BOSTON UNIVERSITY

Lecture 05
Loss Functions
(and probability models)

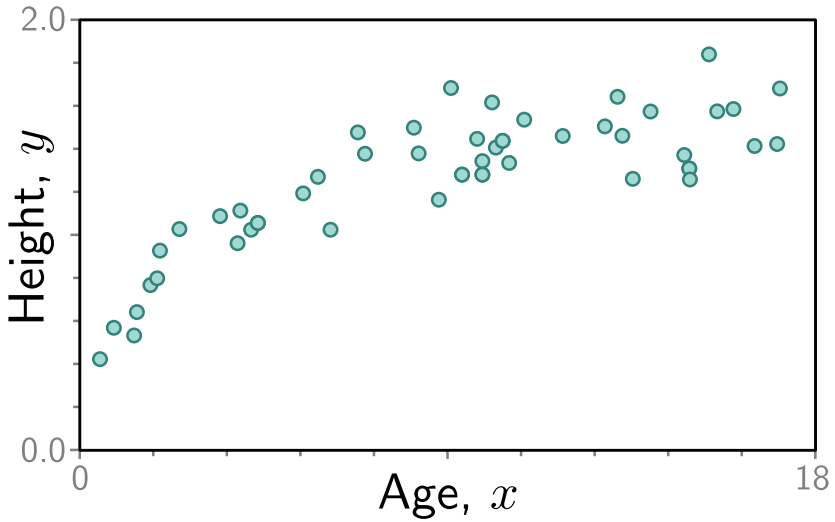
DL4DS – Spring 2026

Recap

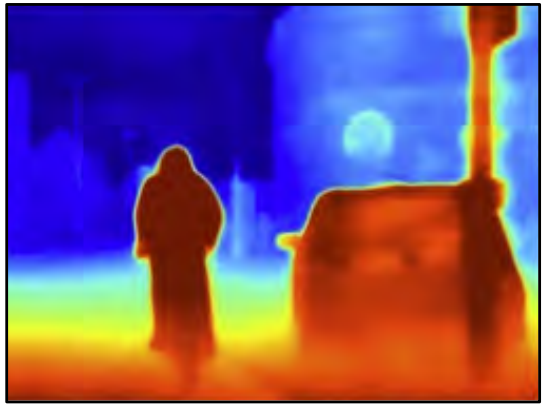
- So far, we talked about *linear regression*, *shallow neural networks* and *deep neural networks*
- Each have parameters, ϕ, that we want to choose for a *best possible mapping between input and output* training data
- A *loss function* or *cost function*, $L[\phi]$, returns a single number that describes a mismatch between $f[x_i, \phi]$and the ground truth outputs, y_i.

We need to find a loss function that works with…

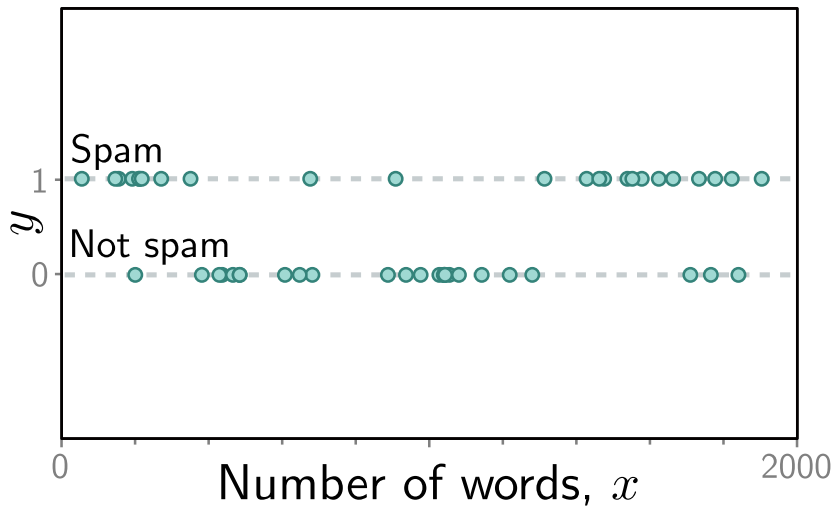
Univariate and Multivariate Regression



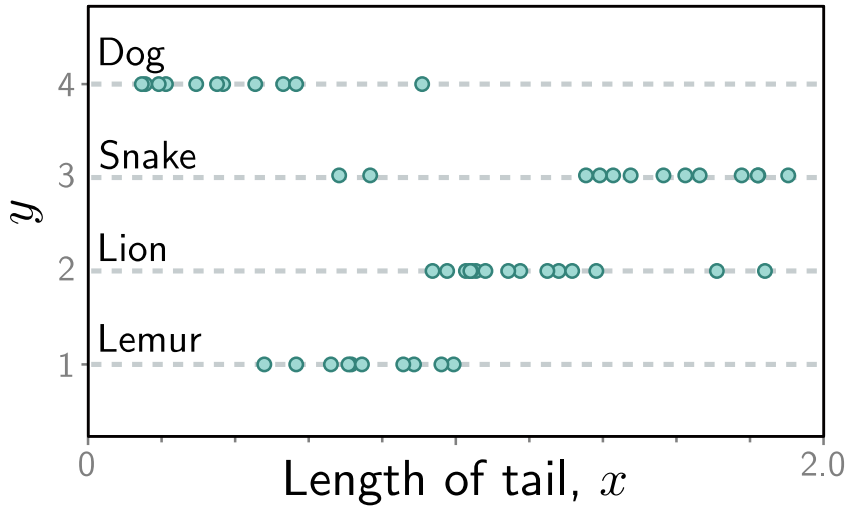
Depth Map

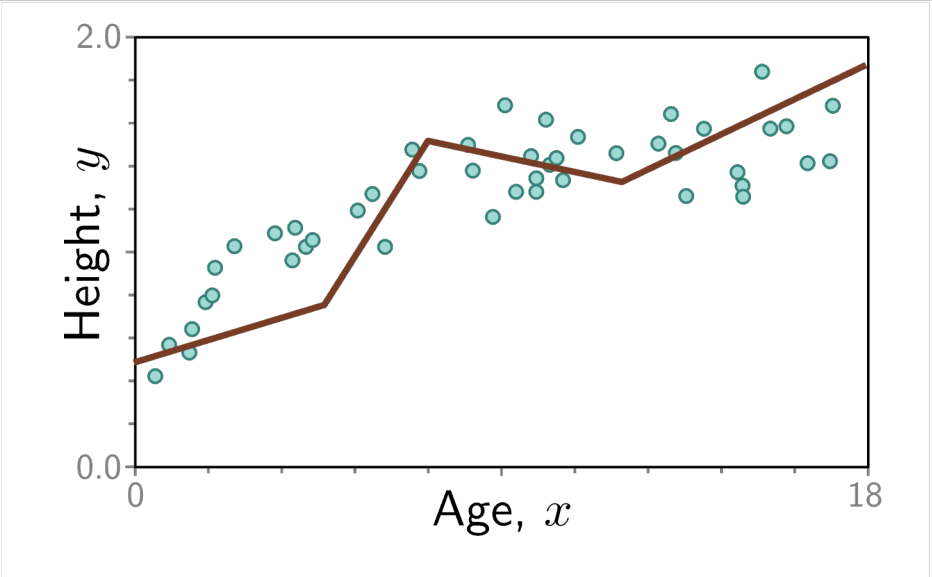


Binary Classification

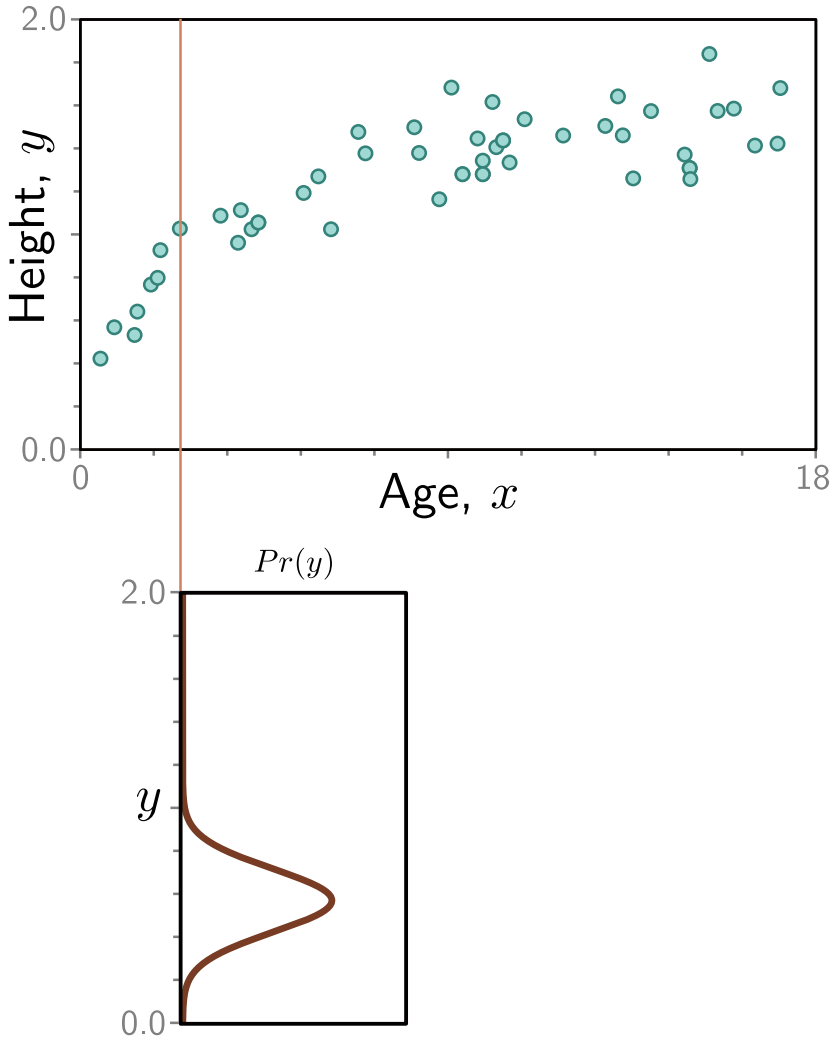


Multiclass Classification





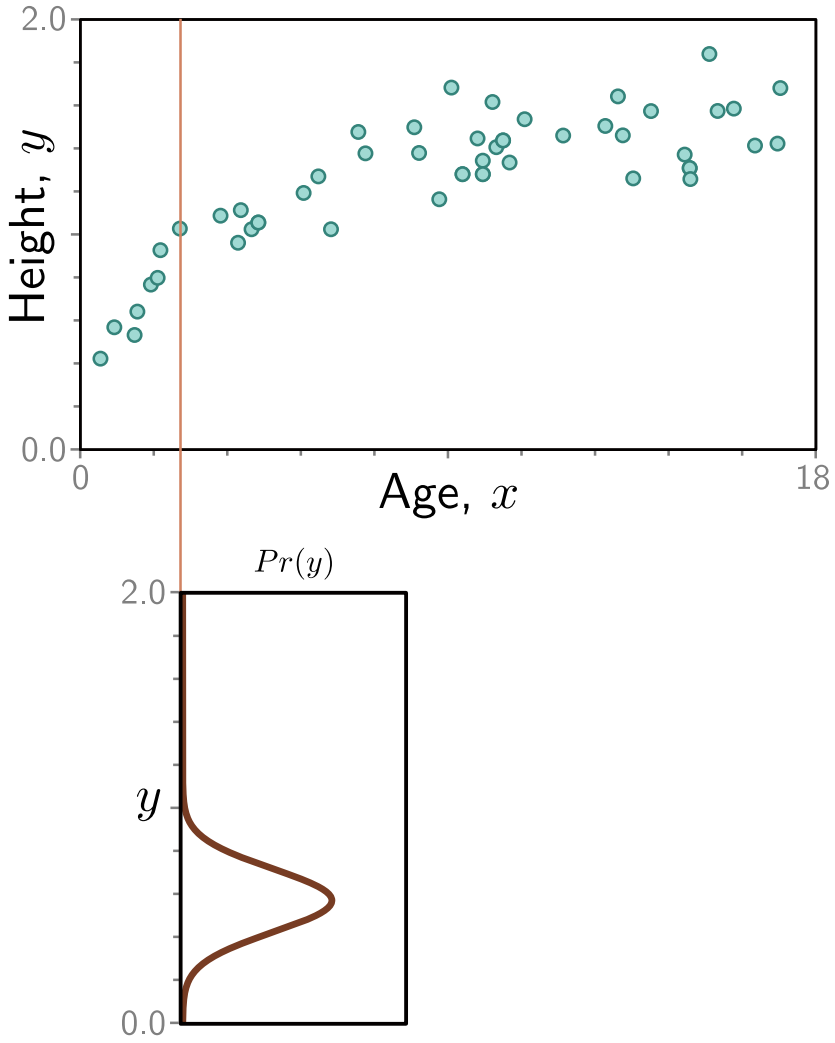
So far, we thought about fitting a model to the data…



Alternatively, we can think about fitting a *probability model* to the data.

$$\Pr(y|x = 2.8)$$

Why?

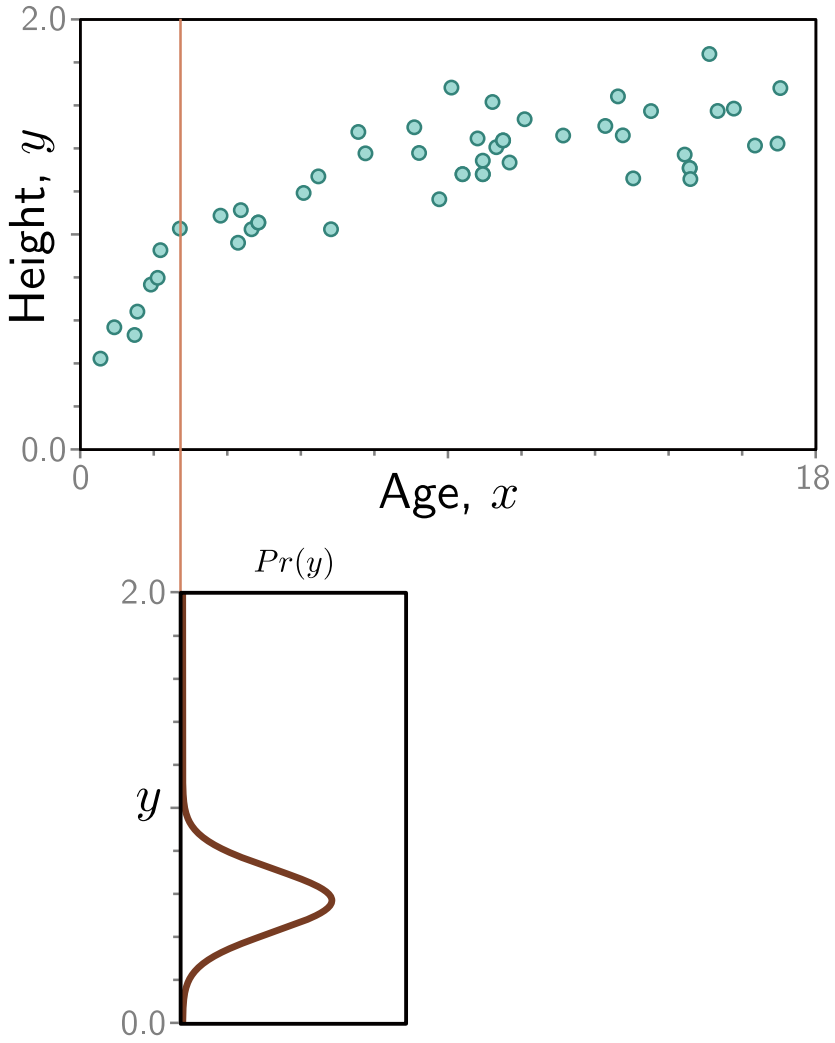


Alternatively, we can think about fitting a *probability model* to the data.

$$\Pr(y|x = 2.8)$$

Why?

Because this provides a *framework* to build loss functions for other prediction types…



Alternatively, we can think about fitting a *probability model* to the data.

$$\Pr(y|x = 2.8)$$

Why?

Because this provides a *framework* to build loss functions for other prediction types…

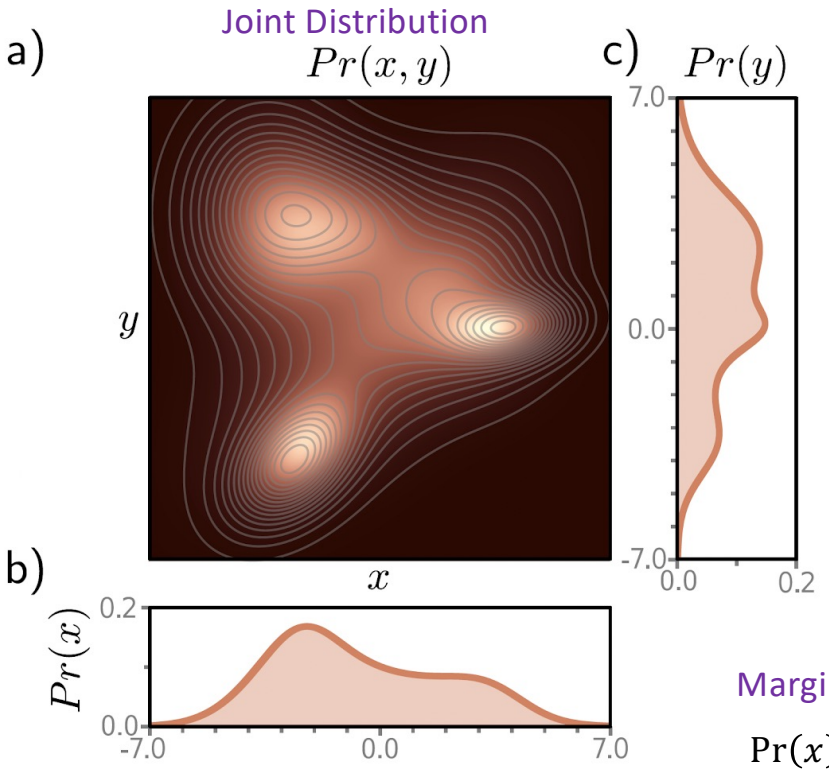
… and justifies least squares for real-valued regression models.

Brief Probability Review

- Random variables, e.g. x and y
- $\Pr(x)$ is a probability distribution over x
- $0 \leq \Pr(x) \leq 1$
- $\int_x \Pr(x)\, dx = 1$ or $\sum_i \Pr(x_i) = 1$
- $\Pr(x, y) = \Pr(x) \cdot \Pr(y)$ when x and y are independent
- $\Pr(x \mid y) \Pr(y) = \Pr(x, y) = \Pr(y \mid x) \Pr(x)$
- And…

See *Understanding Deep Learning*, Appendix C for more probability review

Joint and Marginal Probability Distributions



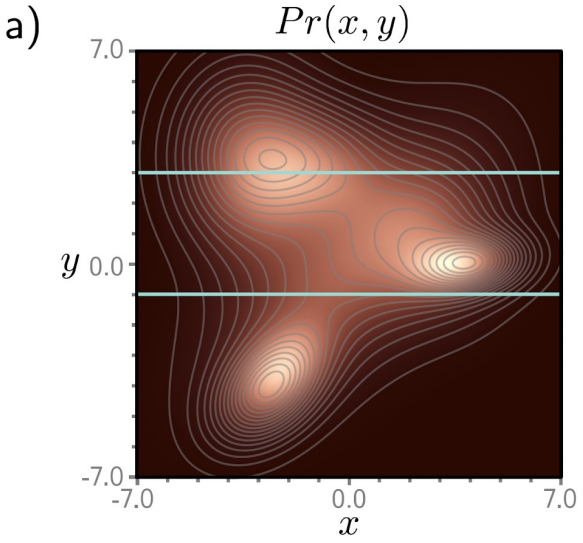
Marginal distribution

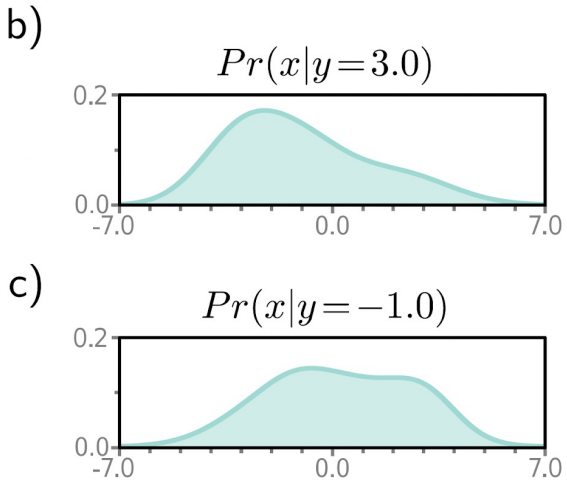
$$\Pr(y) = \int_x \Pr(x, y)\, dx$$

Marginal distribution

$$\Pr(x) = \int_y \Pr(x, y)\, dy$$

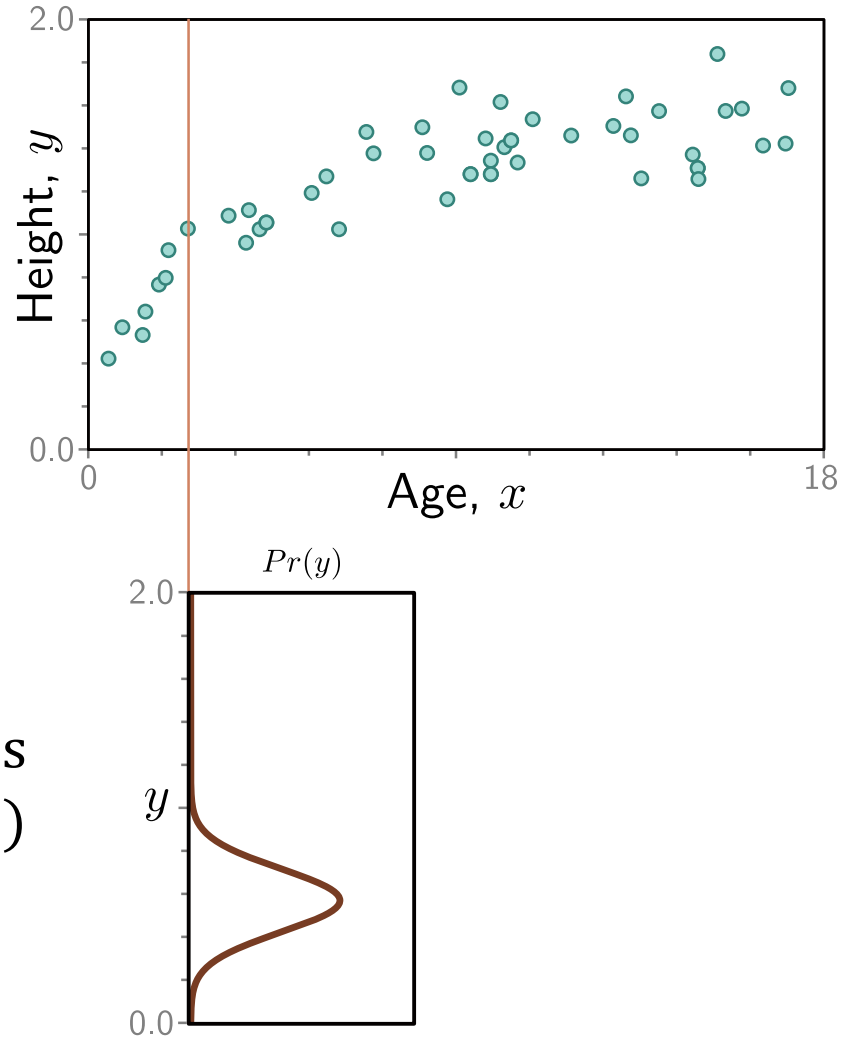
Conditional Probabilities





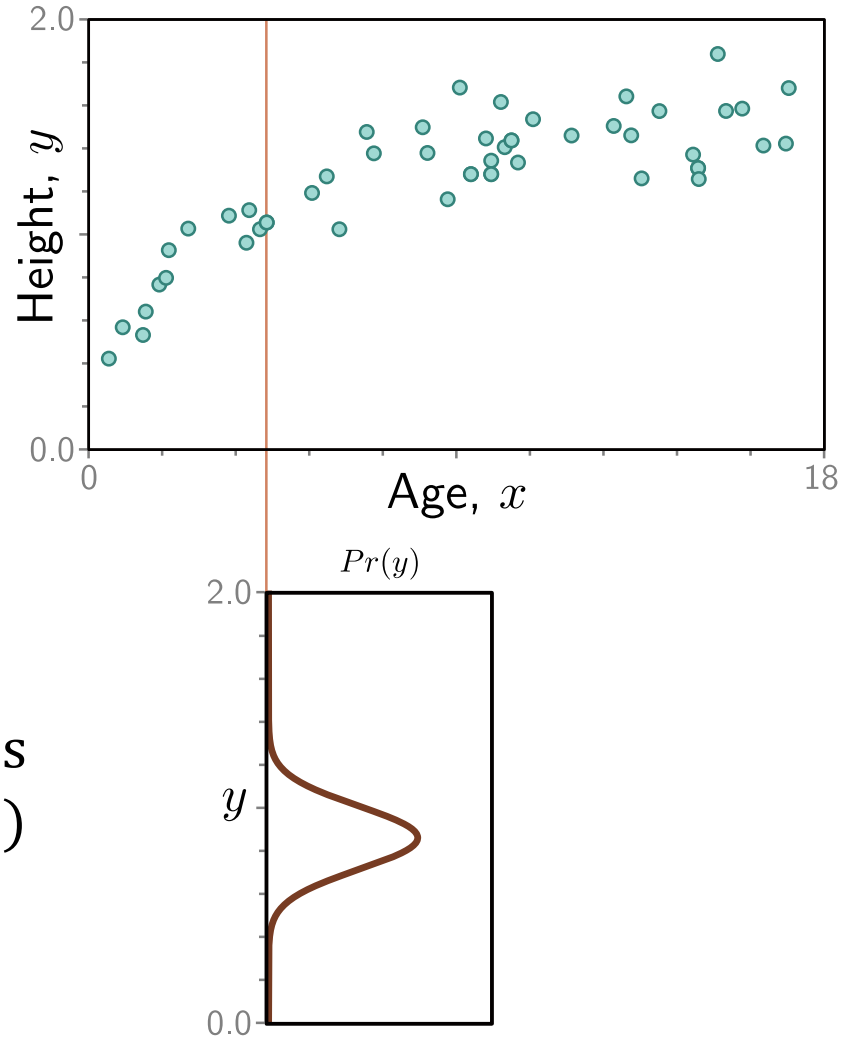
$$\int_x \Pr(x \mid y = 3.0)\, dx = 1$$

$$\int_x \Pr(x \mid y = -1.0)\, dx = 1$$



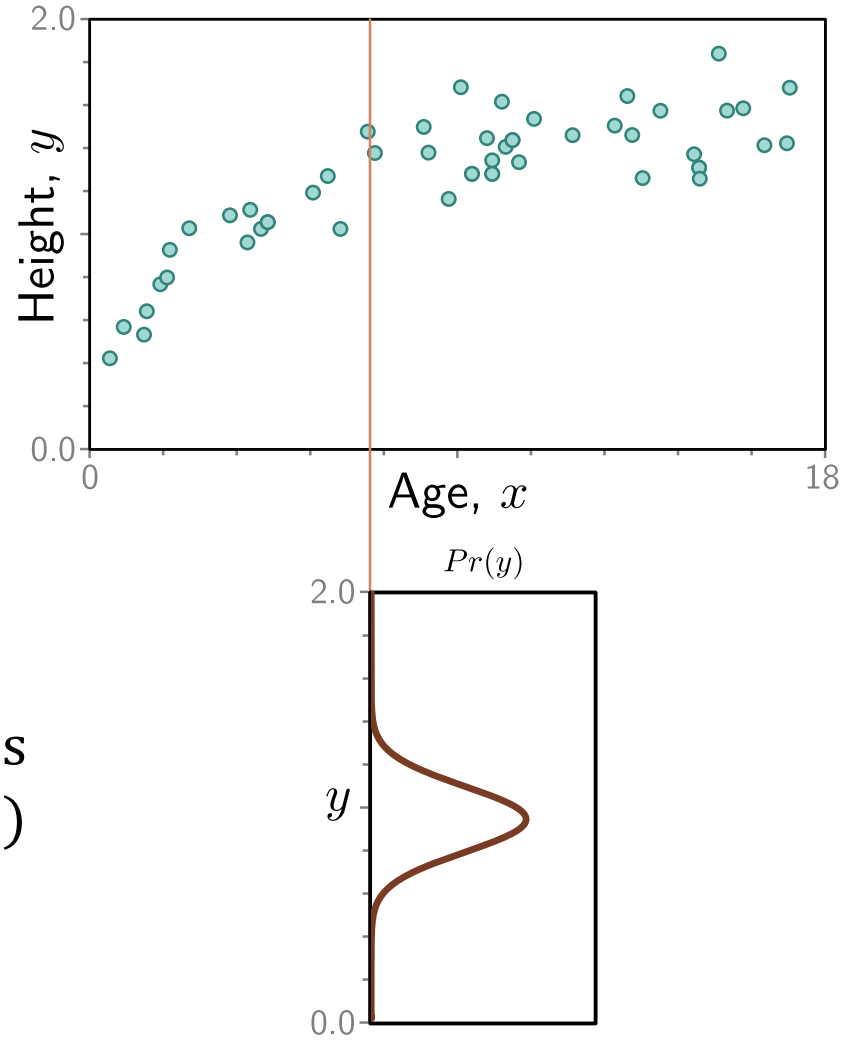
Continuous

$\Pr(y|x = 2.5)$



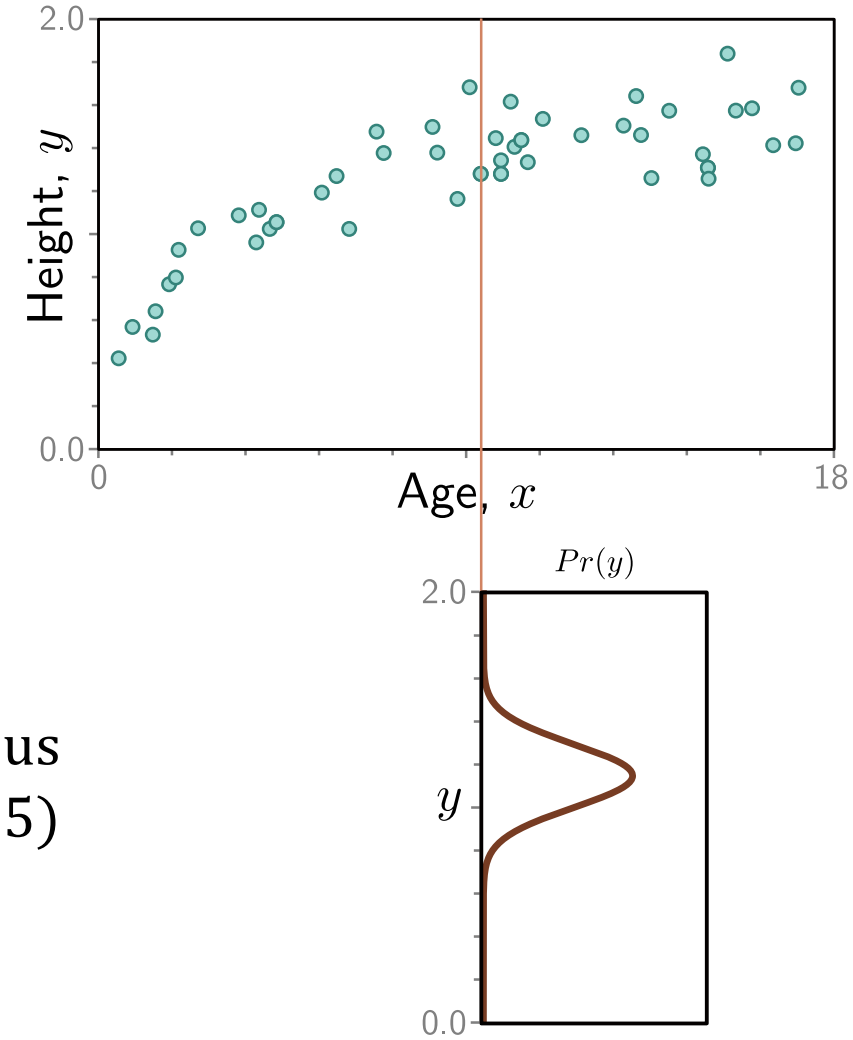
Continuous

$\Pr(y|x = 4.8)$



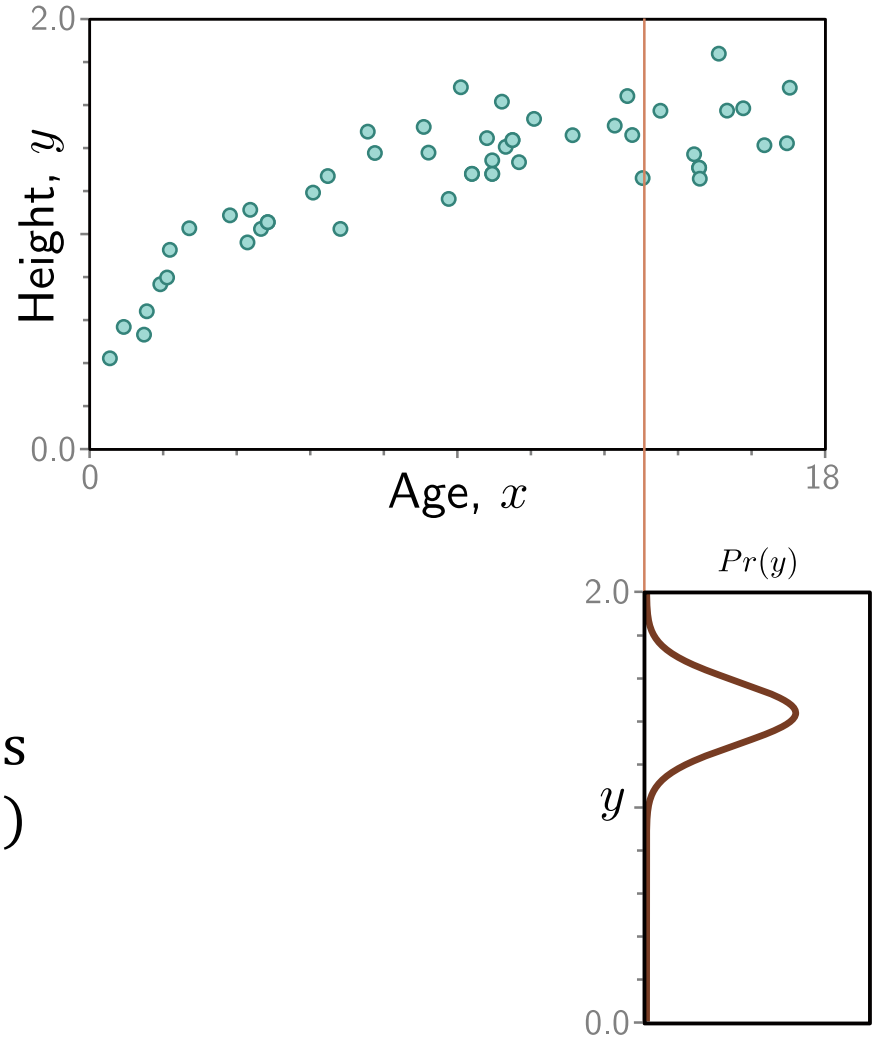
Continuous

$\Pr(y|x = 7.5)$



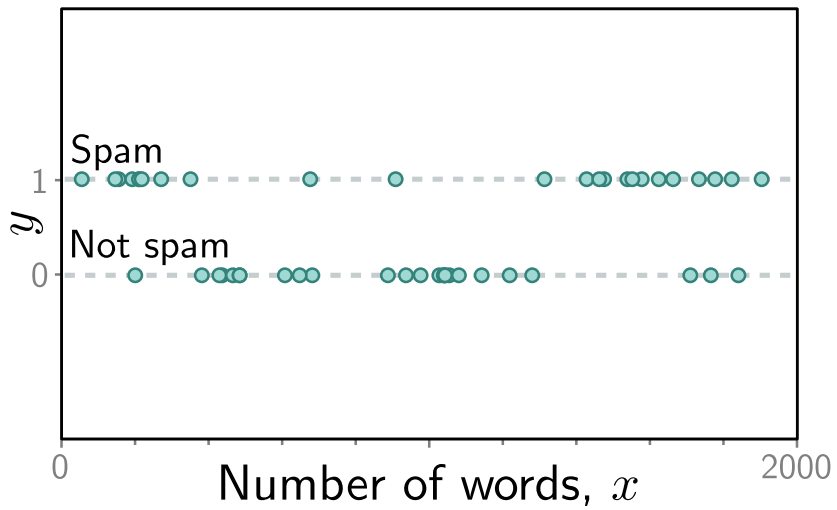
Continuous

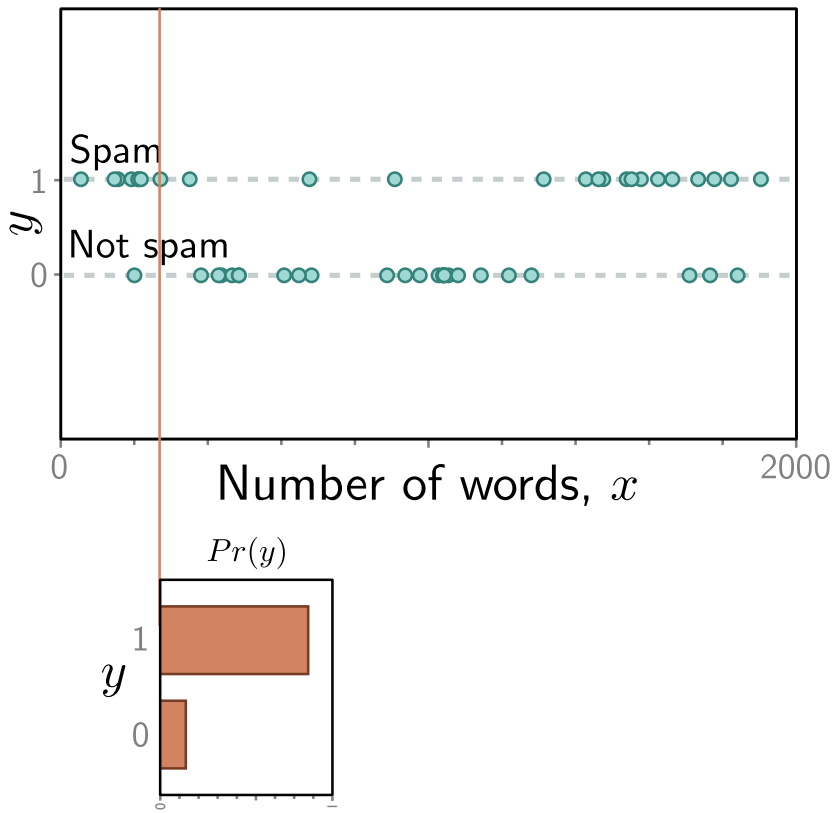
$\Pr(y|x = 10.5)$



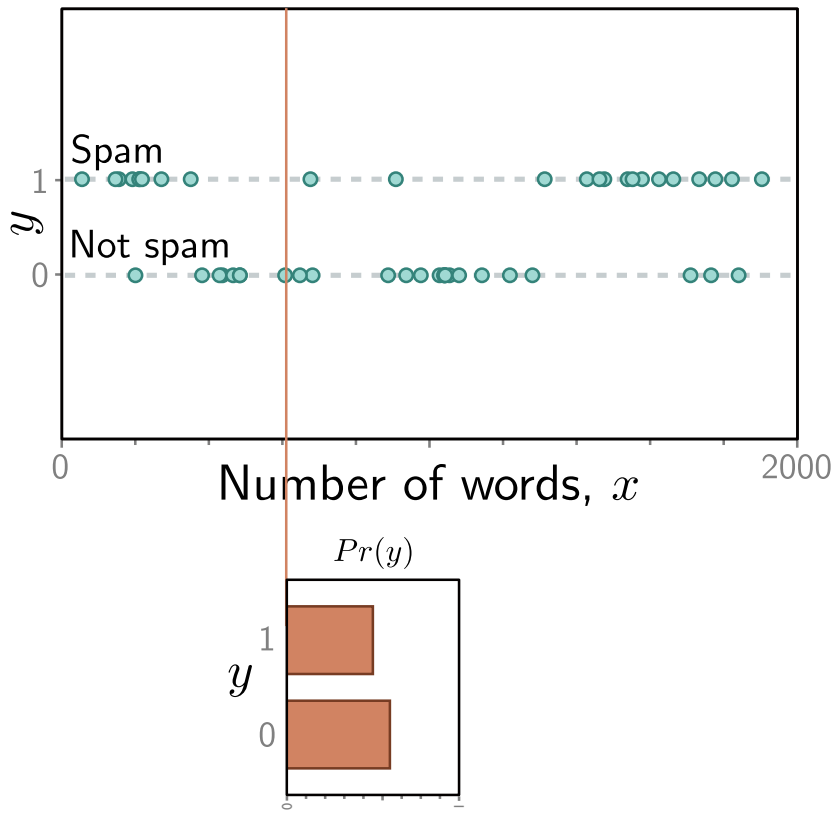
Continuous

$\Pr(y|x = 13.2)$


Spam
Not spam
1
0
y
0
2000
Number of words, x

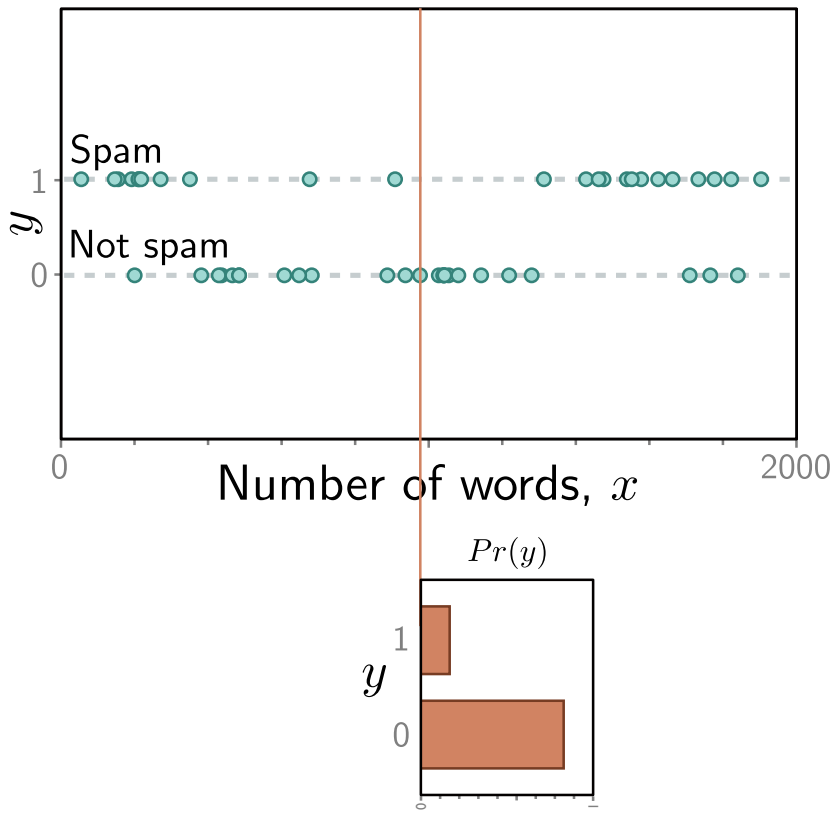


Discrete
$\Pr(y|x = 250)$



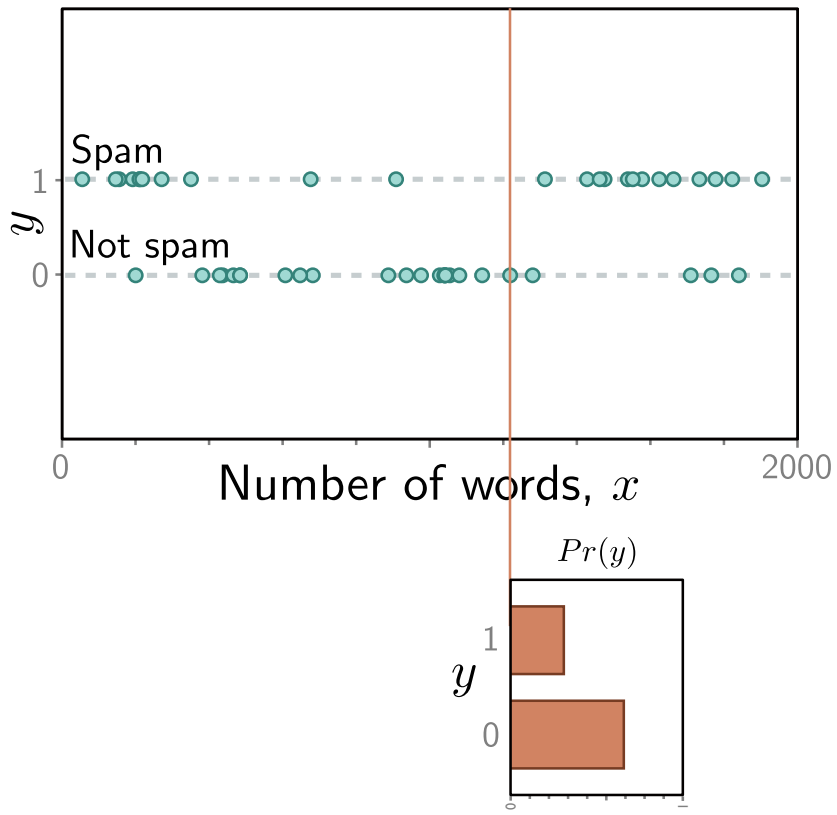
Discrete

$\Pr(y|x = 610)$



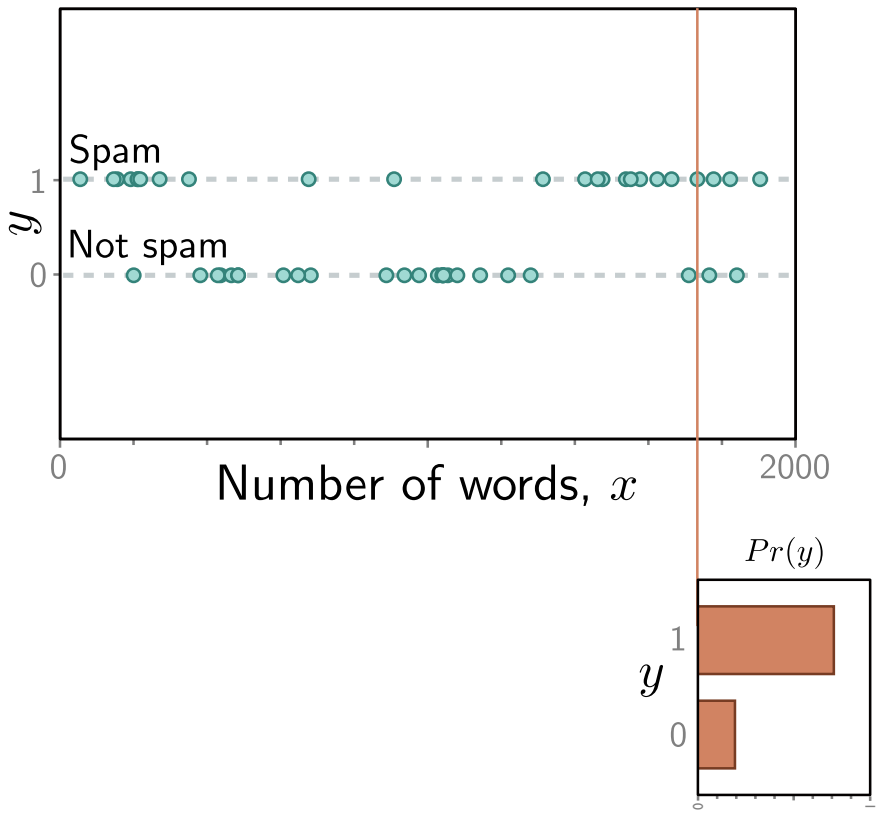
Discrete

$\Pr(y|x = 980)$



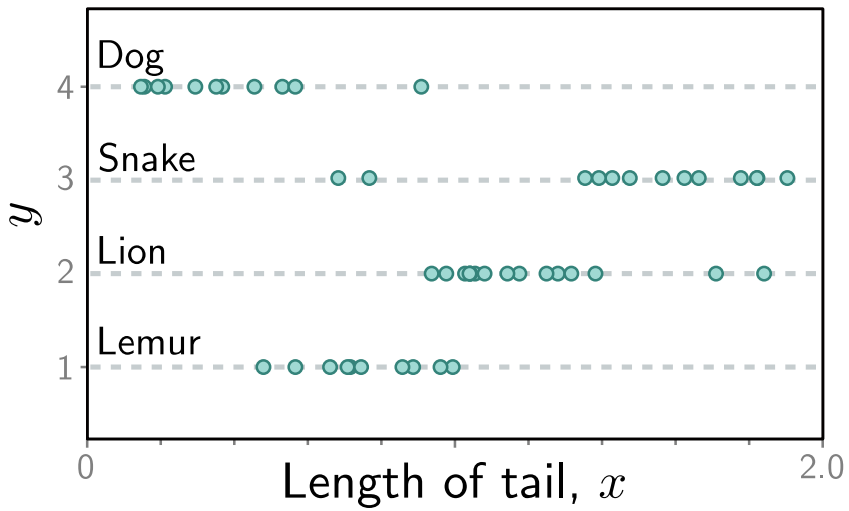
Discrete

$\Pr(y|x = 1220)$



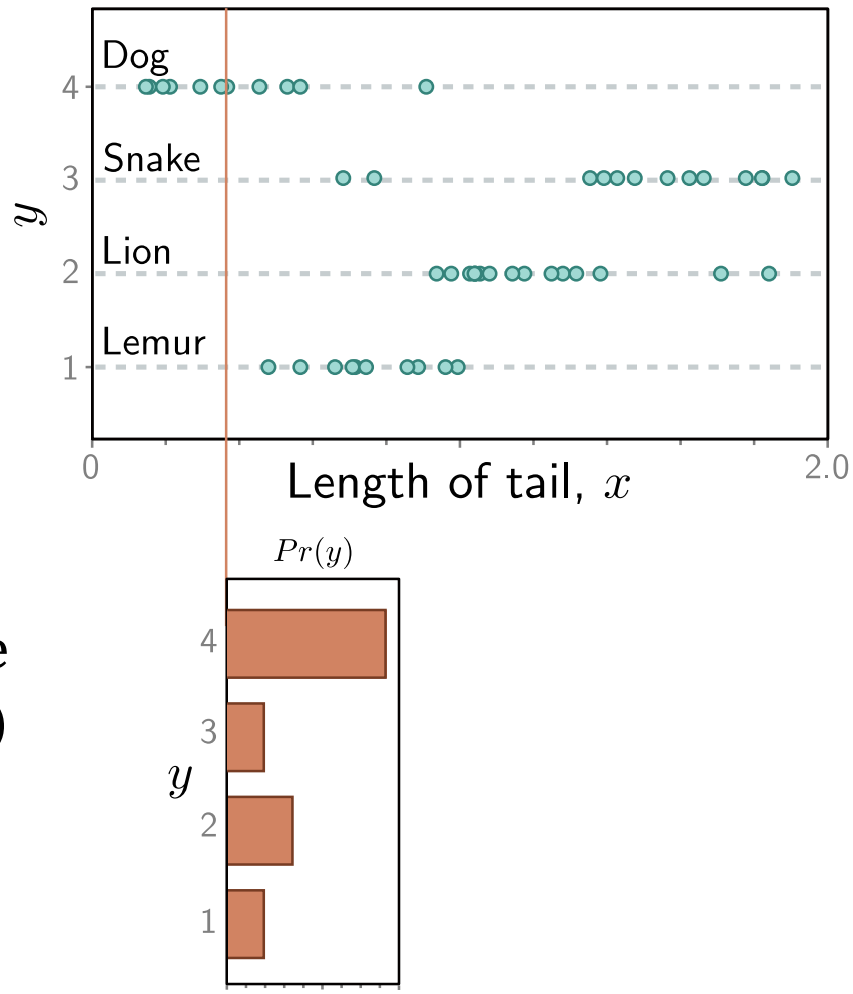
Discrete

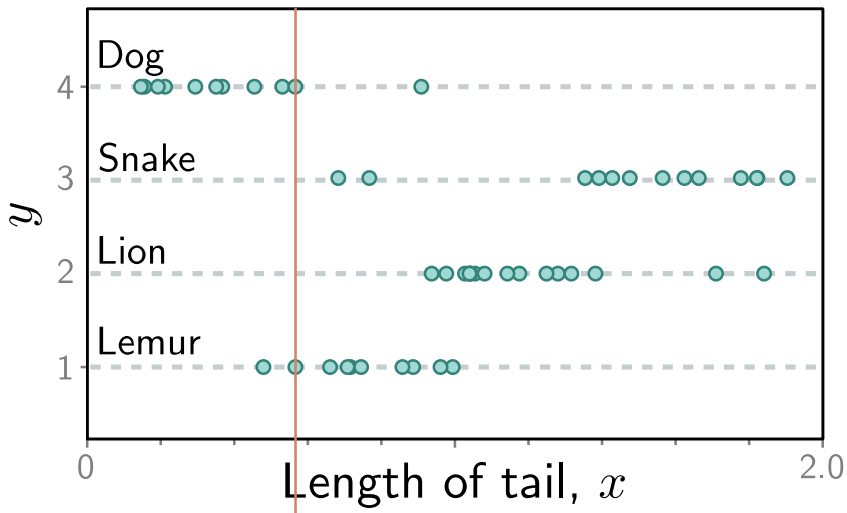
$\Pr(y|x = 1750)$


Dog
Snake
Lion
Lemur
4
3
2
1
y
0
2.0
Length of tail, x

Discrete

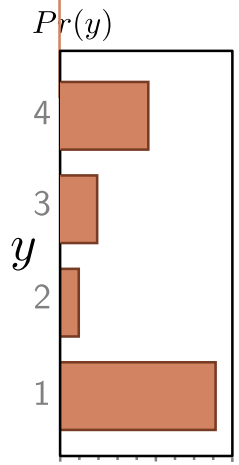
$\Pr(y|x = 0.35)$





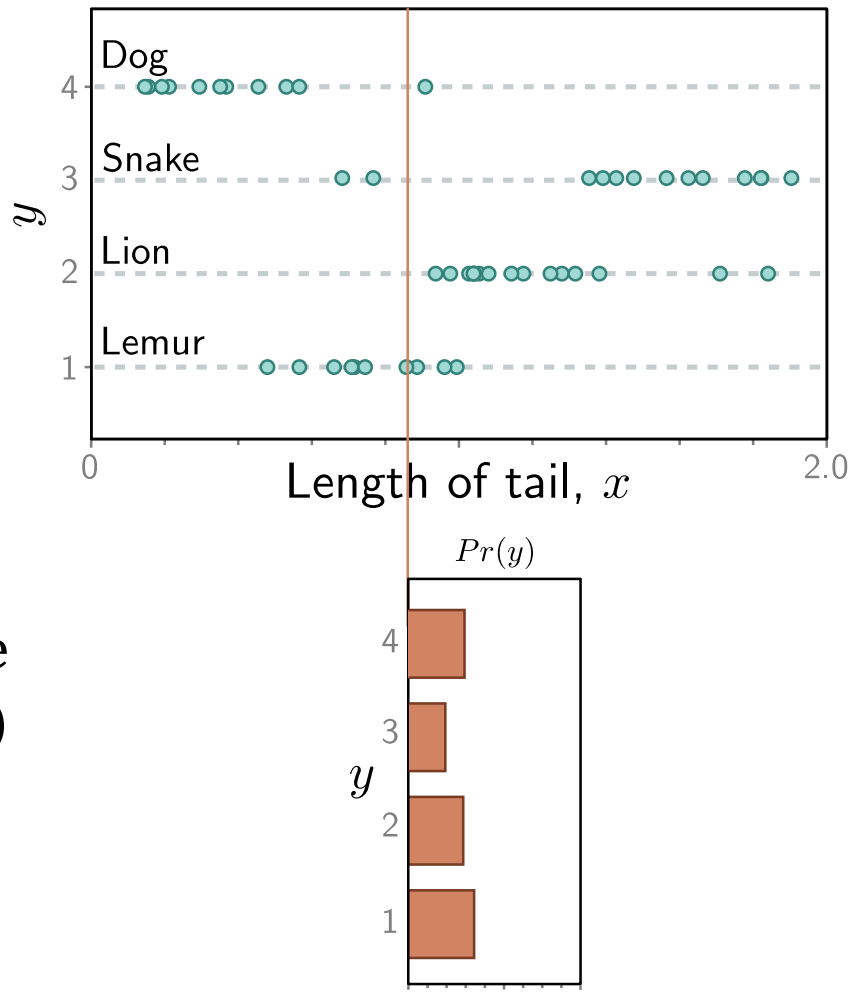
Discrete

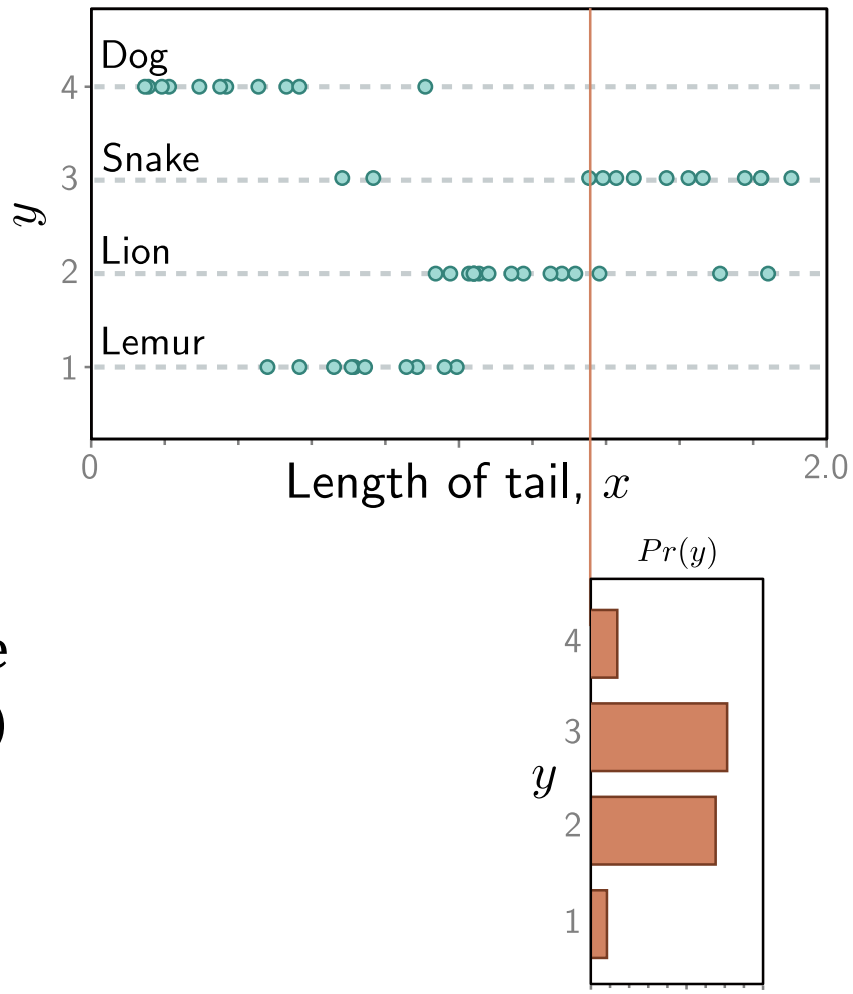
$\Pr(y|x = .55)$



Discrete

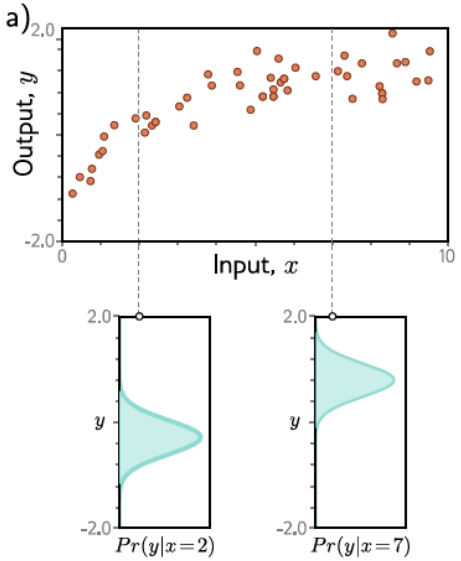
$\Pr(y|x = 0.85)$

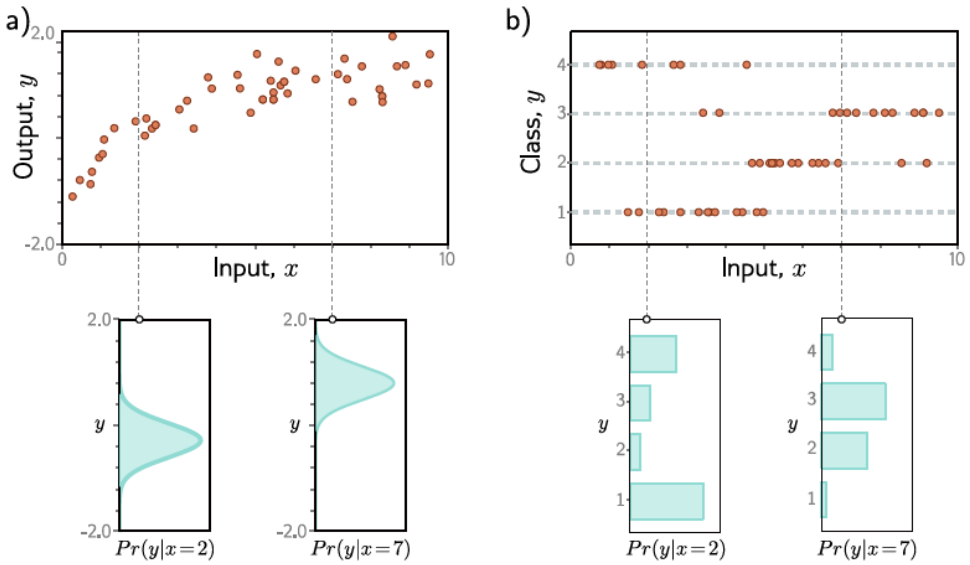


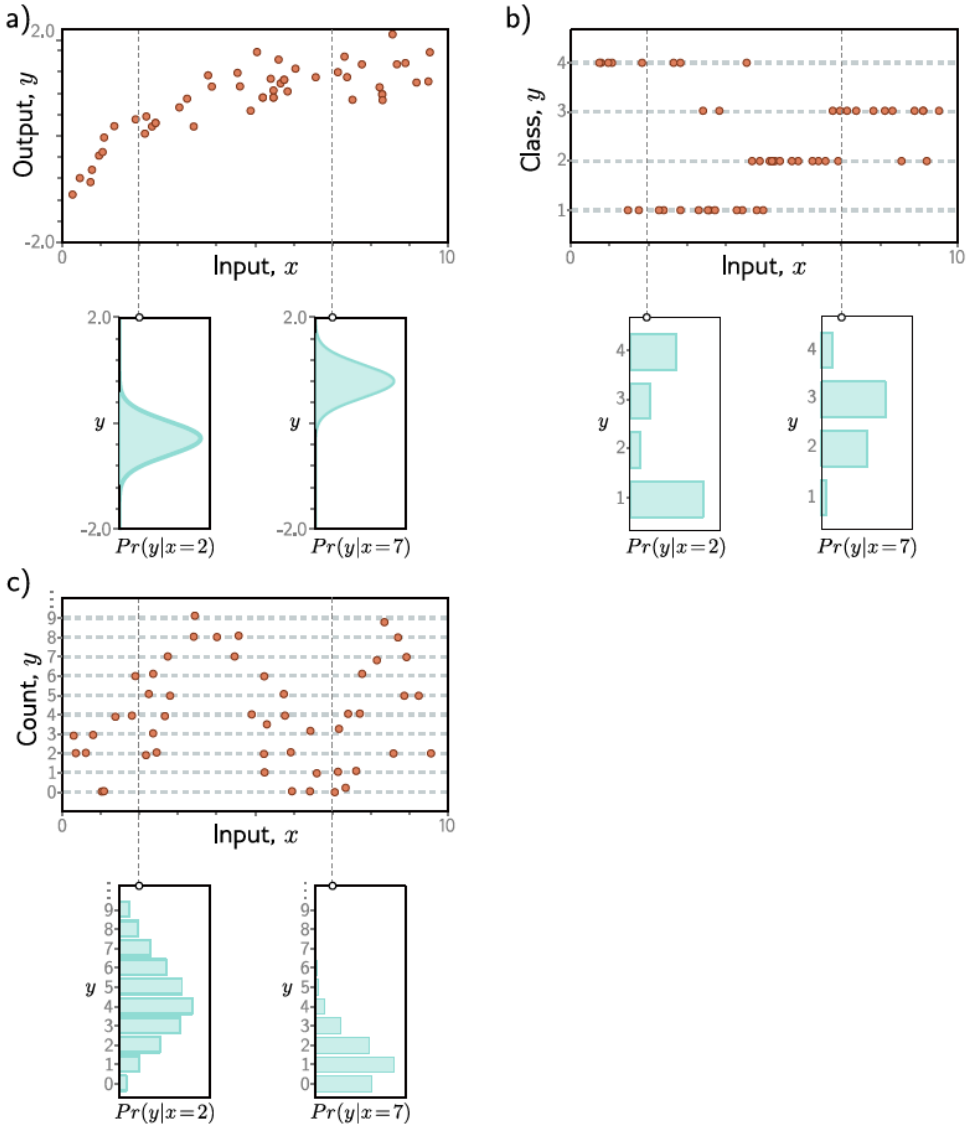


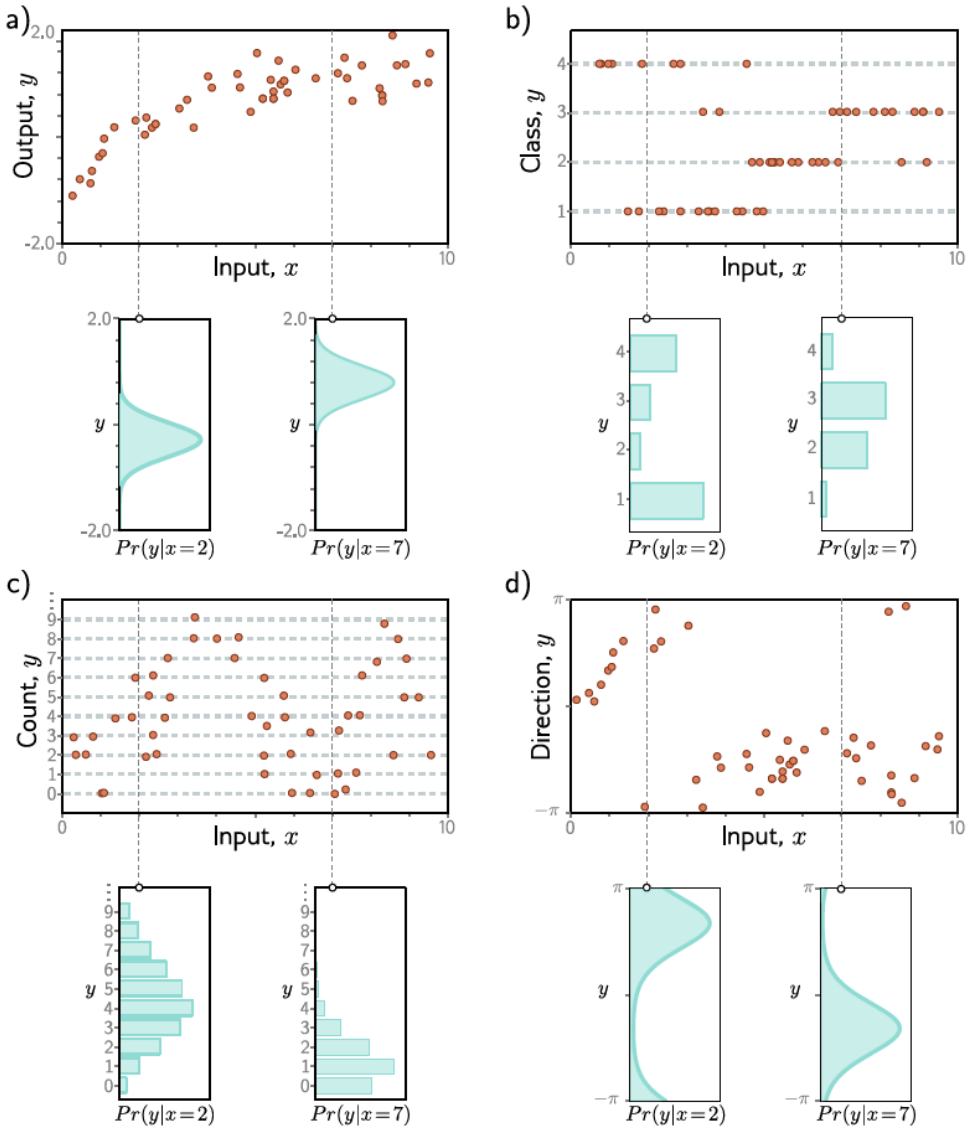
Discrete

$\Pr(y|x = 1.35)$


a)
2.0
Output, y
-2.0
0
Input, x
10
2.0
y
-2.0
$Pr(y|x=2)$
2.0
y
-2.0
$Pr(y|x=7)$


a)
Output, y
2.0
-2.0
0
10
Input, x
2.0
-2.0
y
$Pr(y|x=2)$
$Pr(y|x=7)$
b)
Class, y
4
3
2
1
0
10
Input, x
$Pr(y|x=2)$
$Pr(y|x=7)$


a)
Output, y
Input, x
$Pr(y|x=2)$
$Pr(y|x=7)$
b)
Class, y
Input, x
$Pr(y|x=2)$
$Pr(y|x=7)$
c)
Count, y
Input, x
$Pr(y|x=2)$
$Pr(y|x=7)$


a)
Output, y
Input, x
$Pr(y|x=2)$
$Pr(y|x=7)$
b)
Class, y
Input, x
$Pr(y|x=2)$
$Pr(y|x=7)$
c)
Count, y
Input, x
$Pr(y|x=2)$
$Pr(y|x=7)$
d)
Direction, y
Input, x
$Pr(y|x=2)$
$Pr(y|x=7)$

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:

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

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:

or for short:

$$L[\phi]$$

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

Training

- Loss function:

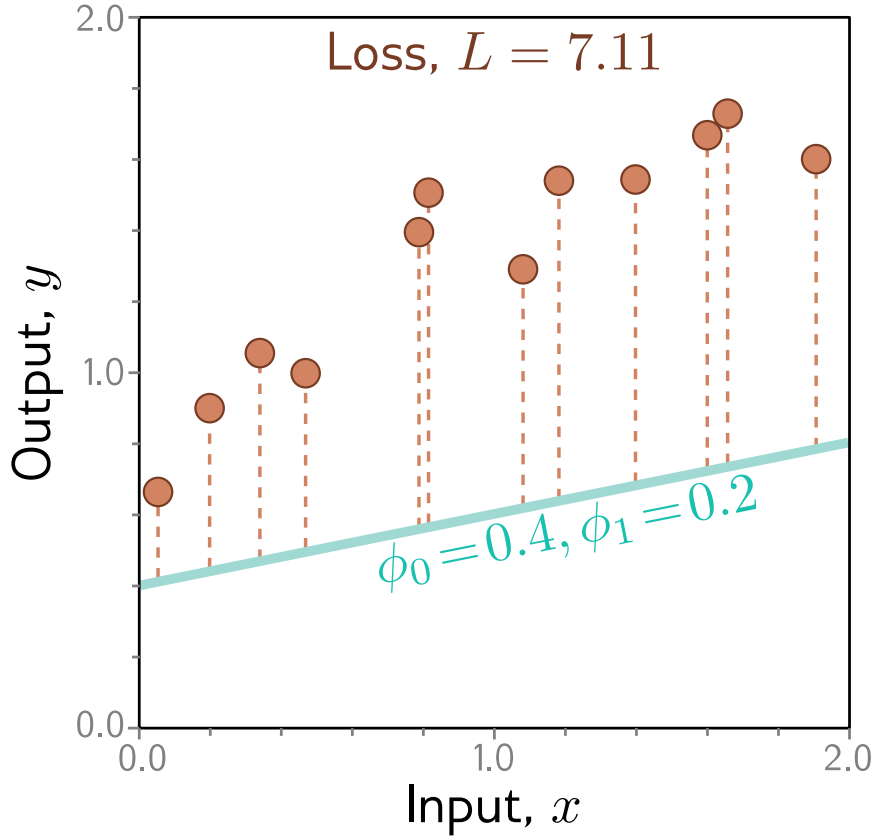
$$L\left[\boldsymbol{\phi}\right]$$

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

- Find the parameters that minimize the loss:

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

Example: 1D Linear regression loss function



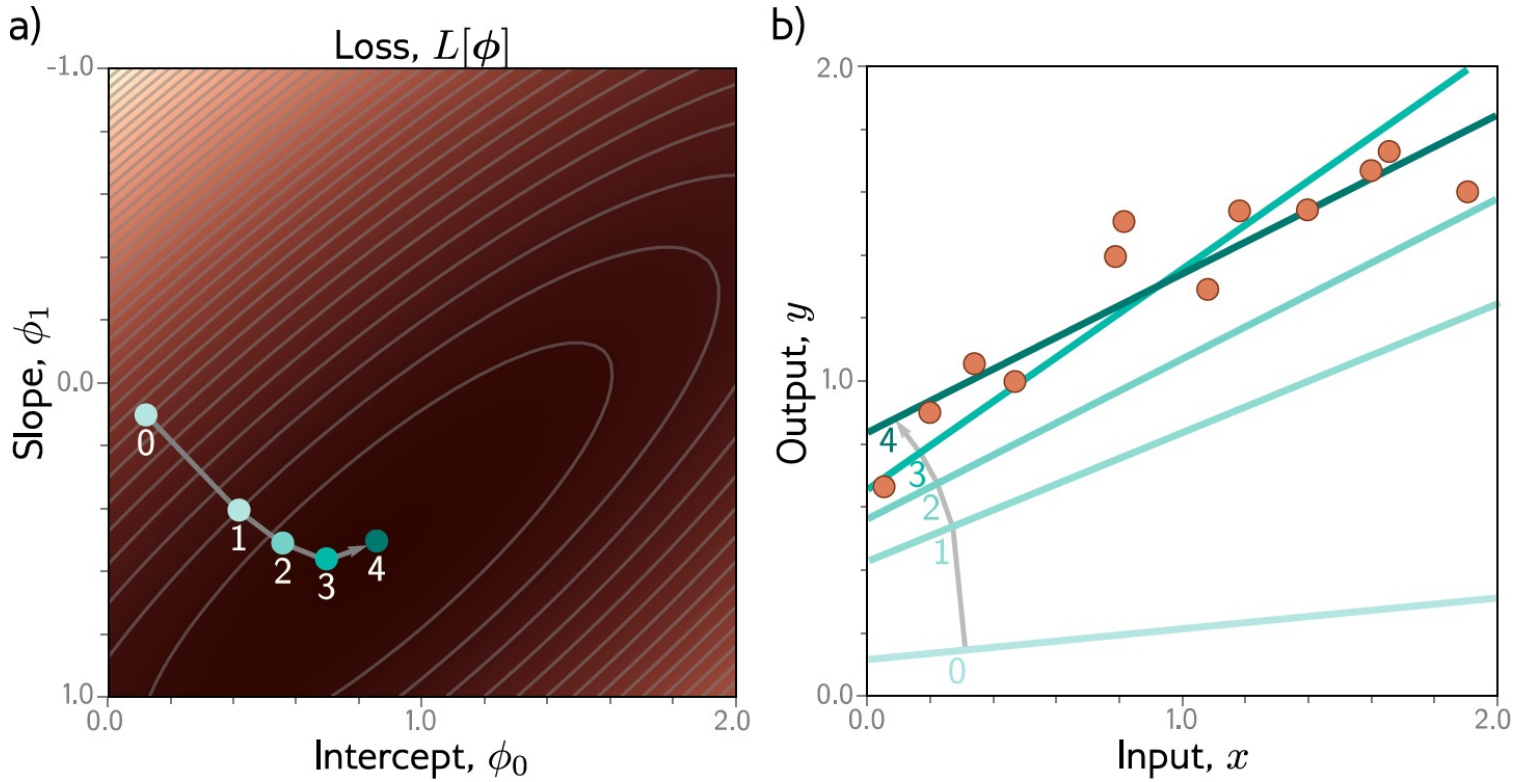
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression training



This technique is known as gradient descent

slido

Please download and install the Slido app on all computers you use

Why do we need a loss function in deep learning?

ⓘ Start presenting to display the poll results on this slide.

Loss functions

- Maximum likelihood
- Recipe for loss functions
- Example 1: univariate regression
- Example 2: binary classification
- Example 3: multiclass classification
- Other types of data
- Multiple outputs
- Cross entropy

Maximum Likelihood Estimation

- In statistics, *maximum likelihood estimation (MLE)* is a method of *estimating the parameters* of an *assumed probability distribution*, *given some observed data*.
- This is achieved by *maximizing a likelihood function* so that, under the assumed statistical model, *the observed data is most probable*.

How do we do this?

- Model predicts output *y* given input *x*

How do we do this?

- ~~Model predicts output *y* given input *x*~~

How do we do this?

- ~~Model predicts output *y* given input *x*~~
- Model predicts a conditional probability distribution:

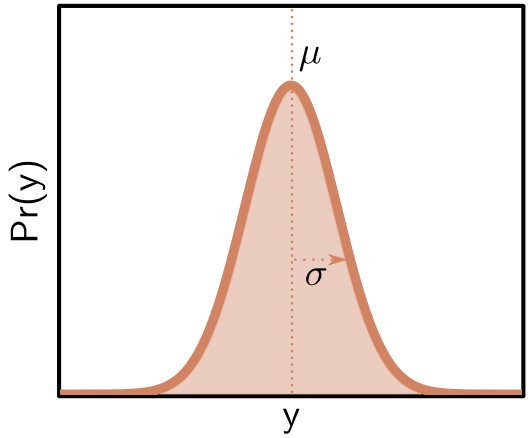
$$Pr(\mathbf{y}|\mathbf{x})$$

  over outputs **y** given inputs **x**.
- Define and minimize a loss function that makes the outputs have high probability

How can a model predict a probability distribution? ➔ Parametric Models

1. Pick a known distribution (e.g., normal distribution) to model output y with parameters $\boldsymbol{\theta}$

   e.g., the normal distribution $\boldsymbol{\theta} = \{\mu, \sigma^2\}$



2. Use model to predict parameters $\boldsymbol{\theta}$ of probability distribution

Maximize the joint, conditional probability

- We know we picked a good model and the right parameters when the joint conditional probability is high for the observed (e.g. training) data.

$$\Pr(y_1, y_2, \ldots, y_I \mid x_1, x_2, \ldots, x_I)$$

Two simplifying assumptions

Identically distributed (the form of the probably distribution is the same for each input/output pair)

$$\Pr(y_1, y_2, \ldots, y_I \mid x_1, x_2, \ldots, x_I) = \prod_{i=1}^{I} \Pr(y_i \mid x_i)$$

Independent

Independent and identically distributed (i.i.d)

Maximum likelihood criterion

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\text{argmax}} \left[\prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{x}_i) \right]$$

$$= \underset{\boldsymbol{\phi}}{\text{argmax}} \left[\prod_{i=1}^{I} Pr(\mathbf{y}_i|\boldsymbol{\theta}_i) \right]$$

$$= \underset{\boldsymbol{\phi}}{\text{argmax}} \left[\prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]) \right]$$

θ_i are the parameters of the probability distribution

ϕ are the parameters of the neural network, e.g.

$$\theta_i = \mathrm{f}[\mathrm{x}_i, \phi]$$

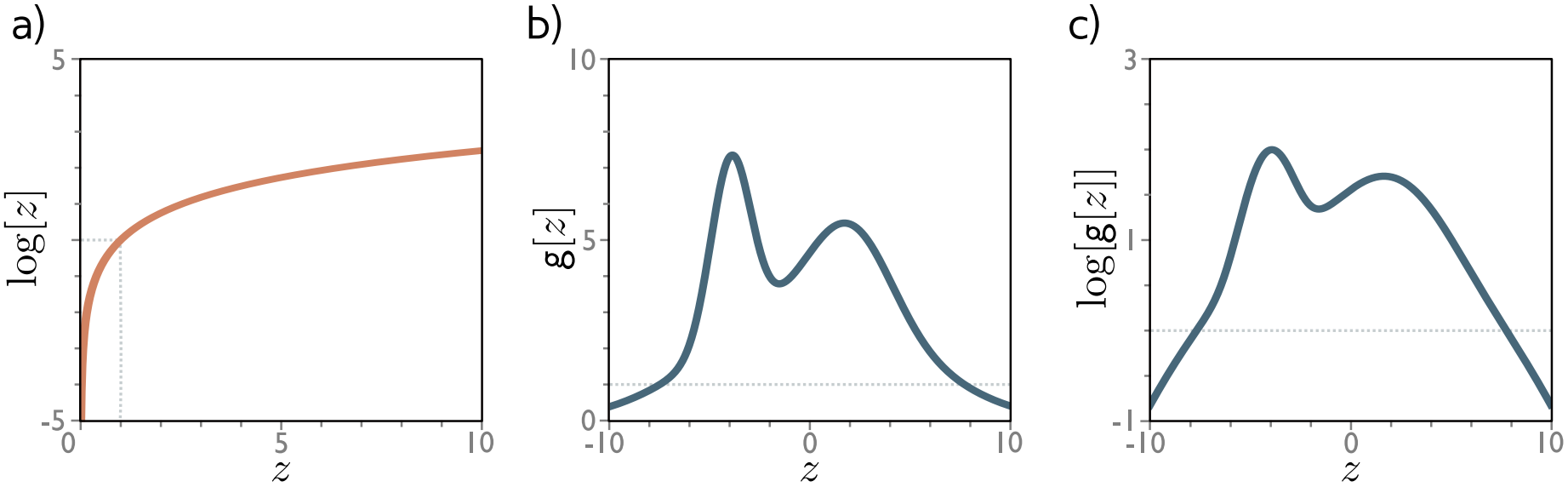
When we consider this probability as a function of the parameters ϕ, we call it a likelihood.

Problem:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \left[\prod_{i=1}^{I} Pr(\mathbf{y}_i | \mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]) \right]$$

- The terms in this product might all be small, $0 \leq \Pr(\cdot) \leq 1$
- The product might get so small that we *can't* easily represent it in fixed precision arithmetic

The log function is monotonic



Maximum of the logarithm of a function is in the same place as maximum of function

Maximum log likelihood

$$\begin{aligned}\hat{\boldsymbol{\phi}} &= \underset{\boldsymbol{\phi}}{\operatorname{argmax}}\left[\prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i,\boldsymbol{\phi}])\right] \\ &= \underset{\boldsymbol{\phi}}{\operatorname{argmax}}\left[\log\left[\prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i,\boldsymbol{\phi}])\right]\right] \\ &= \underset{\boldsymbol{\phi}}{\operatorname{argmax}}\left[\sum_{i=1}^{I} \log\Big[Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i,\boldsymbol{\phi}])\Big]\right]\end{aligned}$$

Now it's a sum of terms, so doesn't matter so much if the terms are small

Minimizing negative log likelihood

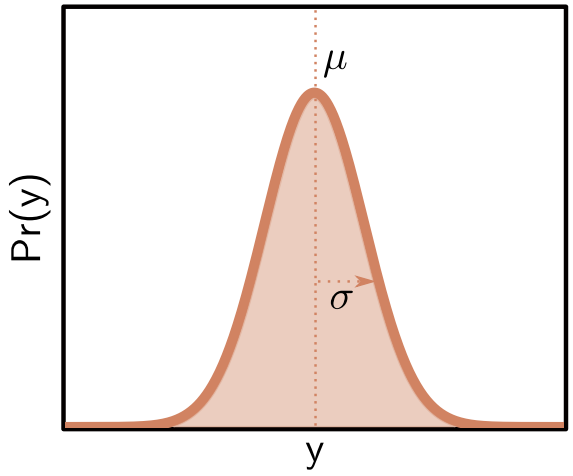
- By convention, we minimize things (i.e., a loss)

$$\begin{aligned}\hat{\boldsymbol{\phi}} &= \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \log\Big[Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}])\Big]\right] \\ &= \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log\Big[Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}])\Big]\right] \\ &= \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \Big[\mathrm{L}[\boldsymbol{\phi}]\Big]\end{aligned}$$

Inference

- But now we predict a probability distribution
- We need an actual prediction (point estimate)
- Find the peak of the probability distribution (i.e., mean for normal)

$$\hat{y} = \hat{\mu} = \underset{y}{\operatorname{argmax}}[\Pr(y|\mathbf{f}[\mathbf{x}, \phi])]]$$



slido

Please download and install the Slido app on all computers you use

Which of the following is true about MLE?

ⓘ Start presenting to display the poll results on this slide.

Loss functions

- Maximum likelihood
- Recipe for loss functions
- Example 1: univariate regression
- Example 2: binary classification
- Example 3: multiclass classification
- Other types of data
- Multiple outputs
- Cross entropy

Recipe for loss functions

1. Choose a suitable probability distribution $Pr(\mathbf{y}|\boldsymbol{\theta})$ that is defined over the domain of the predictions $\mathbf{y}$ and has distribution parameters $\boldsymbol{\theta}$.

Recipe for loss functions

1. Choose a suitable probability distribution $Pr(\mathbf{y}|\boldsymbol{\theta})$ that is defined over the domain of the predictions $\mathbf{y}$ and has distribution parameters $\boldsymbol{\theta}$.
2. Set the machine learning model $\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ to predict one or more of these parameters so $\boldsymbol{\theta} = \mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ and $Pr(\mathbf{y}|\boldsymbol{\theta}) = Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}])$.

Recipe for loss functions

1. Choose a suitable probability distribution $Pr(\mathbf{y}|\boldsymbol{\theta})$ that is defined over the domain of the predictions $\mathbf{y}$ and has distribution parameters $\boldsymbol{\theta}$.
2. Set the machine learning model $\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ to predict one or more of these parameters so $\boldsymbol{\theta} = \mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ and $Pr(\mathbf{y}|\boldsymbol{\theta}) = Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}])$.
3. To train the model, find the network parameters $\hat{\boldsymbol{\phi}}$ that minimize the negative log-likelihood loss function over the training dataset pairs $\{\mathbf{x}_i, \mathbf{y}_i\}$:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[L[\boldsymbol{\phi}]\right] = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log\Big[Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}])\Big] \right]. \tag{5.7}$$

Recipe for loss functions

1. Choose a suitable probability distribution $Pr(\mathbf{y}|\boldsymbol{\theta})$ that is defined over the domain of the predictions $\mathbf{y}$ and has distribution parameters $\boldsymbol{\theta}$.
2. Set the machine learning model $\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ to predict one or more of these parameters so $\boldsymbol{\theta} = \mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ and $Pr(\mathbf{y}|\boldsymbol{\theta}) = Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}])$.
3. To train the model, find the network parameters $\hat{\boldsymbol{\phi}}$ that minimize the negative log-likelihood loss function over the training dataset pairs $\{\mathbf{x}_i, \mathbf{y}_i\}$:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[L[\boldsymbol{\phi}]\right] = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log\Big[Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}])\Big] \right]. \tag{5.7}$$

4. To perform inference for a new test example $\mathbf{x}$, return either the full distribution $Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \hat{\boldsymbol{\phi}}])$ or the maximum of this distribution.

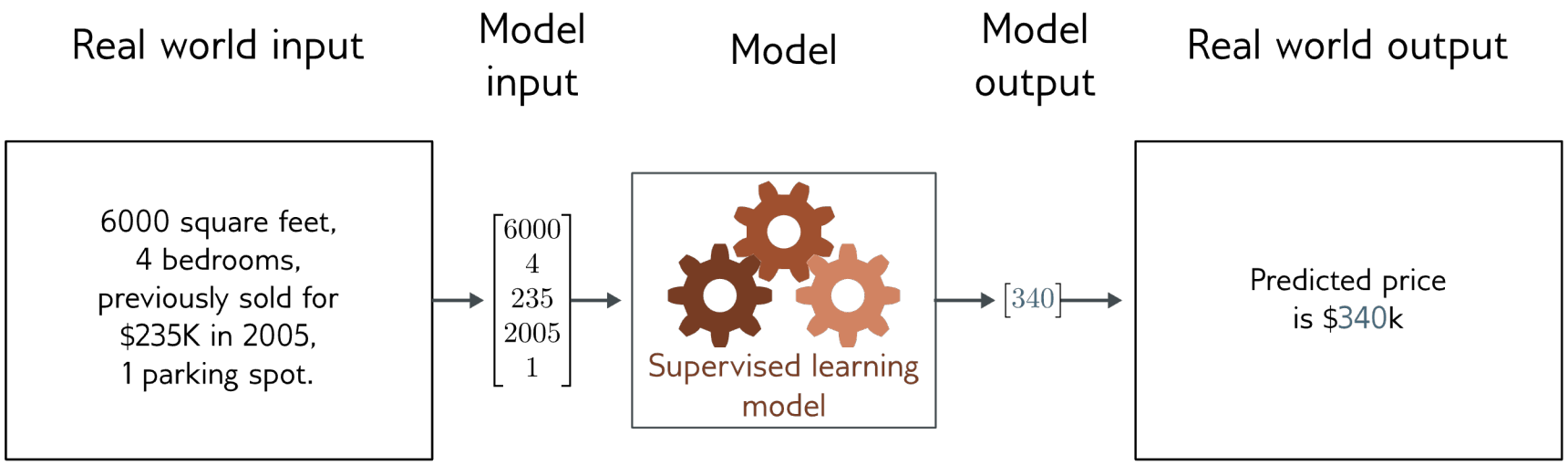
Let’s apply this recipe to

- Example 1: Real valued univariate regression
- Example 2: Binary Classification
- Example 3: Multiclass Classification

Loss functions

- Maximum likelihood
- Recipe for loss functions
- Example 1: univariate regression
- Example 2: binary classification
- Example 3: multiclass classification
- Other types of data
- Multiple outputs
- Cross entropy

Example 1: univariate regression

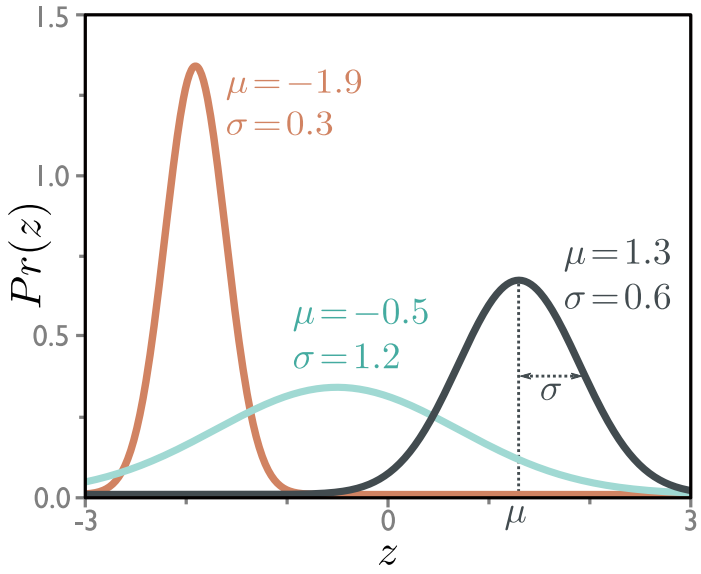


Example 1: univariate regression

1. Choose a suitable probability distribution $Pr(\mathbf{y}|\boldsymbol{\theta})$ that is defined over the domain of the predictions $\mathbf{y}$ and has distribution parameters $\boldsymbol{\theta}$.

- Predict scalar output: $y \in \mathbb{R}$
- Sensible probability distribution:
  - Normal distribution

$$Pr(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y-\mu)^2}{2\sigma^2}\right]$$



Example 1: univariate regression

2. Set the machine learning model $\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ to predict one or more of these parameters so $\boldsymbol{\theta} = \mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ and $Pr(\mathbf{y}|\boldsymbol{\theta}) = Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}])$.

$$Pr(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y-\mu)^2}{2\sigma^2}\right]$$

In this case, just the mean

$$Pr(y|\mathrm{f}[\mathbf{x}, \boldsymbol{\phi}], \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y-\mathrm{f}[\mathbf{x}, \boldsymbol{\phi}])^2}{2\sigma^2}\right]$$

Just learn the mean, μ, and assume the variance is fixed,.

Example 1: univariate regression

3. To train the model, find the network parameters $\hat{\boldsymbol{\phi}}$ that minimize the negative log-likelihood loss function over the training dataset pairs $\{\mathbf{x}_i, \mathbf{y}_i\}$:

$$L[\boldsymbol{\phi}] = -\sum_{i=1}^{I} \log\left[Pr(y_i|\mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}], \sigma^2)\right]$$

$$= -\sum_{i=1}^{I} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y_i - \mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}])^2}{2\sigma^2}\right]\right]$$

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(y_i - \mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}])^2}{2\sigma^2} \right] \right] \right]$$

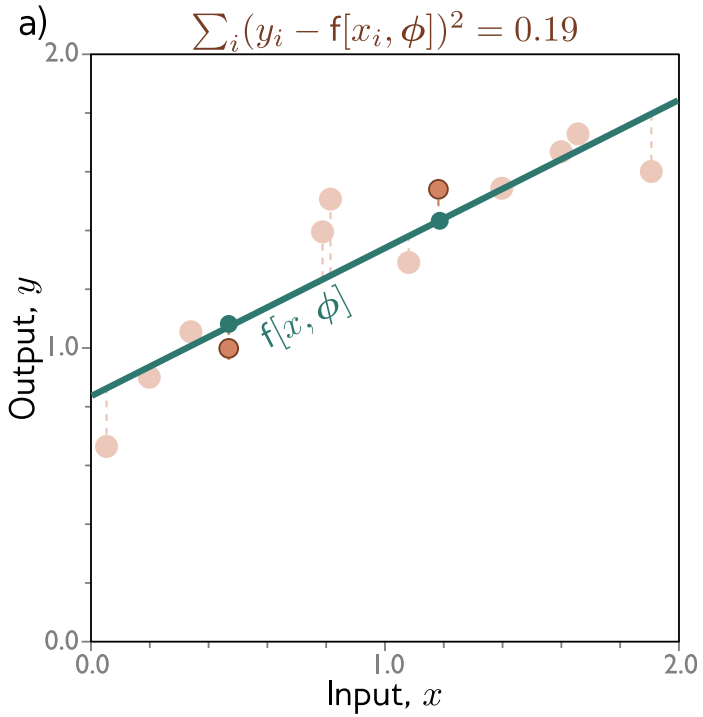
$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(y_i - \mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}])^2}{2\sigma^2} \right] \right] \right]$$

You'll fill in the details in Homework 3:

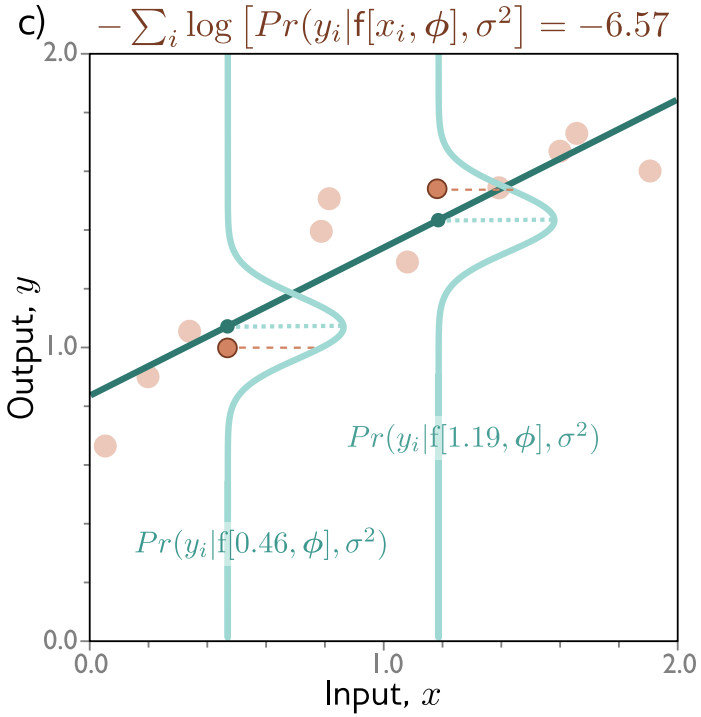
- Log of products is sum of logs
- Log of exp() cancels each other out
- Ignore constants in minimization

$$= \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} (y_i - \mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}])^2 \right], \quad \longleftarrow \text{ Least squares!}$$

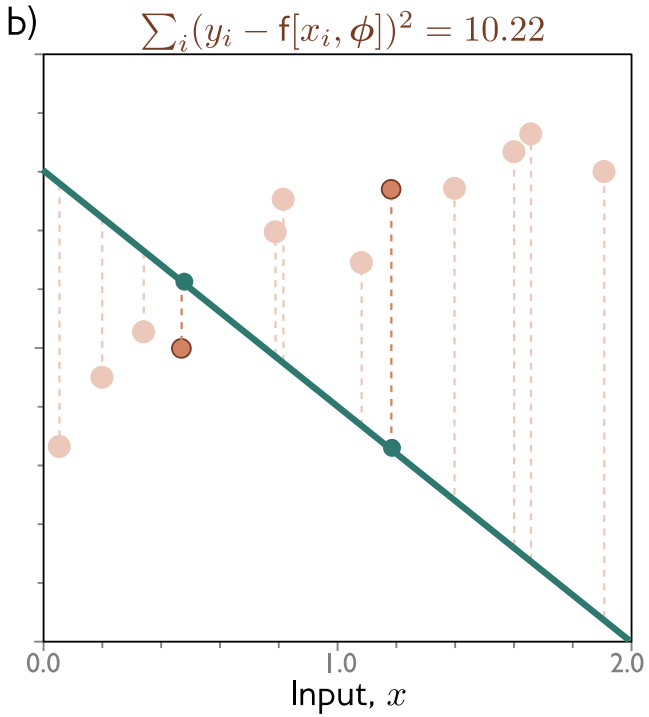
Least squares



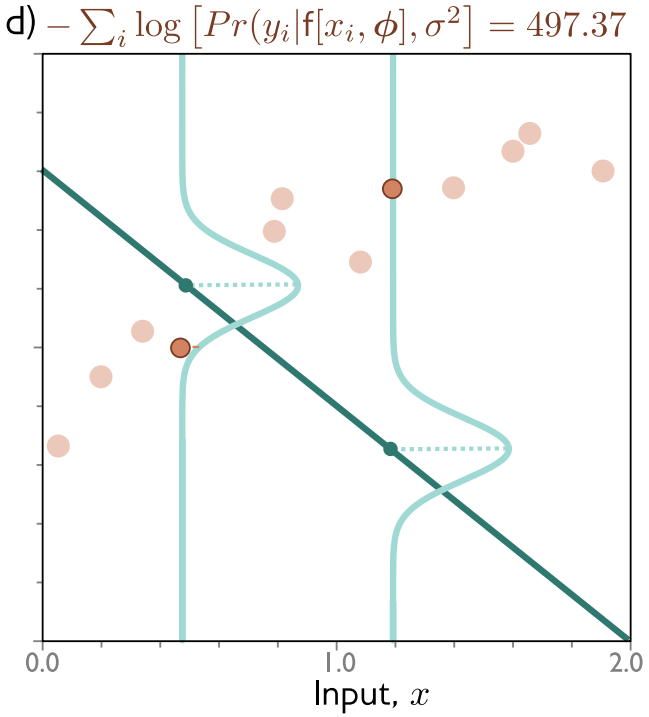
Negative log likelihood



Least squares



Maximum likelihood



Example 1: univariate regression

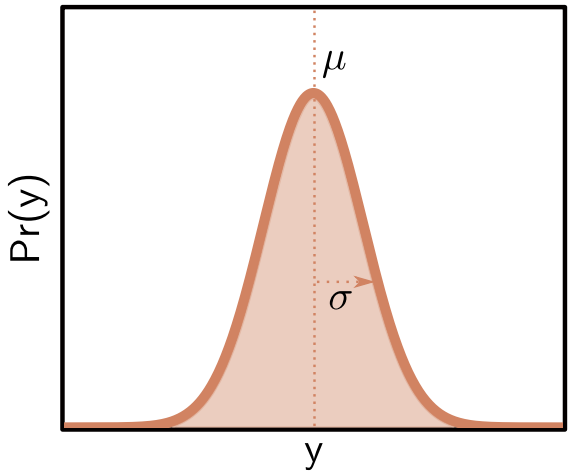
4. To perform inference for a new test example $\mathbf{x}$, return either the full distribution $Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \hat{\boldsymbol{\phi}}])$ or the maximum of this distribution.

Full distribution:

$$Pr(y|\mathrm{f}[\mathbf{x}, \boldsymbol{\phi}], \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y - \mathrm{f}[\mathbf{x}, \boldsymbol{\phi}])^2}{2\sigma^2}\right]$$

Max probability:

$$\hat{y} = \hat{\mu} = \mathrm{f}[\mathrm{x}\,|\boldsymbol{\phi}]$$



Estimating variance

- Perhaps surprisingly, the variance term disappeared:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(y_i - \mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}])^2}{2\sigma^2} \right] \right] \right]$$

$$= \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} (y_i - \mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}])^2 \right]$$

Estimating variance

- But we could learn it during training:

$$\hat{\boldsymbol{\phi}}, \hat{\sigma}^2 = \underset{\boldsymbol{\phi}, \sigma^2}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(y_i - \mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}])^2}{2\sigma^2} \right] \right] \right]$$

- Do gradient descent on both model parameters, $\boldsymbol{\phi}$, and the variance, σ^2

$$\frac{\partial L}{\partial \boldsymbol{\phi}} \text{ and } \frac{\partial L}{\partial \sigma^2}$$

We will explore gradient descent on functions in discussion and future homework.

Heteroscedastic regression

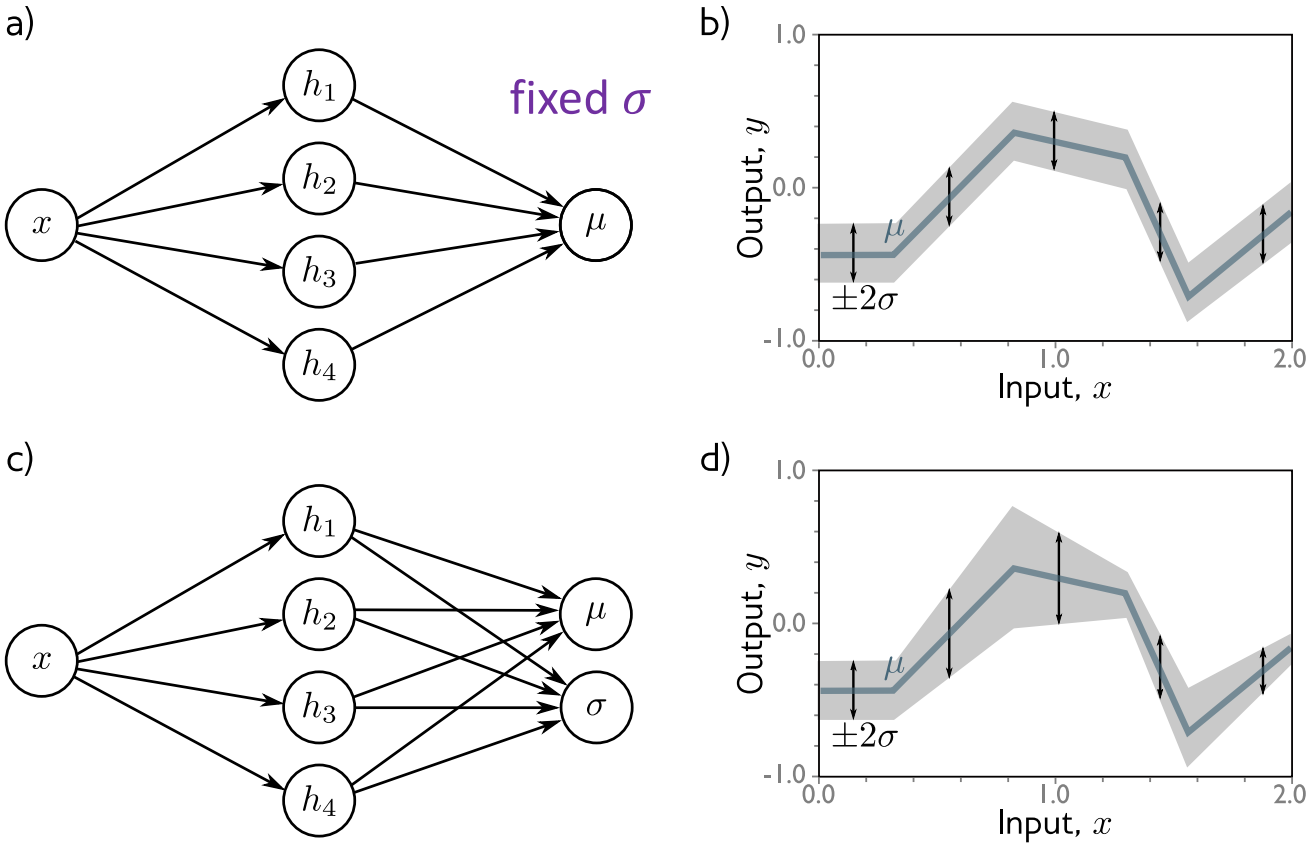
- We were assuming that the noise σ^2 is the same everywhere (homoscedastic).
- But we could make the noise a function of the data *x.*
- Build a model with two outputs:

$$\mu = \mathrm{f}_1[\mathbf{x}, \boldsymbol{\phi}]$$
$$\sigma^2 = \mathrm{f}_2[\mathbf{x}, \boldsymbol{\phi}]^2$$

Squared to ensure it is positive

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\mathrm{argmin}} \left[-\sum_{i=1}^{I} \log \left[\frac{1}{\sqrt{2\pi \mathrm{f}_2[\mathbf{x}_i, \boldsymbol{\phi}]^2}} \right] - \frac{(y_i - \mathrm{f}_1[\mathbf{x}_i, \boldsymbol{\phi}])^2}{2\mathrm{f}_2[\mathbf{x}_i, \boldsymbol{\phi}]^2} \right]$$

Heteroscedastic regression



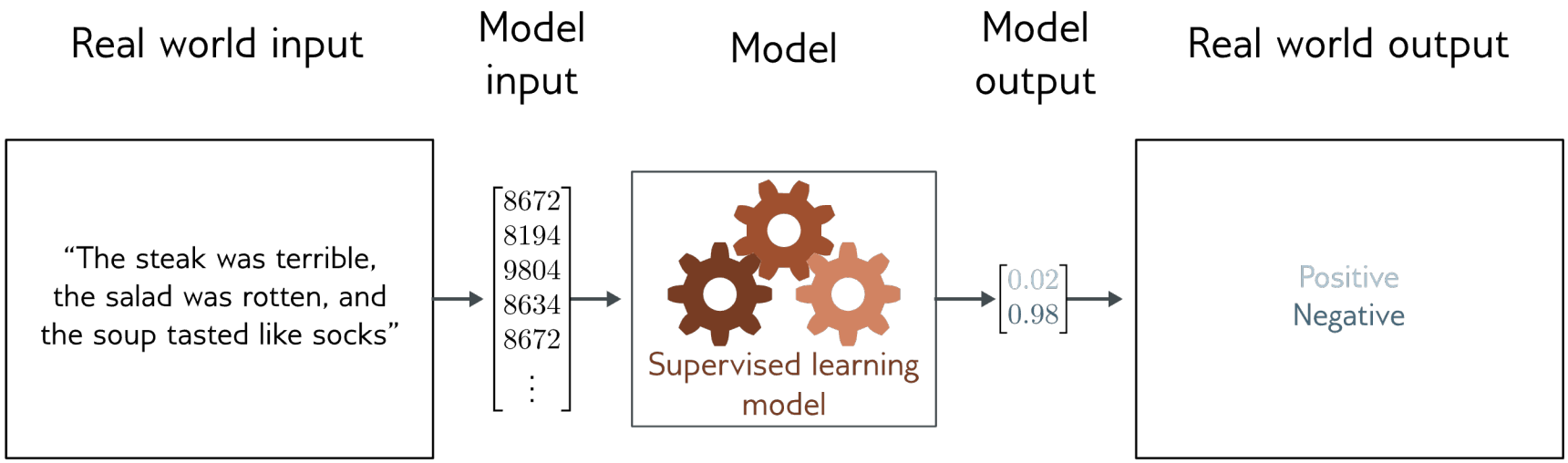
Example 1: Univariate Regression Takeaways

- *Least squares loss* is a good choice assuming normal distribution
- The best prediction is the predicted mean
- We can also estimate global or local variance

Loss functions

- Maximum likelihood
- Recipe for loss functions
- Example 1: univariate regression
- Example 2: binary classification
- Example 3: multiclass classification
- Other types of data
- Multiple outputs
- Cross entropy

Example 2: binary classification



- Goal: predict which of two classes $y \in \{0, 1\}$ the input *x* belongs to

Example 2: binary classification

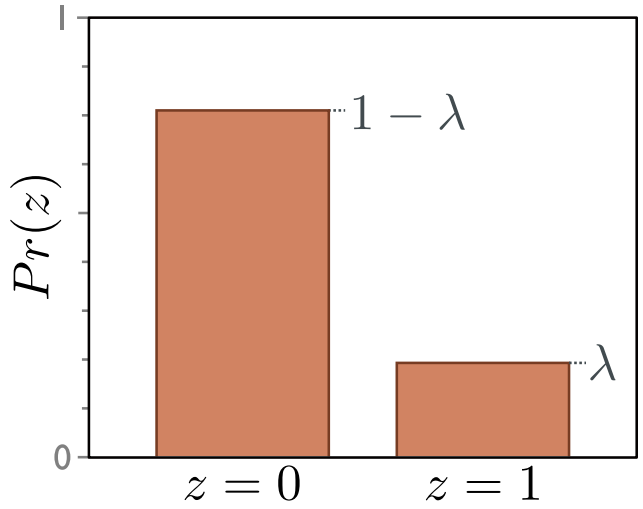
1. Choose a suitable probability distribution $Pr(\mathbf{y}|\boldsymbol{\theta})$ that is defined over the domain of the predictions $\mathbf{y}$ and has distribution parameters $\boldsymbol{\theta}$.

- Domain: $y \in \{0, 1\}$
- Bernoulli distribution
- One parameter $\lambda \in [0,1]$

$$Pr(y|\lambda) = \begin{cases} 1-\lambda & y = 0 \\ \lambda & y = 1 \end{cases}$$

or

$$Pr(y|\lambda) = (1-\lambda)^{1-y} \cdot \lambda^{y}$$



Example 2: binary classification

2. Set the machine learning model $\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ to predict one or more of these parameters so $\boldsymbol{\theta} = \mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ and $Pr(\mathbf{y}|\boldsymbol{\theta}) = Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}])$.

Problem:

- Output of neural network can be anything
- Parameter $\lambda \in [0,1]$

Solution:

- Pass through function that maps "anything" to [0,1]

Example 2: binary classification

2. Set the machine learning model $\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ to predict one or more of these parameters so $\boldsymbol{\theta} = \mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ and $Pr(\mathbf{y}|\boldsymbol{\theta}) = Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}])$.

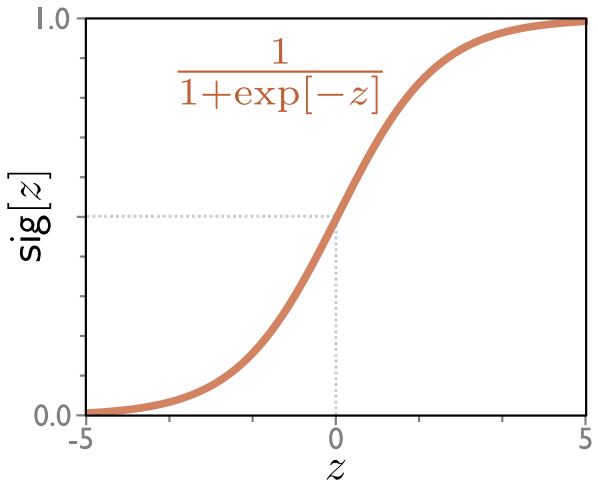
Problem:

- Output of neural network can be anything
- Parameter $\lambda \in [0,1]$

Solution:

- Pass through logistic sigmoid function that maps "anything to [0,1]:

$$\text{sig}[z] = \frac{1}{1+\exp[-z]}$$



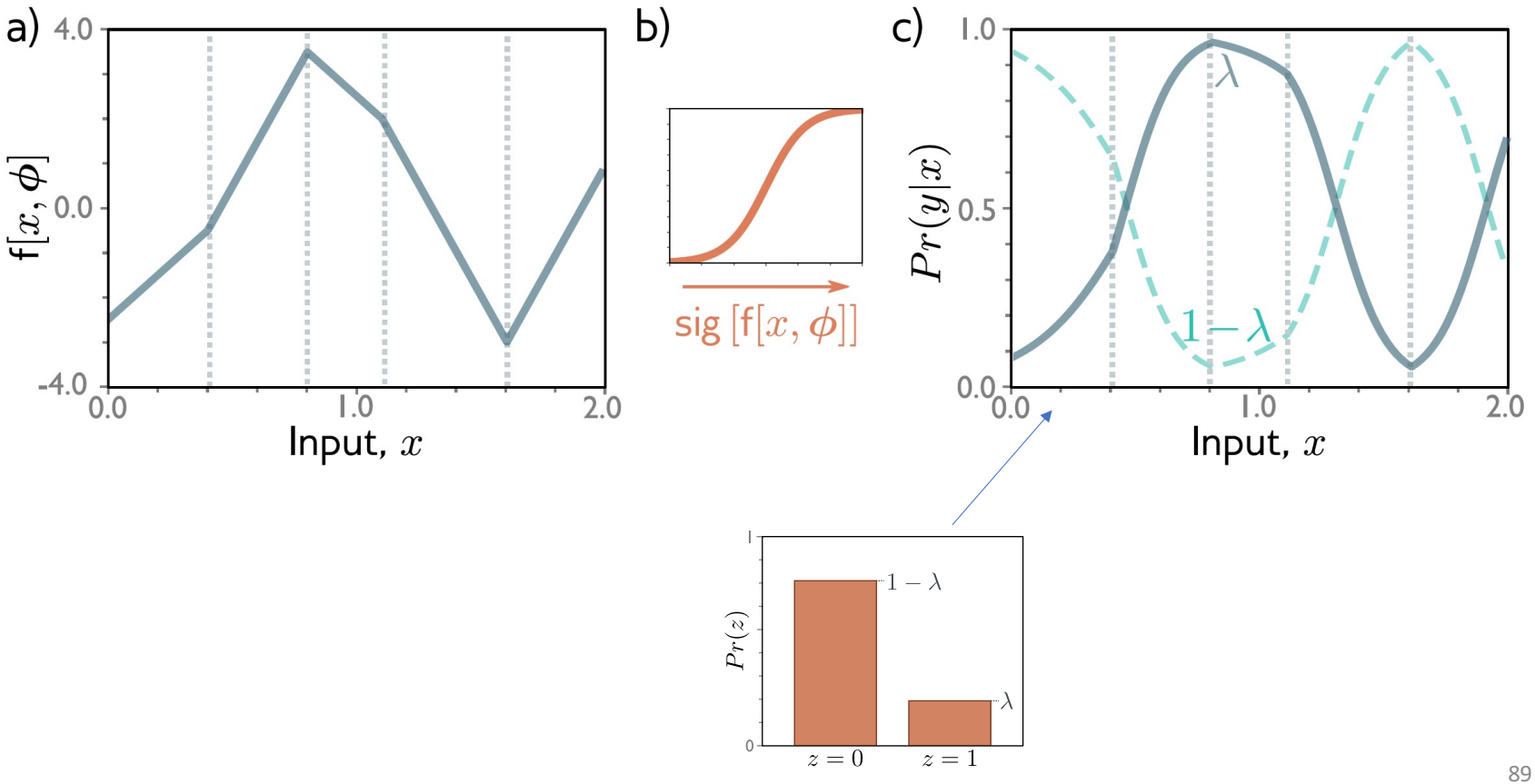
Example 2: binary classification

2. Set the machine learning model $\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ to predict one or more of these parameters so $\boldsymbol{\theta} = \mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ and $Pr(\mathbf{y}|\boldsymbol{\theta}) = Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}])$.

$$Pr(y|\lambda) = (1-\lambda)^{1-y} \cdot \lambda^{y}$$

$$Pr(y|\mathbf{x}) = (1-\text{sig}[\text{f}[\mathbf{x}|\boldsymbol{\phi}]])^{1-y} \cdot \text{sig}[\text{f}[\mathbf{x}|\boldsymbol{\phi}]]^{y}$$

Example 2: binary classification



Example 2: binary classification

3. To train the model, find the network parameters $\hat{\boldsymbol{\phi}}$ that minimize the negative log-likelihood loss function over the training dataset pairs $\{\mathbf{x}_i, \mathbf{y}_i\}$:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[L[\boldsymbol{\phi}]\right] = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log\Big[Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}])\Big]\right]. \tag{5.7}$$

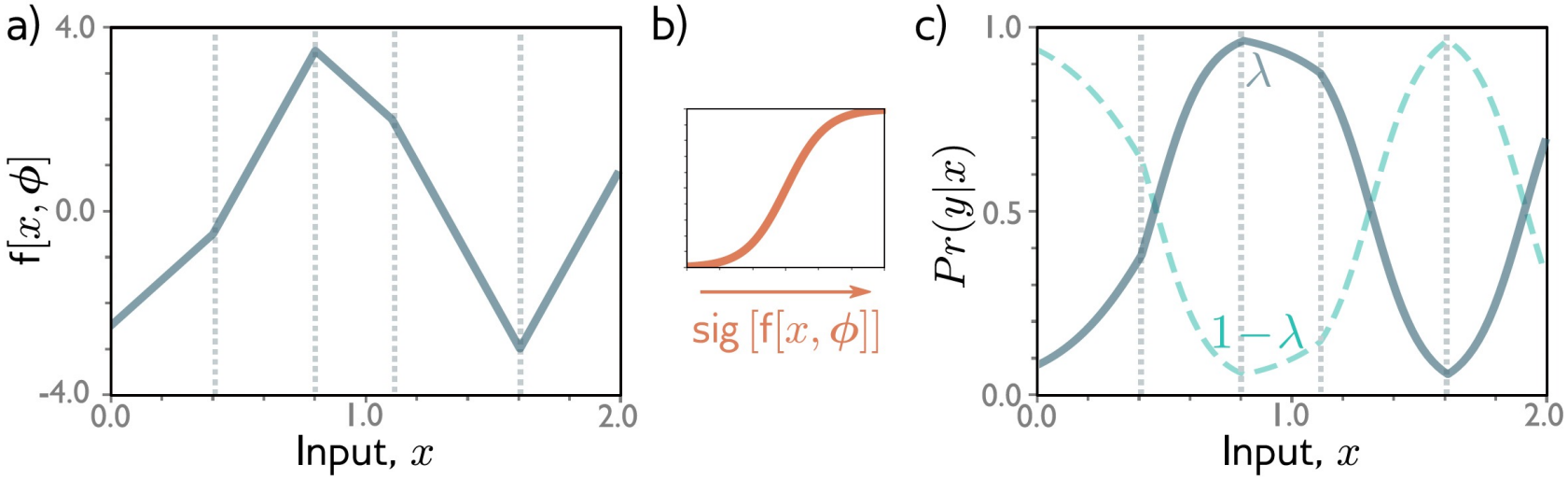
$$Pr(y|\mathbf{x}) = (1 - \text{sig}[\text{f}[\mathbf{x}|\boldsymbol{\phi}]])^{1-y} \cdot \text{sig}[\text{f}[\mathbf{x}|\boldsymbol{\phi}]]^{y}$$

$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} -(1 - y_i) \log\left[1 - \text{sig}[\text{f}[\mathbf{x}_i|\boldsymbol{\phi}]]\right] - y_i \log\left[\text{sig}[\text{f}[\mathbf{x}_i|\boldsymbol{\phi}]]\right]$$

Also called binary cross-entropy loss as it is result from cross-entropy loss calculation – discussed later.

Example 2: binary classification

4. To perform inference for a new test example $\mathbf{x}$, return either the full distribution $Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \hat{\phi}])$ or the maximum of this distribution.



Choose y=1 where λ is greater than 0.5, otherwise 0

And we get a probability estimate!

slido

Please download and install the Slido app on all computers you use

Which statement correctly differentiates least squares and cross-entropy loss?

ⓘ Start presenting to display the poll results on this slide.

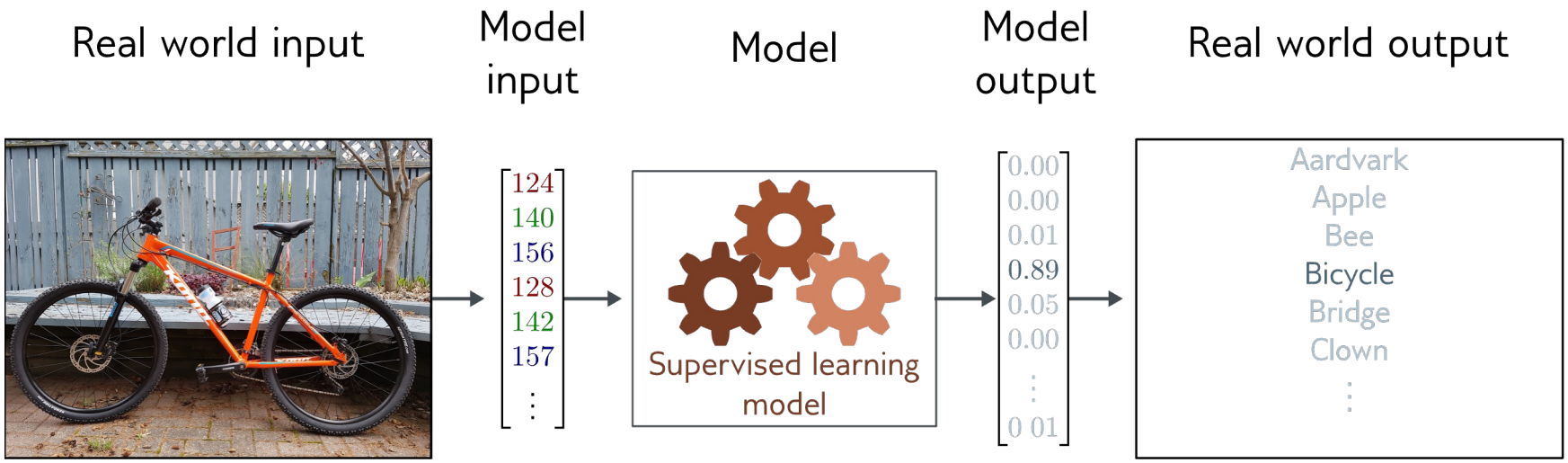
Example 2: Binary Classification Takeaways

- Binary cross entropy loss as the loss function
- Threshold to get prediction
- We also get a probability or “confidence value”

Loss functions

- Maximum likelihood
- Recipe for loss functions
- Example 1: univariate regression
- Example 2: binary classification
- Example 3: multiclass classification
- Other types of data
- Multiple outputs
- Cross entropy

Example 3: multiclass classification



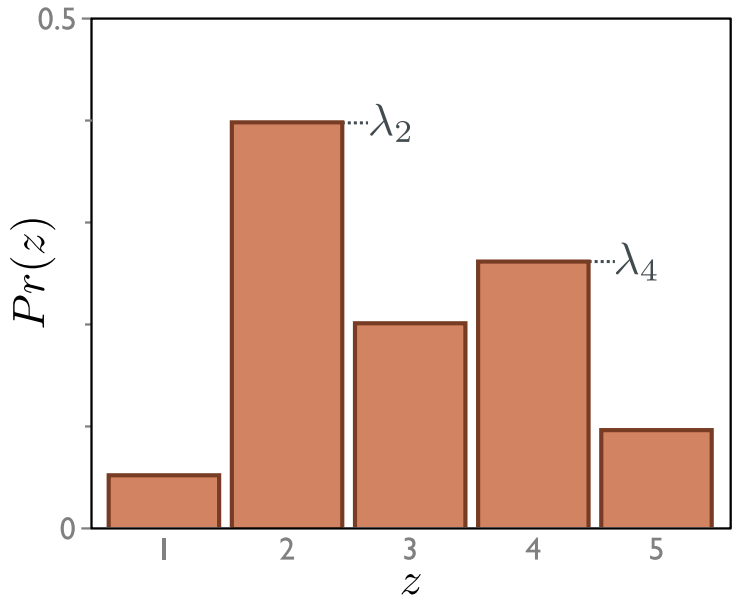
Goal: predict which of K classes $y \in \{1, 2, \ldots, K\}$ the input *x* belongs to

Example 3: multiclass classification

1. Choose a suitable probability distribution $Pr(\mathbf{y}|\boldsymbol{\theta})$ that is defined over the domain of the predictions $\mathbf{y}$ and has distribution parameters $\boldsymbol{\theta}$.

- Domain: $y \in \{1, 2, \ldots, K\}$
- **Categorical distribution**
- K parameters $\lambda_k \in [0,1]$
- $\sum_k \lambda_k = 1$

$$Pr(y = k) = \lambda_k$$



Example 3: multiclass classification

2. Set the machine learning model $\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ to predict one or more of these parameters so $\boldsymbol{\theta} = \mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ and $Pr(\mathbf{y}|\boldsymbol{\theta}) = Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}])$.

Problem:

- Output of neural network can be anything
- Parameters $\lambda_k \in [0,1]$, sum to one

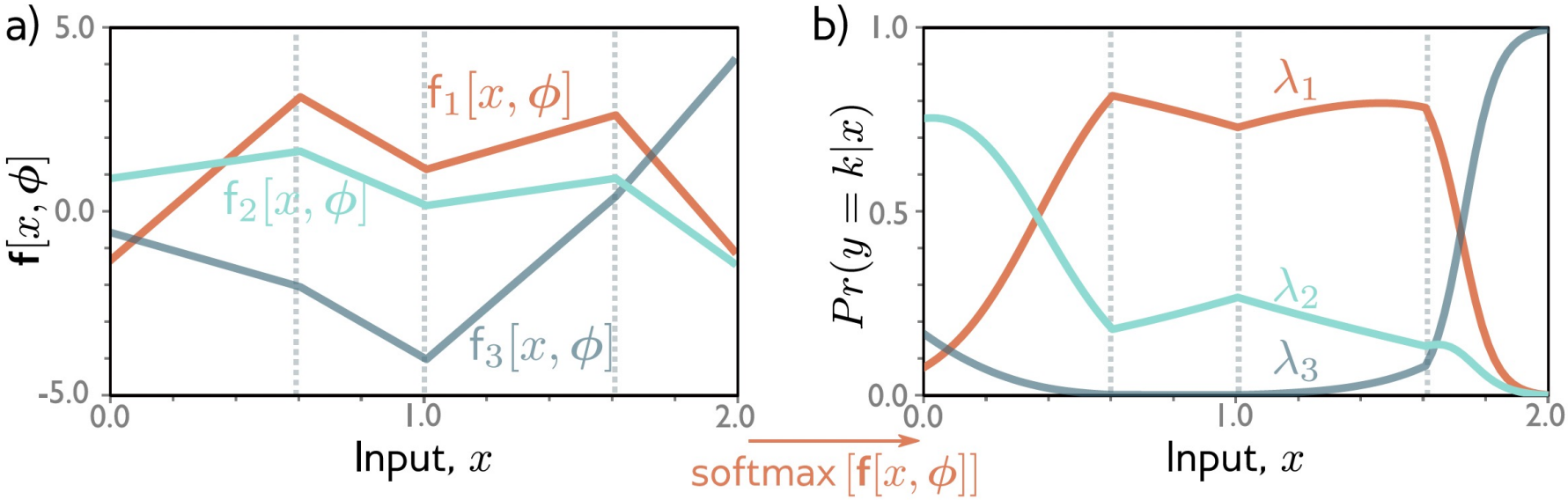
$$\text{softmax}_k[\mathbf{z}] = \frac{\exp[z_k]}{\sum_{k'=1}^{K} \exp[z_{k'}]}$$

Solution:

- Pass through function that maps "anything" to [0,1] and sums to one

$$Pr(y = k|\mathbf{x}) = \text{softmax}_k[\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]]$$

Example 3: multiclass classification



$$Pr(y = k|\mathbf{x}) = \text{softmax}_k[\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]]$$

Example 3: multiclass classification

3. To train the model, find the network parameters $\hat{\boldsymbol{\phi}}$ that minimize the negative log-likelihood loss function over the training dataset pairs $\{\mathbf{x}_i, \mathbf{y}_i\}$:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[L[\boldsymbol{\phi}]\right] = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log\Big[Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}])\Big]\right]. \tag{5.7}$$

$$L[\boldsymbol{\phi}] = -\sum_{i=1}^{I} \log\left[\text{softmax}_{y_i}\left[\mathbf{f}\left[\mathbf{x}_i, \boldsymbol{\phi}\right]\right]\right]$$

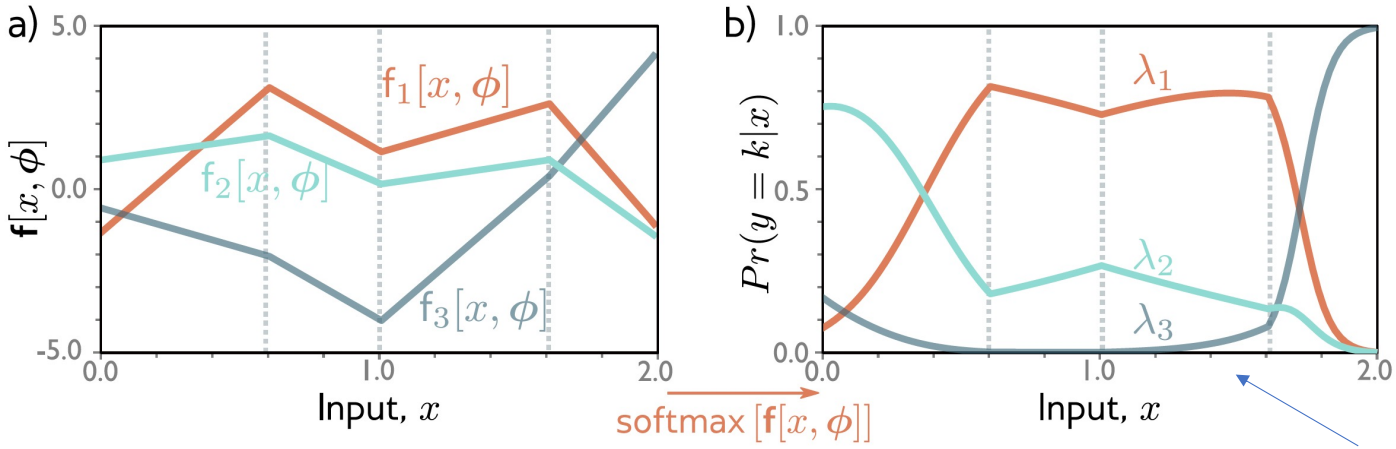
$$= -\sum_{i=1}^{I} \text{f}_{y_i}\left[\mathbf{x}_i, \boldsymbol{\phi}\right] - \log\left[\sum_{k=1}^{K} \exp\left[\ \text{f}_k\left[\mathbf{x}_i, \boldsymbol{\phi}\right]\right]\right]$$

$$\text{softmax}_k[\mathbf{z}] = \frac{\exp[z_k]}{\sum_{k'=1}^{K} \exp[z_{k'}]}$$

Multiclass cross-entropy loss

Example 3: multiclass classification

4. To perform inference for a new test example $\mathbf{x}$, return either the full distribution $Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \hat{\phi}])$ or the maximum of this distribution.



Choose the class with the largest probability

We also get probability or "confidence"

Loss functions

- Maximum likelihood
- Recipe for loss functions
- Example 1: univariate regression
- Example 2: binary classification
- Example 3: multiclass classification
- Other types of data
- Multiple outputs
- Cross entropy

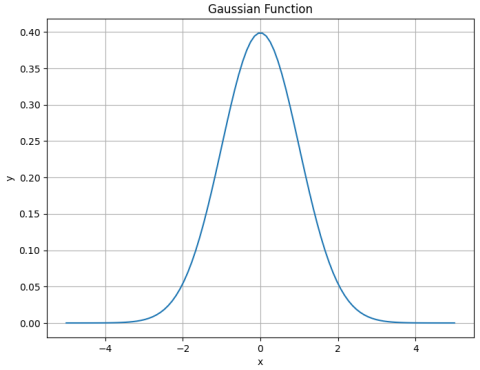
Other data types

Data Type	Domain	Distribution	Use
univariate, continuous, unbounded	$y \in \mathbb{R}$	univariate normal	regression
univariate, continuous, unbounded	$y \in \mathbb{R}$	Laplace or t-distribution	robust regression
univariate, continuous, unbounded	$y \in \mathbb{R}$	mixture of Gaussians	multimodal regression
univariate, continuous, bounded below	$y \in \mathbb{R}^+$	exponential or gamma	predicting magnitude
univariate, continuous, bounded	$y \in [0, 1]$	beta	predicting proportions
multivariate, continuous, unbounded	$\mathbf{y} \in \mathbb{R}^K$	multivariate normal	multivariate regression
univariate, continuous, circular	$y \in (-\pi, \pi]$	von Mises	predicting direction
univariate, discrete, binary	$y \in \{0, 1\}$	Bernoulli	binary classification
univariate, discrete, bounded	$y \in \{1, 2, \ldots, K\}$	categorical	multiclass classification
univariate, discrete, bounded below	$y \in [0, 1, 2, 3, \ldots]$	Poisson	predicting event counts
multivariate, discrete, permutation	$\mathbf{y} \in \text{Perm}[1, 2, \ldots, K]$	Plackett-Luce	ranking

Figure 5.11 Distributions for loss functions for different prediction types.

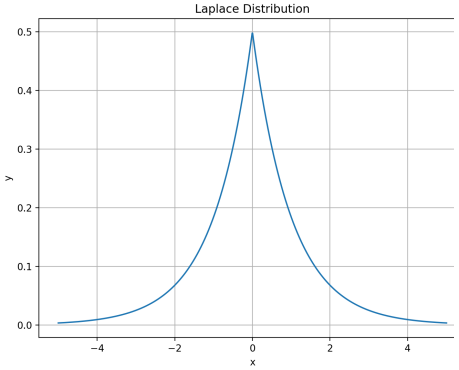
Other Distributions

Gaussian



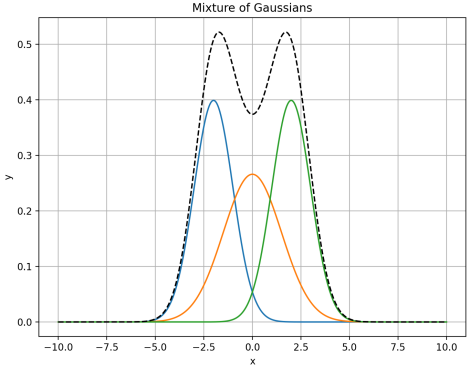
$y \in \mathbb{R}$ Regression

Laplace



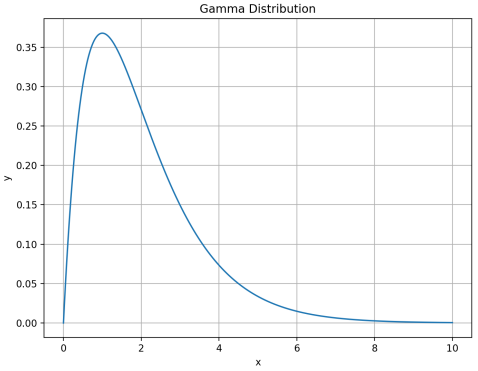
$y \in \mathbb{R}$ Robust Regression

Mixture of Gaussians



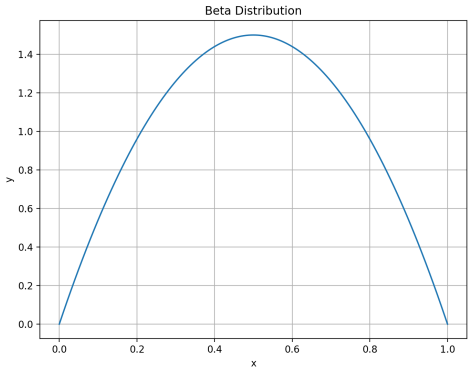
$y \in \mathbb{R}$ Multimodal Regression

Gamma



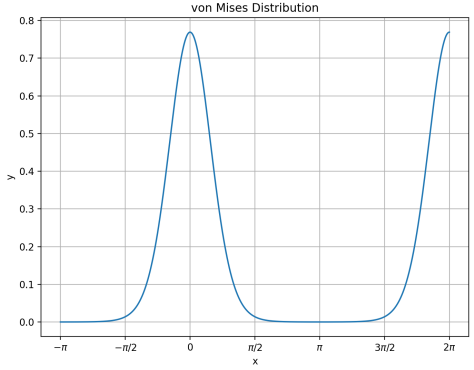
$y \in \mathbb{R}^+$ Predict Magnitude

Beta



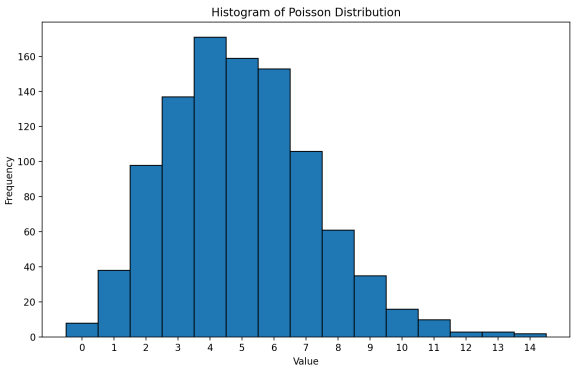
$y \in [0,1]$Predict Proportions

Von Mises



$y \in (-\pi, \pi]$Predict Directions

Poisson



$y \in [0,1,2, \ldots]$Predict Event Counts

Loss functions

- Maximum likelihood
- Recipe for loss functions
- Example 1: univariate regression
- Example 2: binary classification
- Example 3: multiclass classification
- Other types of data
- Multiple outputs
- Cross entropy

Multiple outputs

- Treat each output y_d as *independent*:

$$Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]) = \prod_d Pr(y_d|\mathbf{f}_d[\mathbf{x}_i, \boldsymbol{\phi}])$$

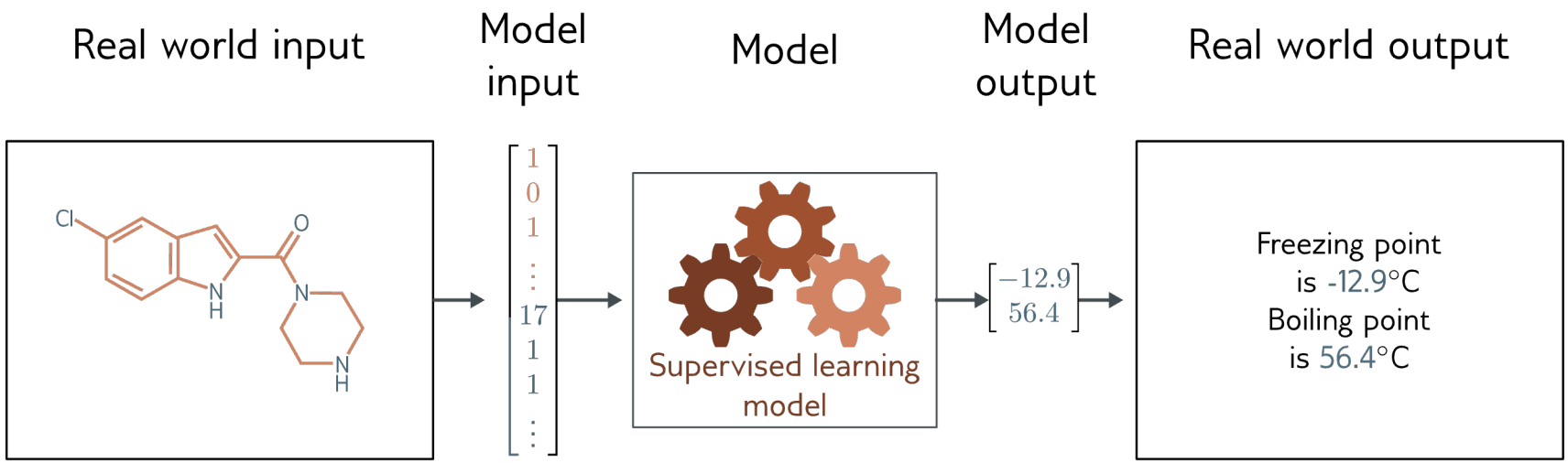
where $\mathbf{f}_d[\mathbf{x}, \boldsymbol{\phi}]$ is the d^{th} set of network outputs

- Negative log likelihood becomes sum of terms:

$$L[\boldsymbol{\phi}] = -\sum_{i=1}^{I} \log\Big[Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}])\Big] = -\sum_{i=1}^{I}\sum_d \log\Big[\underbrace{Pr(y_{id}|\mathbf{f}_d[\mathbf{x}_i, \boldsymbol{\phi}])}\Big]$$

d^{th} output of the i^{th} training example

Example 4: multivariate regression



Example 4: multivariate regression

- Goal: to predict a multivariate target $\mathbf{y} \in \mathbb{R}^{D_o}$
- Solution treat each dimension independently

$$\begin{aligned} Pr(\mathbf{y}|\boldsymbol{\mu}, \sigma^2) &= \prod_{d=1}^{D_o} Pr(y_d|\mu_d, \sigma^2) \\ &= \prod_{d=1}^{D_o} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y_d - \mu_d)^2}{2\sigma^2}\right] \end{aligned}$$

- Make network with D_o outputs to predict means

$$Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}], \sigma^2) = \prod_{d=1}^{D_o} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y_d - \mathrm{f}_d[\mathbf{x}, \boldsymbol{\phi}])^2}{2\sigma^2}\right]$$

Example 4: multivariate regression

- What if the outputs vary in magnitude
  - E.g., predict weight in kilos and height in meters
  - One dimension has much bigger numbers than others
- Could learn a separate variance for each…
- …or rescale before training, and then rescale output in opposite way

Loss functions

- Maximum likelihood
- Recipe for loss functions
- Example 1: univariate regression
- Example 2: binary classification
- Example 3: multiclass classification
- Other types of data
- Multiple outputs
- Cross entropy

Cross-entropy loss

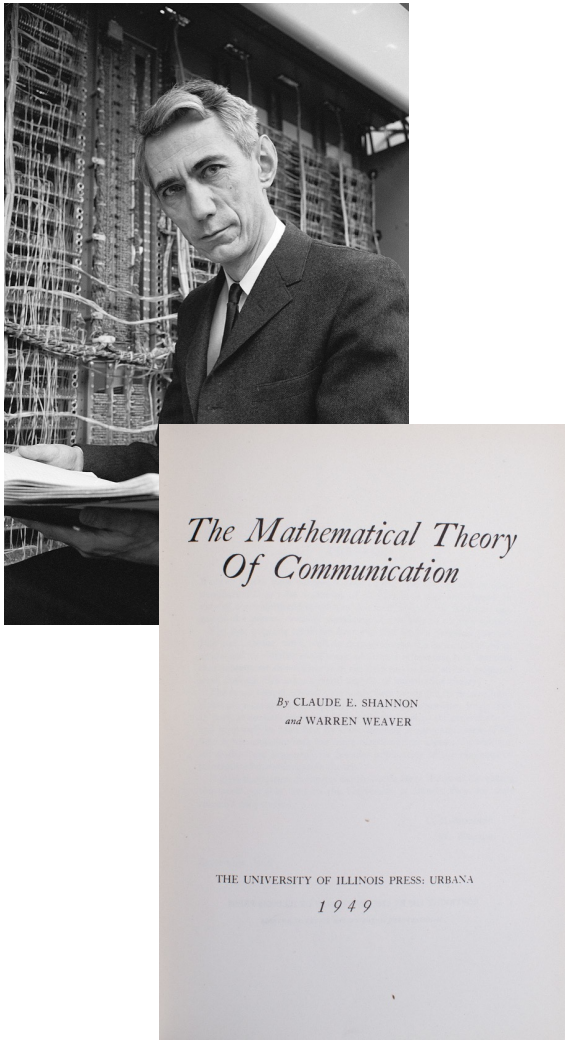
- So far we defined loss functions that minimize negative log-likelihood.
- Cross-entropy loss is common in neural network training.
- We can show that it is equivalent to negative log-likelihood

One can approach problems from different mathematical formulations.

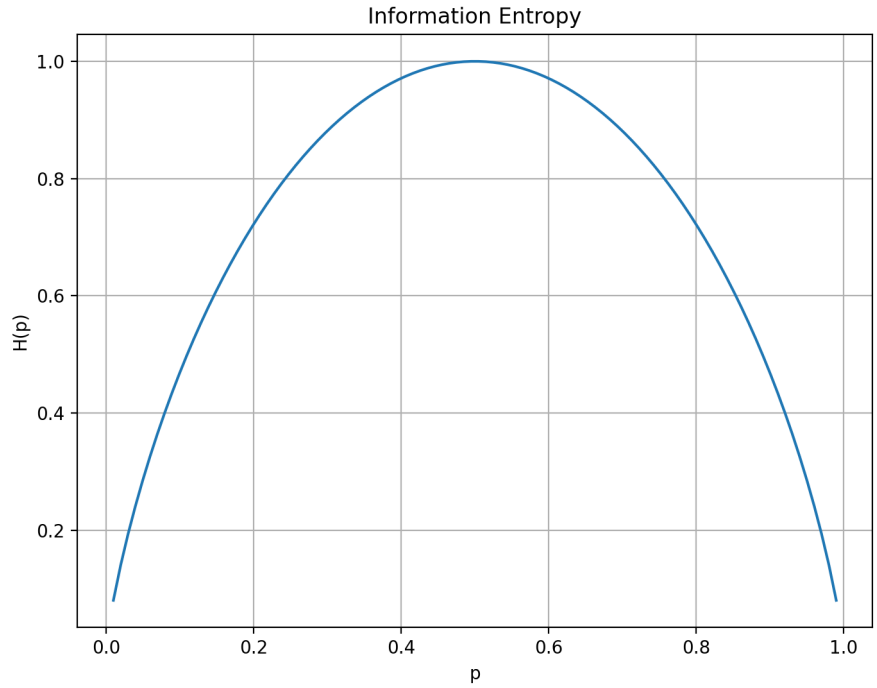
Information Theory and Entropy

- **Claude Shannon:** the "father of information theory," was an American mathematician, electrical engineer, and cryptographer
- **Theory of Communication:** In his landmark 1948 paper, "A Mathematical Theory of Communication," Shannon introduced a formal framework for the transmission, processing, and storage of information.
- **Information Theory:** Quantified information, allowing for the measurement of information content in messages, which is crucial for data compression, error detection and correction, and more.
- **Concept of Information Entropy:** introduced entropy as a measure of the uncertainty or randomness in a set of possible messages, providing a limit on the best possible lossless compression of any communication.

$$H(x) = -\sum_{x} P(x) \log_2(P(x))$$



Entropy for a Binary Event $x \in \{0,1\}$



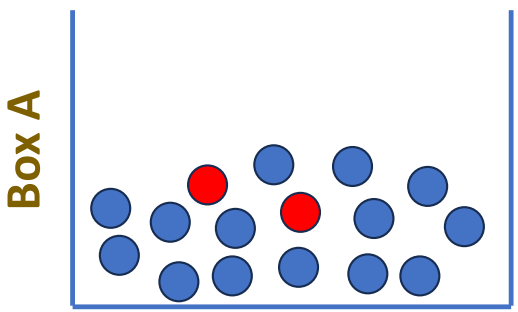
True with a probability of p

Peaks at 50/50.

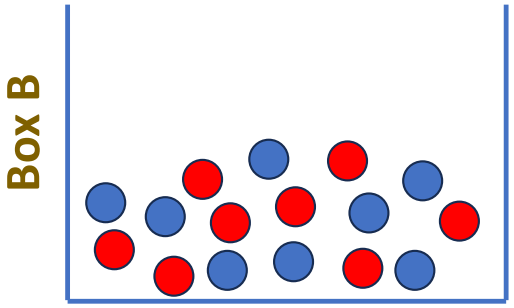
$$H(x) = -\sum_{x} P(x)\ \log_2\big(P(x)\big) = -p\log_2(p) - (1-p)\log_2(1-p)$$

Entropy is a measure of *surprise* or *uncertainty on average*

Randomly pick a ball from the box



Low or High Entropy?



Low or High Entropy?

In class poll: https://piazza.com/class/mkkiwlgx1413pj/post/23

Connection to Deep Learning

Cross-Entropy Loss

- If a neural network predicts **(0.25, 0.25, 0.25, 0.25)** for four possible classes, **high entropy** → uncertain.
- If it predicts **(0.99, 0.01, 0, 0)**, **low entropy** → confident.

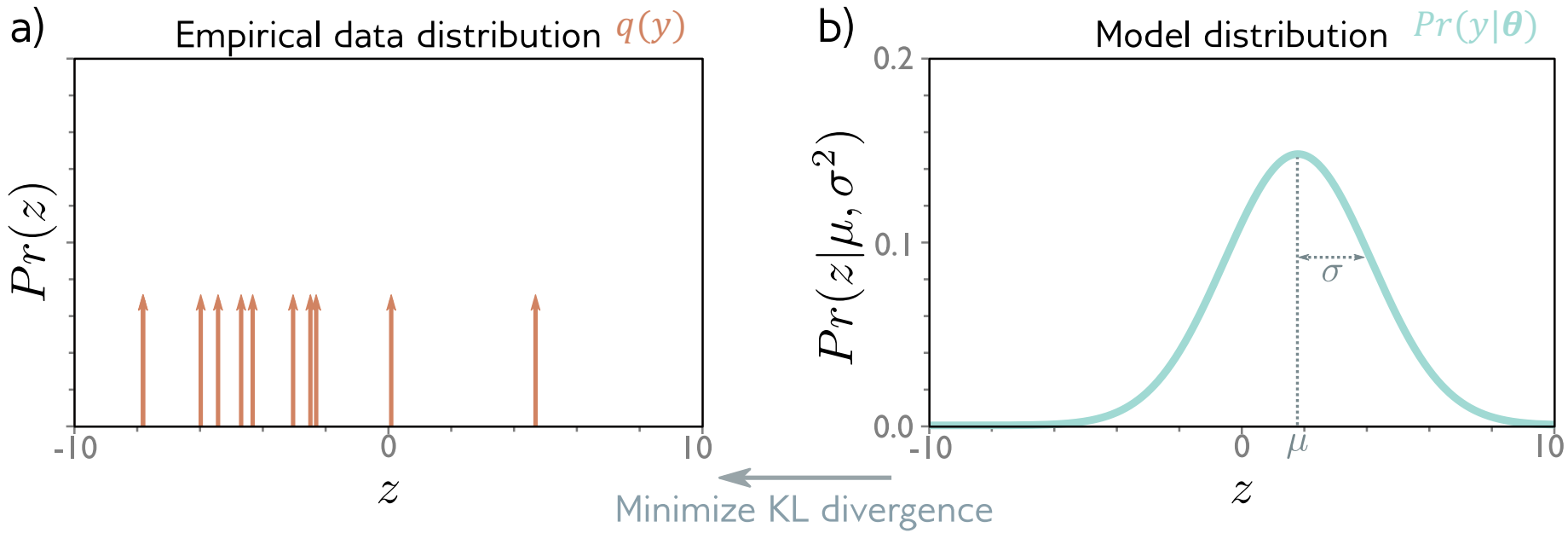
Regularization & Overfitting

- A **high-entropy** model makes diverse predictions → good for exploration.
- A **low-entropy** model could be overconfident → might overfit.

"Raise the temperature in LLMs."

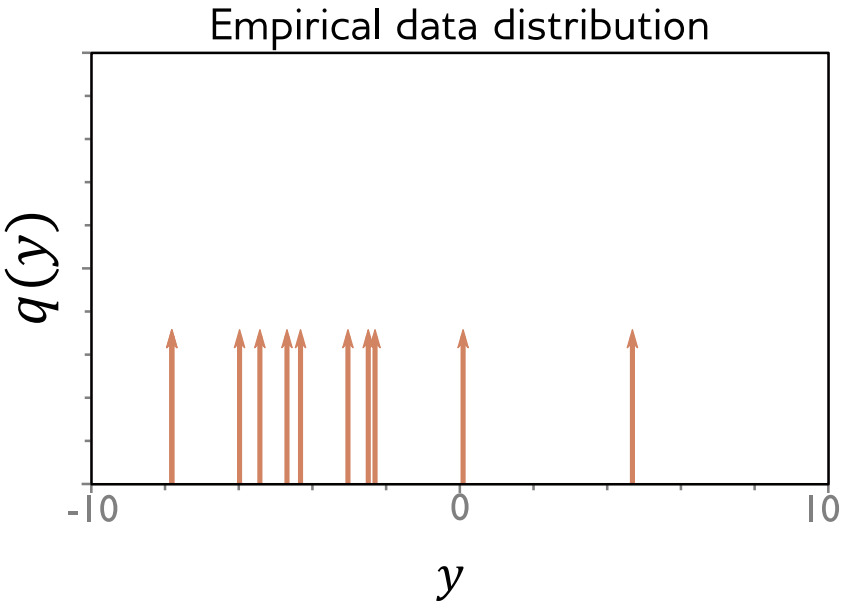
Cross Entropy – Concept from Information Theory

Measures the difference between the empirical distribution, $q(y)$, and a model distribution, $\Pr(y|\theta)$.



Kullback-Leibler Divergence -- a measure between probability distributions

Empirical Distribution – Collection of samples



Each sample represented by a shifted Dirac delta function.

$$\int \delta[x - x_0]dx = 1$$

$$\int f[x]\,\delta[x - x_0]\,dx = f[x_0]$$

So we say empirical distribution is

$$q(y) = \frac{1}{I}\sum_{i=1}^{I} \delta[\,y - y_i]$$

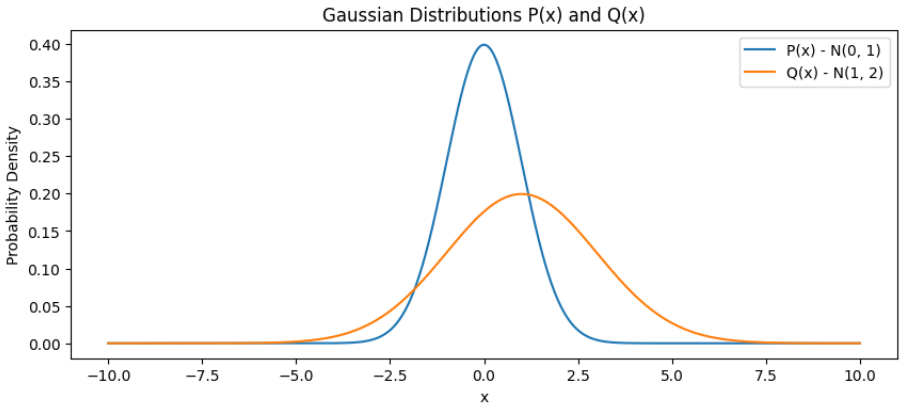
which will be helpful formulation in a moment.

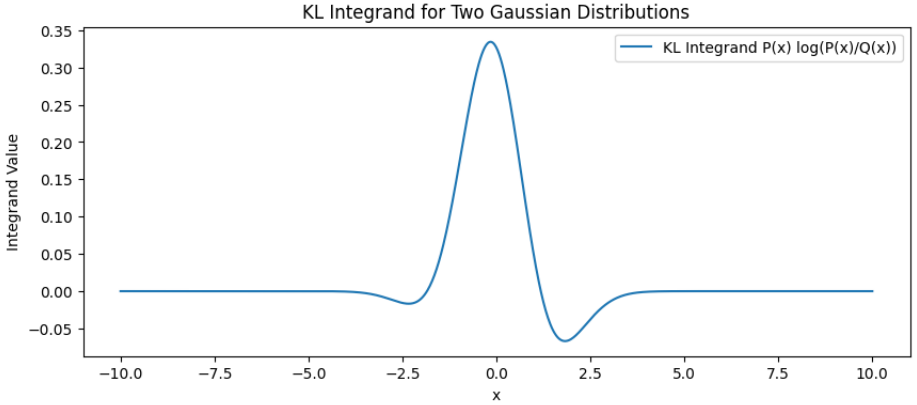
Kullback-Leibler (KL) divergence

How much a model distribution, Q, is different from a true probability distribution, P.

$$D_{KL}[q(z) \parallel p(z)] = \int q(z) \log \frac{q(z)}{p(z)} dz$$

$$= \int_{-\infty}^{\infty} q(z)\ log[q(z)]dz - q(z)log[p(z)]dz$$



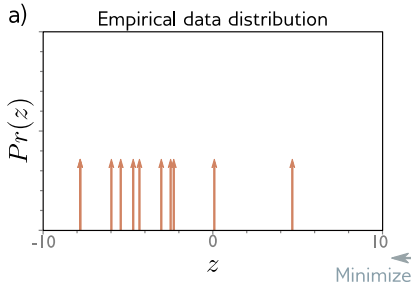


Interactive Colab Notebook

KL Divergence: 0.4431

Derivation

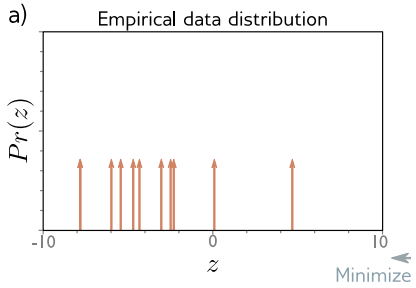
$$q(y) = \frac{1}{I}\sum_{i=1}^{I}\delta[y - y_i],$$



Training dataset as collection of Dirac delta functions.

Derivation

$$q(y) = \frac{1}{I}\sum_{i=1}^{I}\delta[y - y_i],$$



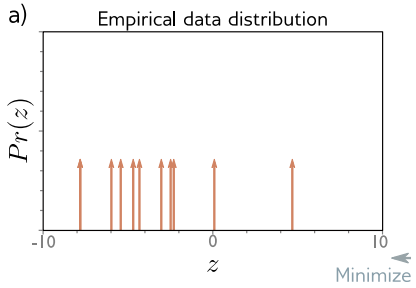
Training dataset as collection of Dirac delta functions.

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[\int_{-\infty}^{\infty} q(y)\log\left[q(y)\right]dy - \int_{-\infty}^{\infty} q(y)\log\left[Pr(y|\boldsymbol{\theta})\right]dy\right]$$

Minimize KL divergence.

Derivation

$$q(y) = \frac{1}{I}\sum_{i=1}^{I}\delta[y - y_i],$$



Training dataset as collection of Dirac delta functions.

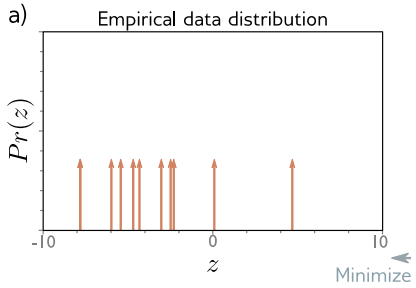
$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[\int_{-\infty}^{\infty} q(y)\log\left[q(y)\right]dy - \int_{-\infty}^{\infty} q(y)\log\left[Pr(y|\boldsymbol{\theta})\right]dy\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[-\int_{-\infty}^{\infty} q(y)\log\left[Pr(y|\boldsymbol{\theta})\right]dy\right],\end{aligned}$$

Minimize KL divergence.

1st term not dependent on θ.

Derivation

$$q(y) = \frac{1}{I}\sum_{i=1}^{I}\delta[y - y_i],$$



Training dataset as collection of Dirac delta functions.

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[\int_{-\infty}^{\infty} q(y)\log\left[q(y)\right]dy - \int_{-\infty}^{\infty} q(y)\log\left[Pr(y|\boldsymbol{\theta})\right]dy\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[-\int_{-\infty}^{\infty} q(y)\log\left[Pr(y|\boldsymbol{\theta})\right]dy\right],\end{aligned}$$

Minimize KL divergence.

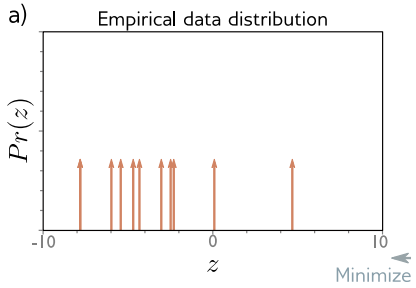
1st term not dependent on θ.

$$\hat{\boldsymbol{\theta}} = \underset{\theta}{\operatorname{argmin}}\left[-\int_{-\infty}^{\infty}\left(\frac{1}{I}\sum_{i=1}^{I}\delta[y - y_i]\right)\log\left[Pr(y|\boldsymbol{\theta})\right]dy\right]$$

Substituting for $q(y)$.

Derivation

$$q(y) = \frac{1}{I}\sum_{i=1}^{I}\delta[y - y_i],$$



Training dataset as collection of Dirac delta functions.

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[\int_{-\infty}^{\infty} q(y)\log\left[q(y)\right]dy - \int_{-\infty}^{\infty} q(y)\log\left[Pr(y|\boldsymbol{\theta})\right]dy\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[-\int_{-\infty}^{\infty} q(y)\log\left[Pr(y|\boldsymbol{\theta})\right]dy\right],\end{aligned}$$

Minimize KL divergence.

1st term not dependent on θ.

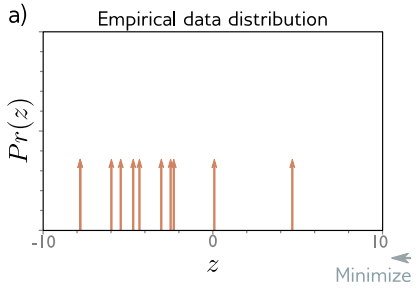
$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\theta}{\operatorname{argmin}}\left[-\int_{-\infty}^{\infty}\left(\frac{1}{I}\sum_{i=1}^{I}\delta[y - y_i]\right)\log\left[Pr(y|\boldsymbol{\theta})\right]dy\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[-\frac{1}{I}\sum_{i=1}^{I}\log\left[Pr(y_i|\boldsymbol{\theta})\right]\right]\end{aligned}$$

Substituting for $q(y)$.

Property of the Dirac delta function.

Derivation

$$q(y) = \frac{1}{I}\sum_{i=1}^{I}\delta[y - y_i],$$



Training dataset as collection of Dirac delta functions.

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[\int_{-\infty}^{\infty} q(y)\log\left[q(y)\right]dy - \int_{-\infty}^{\infty} q(y)\log\left[Pr(y|\boldsymbol{\theta})\right]dy\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[-\int_{-\infty}^{\infty} q(y)\log\left[Pr(y|\boldsymbol{\theta})\right]dy\right],\end{aligned}$$

Minimize KL divergence.

1st term not dependent on θ.

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\theta}{\operatorname{argmin}}\left[-\int_{-\infty}^{\infty}\left(\frac{1}{I}\sum_{i=1}^{I}\delta[y - y_i]\right)\log\left[Pr(y|\boldsymbol{\theta})\right]dy\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[-\frac{1}{I}\sum_{i=1}^{I}\log\left[Pr(y_i|\boldsymbol{\theta})\right]\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[-\sum_{i=1}^{I}\log\left[Pr(y_i|\boldsymbol{\theta})\right]\right].\end{aligned}$$

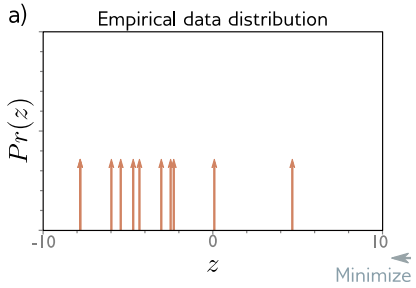
Substituting for $q(y)$.

Property of the Dirac delta function.

$^1/_I$ is just a constant, so ignore.

Derivation

$$q(y) = \frac{1}{I}\sum_{i=1}^{I}\delta[y - y_i],$$



Training dataset as collection of Dirac delta functions.

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[\int_{-\infty}^{\infty} q(y)\log\left[q(y)\right]dy - \int_{-\infty}^{\infty} q(y)\log\left[Pr(y|\boldsymbol{\theta})\right]dy\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[-\int_{-\infty}^{\infty} q(y)\log\left[Pr(y|\boldsymbol{\theta})\right]dy\right],\end{aligned}$$

Minimize KL divergence.

1st term not dependent on θ.

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\theta}{\operatorname{argmin}}\left[-\int_{-\infty}^{\infty}\left(\frac{1}{I}\sum_{i=1}^{I}\delta[y - y_i]\right)\log\left[Pr(y|\boldsymbol{\theta})\right]dy\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[-\frac{1}{I}\sum_{i=1}^{I}\log\left[Pr(y_i|\boldsymbol{\theta})\right]\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[-\sum_{i=1}^{I}\log\left[Pr(y_i|\boldsymbol{\theta})\right]\right].\end{aligned}$$

Substituting for $q(y)$.

Property of the Dirac delta function.

$1/I$ is just a constant, so ignore.

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\left[-\sum_{i=1}^{I}\log\left[Pr(y_i|\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}])\right]\right]$$

Model is predicting θ ➔ **Negative Log Likelihood!!**

Recap

- Reconsidered loss functions as fitting a parametric probability model
- Introduced Maximum Likelihood criterion for finding parameters to making the training data most probably under that model
- Introduced a 4-step recipe for (1) picking a suitable parametric probability distribution, (2) defining the model to pick one or more of the parameters, (3) training the model and (4) doing inference
- Derived loss functions for univariate regression, binary and multiclass classification
- Briefly reviewed parametric probability models for other types of data
- Discussed how this is the same as Cross Entropy from Information Theory

Minimizing Negative Log Likelihood (or equivalently KL Divergence)

$$\hat{\phi} = \underset{\phi}{\text{argmin}}\left[-\sum_{i=1}^{I} \log[\Pr(y_i|\, f[x_i, \phi])]\right]$$

$$= \underset{\phi}{\text{argmin}}\left[L[\phi]\right]$$

Recipe for loss functions

1. Choose a suitable probability distribution $\Pr(\mathbf{y}|\boldsymbol{\theta})$ that is defined over the domain of the predictions $\mathbf{y}$ and has distribution parameters $\boldsymbol{\theta}$.
2. Set the machine learning model $\mathbf{f}[\mathbf{x}, \phi]$ to predict one or more of these parameters so $\boldsymbol{\theta} = \mathbf{f}[\mathbf{x}, \phi]$ and $\Pr(\mathbf{y} \,|\, \mathbf{f}[\mathbf{x}, \phi])$.
3. To train the model, find the network parameters $\hat{\phi}$ that minimize the negative log-likelihood loss function over the training dataset pairs $\{\mathbf{x}_i, \mathbf{y}_i\}$:

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}}\big[L[\phi]\big] = \underset{\phi}{\operatorname{argmin}}\left[-\sum_{i=1}^{I} \log[\Pr(\mathrm{y}_i|\, \mathrm{f}[\mathrm{x}_i, \phi])]\right]$$

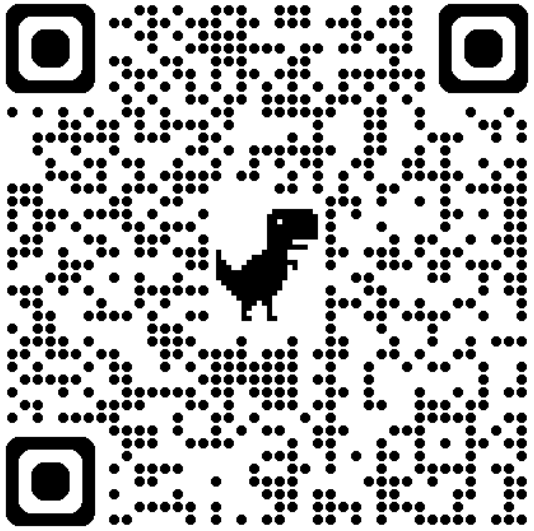
4. To perform inference for a new test example x, return either the full distribution $\Pr(\mathbf{y} \,|\, \mathbf{f}[\mathbf{x}, \phi])$ or the maximum of this distribution.

Next up

- Now let's find the parameters that give the smallest loss
  - ➔ Training the model

Feedback?

Lecture Feedback



https://forms.gle/pXHM5nx1Ti9aFmpw6