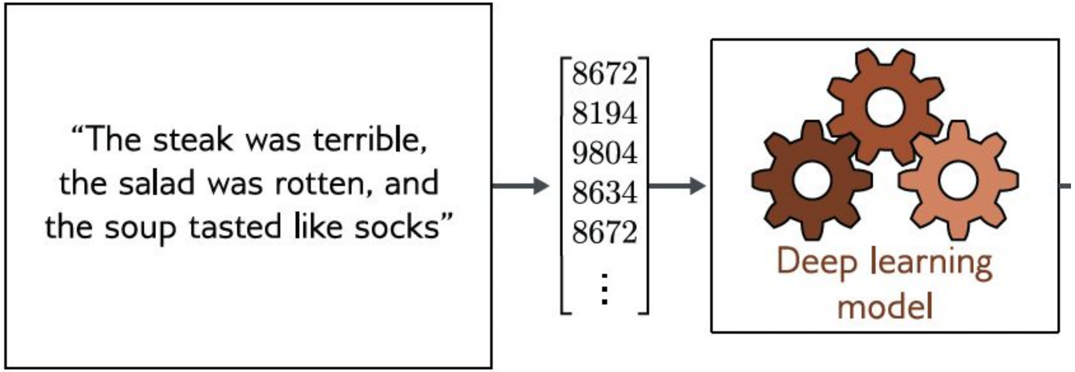
```
[4] encoding = tiktoken.encoding_for_model("gpt-4o")
```

```
[6] tokens = encoding.encode("The steak was terrible, the salad was rotten, and the soup tasted like socks")
    tokens
```

```
[976,
 67314,
 673,
 28380,
 11,
 290,
 38312,
 673,
 146652,
 11,
 326,
 290,
 29684,
 88244,
 1299,
 54699]
```

```
[8] [encoding.decode([t]) for t in tokens]
```

```
['The',
 ' steak',
 ' was',
 ' terrible',
 ',',
 ' the',
 ' salad',
 ' was',
 ' rotten',
 ',',
 ' and',
 ' the',
 ' soup',
 ' tasted',
 ' like',
 ' socks']
```



Lecture Outline

- Supervised Learning
- Preparing Data for Learning
- Where are We Going Next?

Where are we going next?

- Shallow neural networks
  - Universal approximation
- Deep neural networks
  - More flexibility with fewer parameters
- Loss functions
  - How do we decide what parameters are better?
  - Where did least squares come from?
  - When should we use other loss functions?
- Fitting models / Gradients / Measuring / Regularization
  - How we actually train these neural networks
  - And encourage them to generalize…

Feedback?