Loss Functions

January 16-17th, 2025

Deep Learning (CS60010)

**Slides adapted from http://udlbook.com*

Loss function

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

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

→ I examples

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

$$L\left[\boldsymbol{\phi}, \mathrm{f}[\mathbf{x}, \boldsymbol{\phi}], \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}\right]$$

model train data

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:

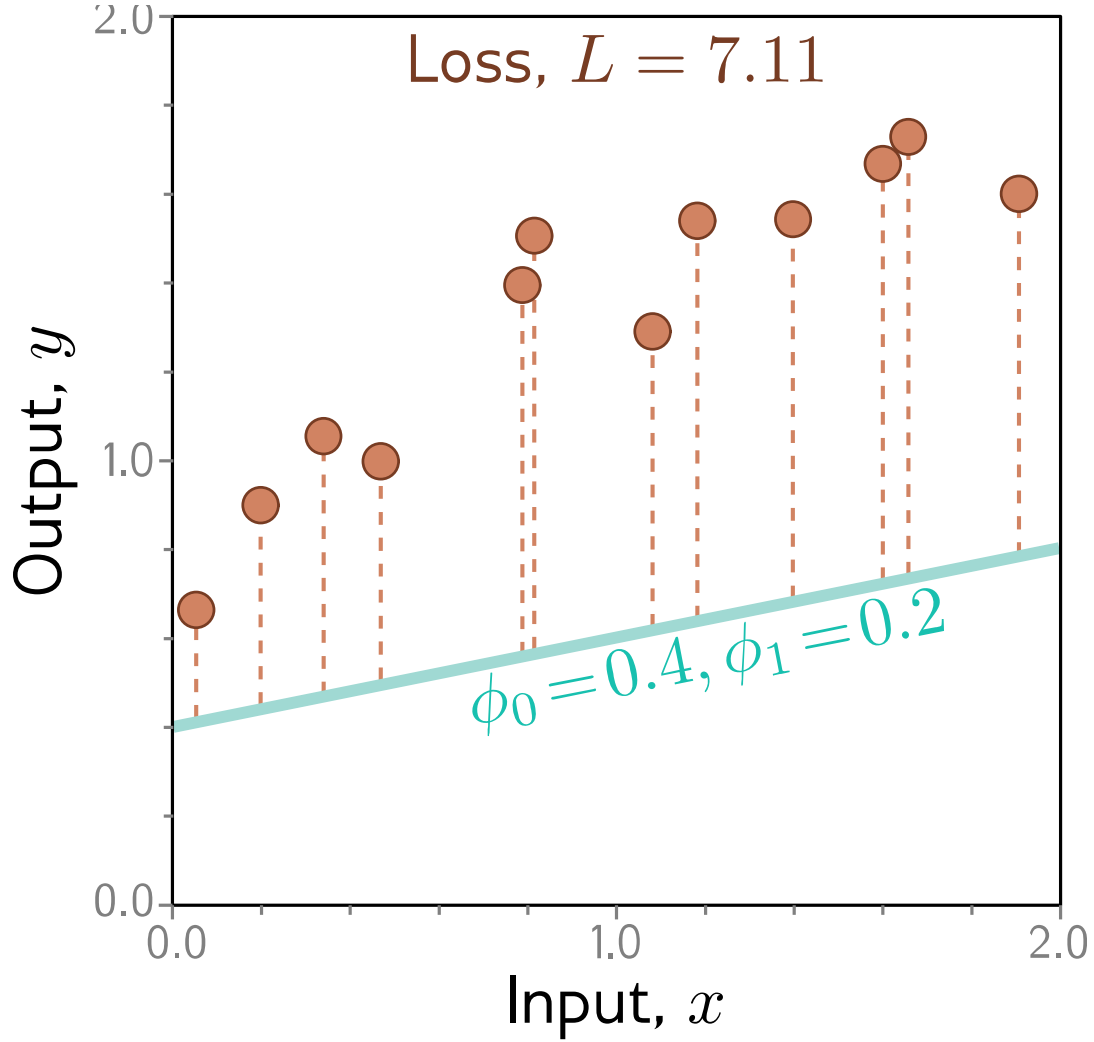
$$L[\phi]$$

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

- Find the parameters that minimize the loss:

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

Example: 1D Linear regression loss function



Loss function:

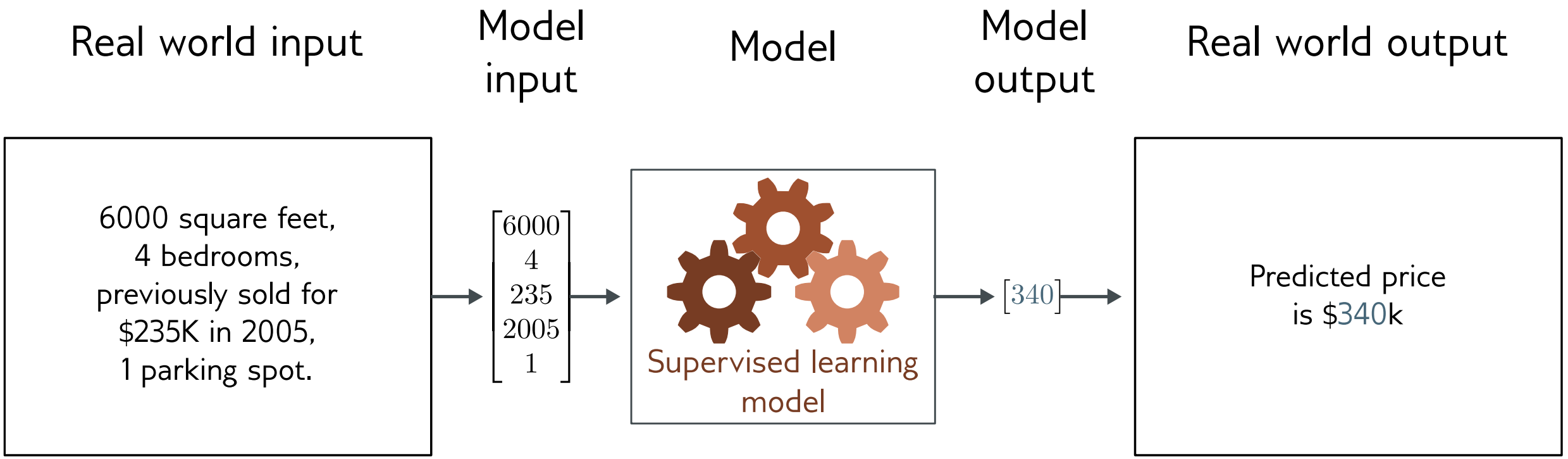
$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

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

"Least squares loss function"

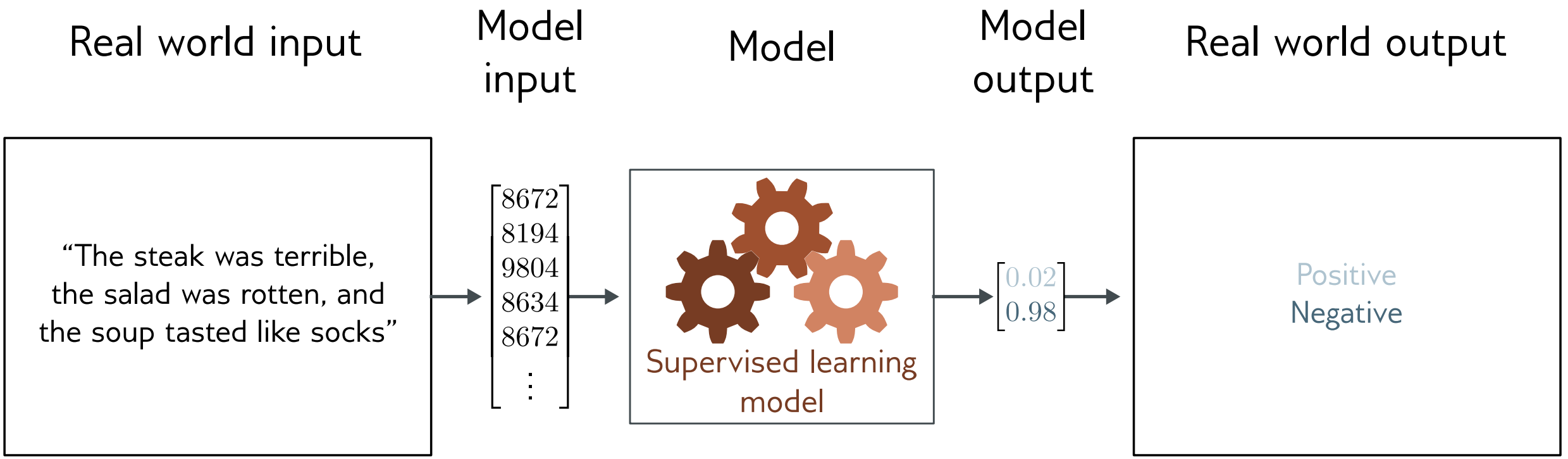
But why? How to construct loss functions?

Regression



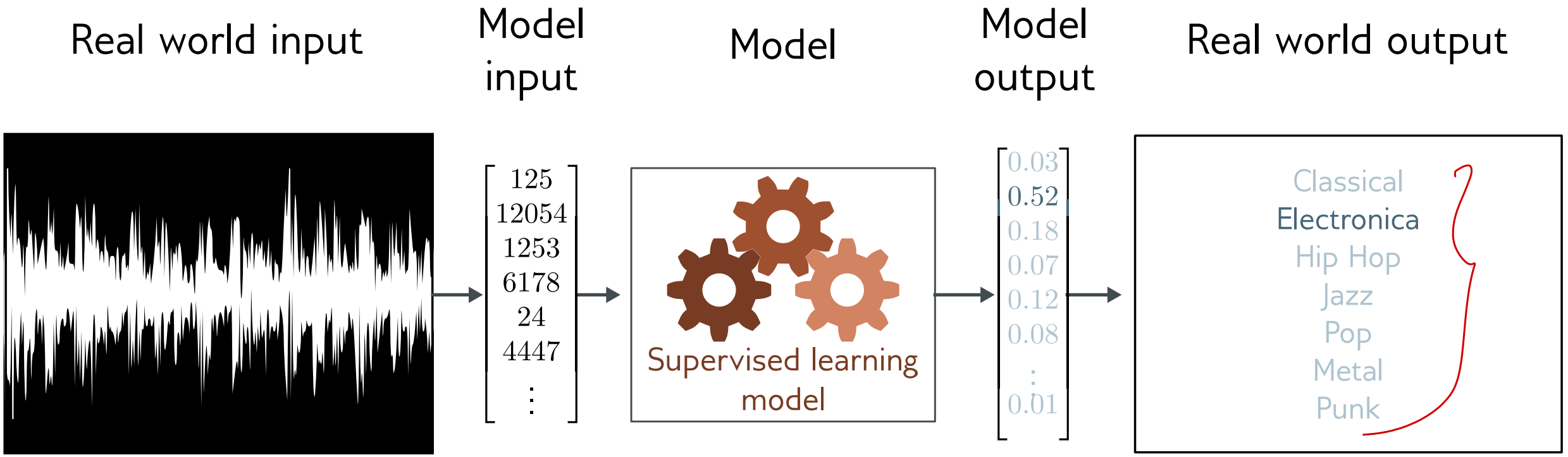
- Univariate regression problem (one output, real value)

Text classification

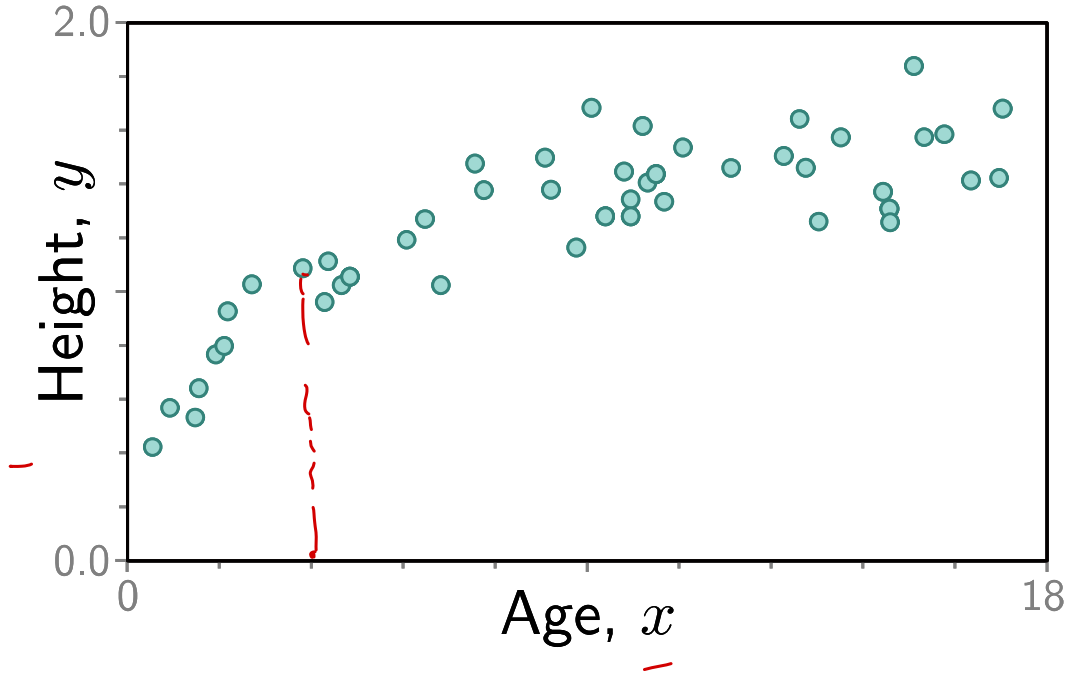


- Binary classification problem (two discrete classes)

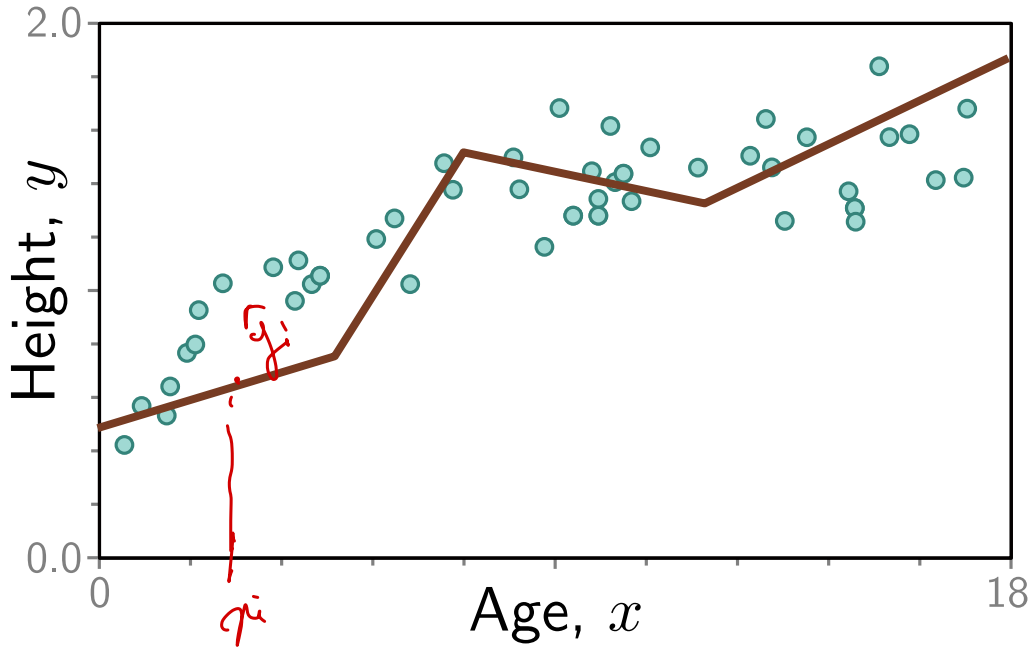
Music genre classification



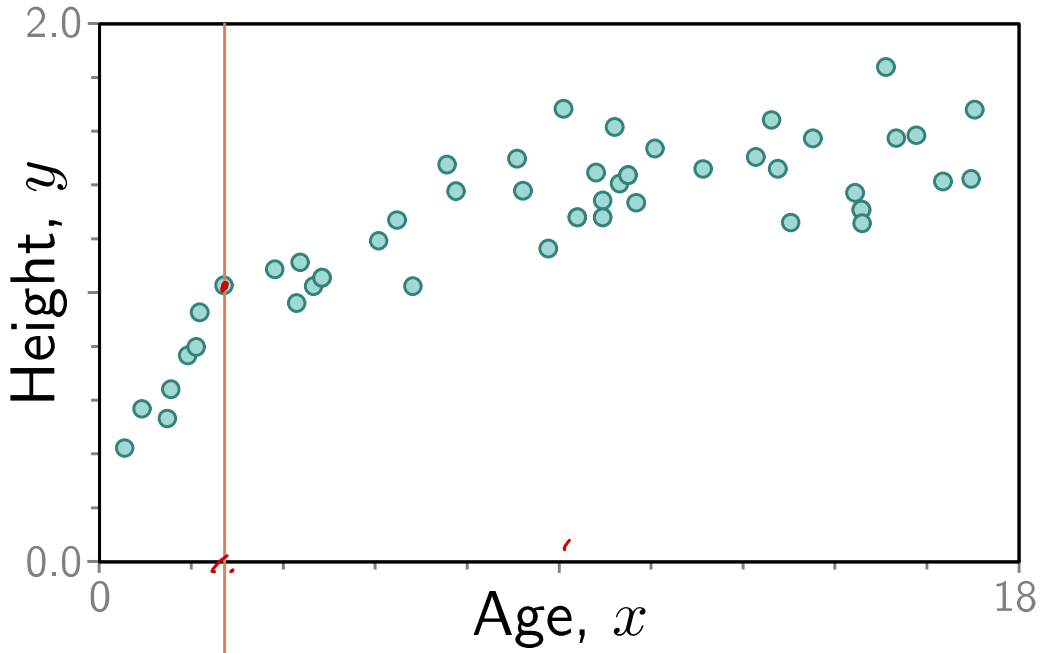
- Multiclass classification problem (discrete classes, >2 possible values)

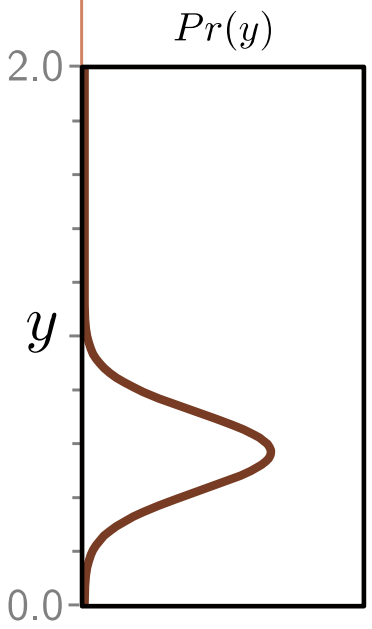


$f(x_i; \phi) \approx y_i$



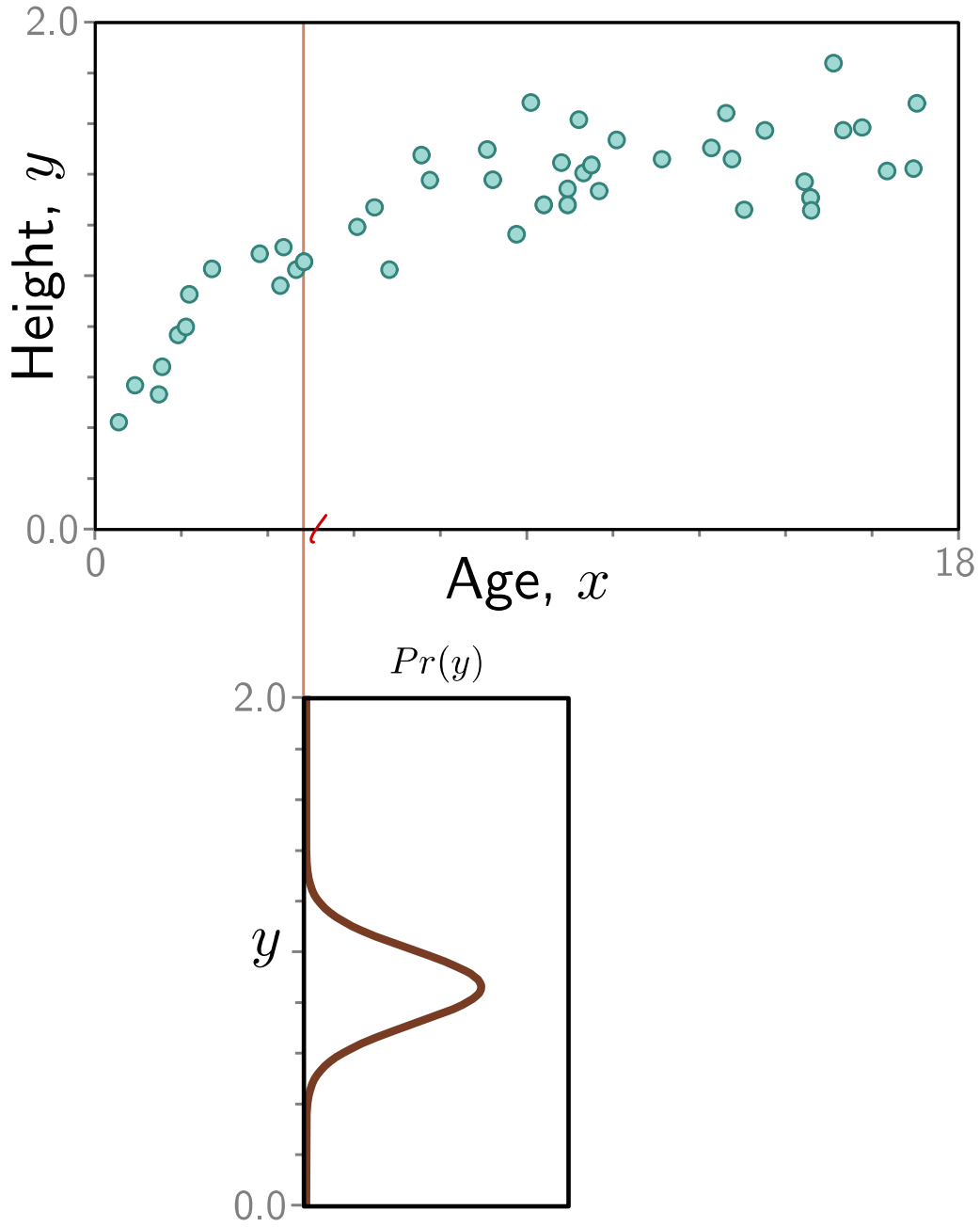
$P(y_i|x_i)$

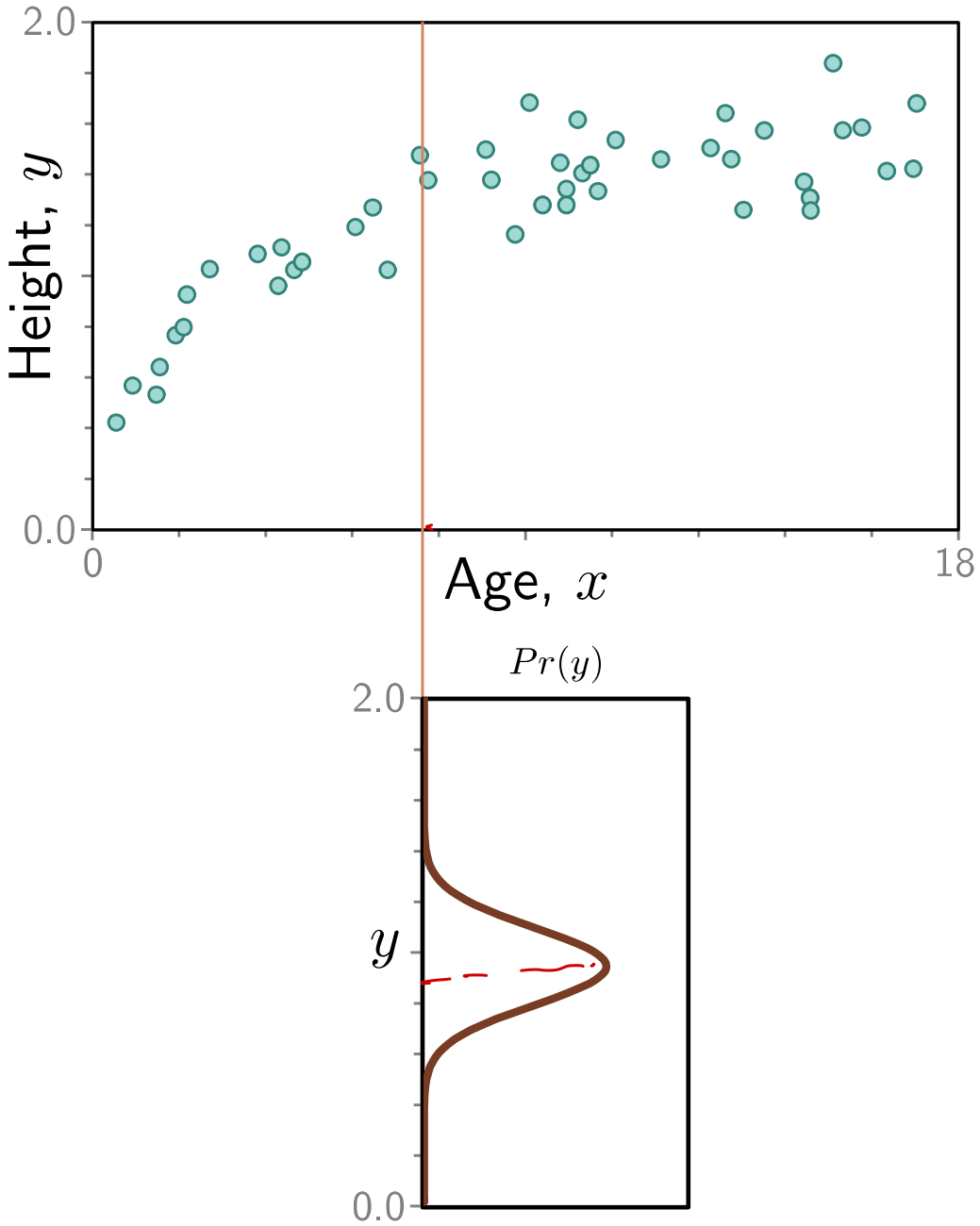


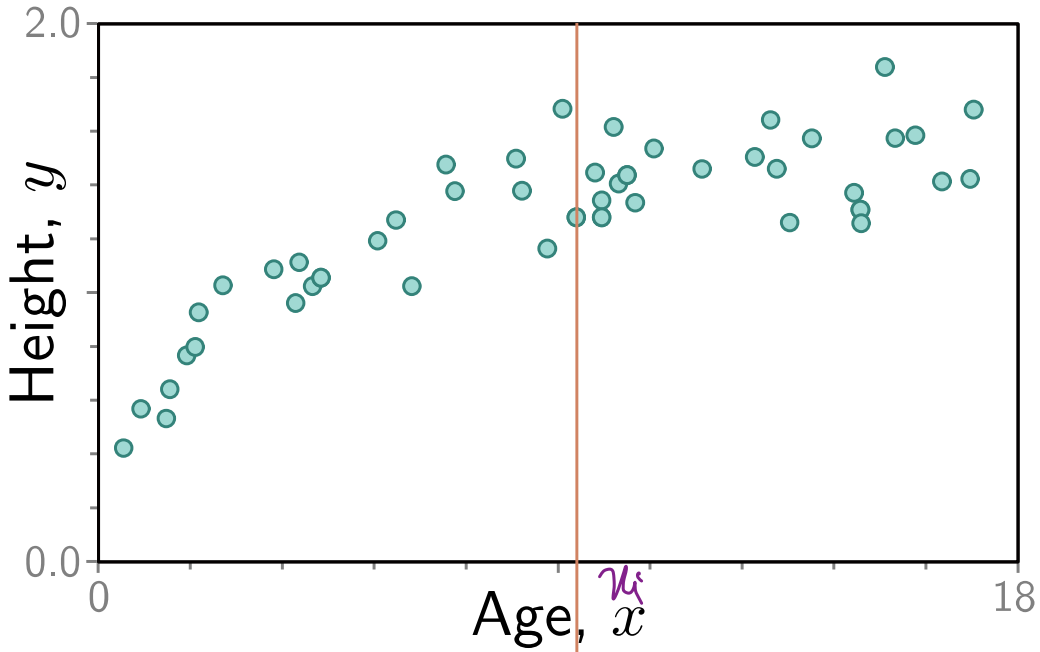


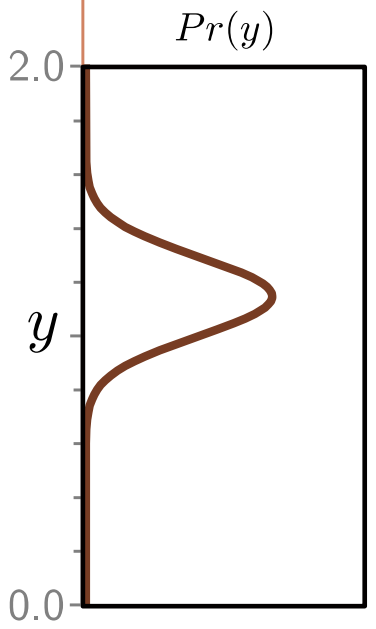
Shift perspective, consider model as computing a conditional probability distribution over possible outputs given an input

$Pr(y|x_i)$


2.0
0.0
Height, y
0
18
Age, x
$Pr(y)$
2.0
y
0.0


2.0
0.0
Height, y
0
18
Age, x
$Pr(y)$
2.0
y
0.0



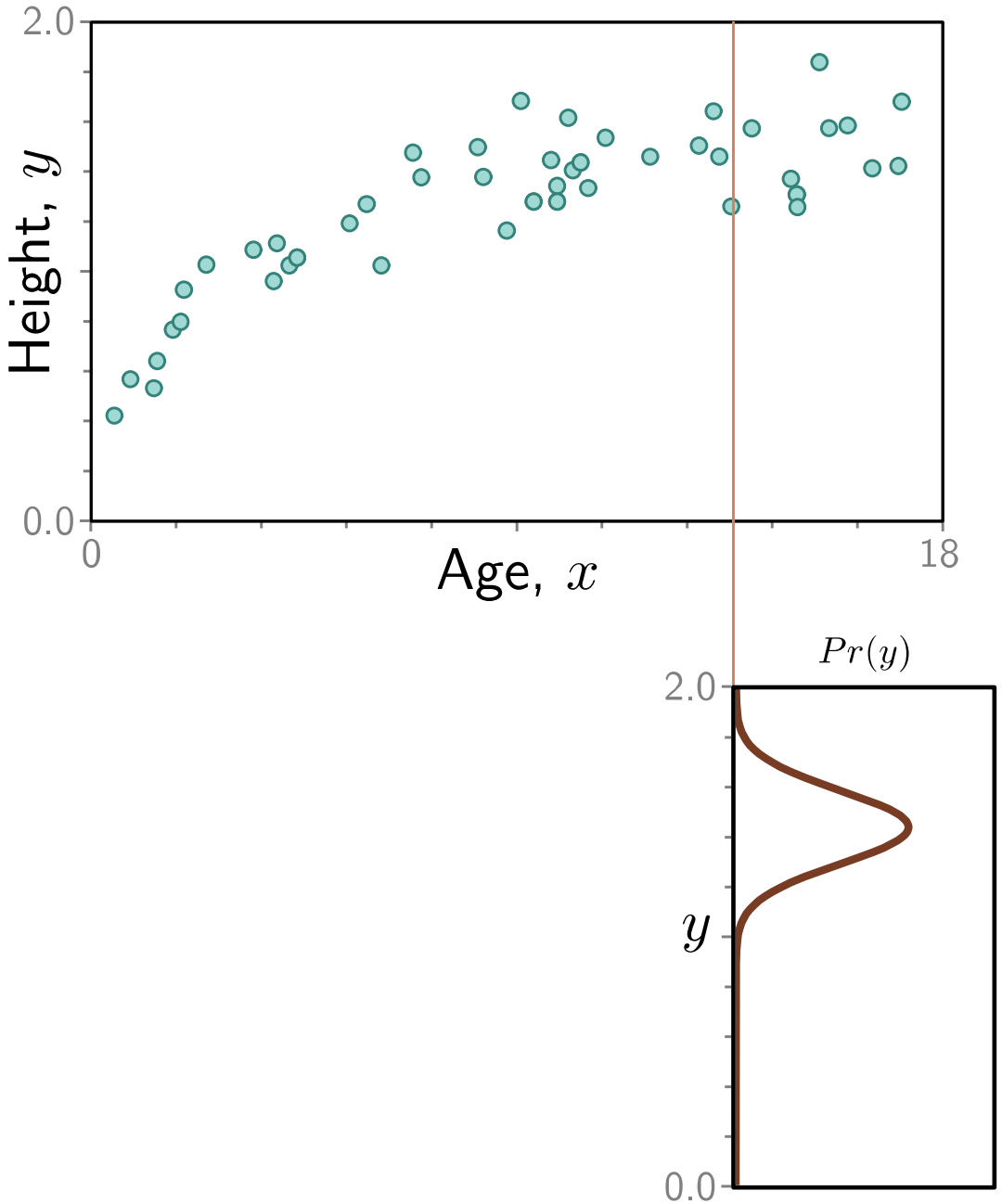


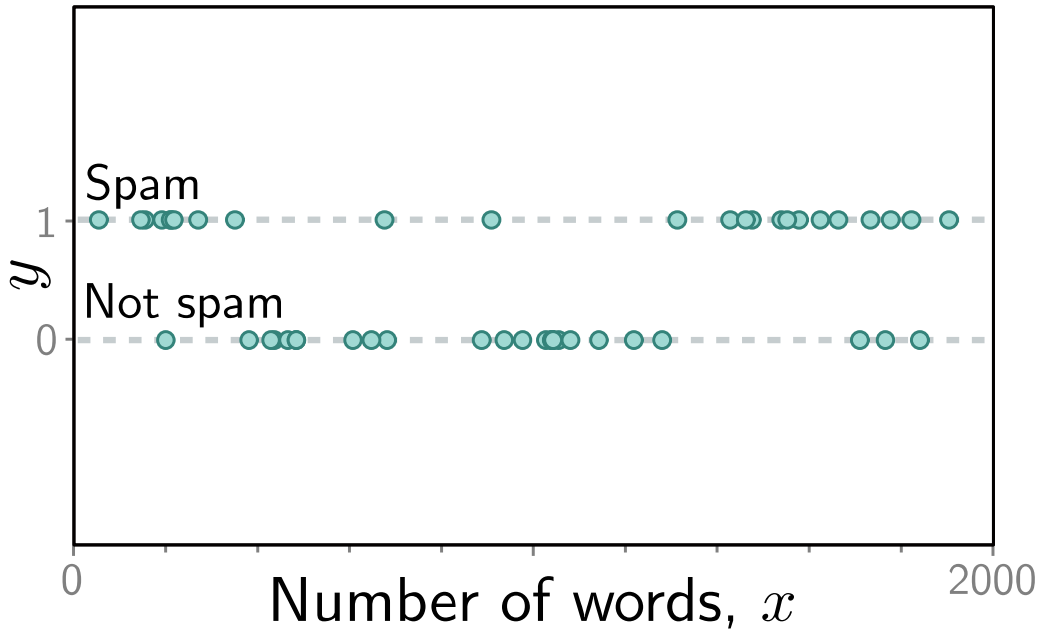
θ_i

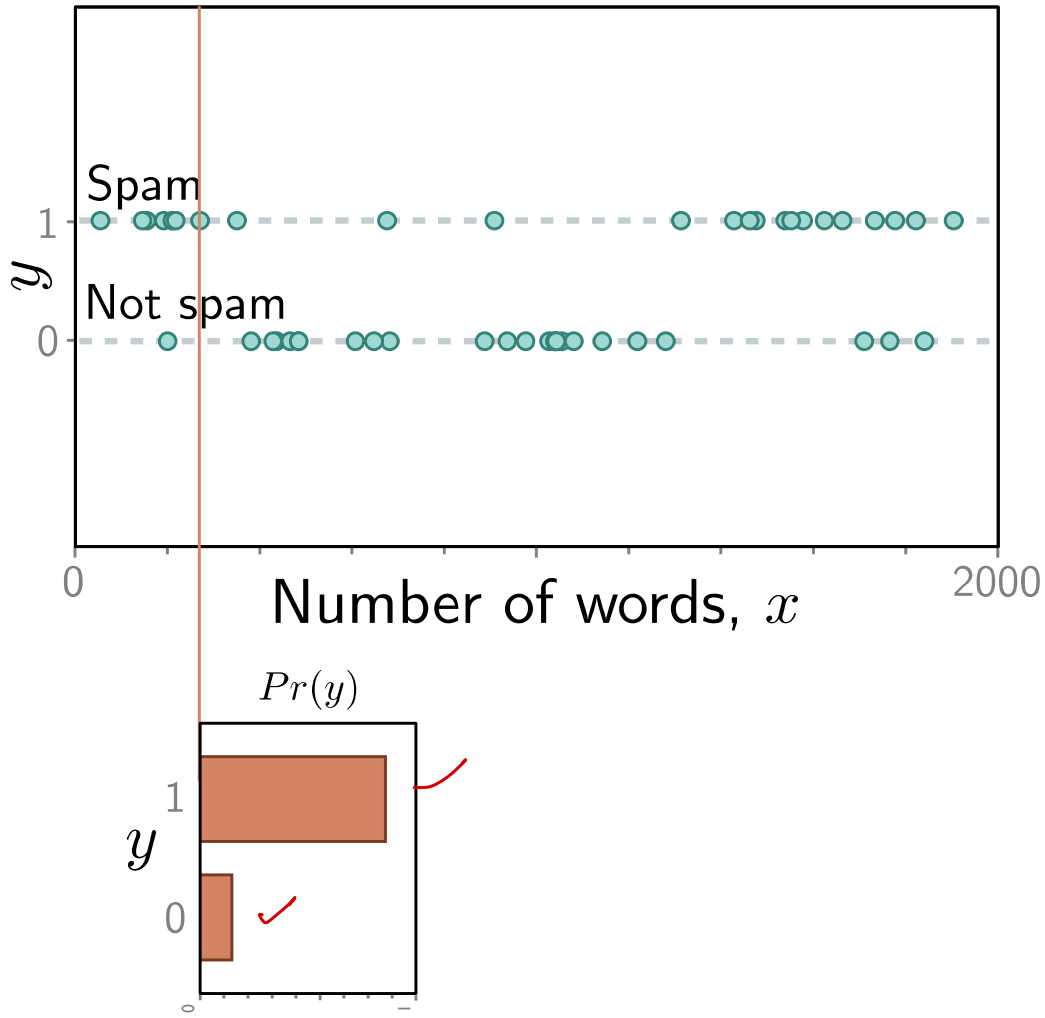
$Pr(y_i|x_i)$

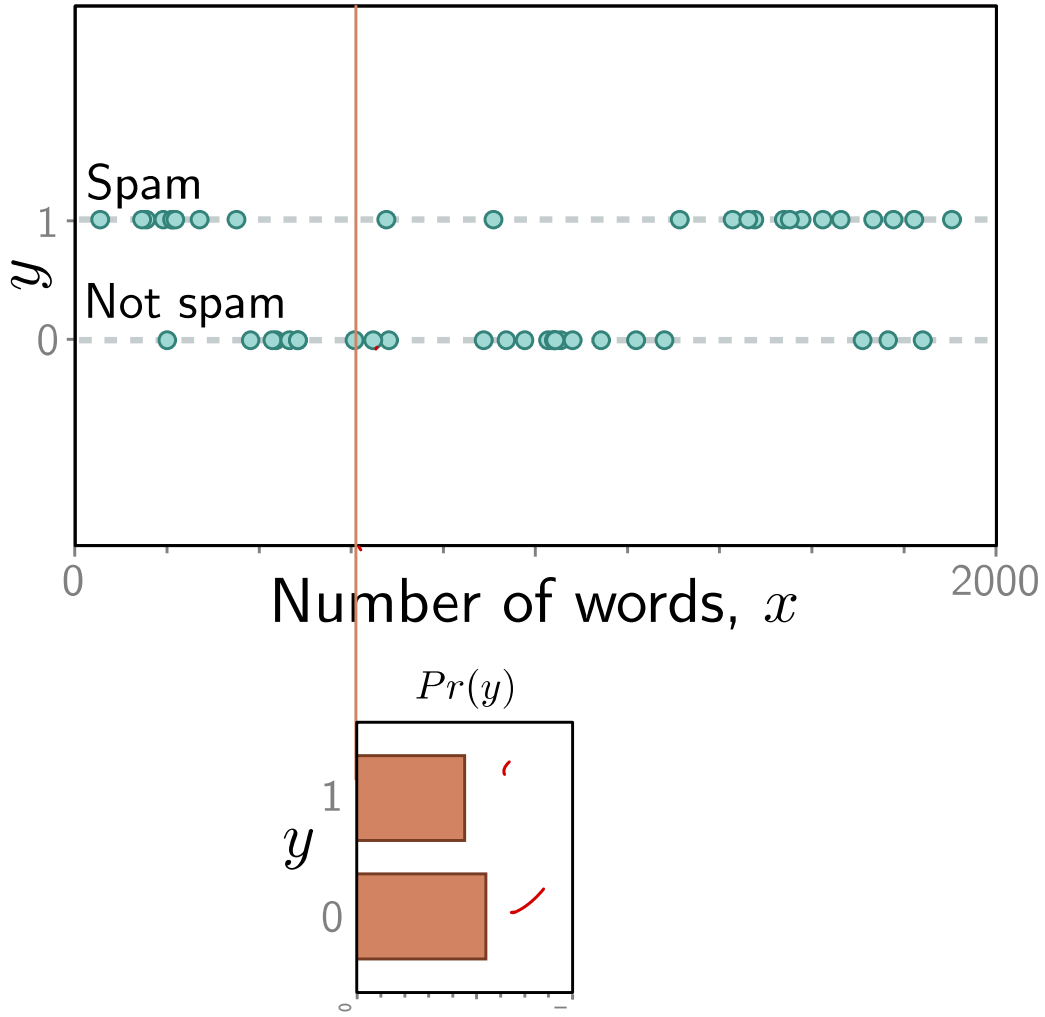
$Pr(y_i|\theta_i)$

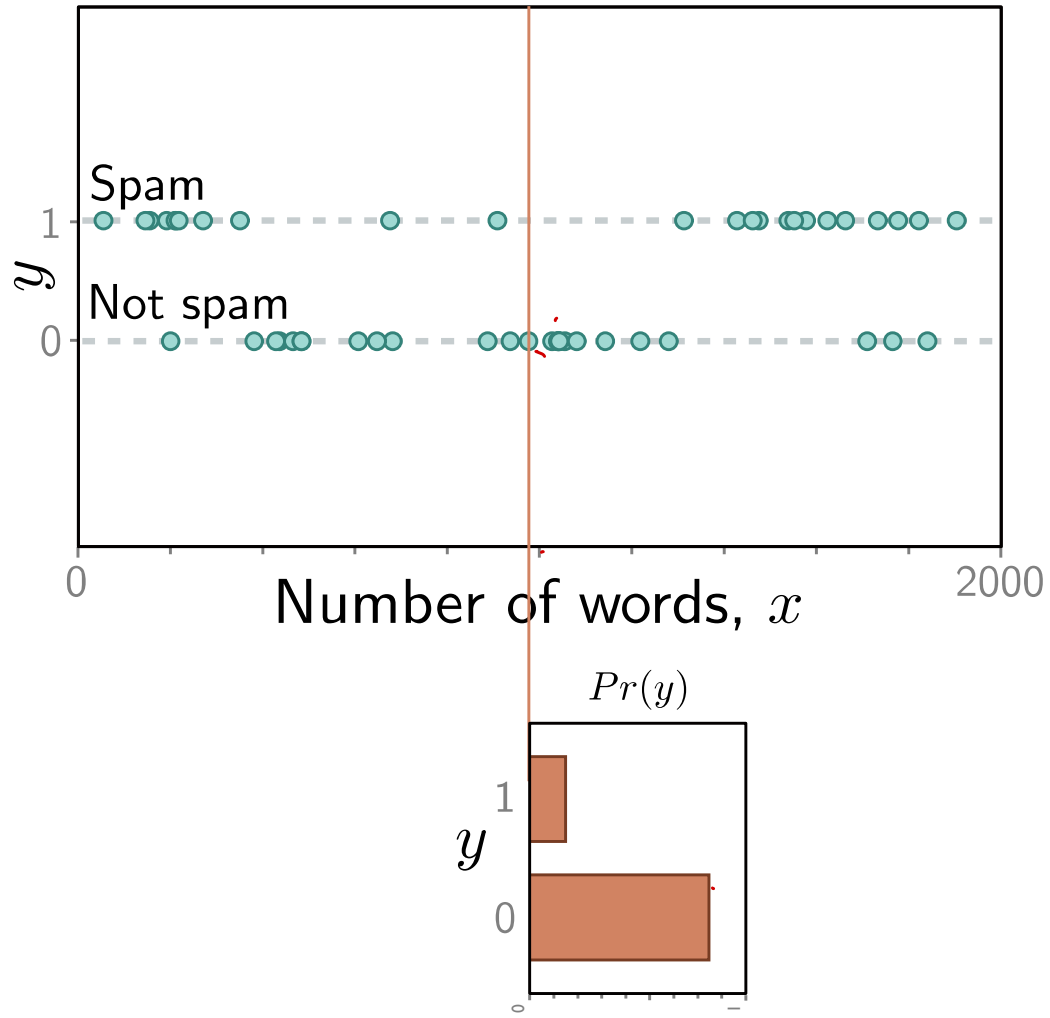
$Pr(y_i|f(x_i,\phi))$

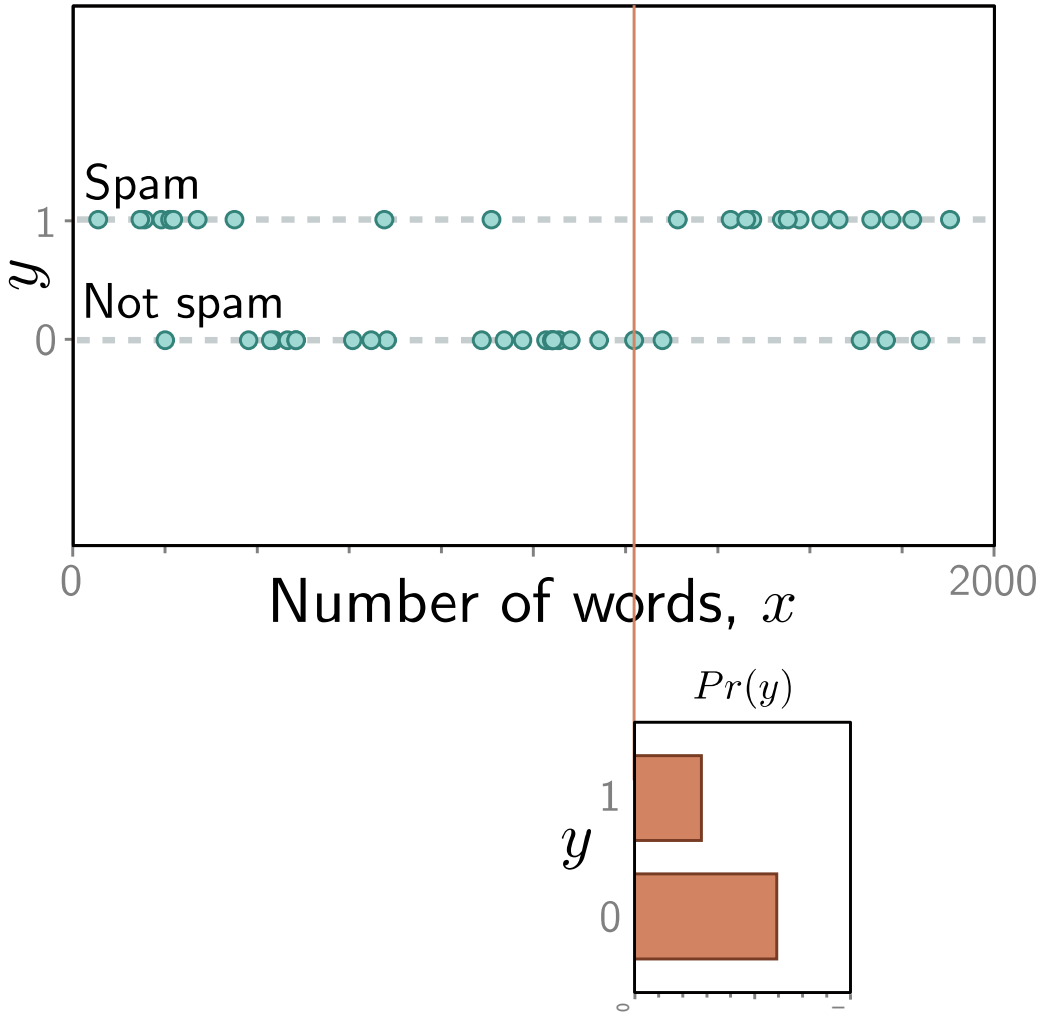

2.0
0.0
Height, y
0
18
Age, x
$Pr(y)$
2.0
y
0.0

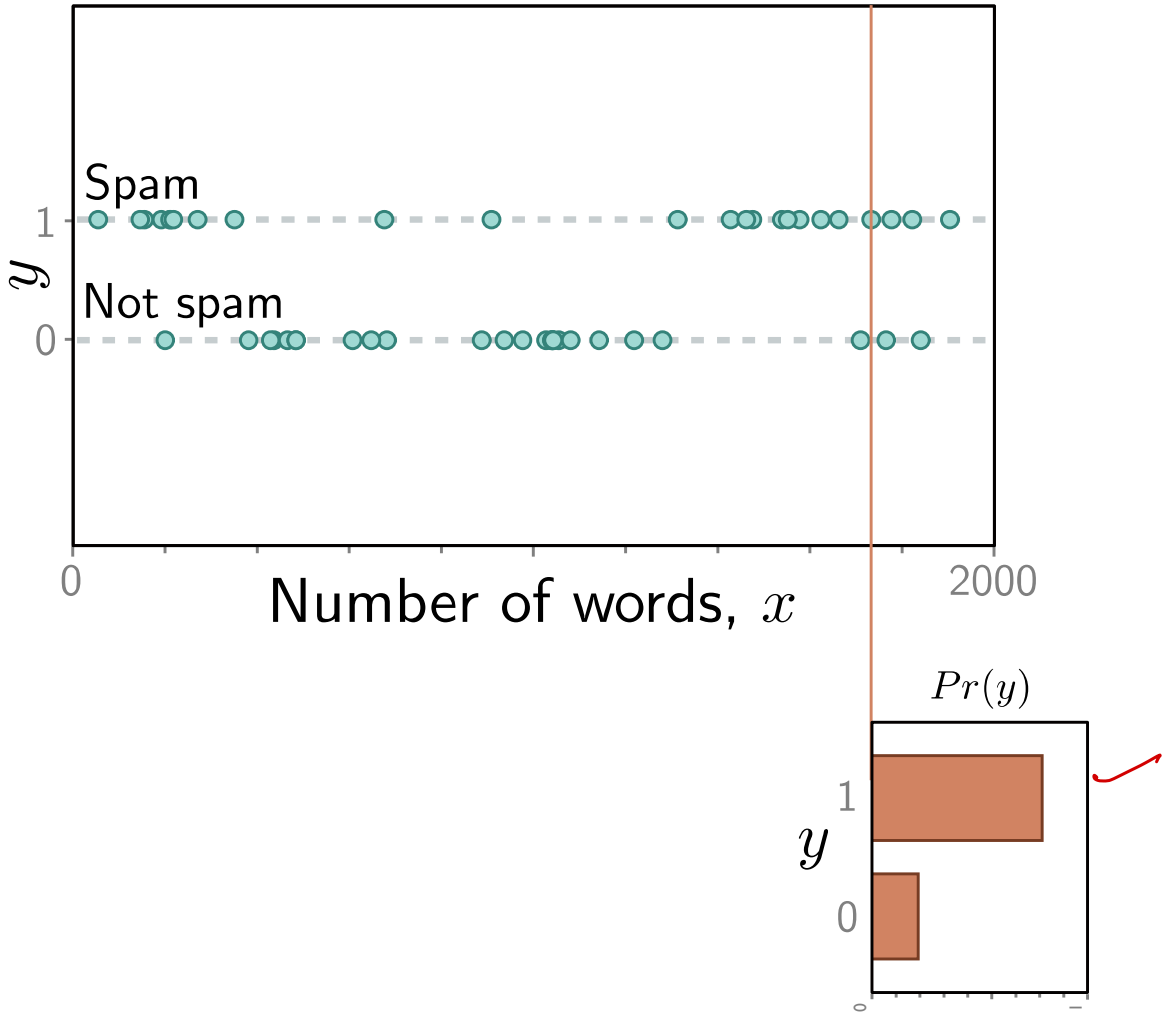

Spam
Not spam
1
0
y
0
2000
Number of words, x

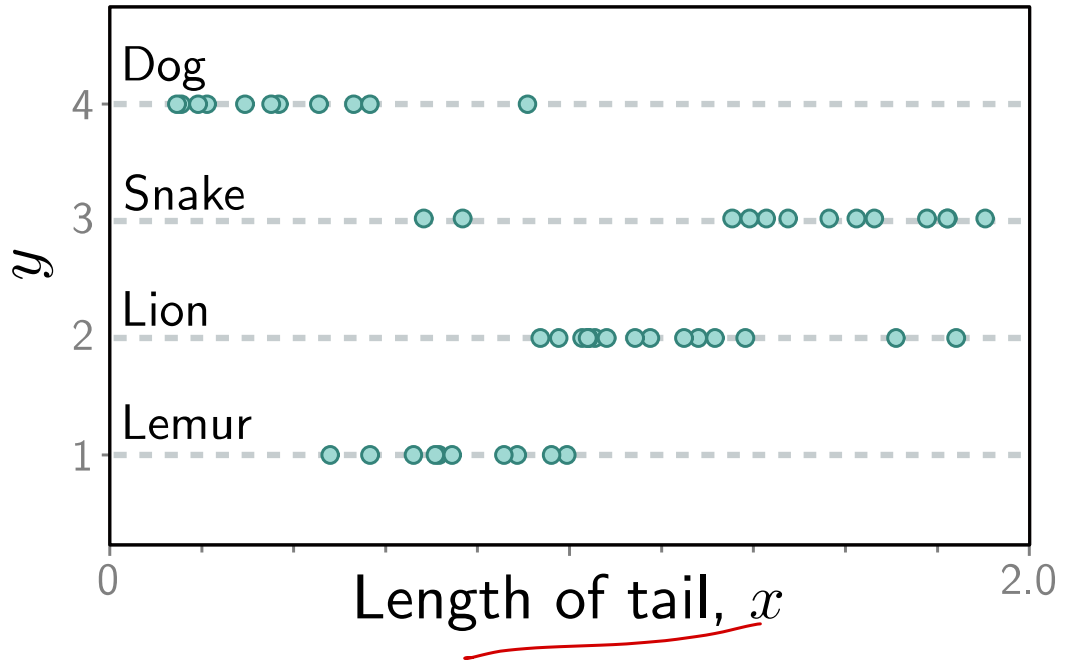

Spam
Not spam
1
0
y
0
2000
Number of words, x
$Pr(y)$
1
0
y

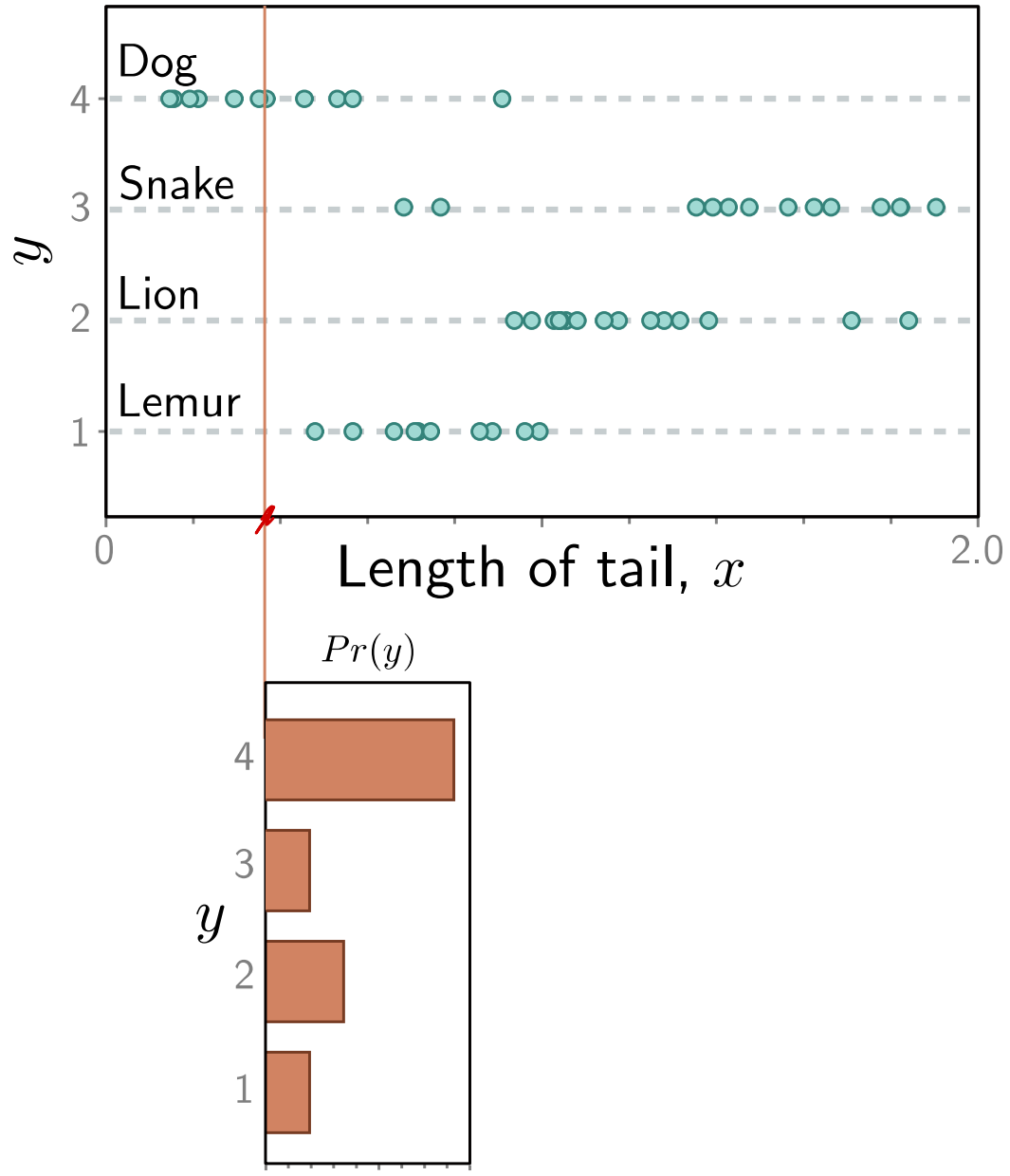

Spam
Not spam
1
0
y
0
2000
Number of words, x
$Pr(y)$
1
0
y

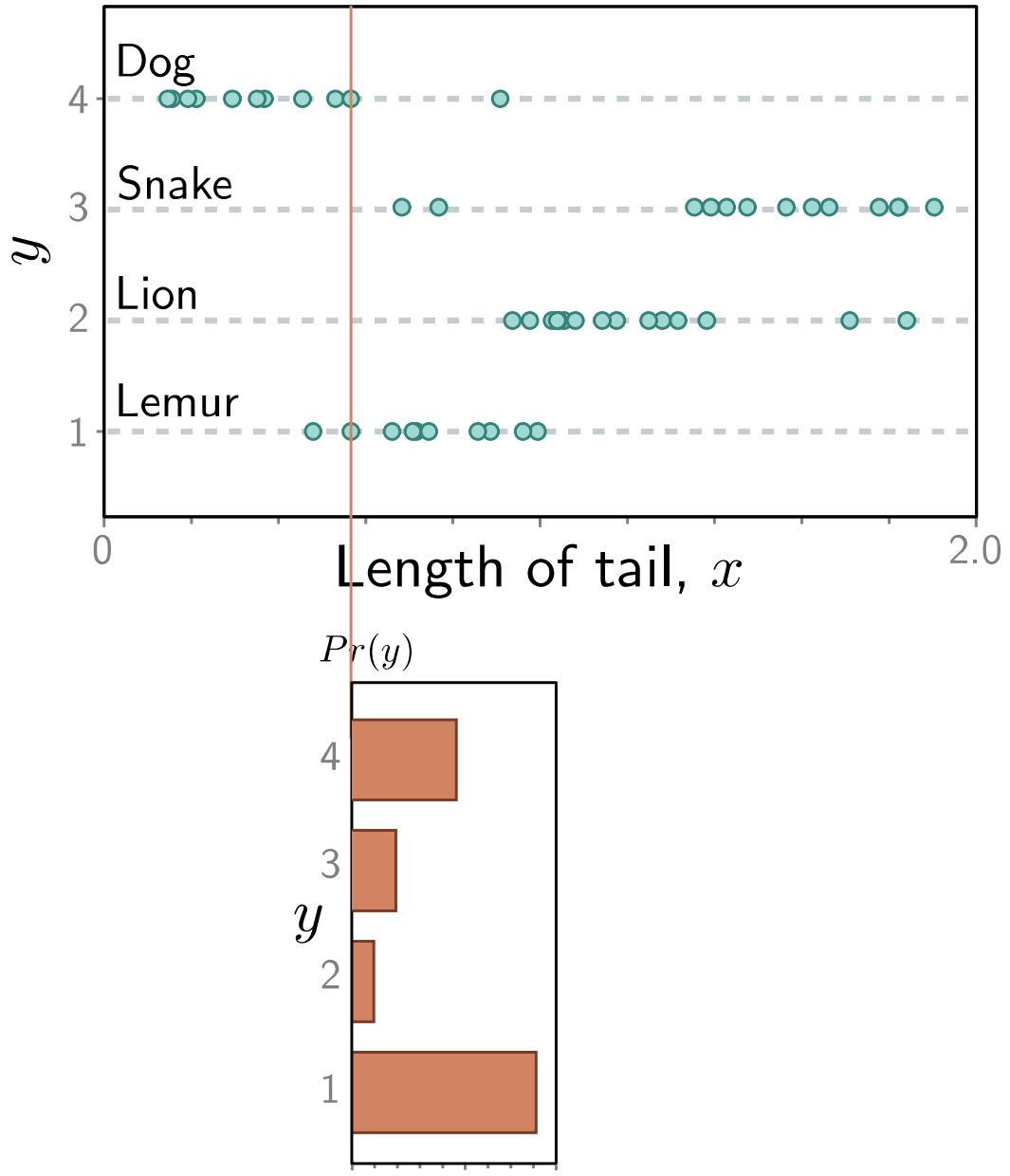

Spam
Not spam
1
0
y
0
2000
Number of words, x
$Pr(y)$
1
0
y

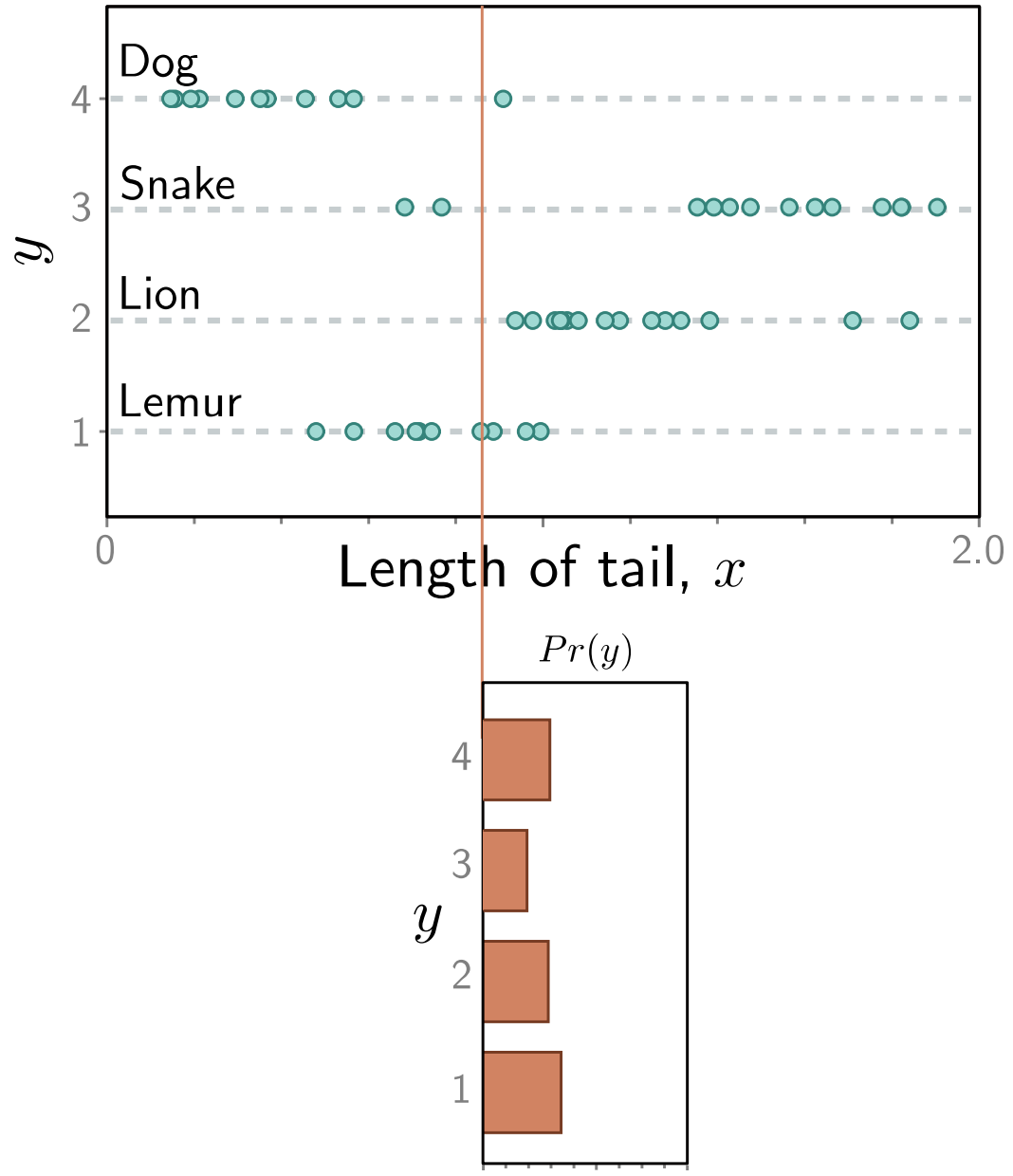

Spam
Not spam
1
0
y
0
2000
Number of words, x
$Pr(y)$
1
0
y

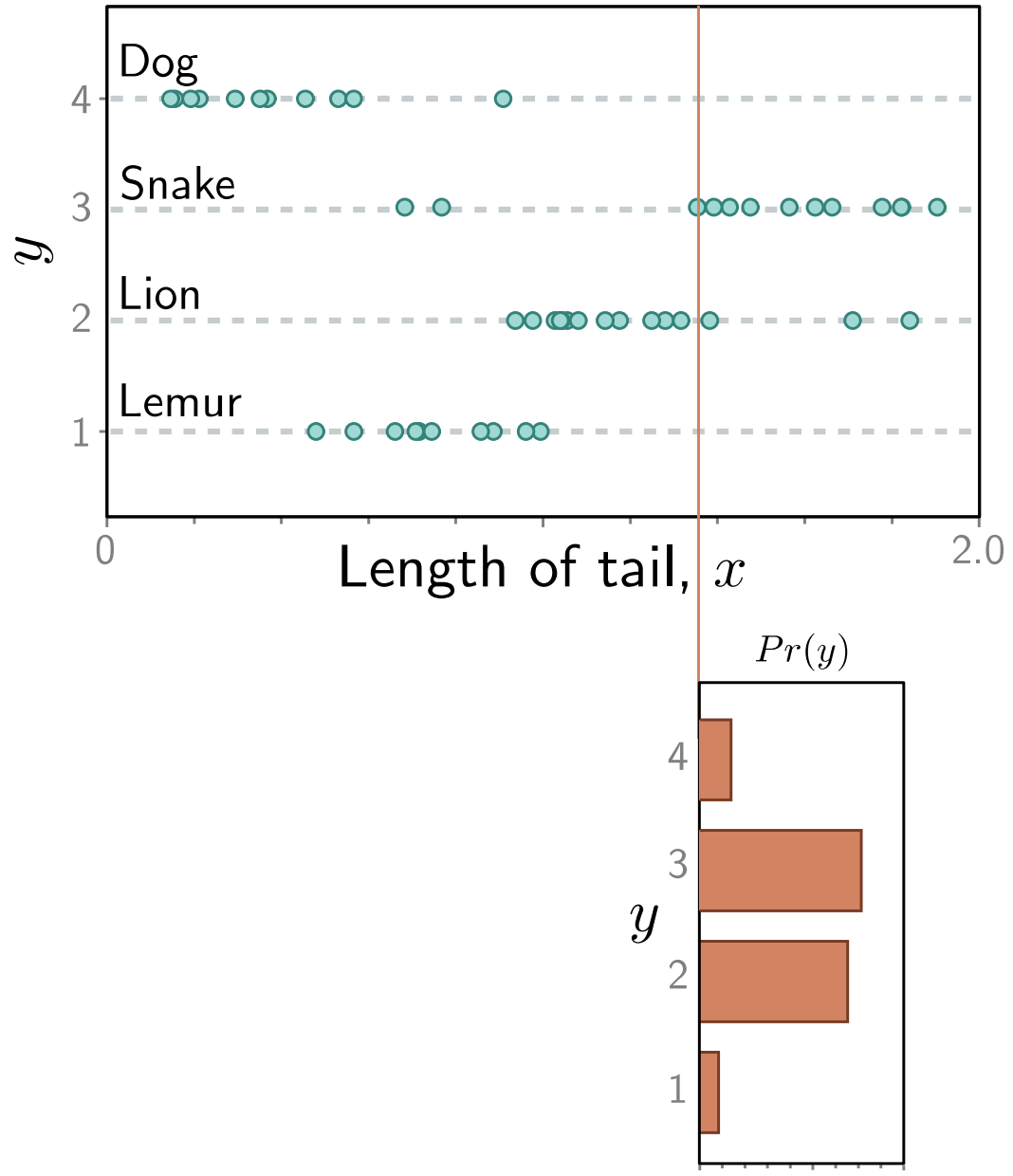

Spam
Not spam
1
0
y
0
2000
Number of words, x
$Pr(y)$
1
0
y

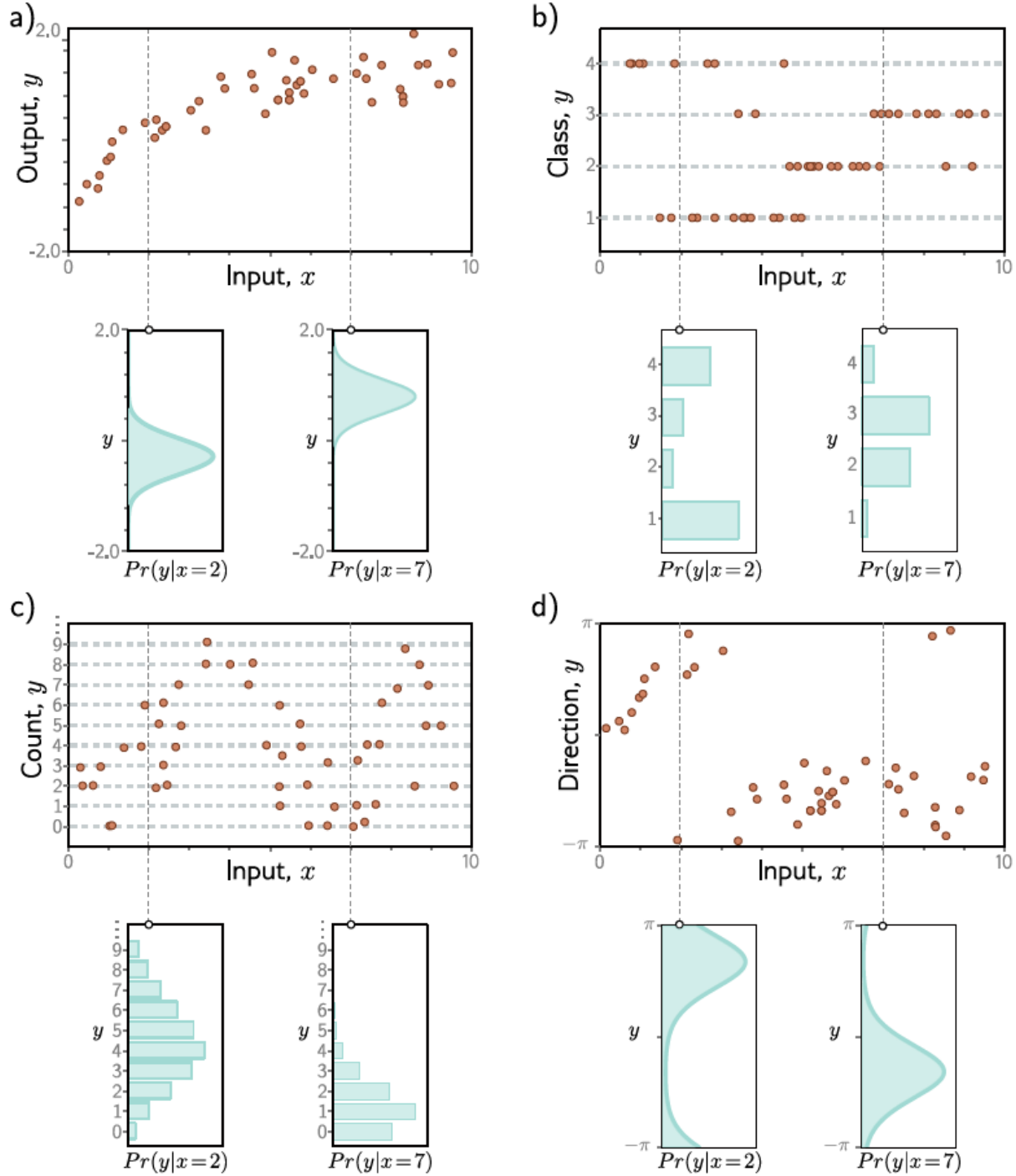

Dog
Snake
Lion
Lemur
4
3
2
1
y
0
2.0
Length of tail, x


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


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


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


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


a)
Output, y
2.0
-2.0
0
10
Input, x
y
$Pr(y|x=2)$
$Pr(y|x=7)$
b)
Class, y
1
2
3
4
c)
Count, y
0
1
2
3
4
5
6
7
8
9
d)
Direction, y
π
$-\pi$

How to construct loss functions

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

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

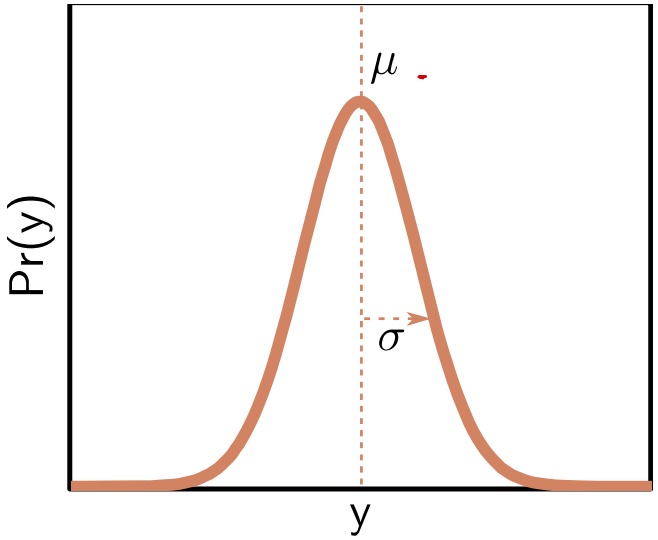
  over outputs **y** given inputs **x**.
- Loss function aims to make the outputs have high probability

How can a model predict a probability distribution?

How can a model predict a probability distribution?

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

Different parameters for each training input

Maximum likelihood criterion

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

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

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

$Pr(y_i|x_i)$

$= Pr(y_i|\theta_i)$ → dist. parameters

$= Pr(y_i|f(x_i;\phi))$

ϕ → Model params

θ → dist. parameters

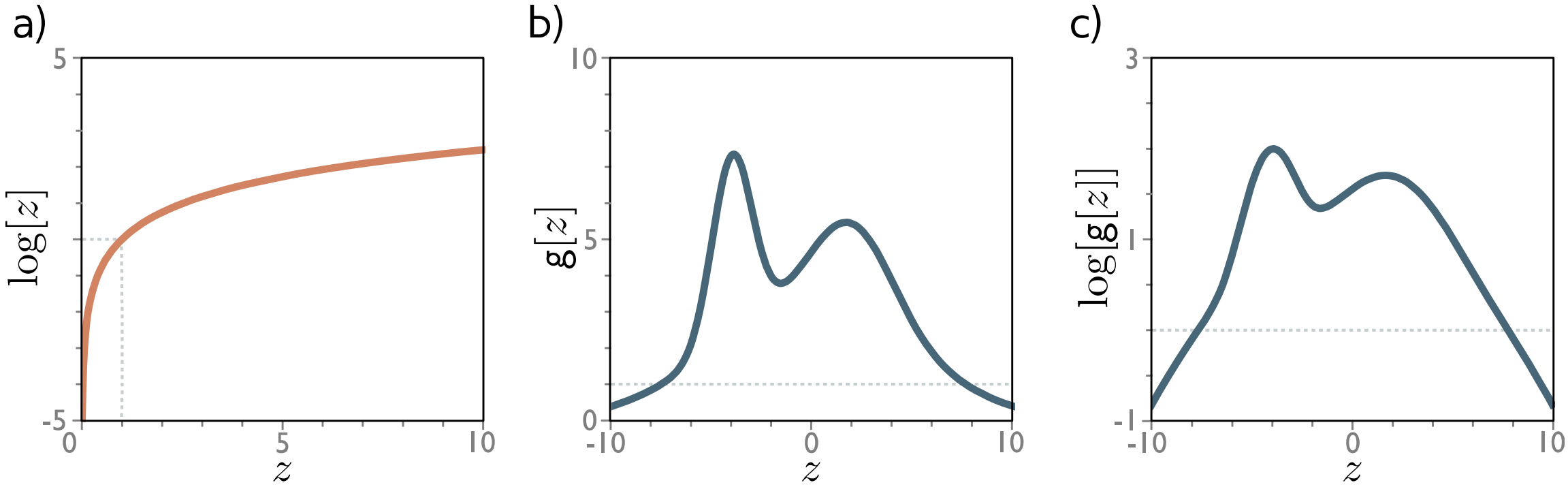
When we consider this probability as a function of the parameters ϕ, we call it a likelihood.

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
- The product might get so small that we can't easily represent it

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

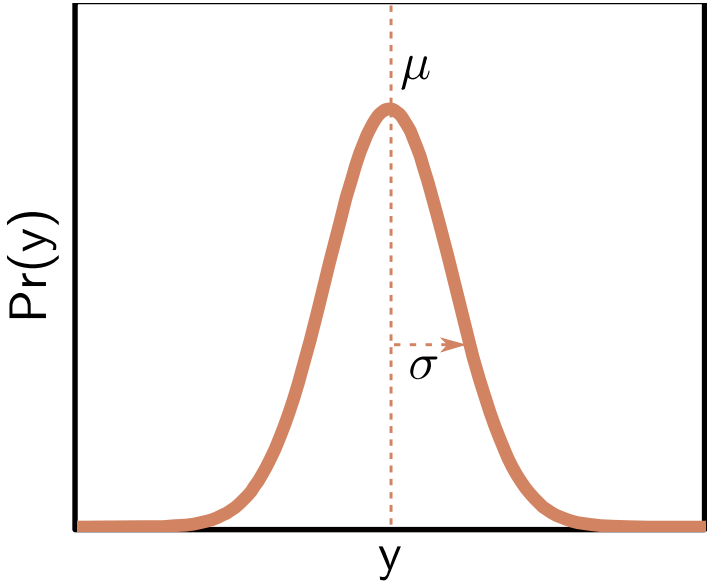
- By convention, we minimize things (i.e., a loss)

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

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)

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



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 $\{\mathbf{x}_i, \mathbf{y}_i\}$:

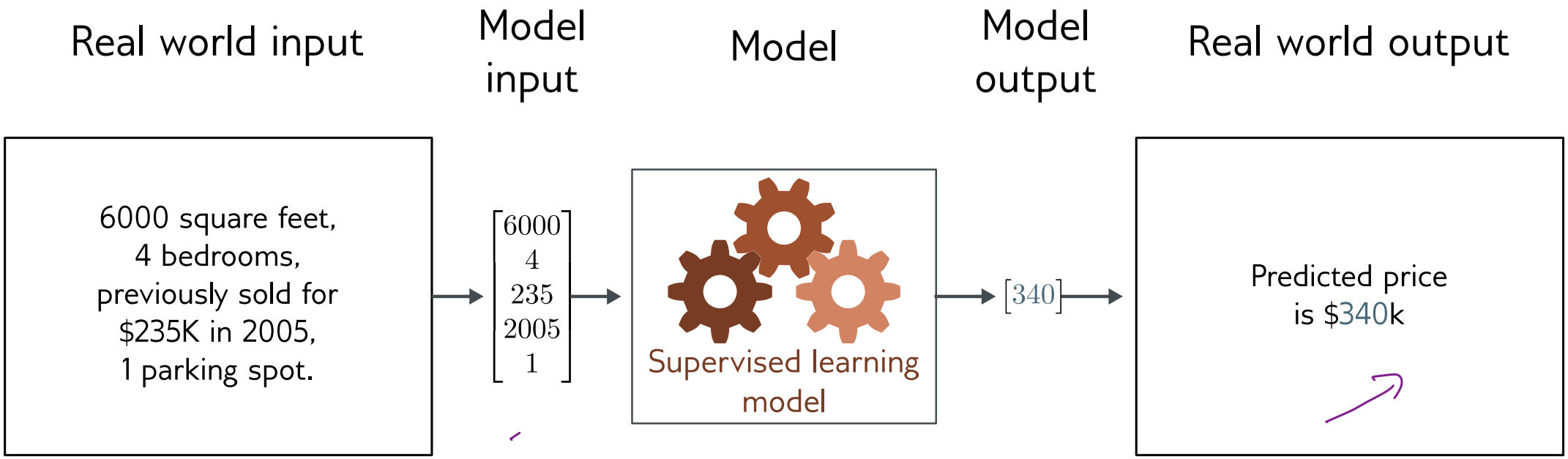
$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[L[\boldsymbol{\phi}]\right] = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log\Big[Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}])\Big] \right]. \tag{5.7}$$

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.

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

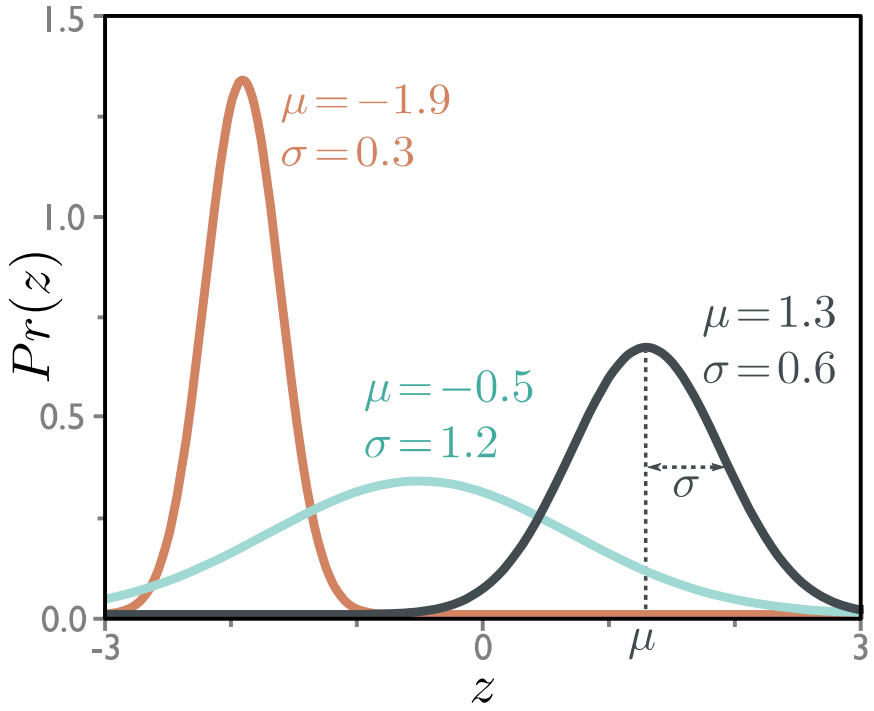


Example 1: univariate regression

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 $Pr(\mathbf{y}|\boldsymbol{\theta}) = Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}])$.

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

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

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 $\{\mathbf{x}_i, \mathbf{y}_i\}$:

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

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

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

$$\underset{\phi}{\operatorname{argmin}} \left[\sum_{i=1}^{I} -\log \frac{1}{\sqrt{2\pi\sigma^2}} + \frac{(y_i - f(x_i, \phi))^2}{2\sigma^2} \right]$$

$$= \underset{\phi}{\operatorname{argmin}} \sum_{i=1}^{I} \frac{(y_i - f(x_i, \phi))^2}{2\sigma^2}$$

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

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

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

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

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

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

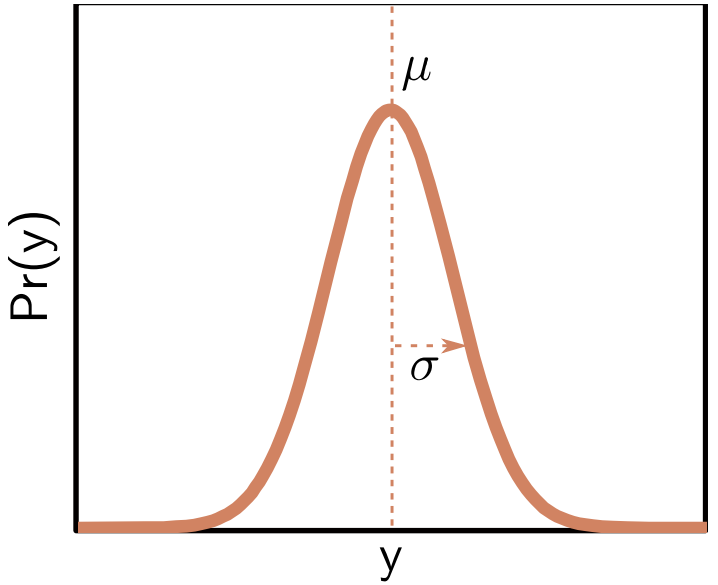
Least squares!

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.

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

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



Estimating variance

- Perhaps surprisingly, the variance term disappeared:

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

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

- But we could learn it:

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

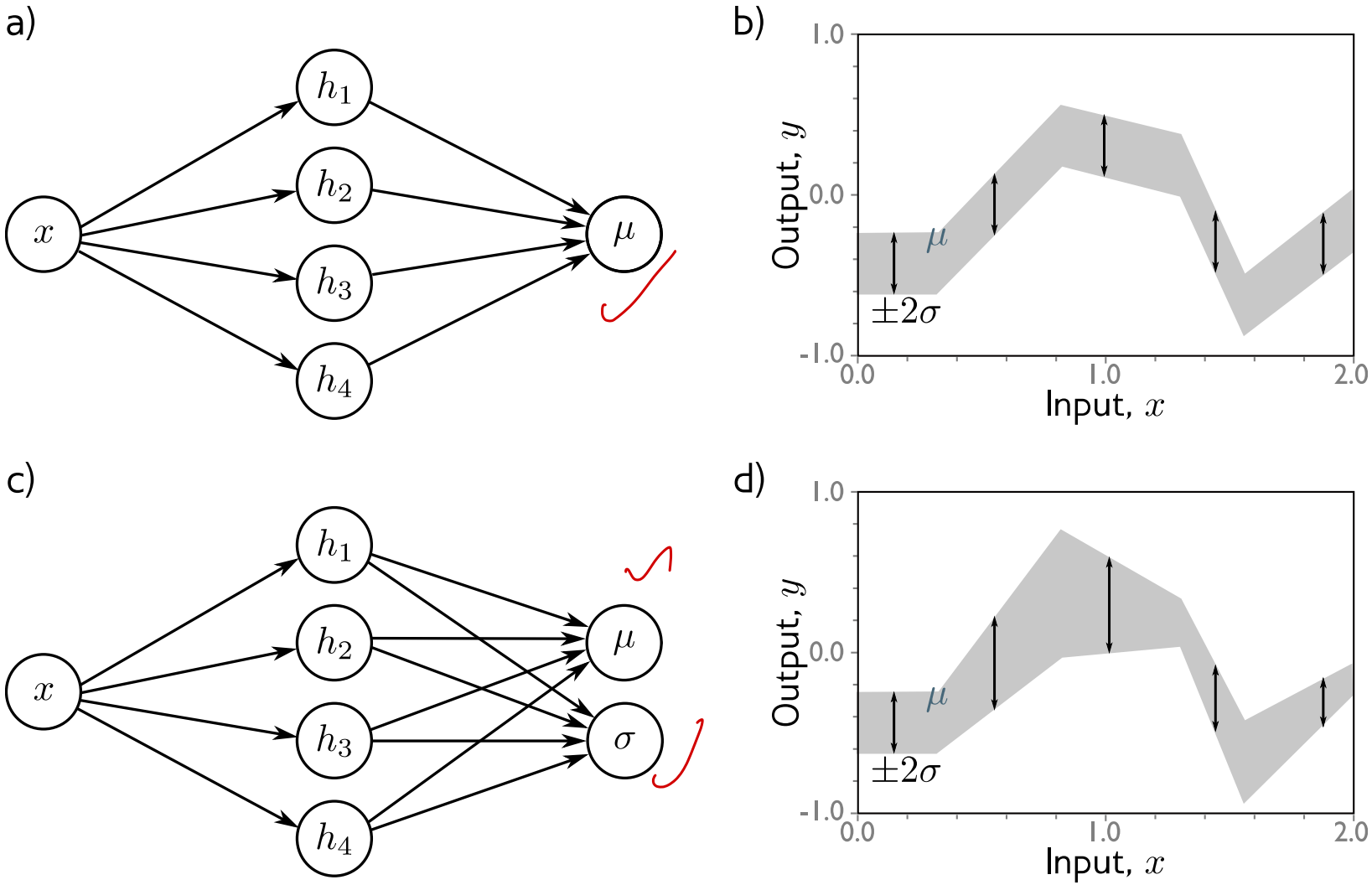
Heteroscedastic regression

- Assume that the noise σ^2 is the same everywhere.
- But we could make the noise a function of the data *x*.
- Build a model with two outputs:

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

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

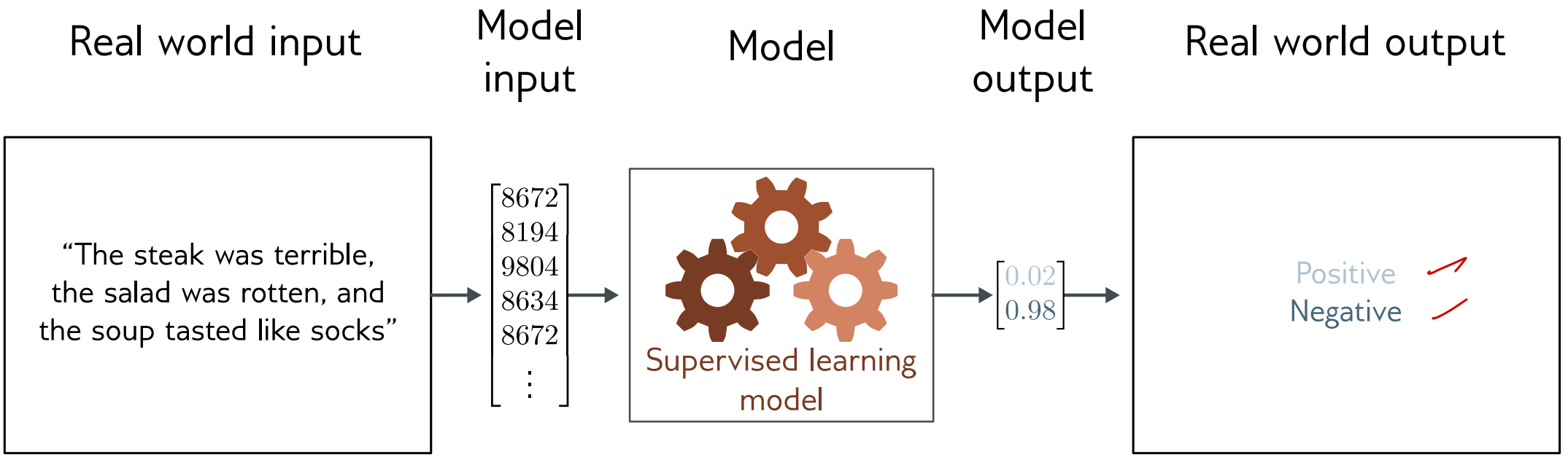
Heteroscedastic regression



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

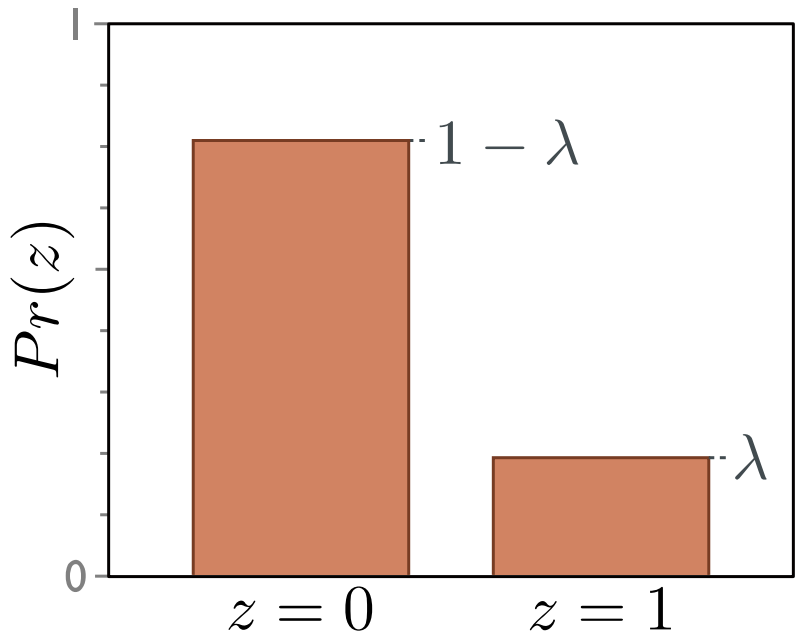
Example 2: binary classification

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 $\lambda \in [0,1]$

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

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



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 $Pr(\mathbf{y}|\boldsymbol{\theta}) = Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}])$.

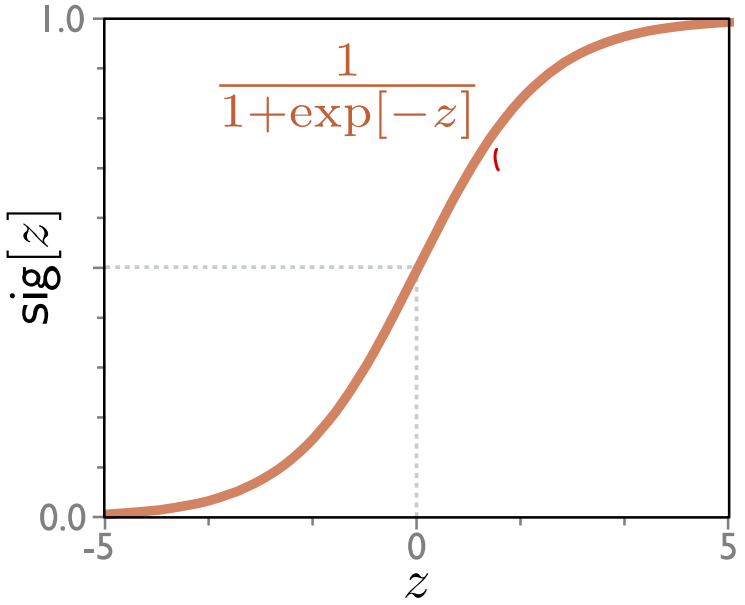
Problem:

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

Solution:

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

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



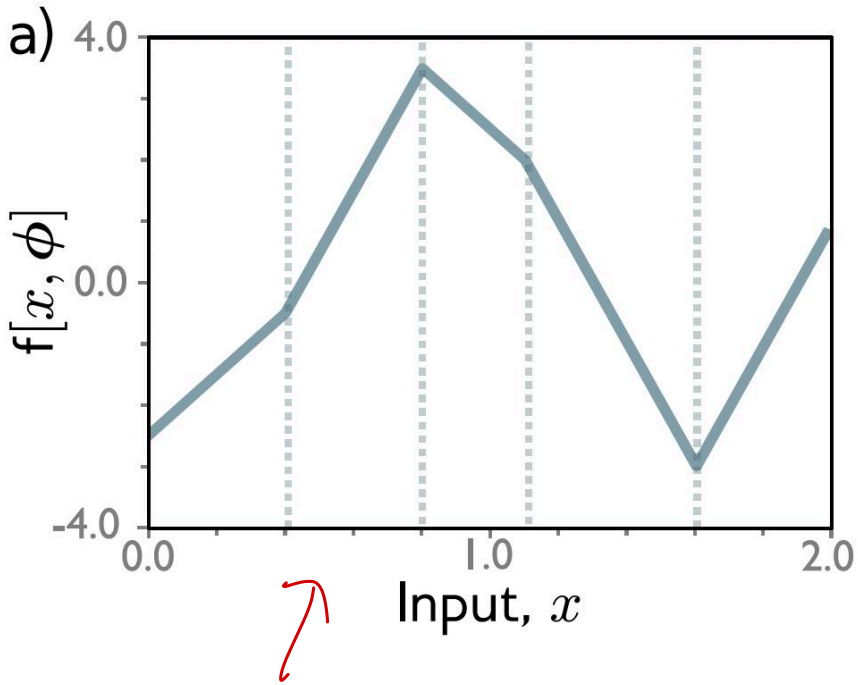
Example 2: binary classification

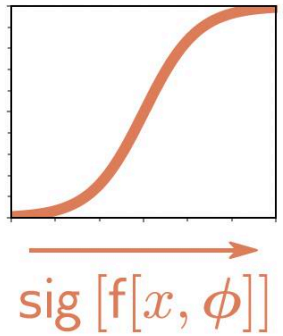
2. Set the machine learning model $\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ to predict one or more of these parameters so $\boldsymbol{\theta} = \mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ and $Pr(\mathbf{y}|\boldsymbol{\theta}) = Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}])$.

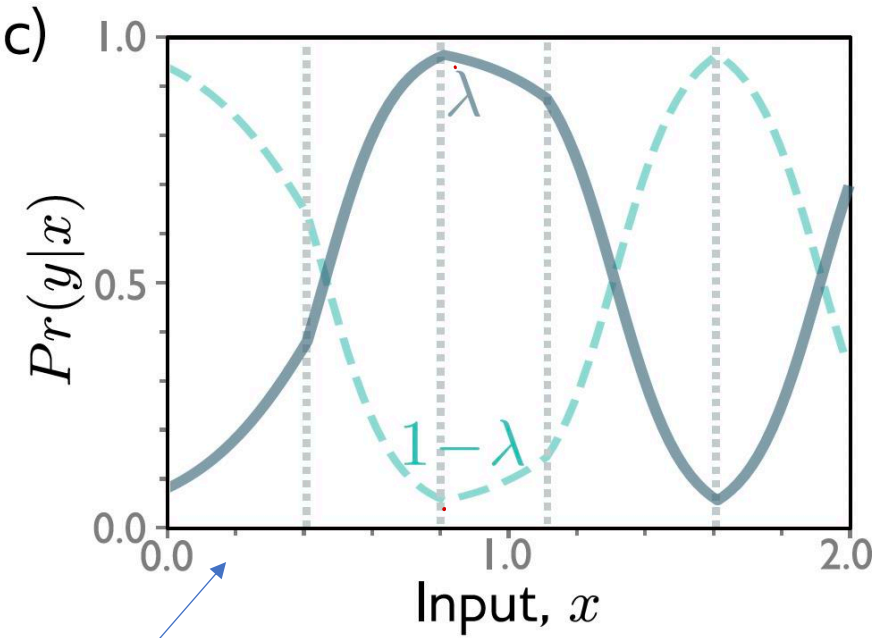
$$Pr(y|\lambda) = (1-\lambda)^{1-y} \cdot \lambda^{y}$$

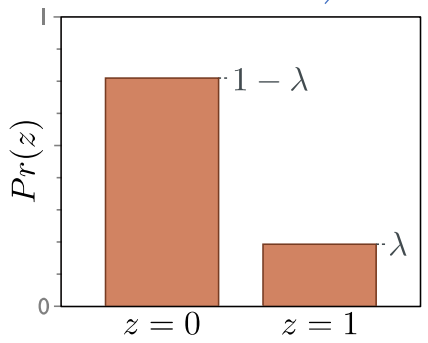
$$Pr(y|\mathbf{x}) = (1 - \text{sig}[\text{f}[\mathbf{x}|\boldsymbol{\phi}]])^{1-y} \cdot \text{sig}[\text{f}[\mathbf{x}|\boldsymbol{\phi}]]^{y}$$

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 $\{\mathbf{x}_i, \mathbf{y}_i\}$:

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

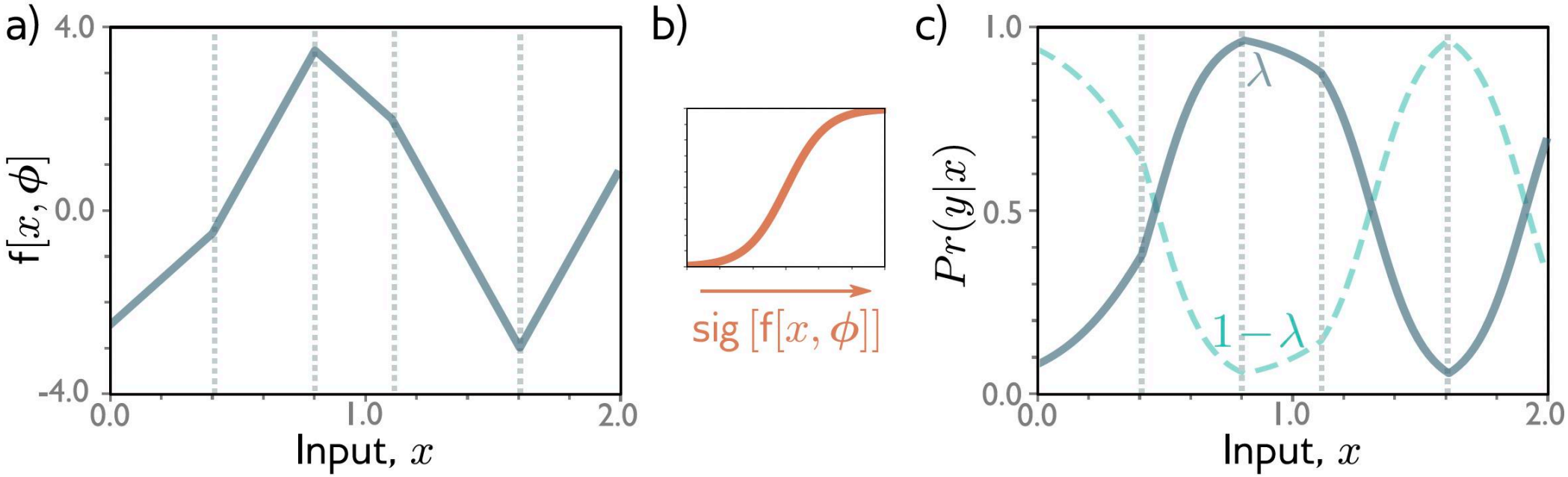
$$Pr(y|\mathbf{x}) = (1 - \text{sig}[\text{f}[\mathbf{x}|\boldsymbol{\phi}]])^{1-y} \cdot \text{sig}[\text{f}[\mathbf{x}|\boldsymbol{\phi}]]^{y}$$

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

Binary cross-entropy loss

Example 2: binary classification

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

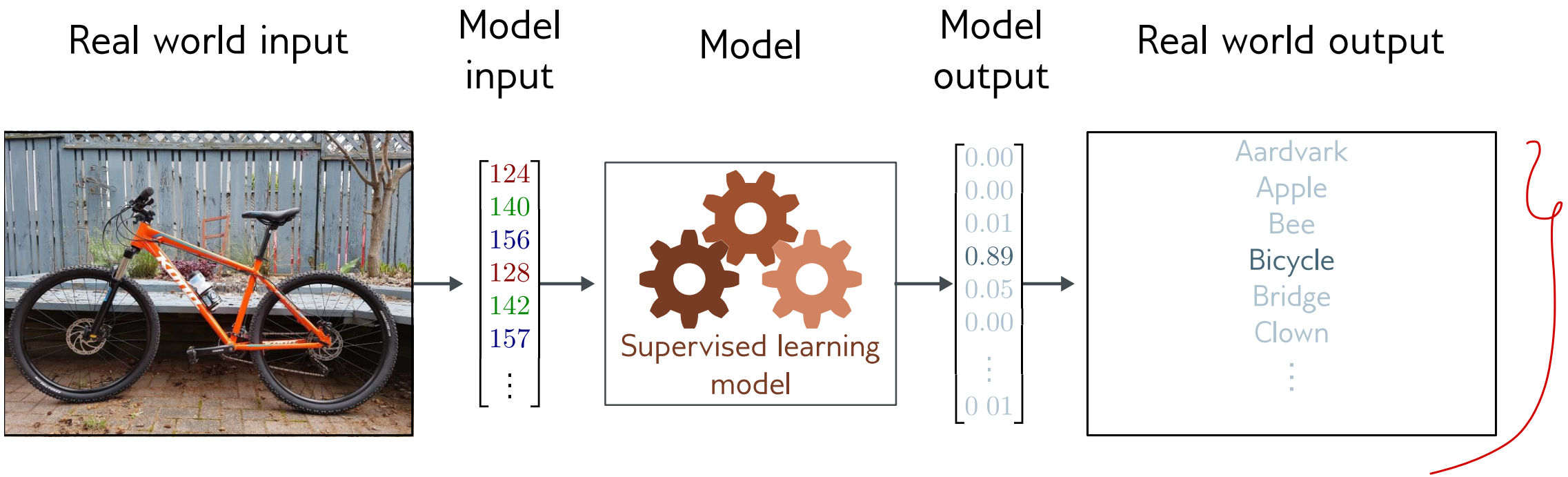


Choose y=1 where λ is greater than 0.5, otherwise 0

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



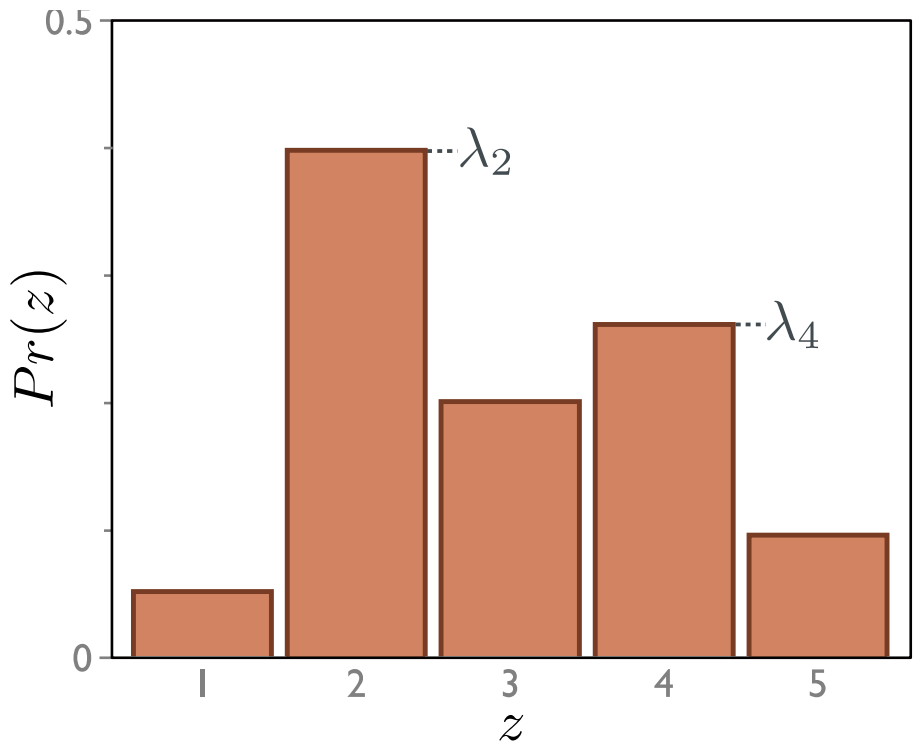
Goal: predict which of K classes $y \in \{1, 2, \ldots, K\}$ the input *x* belongs to

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 of all parameters = 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

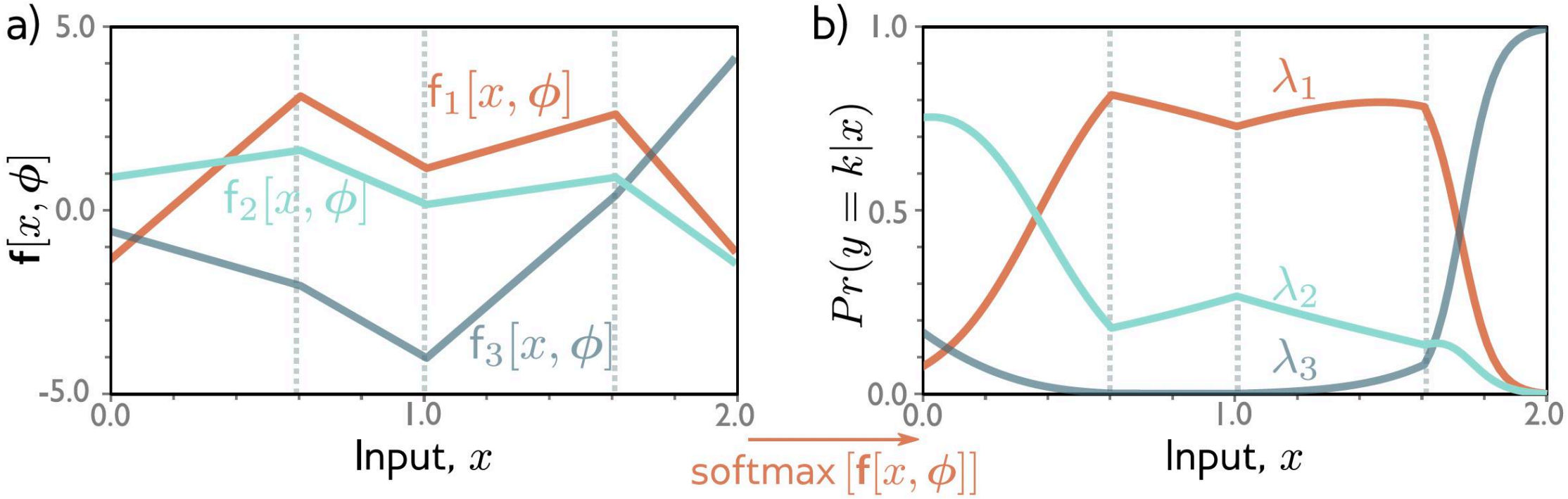
Solution:

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

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

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

Example 3: multiclass classification



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

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 $\{\mathbf{x}_i, \mathbf{y}_i\}$:

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

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

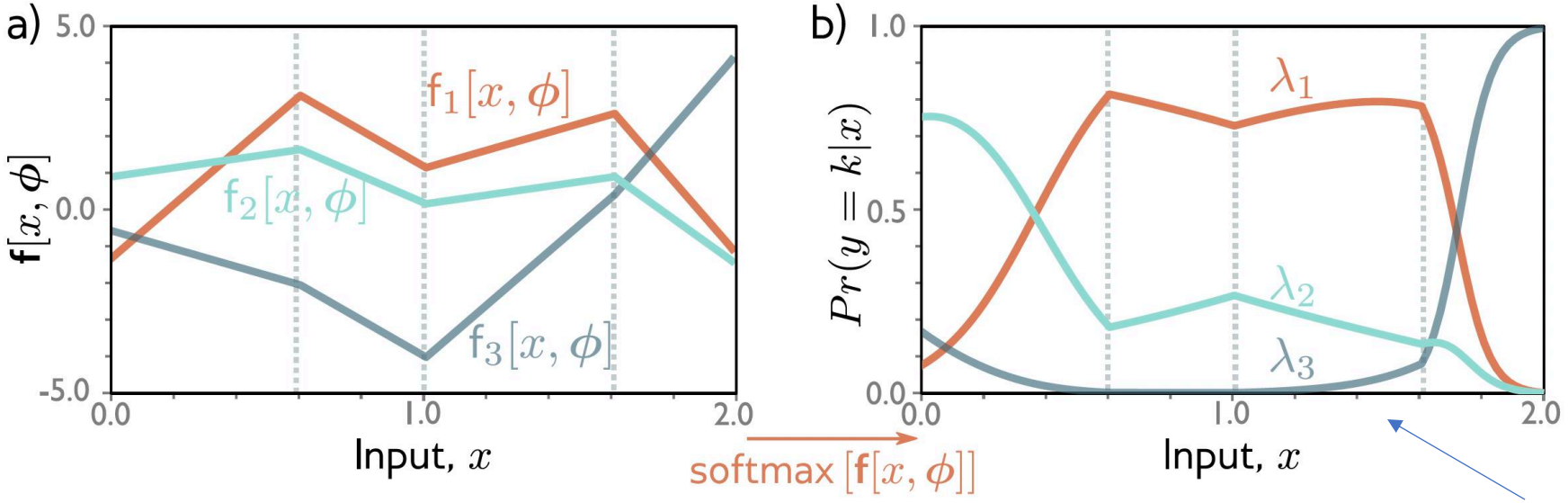
$$\text{softmax}_k[\mathbf{z}] = \frac{\exp[z_k]}{\sum_{k'=1}^{K} \exp[z_{k'}]}$$

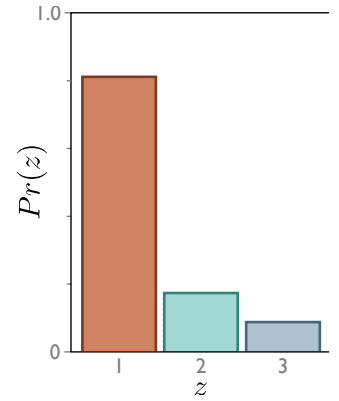
$$= -\sum_{i=1}^{I} \mathrm{f}_{y_i}\left[\mathbf{x}_i, \boldsymbol{\phi}\right] - \log\left[\sum_{k=1}^{K} \exp\left[\ \mathrm{f}_k\left[\mathbf{x}_i, \boldsymbol{\phi}\right]\right]\right]$$

Multiclass cross-entropy loss

Example 3: multiclass classification

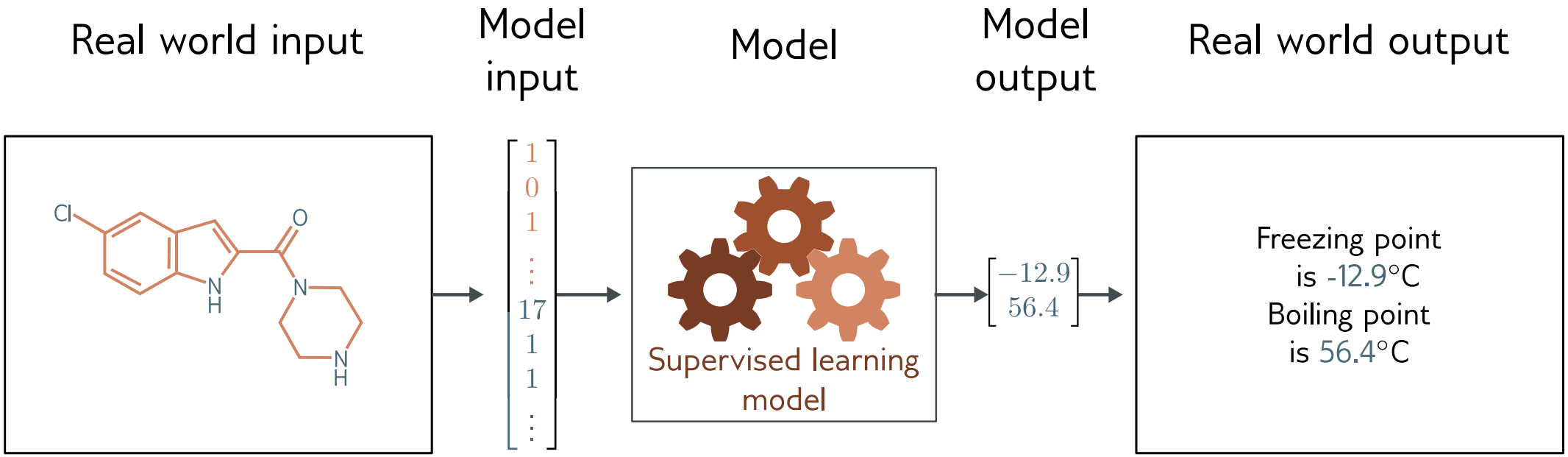
4. To perform inference for a new test example $\mathbf{x}$, return either the full distribution $Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \hat{\boldsymbol{\phi}}])$ or the maximum of this distribution.





Choose the class with the largest probability

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

$$Pr(\mathbf{y}|\boldsymbol{\mu}, \sigma^2) = \prod_{d=1}^{D_o} Pr(y_d|\mu_d, \sigma^2)$$

$$= \prod_{d=1}^{D_o} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y_d - \mu_d)^2}{2\sigma^2}\right]$$

- Make network with D_o outputs to predict means

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

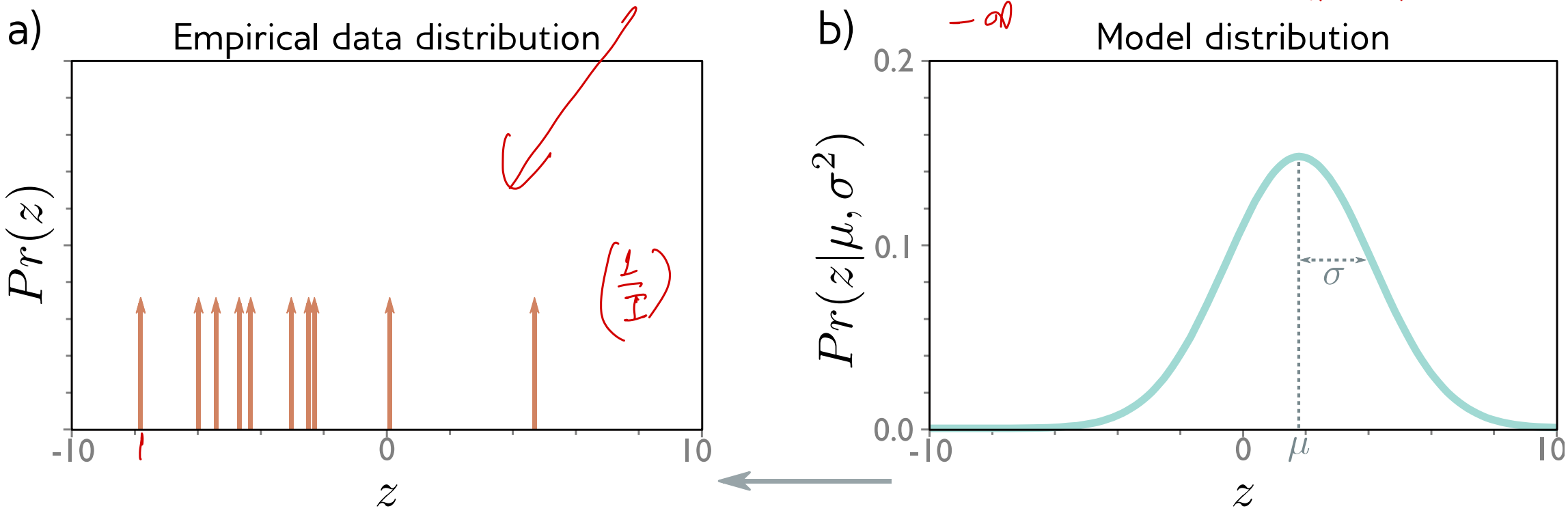
Example 4: multivariate regression

- 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

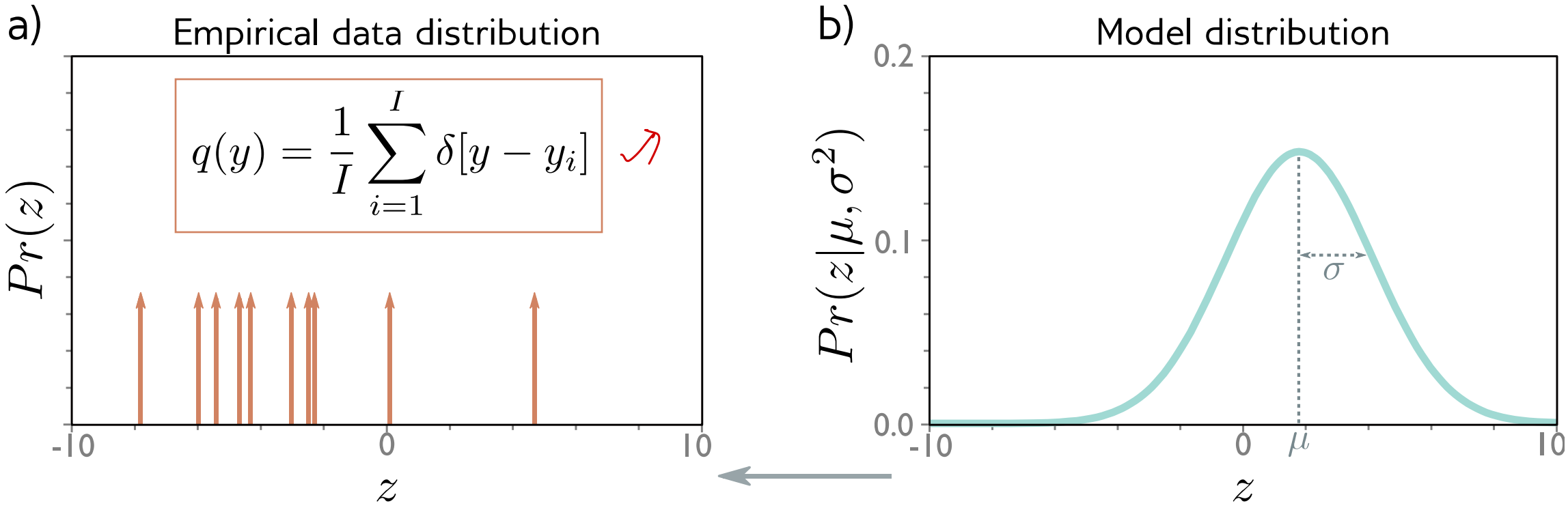
Cross Entropy



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

Kullback-Leibler Divergence -- a measure between probability distributions

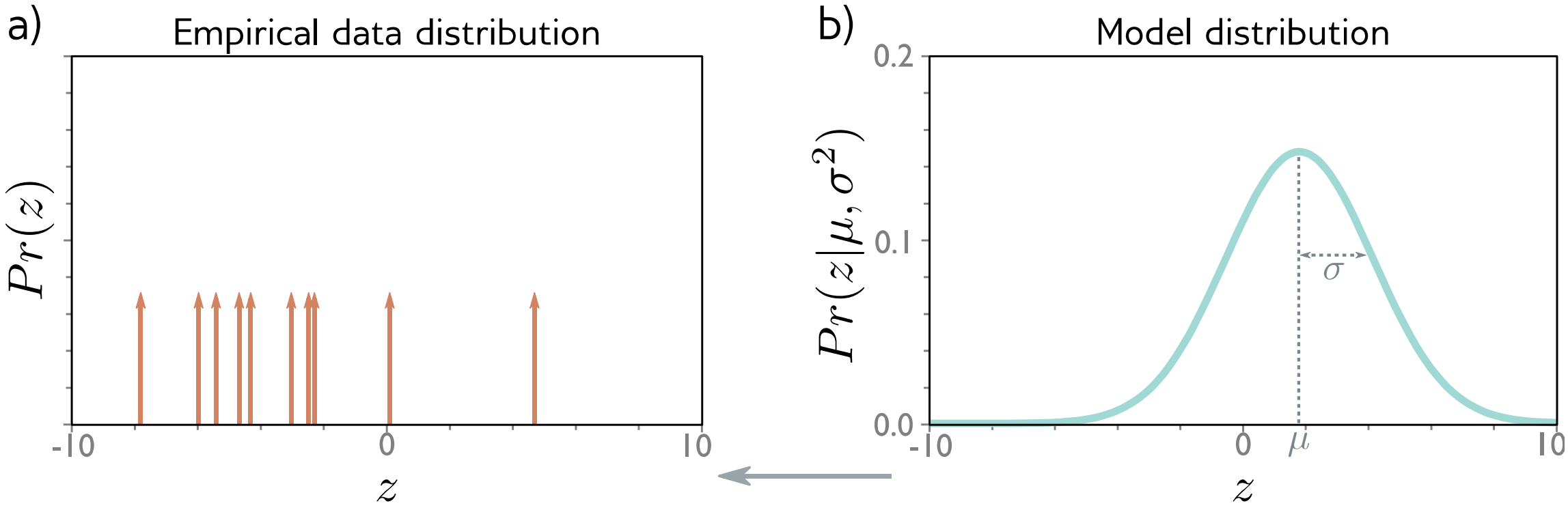
Cross Entropy



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

Kullback-Leibler Divergence -- a measure between probability distributions

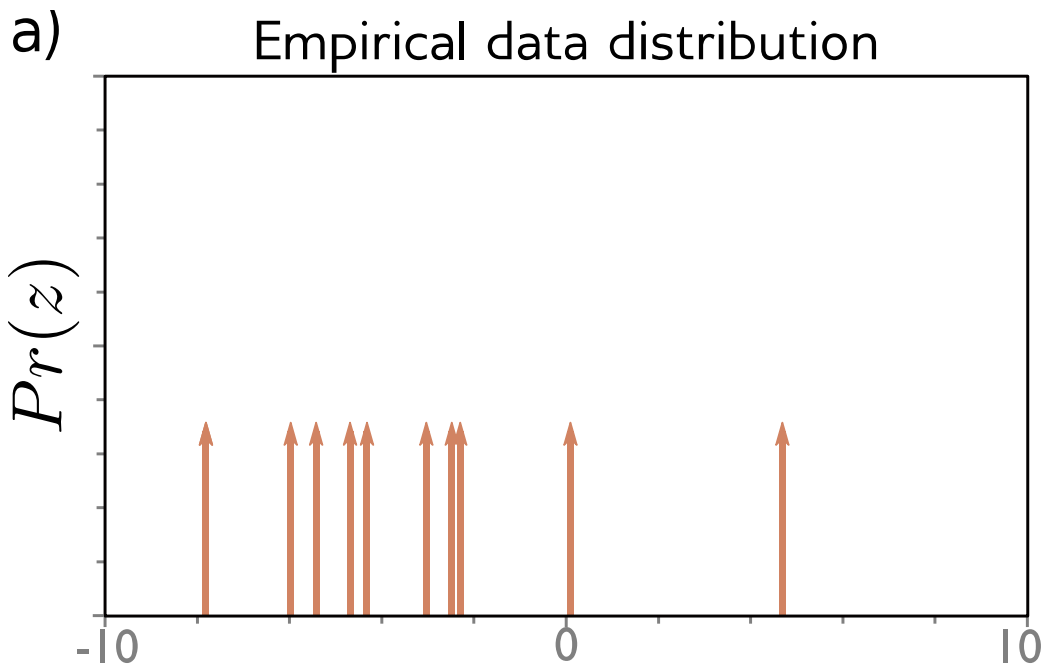
Cross Entropy

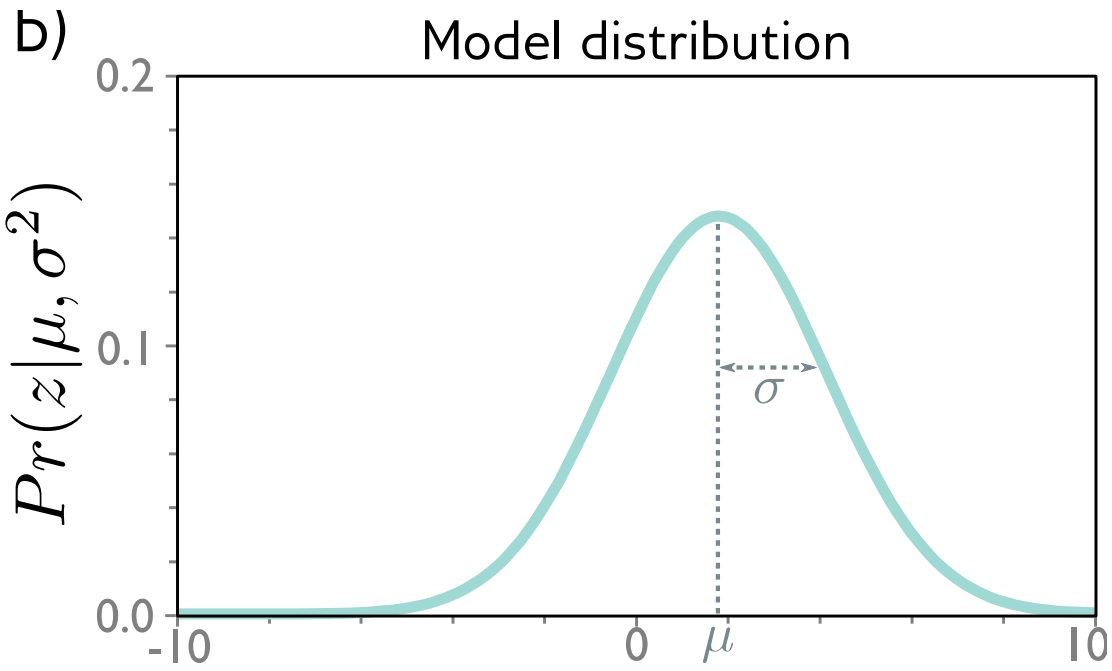


$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left[\int_{-\infty}^{\infty} q(y) \log[q(y)] dy - \int_{-\infty}^{\infty} q(y) \log \left[Pr(y|\boldsymbol{\theta}) \right] dy \right]$$

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

Cross Entropy





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

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

Minimum negative log likelihood

Cross entropy in machine learning

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

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

Minimum negative log likelihood

In machine learning:

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

Next up

- We have models with parameters!
- We have loss functions!
- Now let’s find the parameters that give the smallest loss
  - Training, learning, or fitting the model