

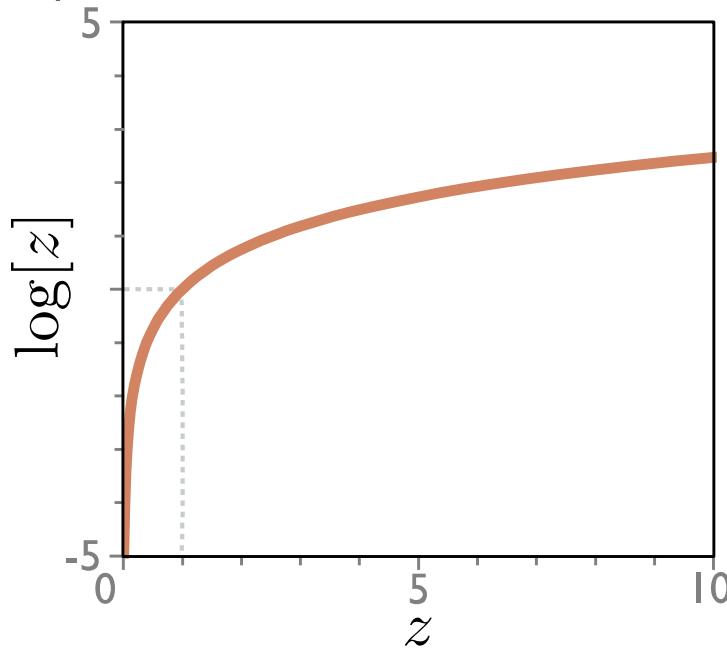
CM20315 - Machine Learning

Prof. Simon Prince
5. Loss functions

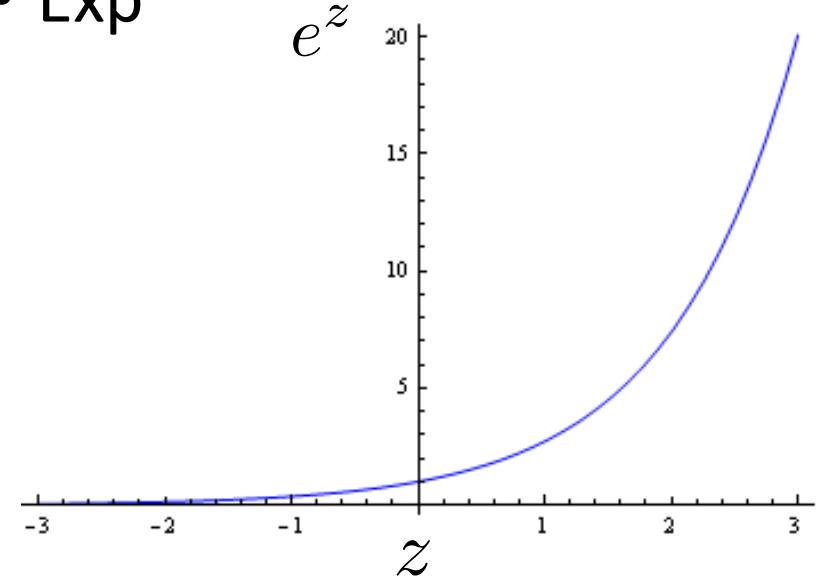


Log and exp functions

- Log



- Exp



- Two rules:

$$\log[\exp[z]] = z$$

$$\log[a \cdot b] = \log[a] + \log[b]$$

Regression

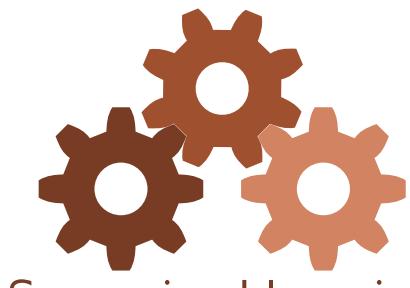
Real world input

6000 square feet,
4 bedrooms,
previously sold for
\$235K in 2005,
1 parking spot.

Model
input

$$\begin{bmatrix} 6000 \\ 4 \\ 235 \\ 2005 \\ 1 \end{bmatrix}$$

Model



Supervised learning
model

Model
output

$$[340]$$

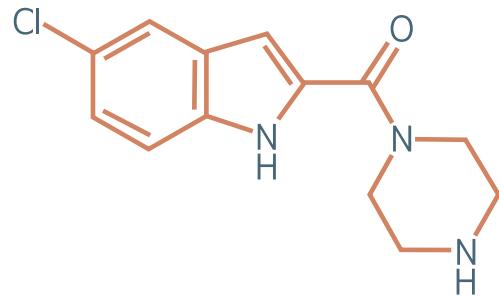
Real world output

Predicted price
is \$340k

- Univariate regression problem (one output, real value)

Graph regression

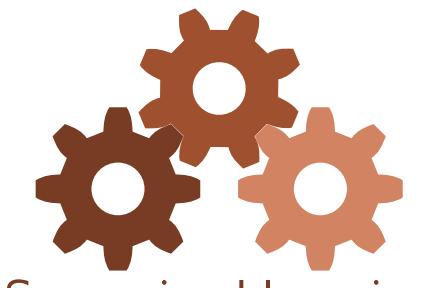
Real world input



Model
input

$$\begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ 17 \\ 1 \\ 1 \\ \vdots \end{bmatrix}$$

Model



Supervised learning
model

Model
output

$$\begin{bmatrix} -12.9 \\ 56.4 \end{bmatrix}$$

Real world output

Freezing point
is -12.9°C
Boiling point
is 56.4°C

- Multivariate regression problem (>1 output, real value)

Text classification

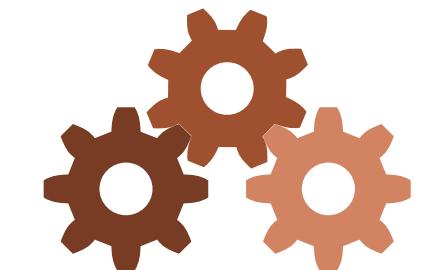
Real world input

“The steak was terrible,
the salad was rotten, and
the soup tasted like socks”

Model
input

$$\begin{bmatrix} 8672 \\ 8194 \\ 9804 \\ 8634 \\ 8672 \\ \vdots \end{bmatrix}$$

Model



Supervised learning
model

Model
output

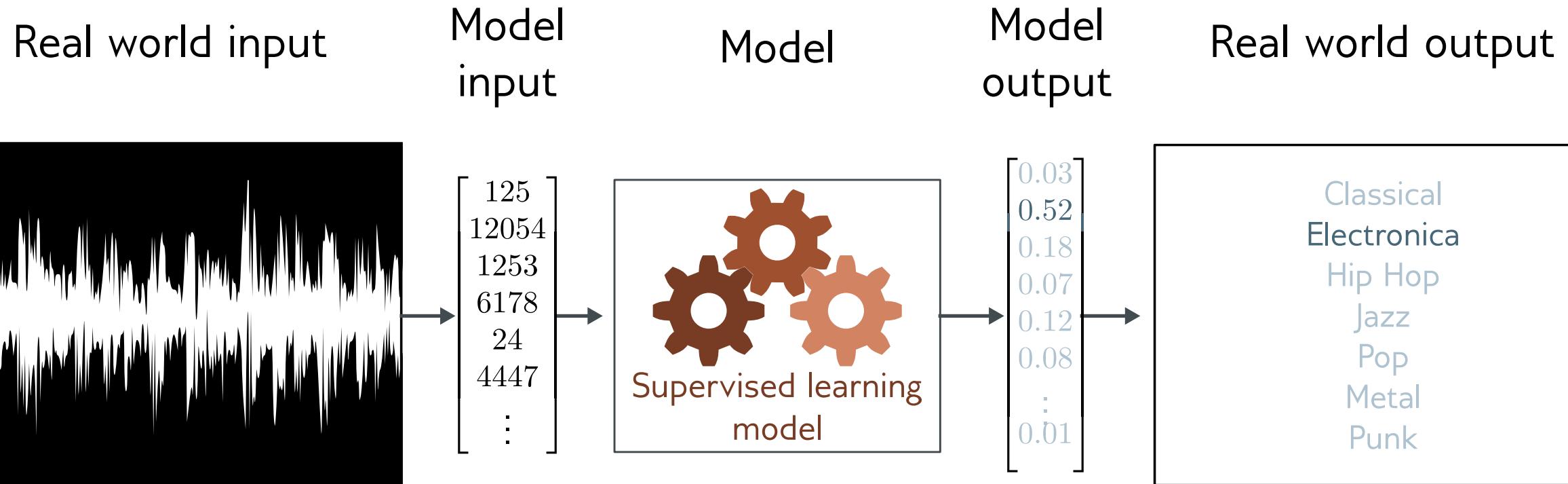
$$\begin{bmatrix} 0.02 \\ 0.98 \end{bmatrix}$$

Real world output

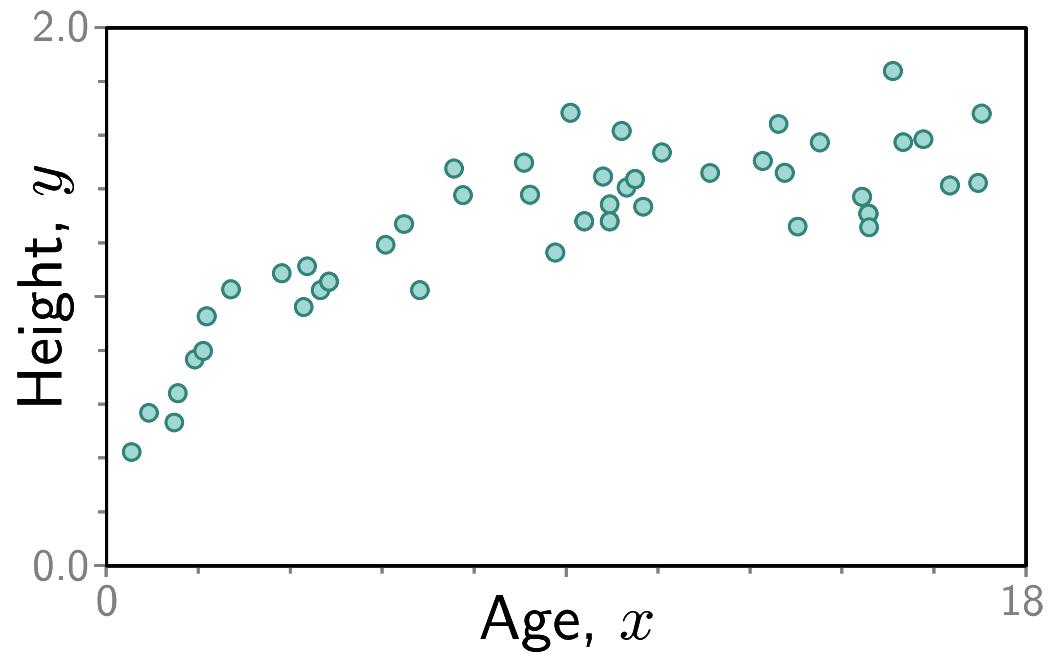
Positive
Negative

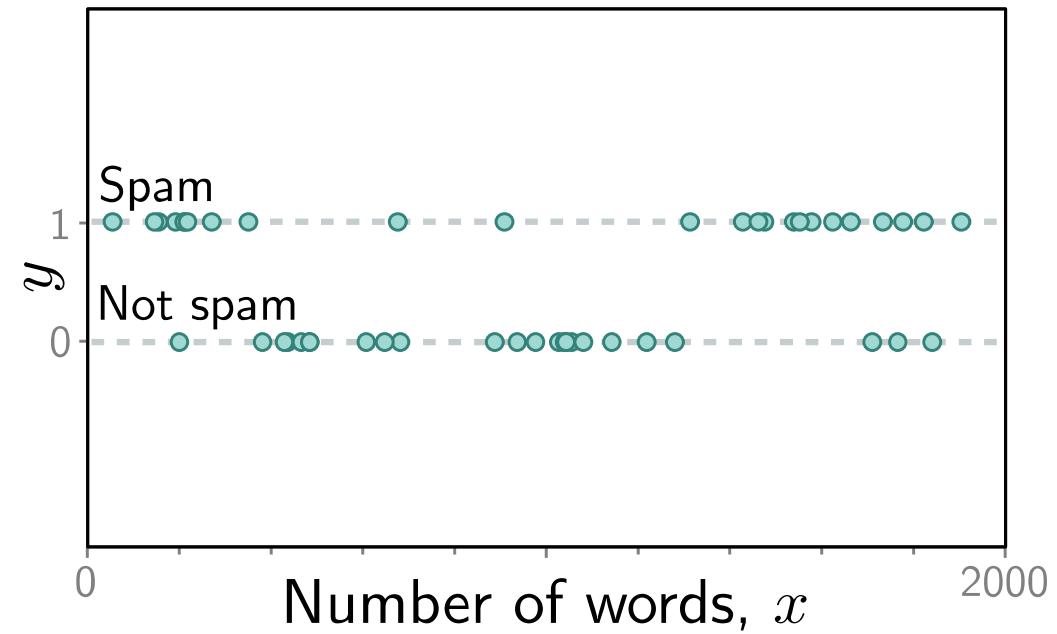
- Binary classification problem (two discrete classes)

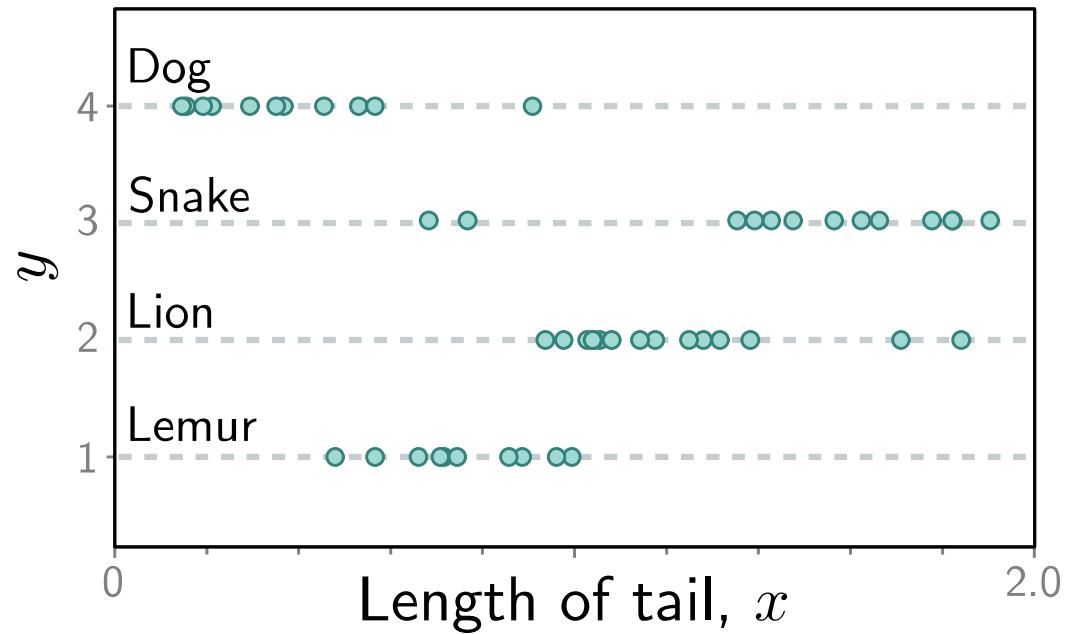
Music genre classification

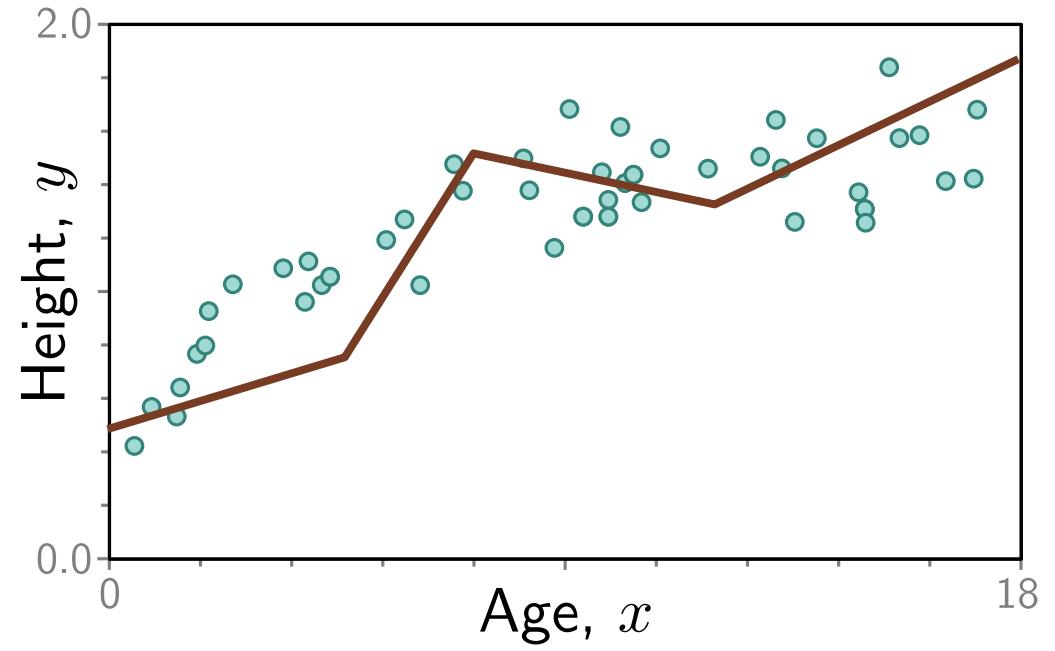


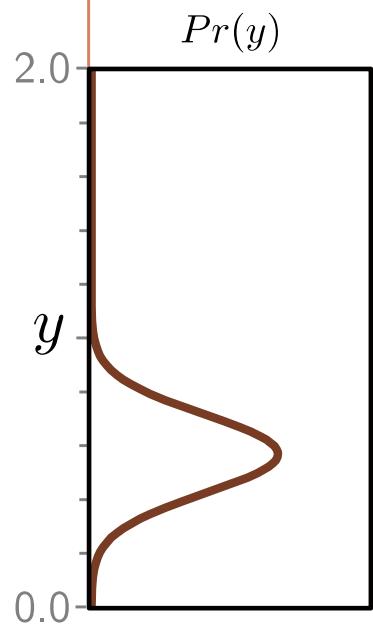
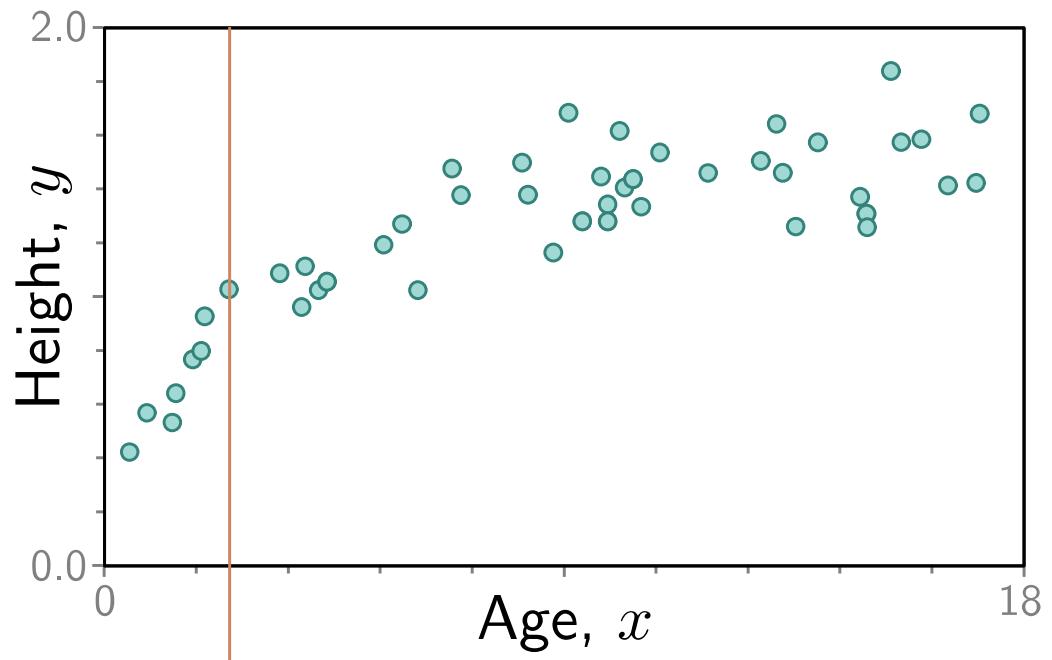
- Multiclass classification problem (discrete classes, >2 possible values)

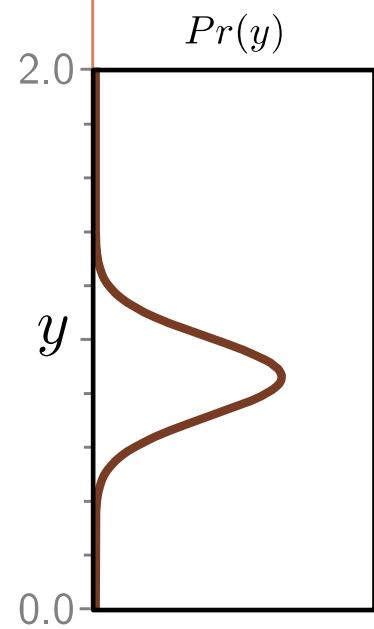
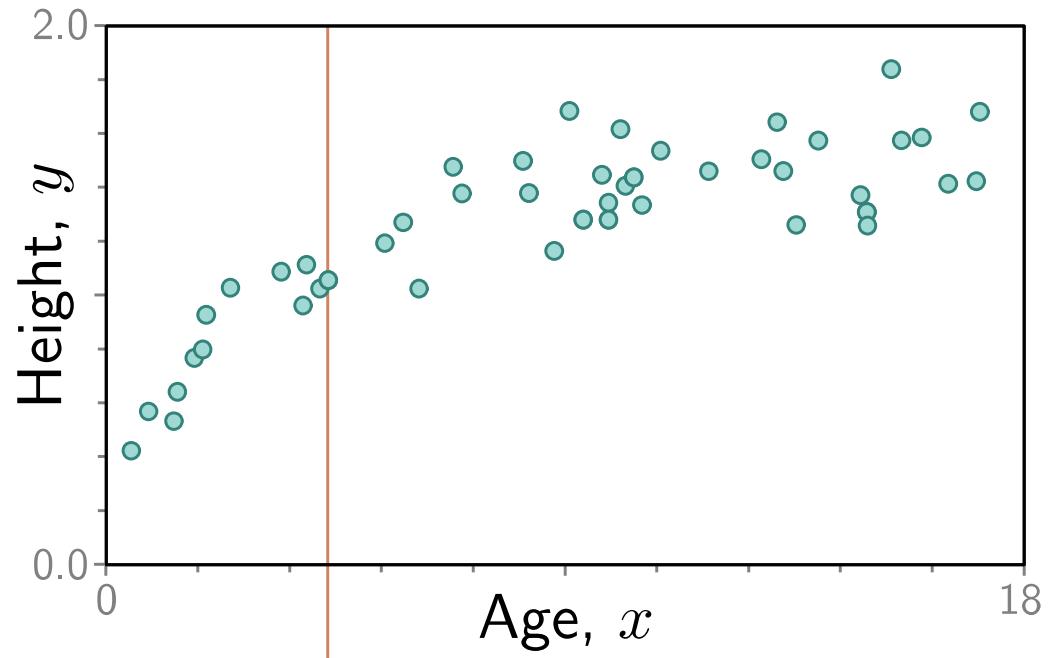


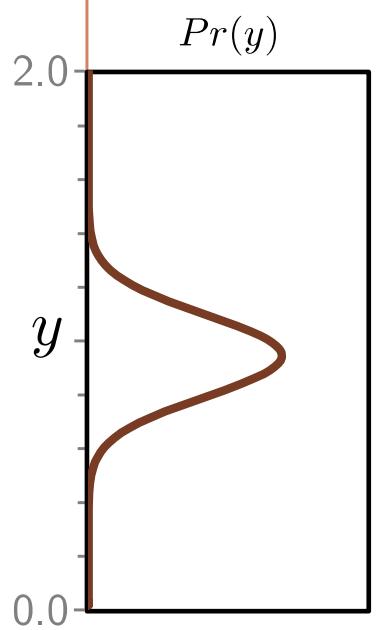
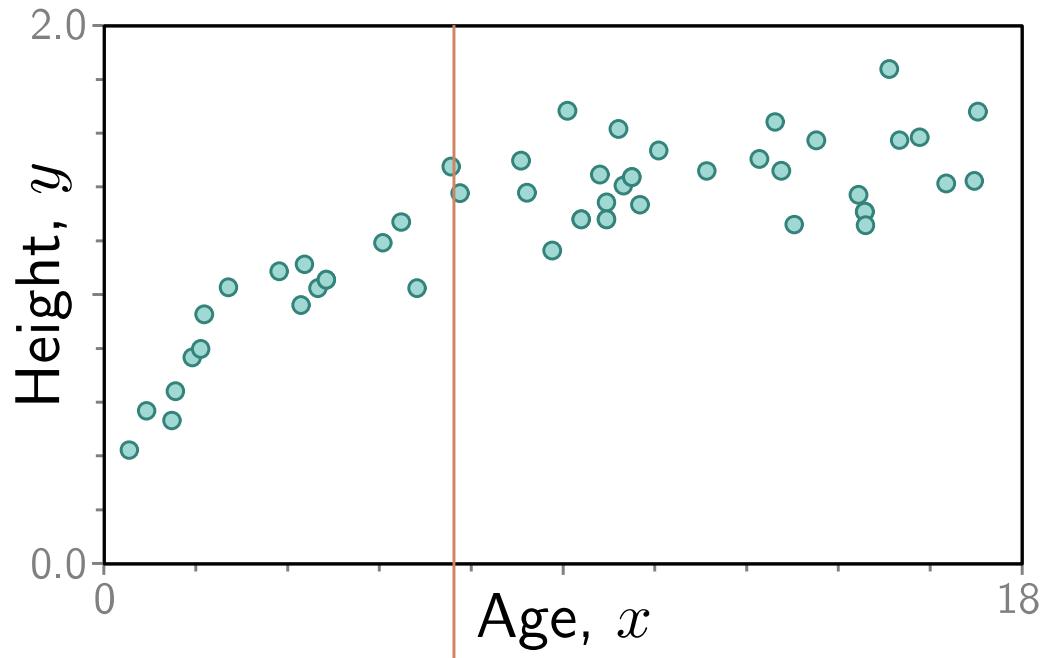


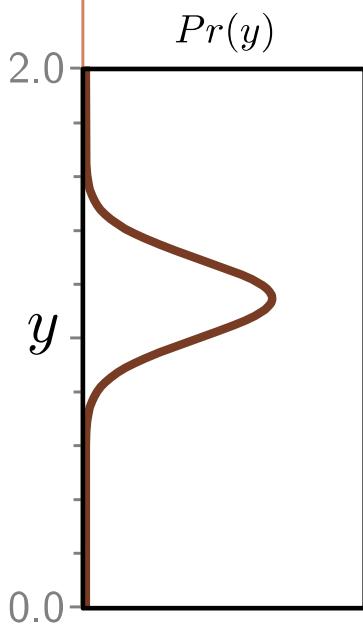
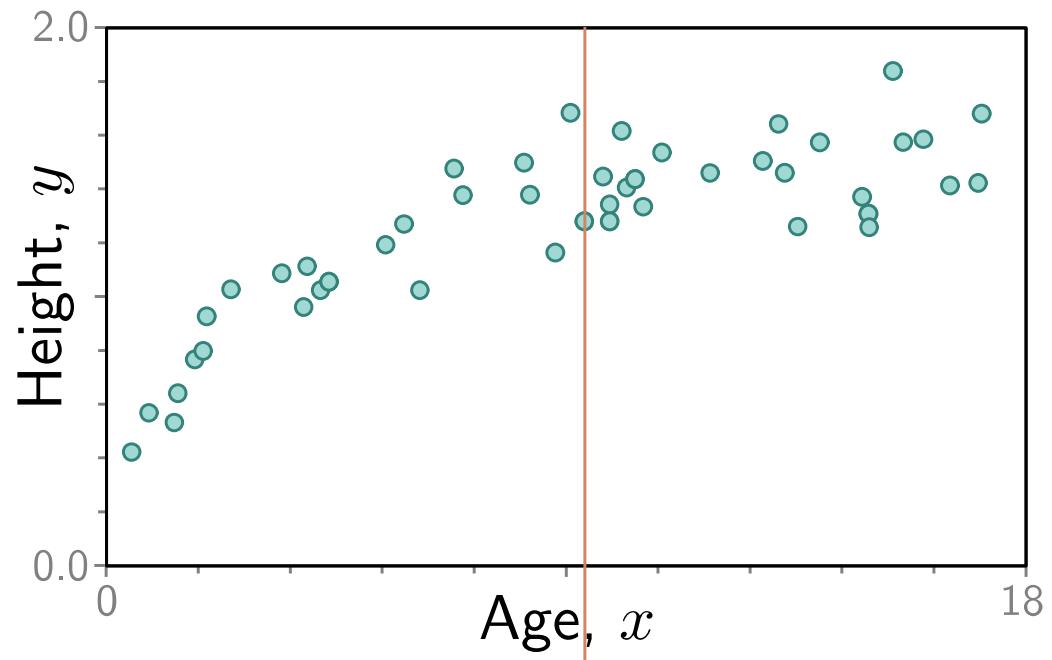


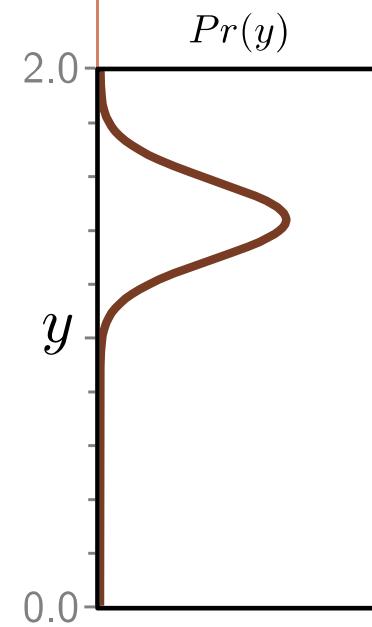
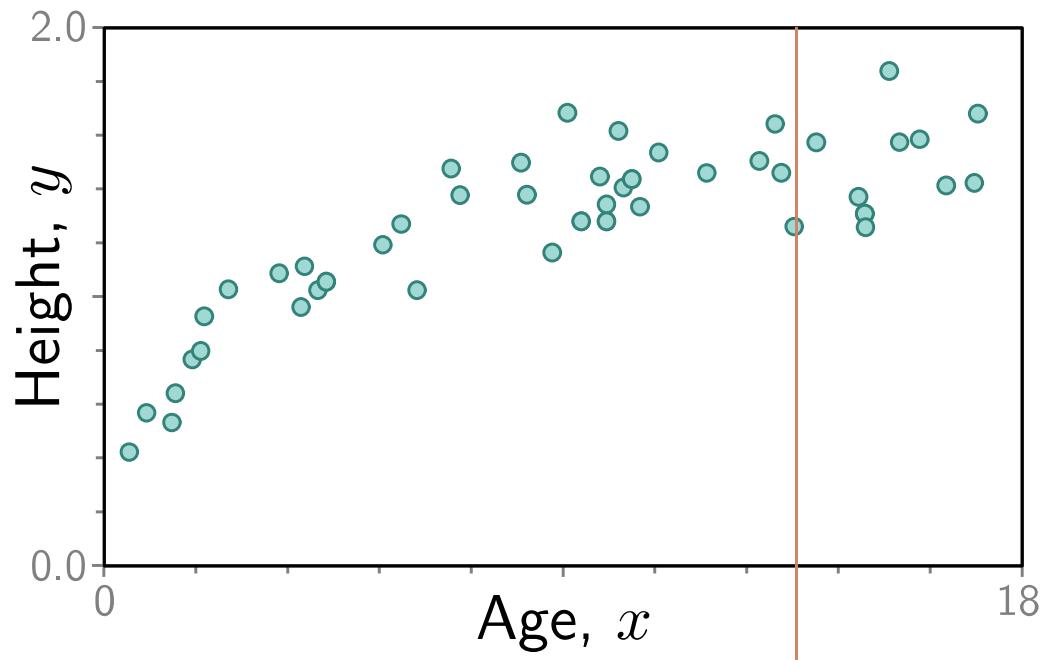


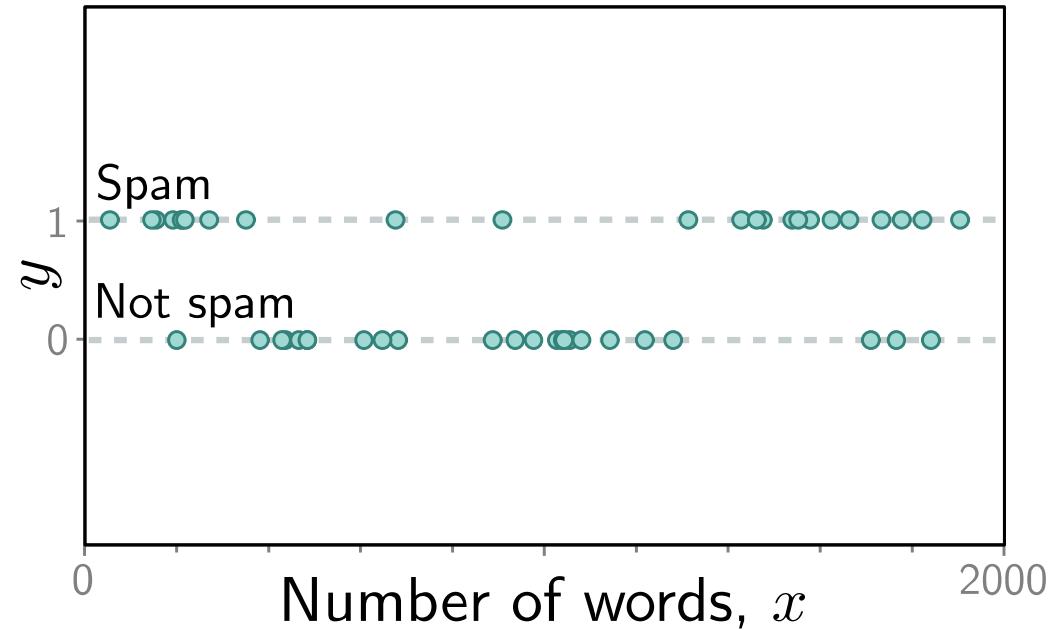


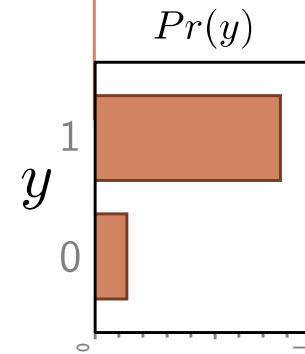
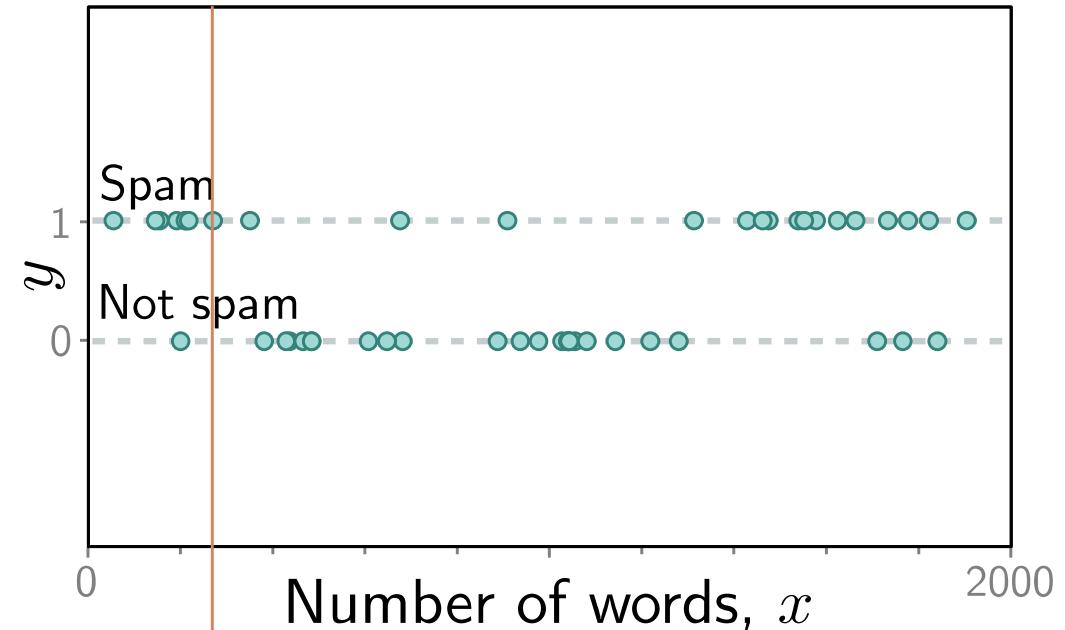


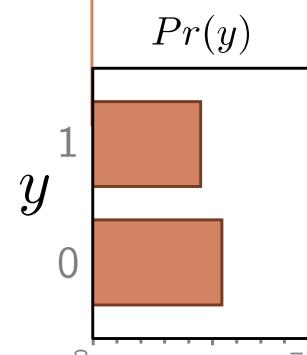
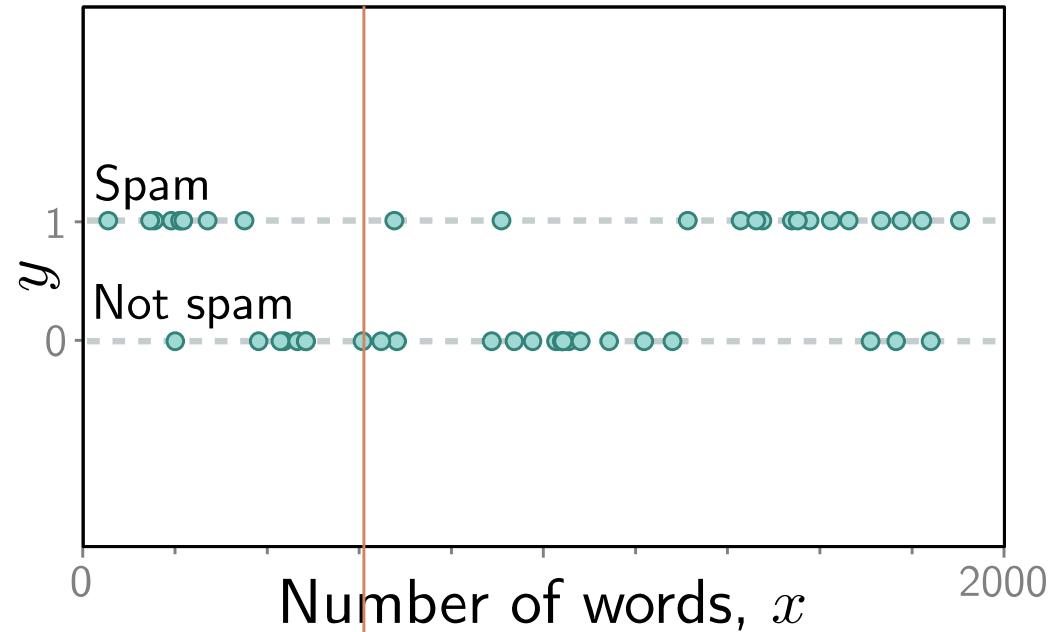


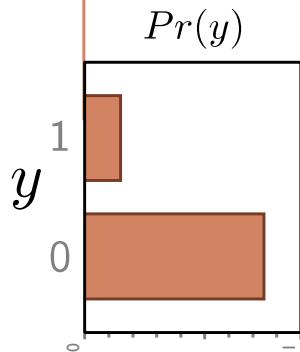
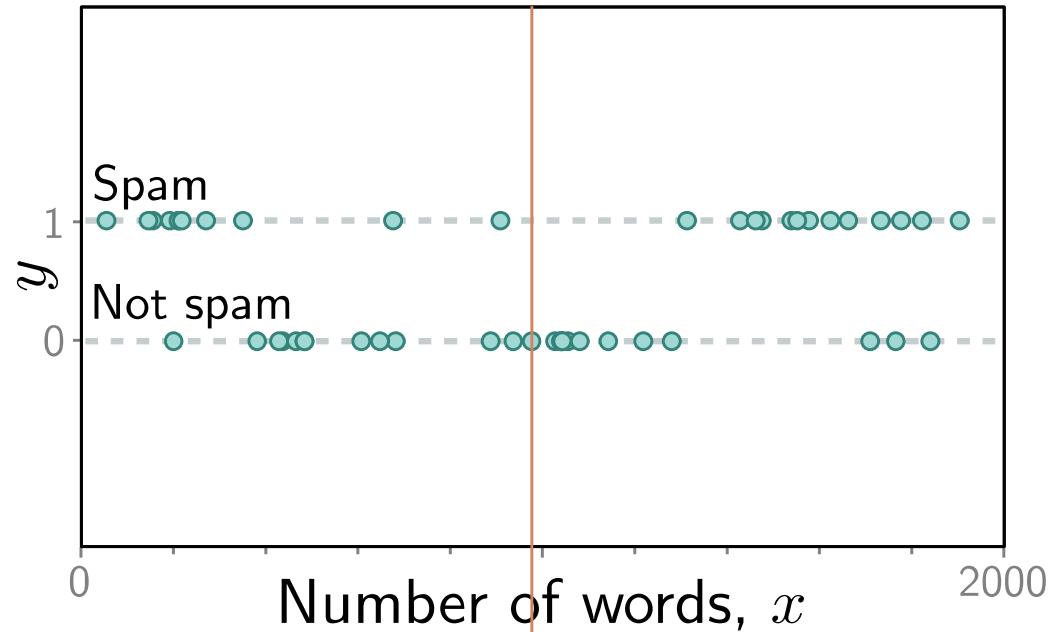


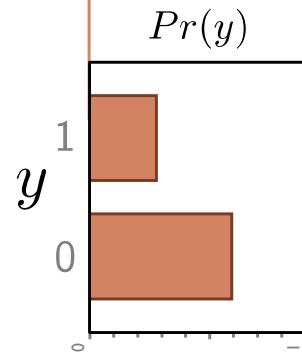
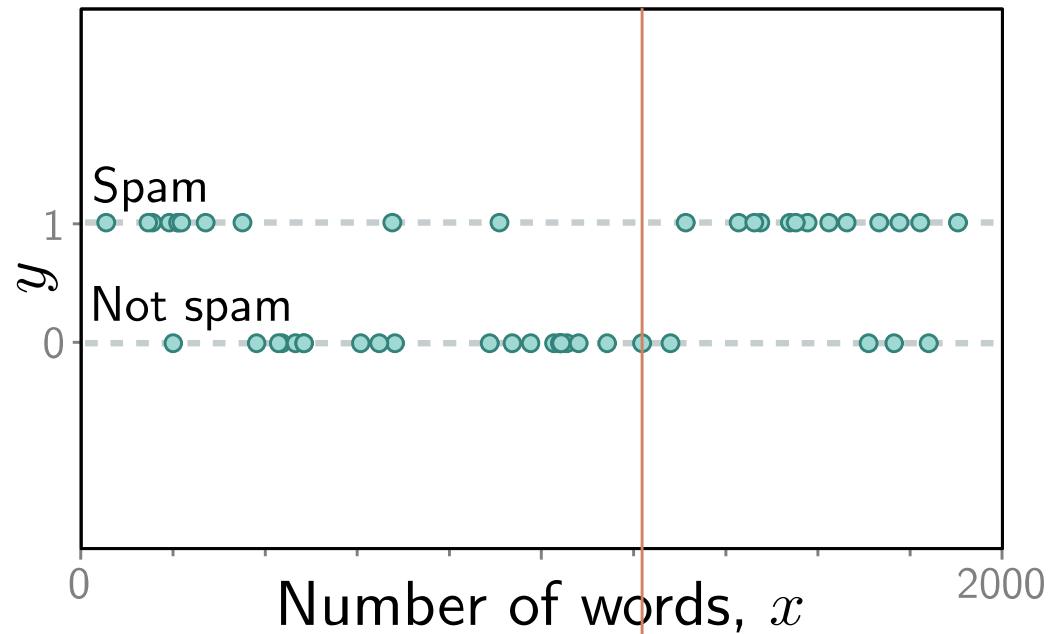


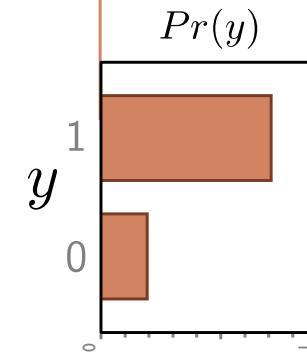
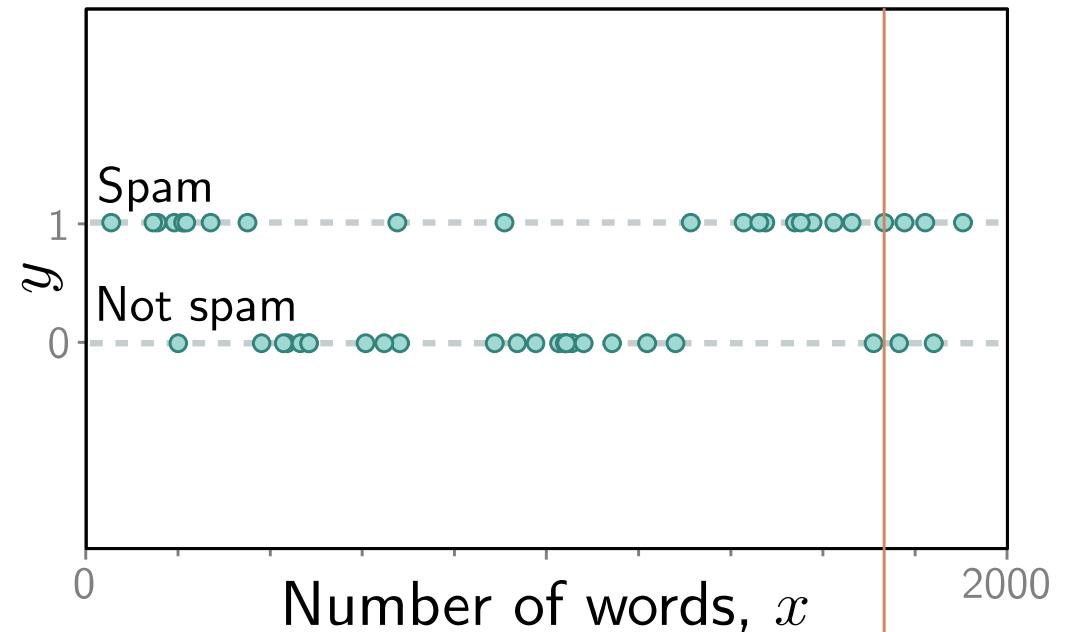


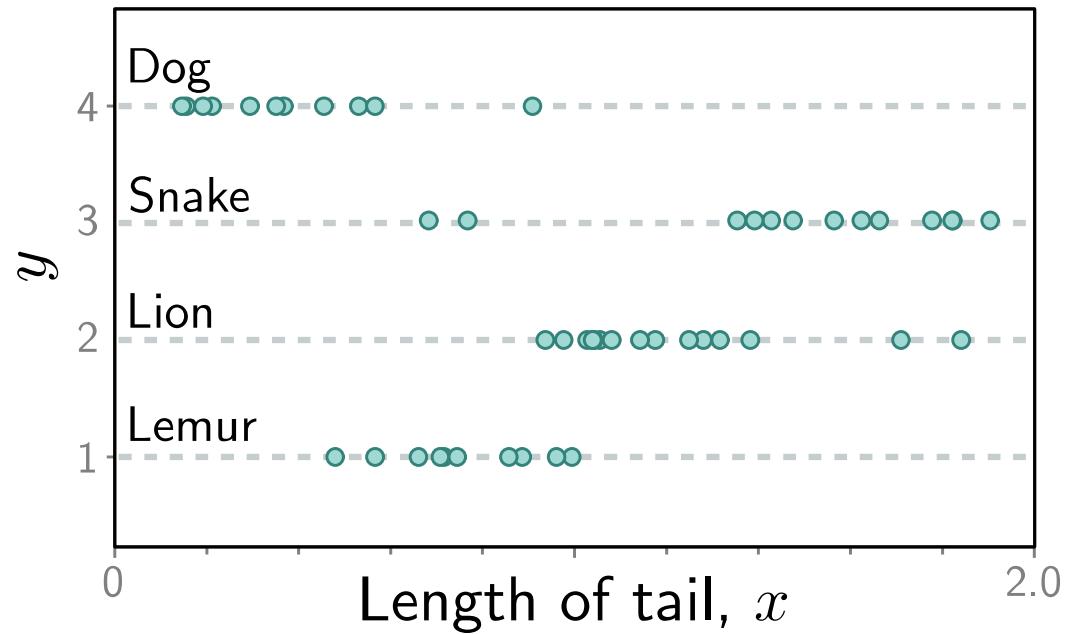


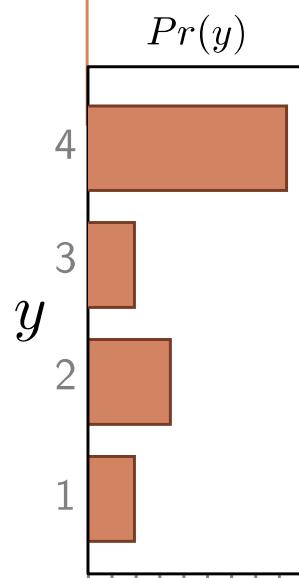
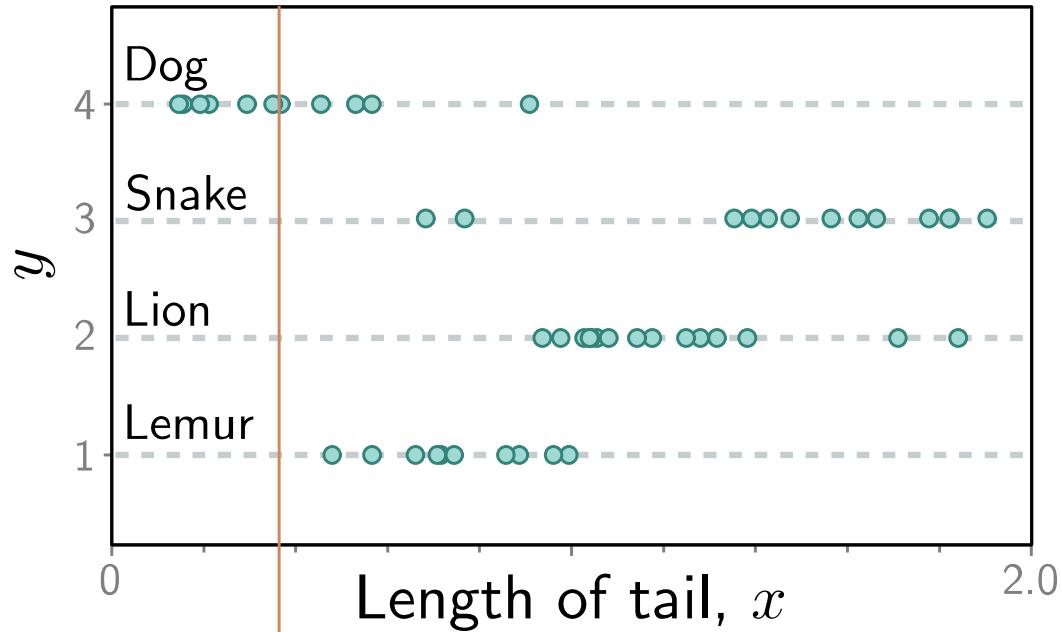


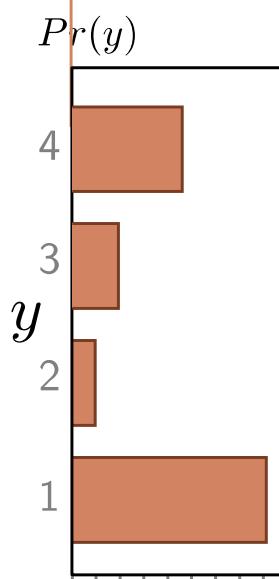
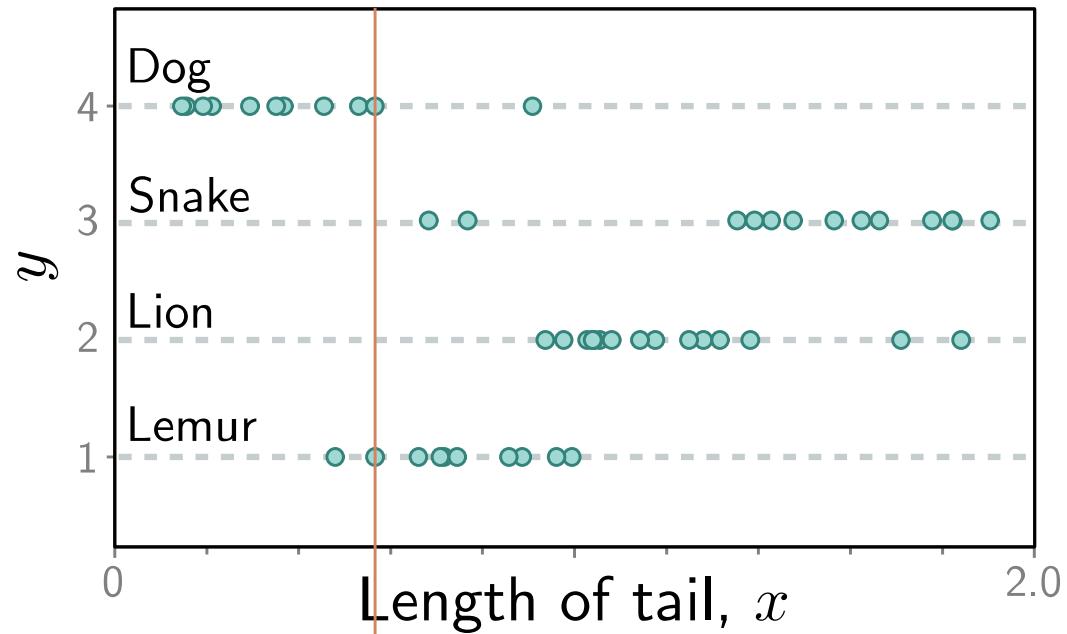


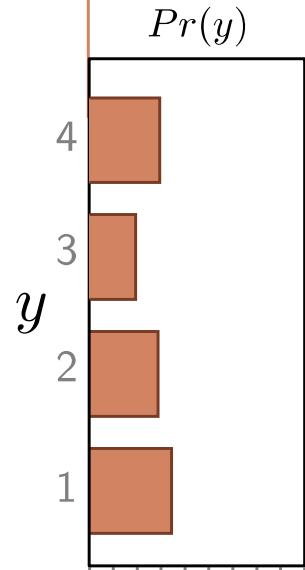
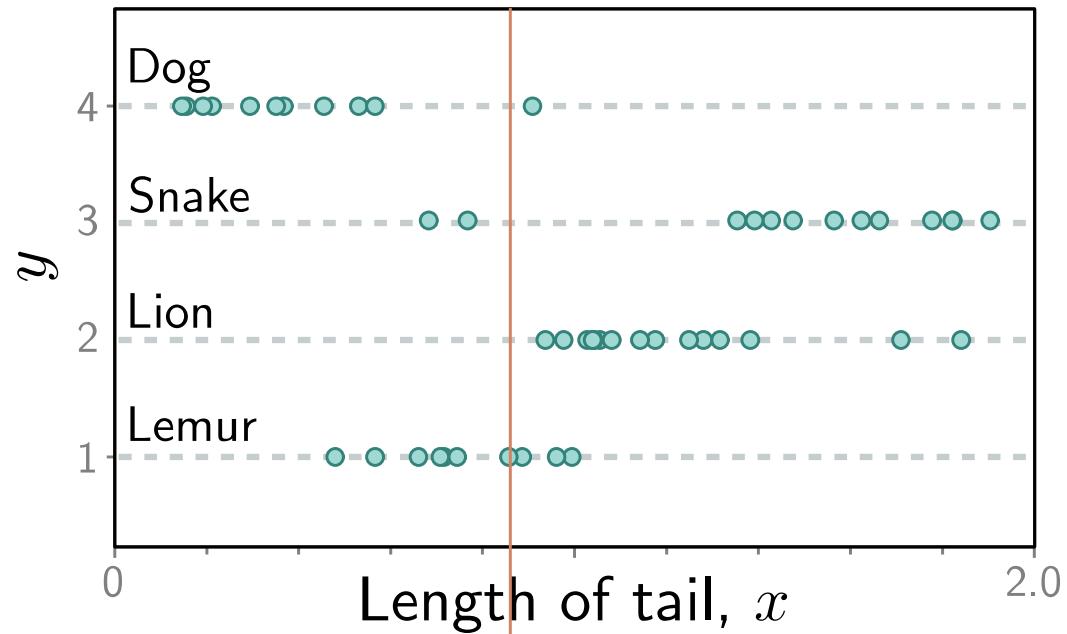


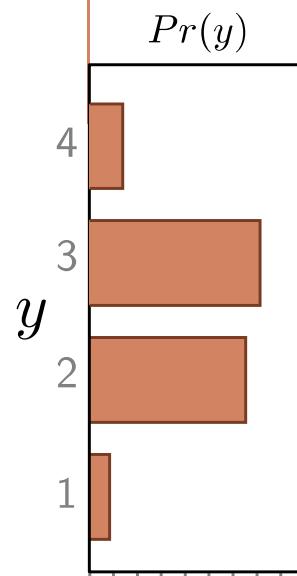
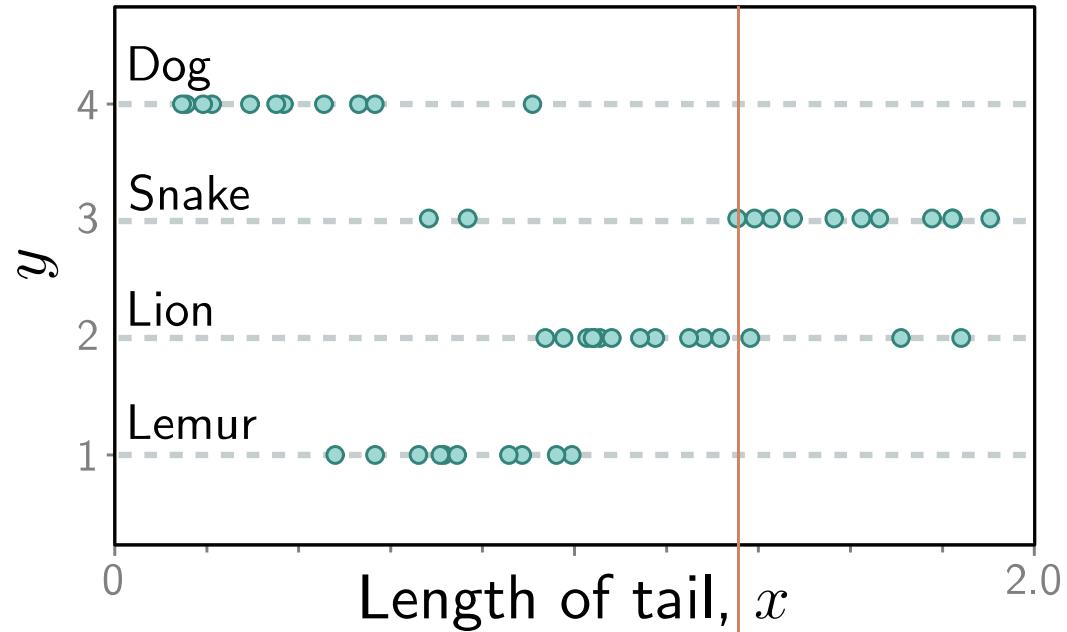


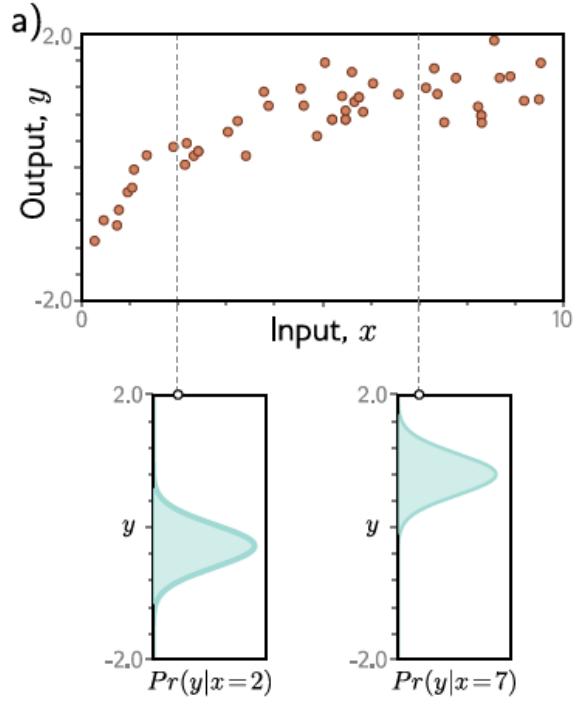


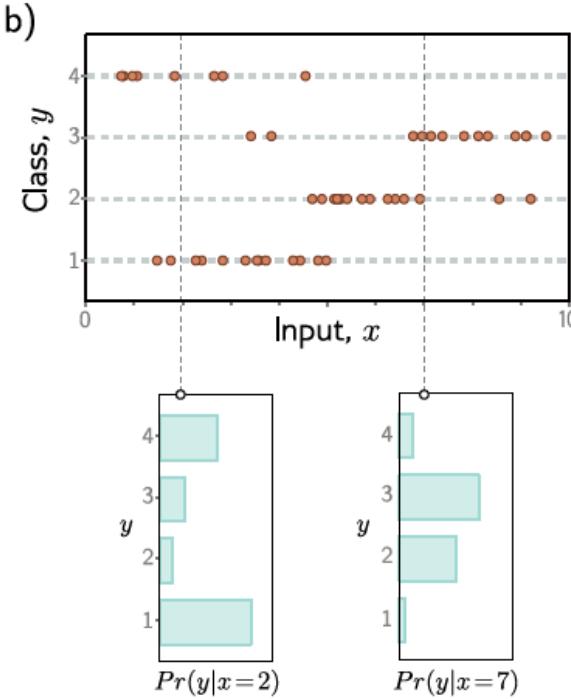
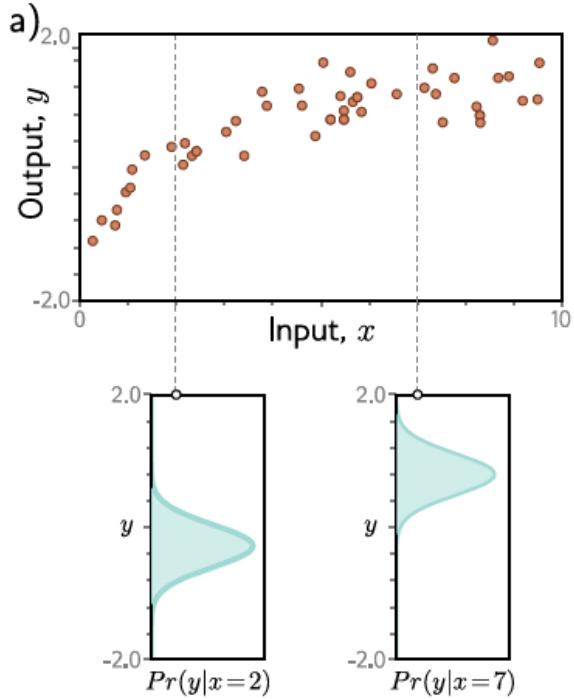


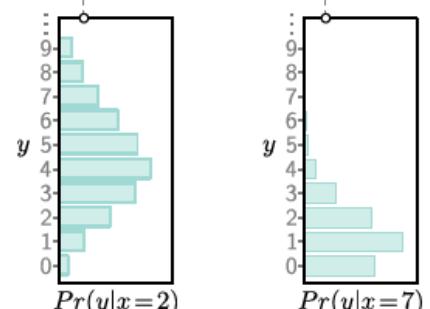
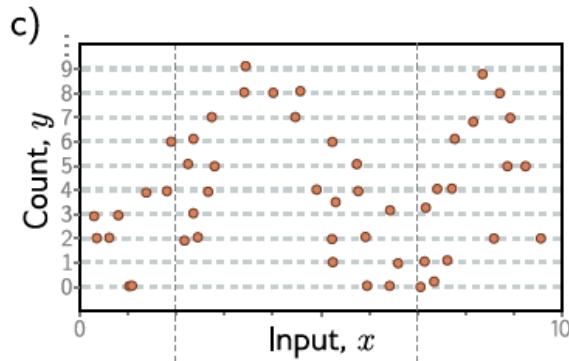
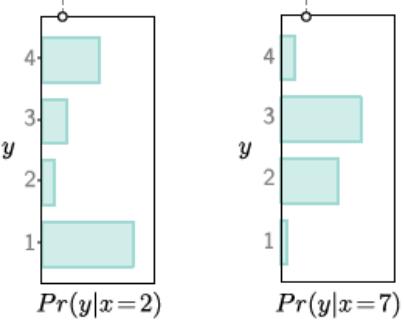
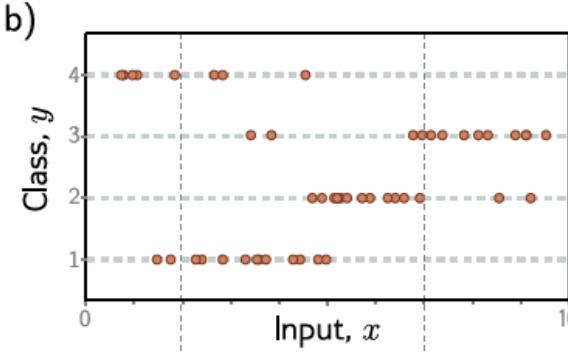
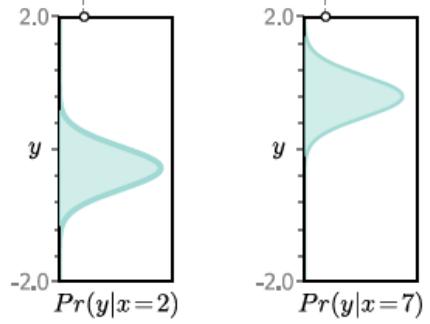
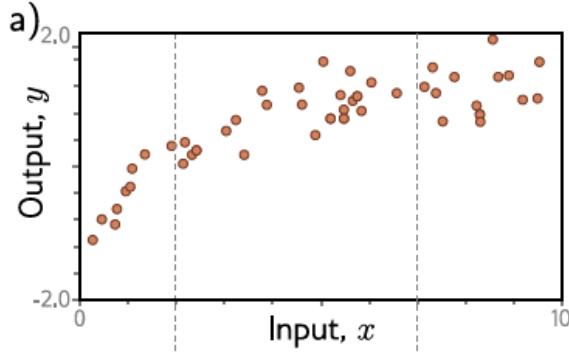


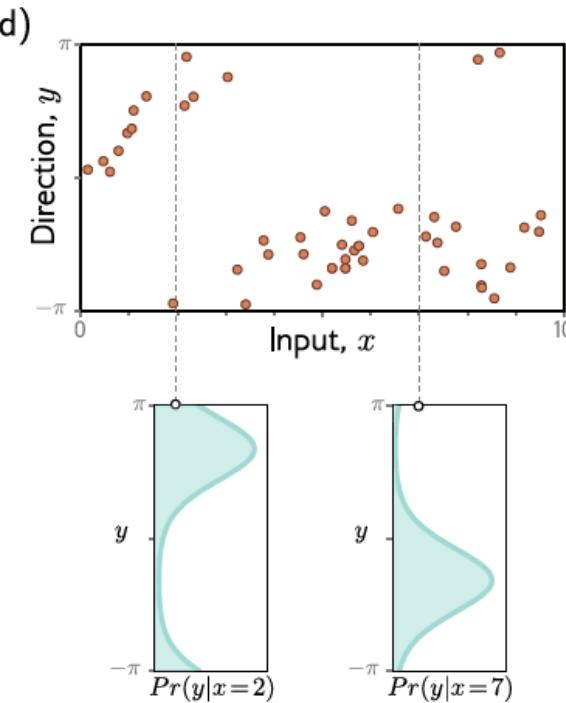
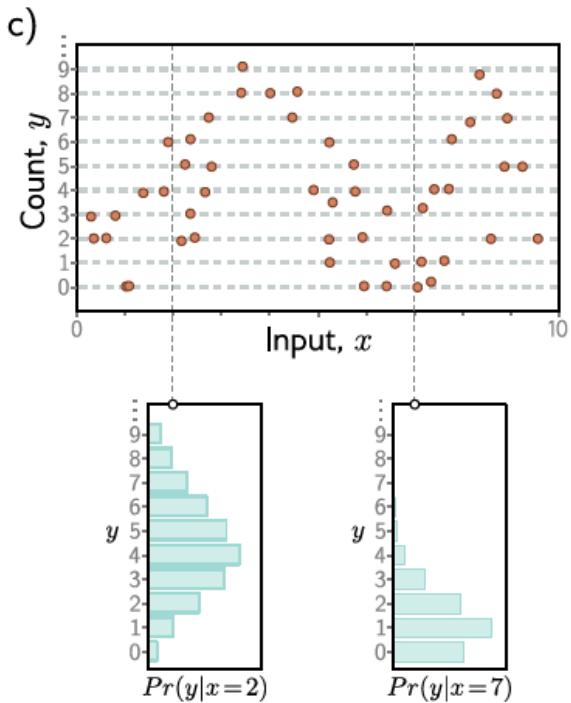
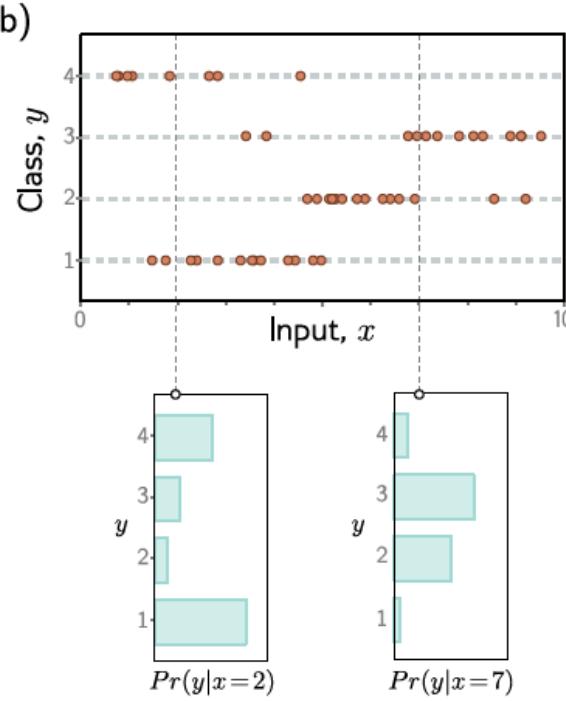
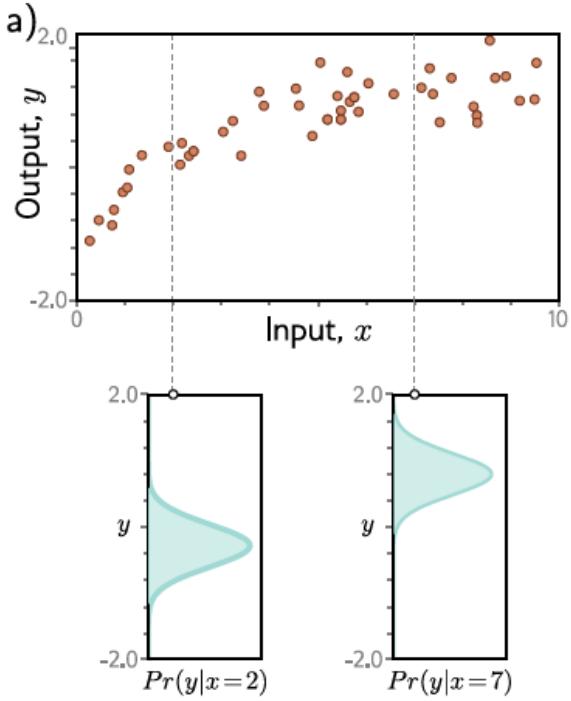












Loss function

- 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:

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



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:

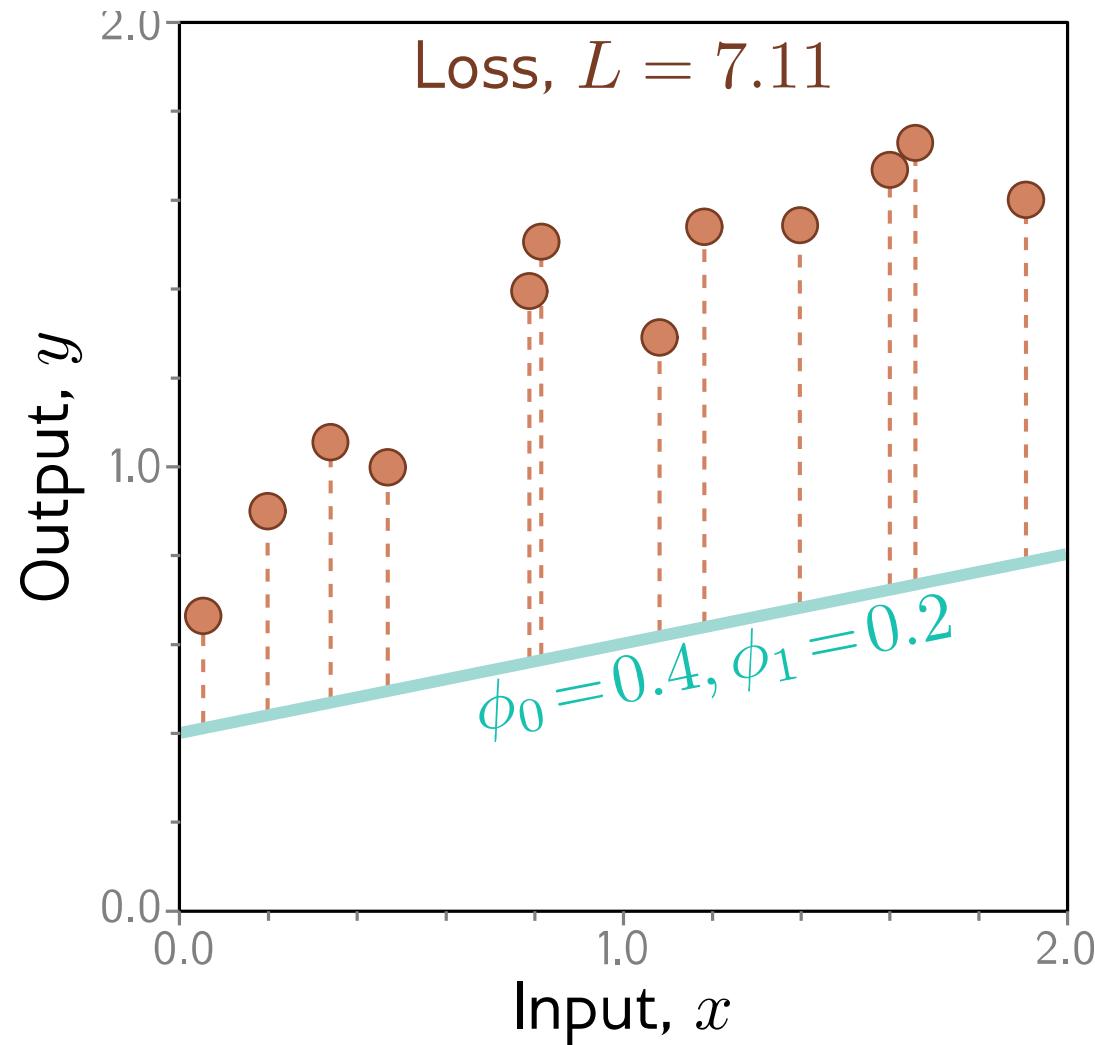
$$L [\phi]$$

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

- Find the parameters that minimize the loss:

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

Example: 1D Linear regression loss function

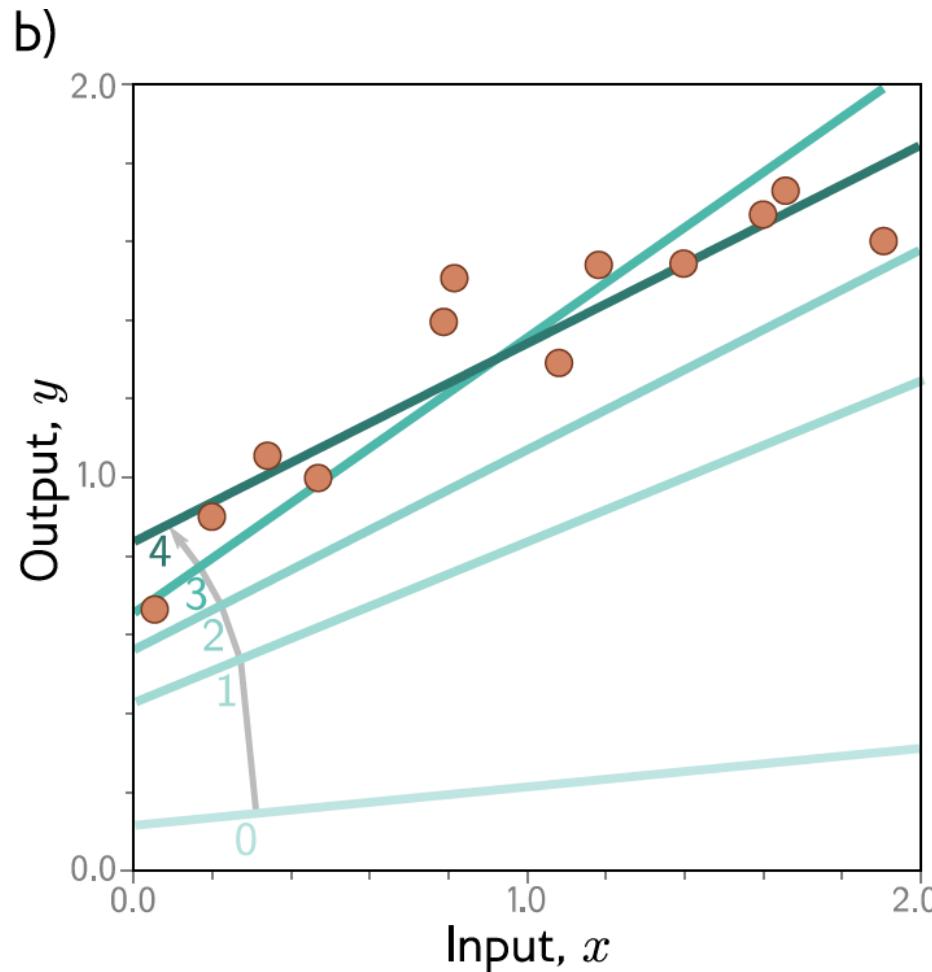
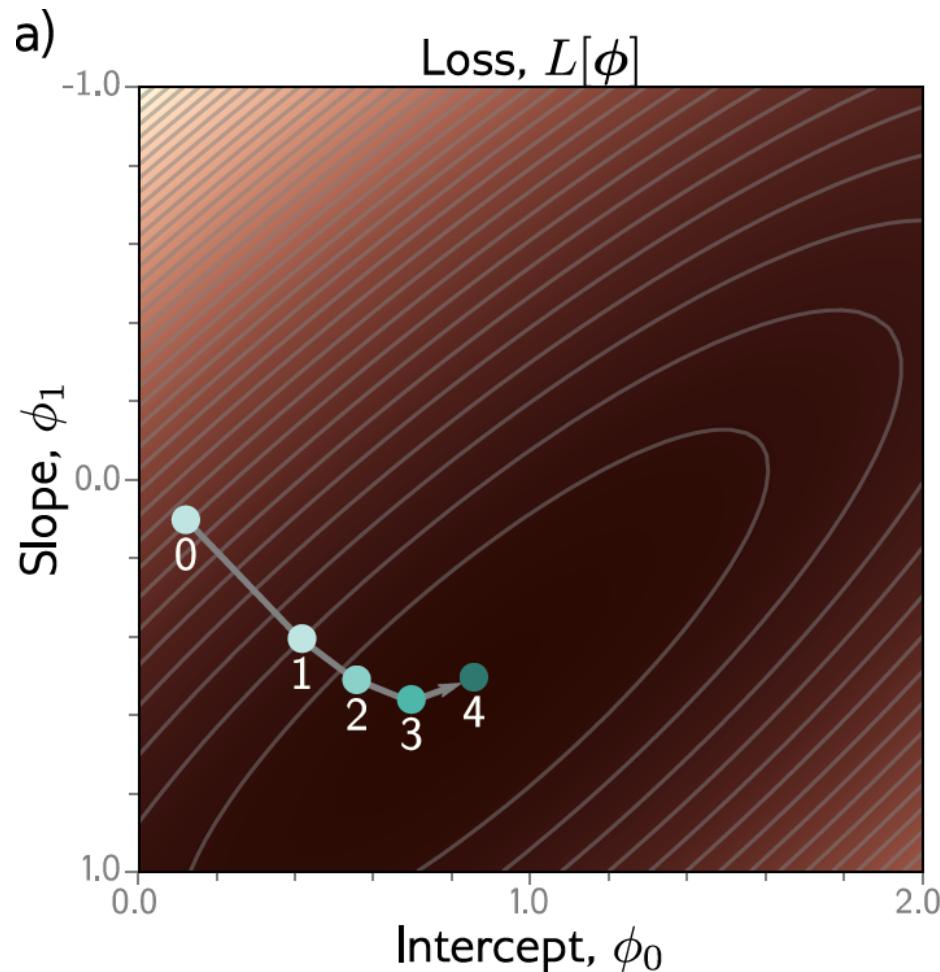


Loss function:

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

“Least squares loss function”

Example: 1D Linear regression training



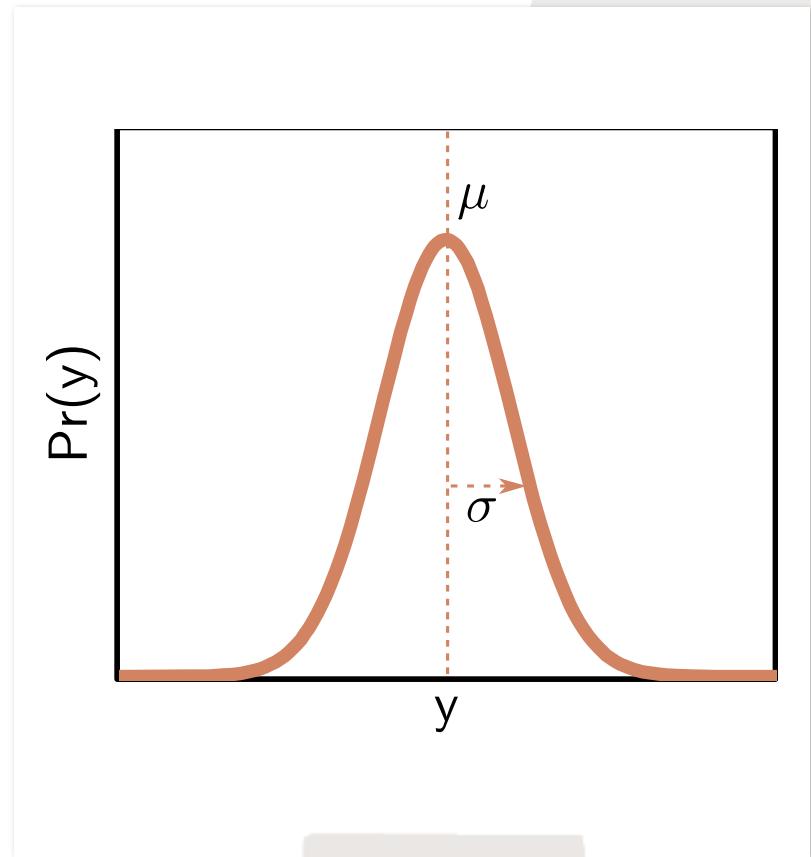
This technique is known as **gradient descent**

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

Some Statistics

- What Is a Probability Distribution?
- A **probability distribution** describes **how likely different values** of a random variable are.
 - Which values are more likely to occur?
 - Which values are less likely?
- The **normal distribution**
 - bell-shaped curve:
 - Most values are near the middle (mean)
 - Fewer values occur far from the mean
 - It is **symmetric** around the center
- If we measure the heights of a large group of people, most people will be **around the average height**, say **170 cm**.
 - Very few people will be extremely short or extremely tall.



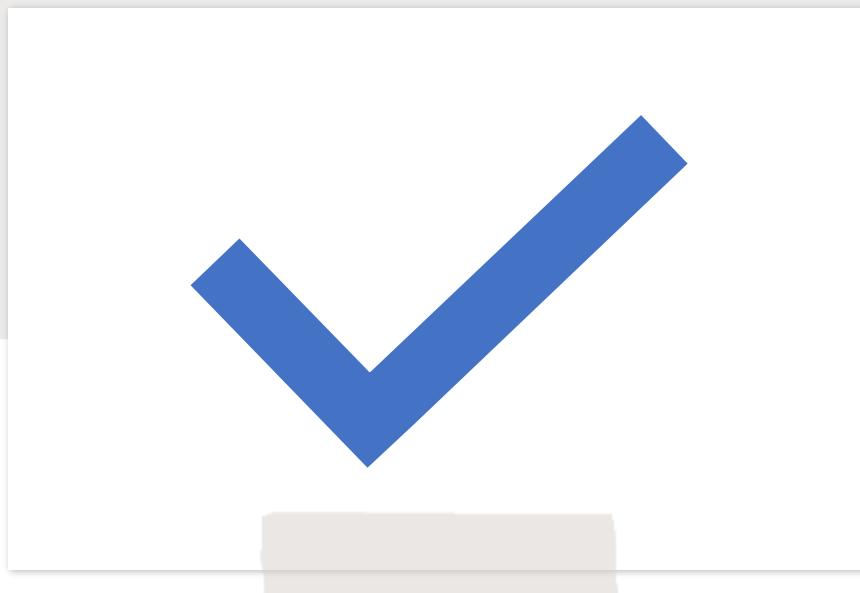
Some Statistics

- The **log-likelihood** tells us **how well the model fits the data** by measuring the total probability of observing the given dataset under the model's predictions.
- **Why Do We Multiply Probabilitie** $L(w) = P(y_1|x_1, w) \times P(y_2|x_2, w) \times \cdots \times P(y_N|x_N, w)$
- Each data point has an **independent probability** under the model.
- Suppose we are predicting **whether an email is spam** y given some features x .
- Our model estimates a probability for each email. $P(y_i|x_i, w)$
- Since each email is **independent**, the probability of observing the entire dataset is the **product** of all individual probabilities:

How to construct loss functions

Model predicts output y given
input x

How to construct loss functions



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

How to construct loss functions

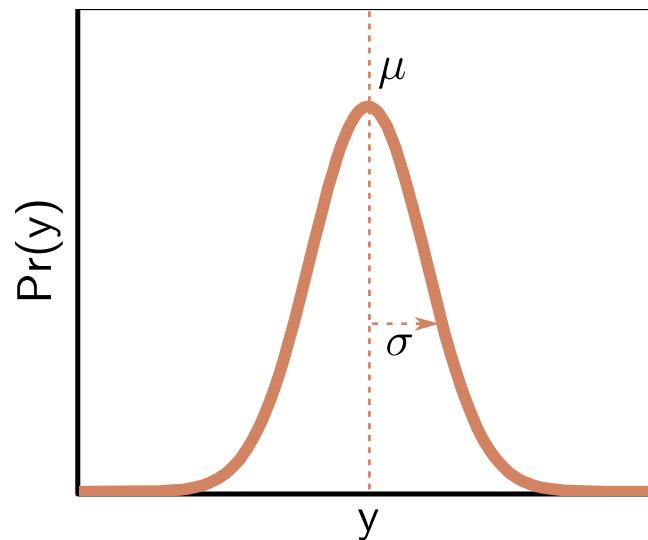
$$Pr(y|x)$$

- Model predicts output y given input x
- Model predicts a conditional probability distribution:
over outputs y given inputs x .
- Loss function aims to make the outputs have high probability
- Why prob not single value?
 - Uncertainty in Predictions (weight is 170 ± 5) -> learn the **distribution** of possible weights.
 - Can derive the loss function

How can a model predict a probability distribution?

1. Pick a known distribution (e.g., normal distribution) to model output y with parameters θ

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



2. Use model to predict parameters θ of probability distribution

Maximum likelihood criterion

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

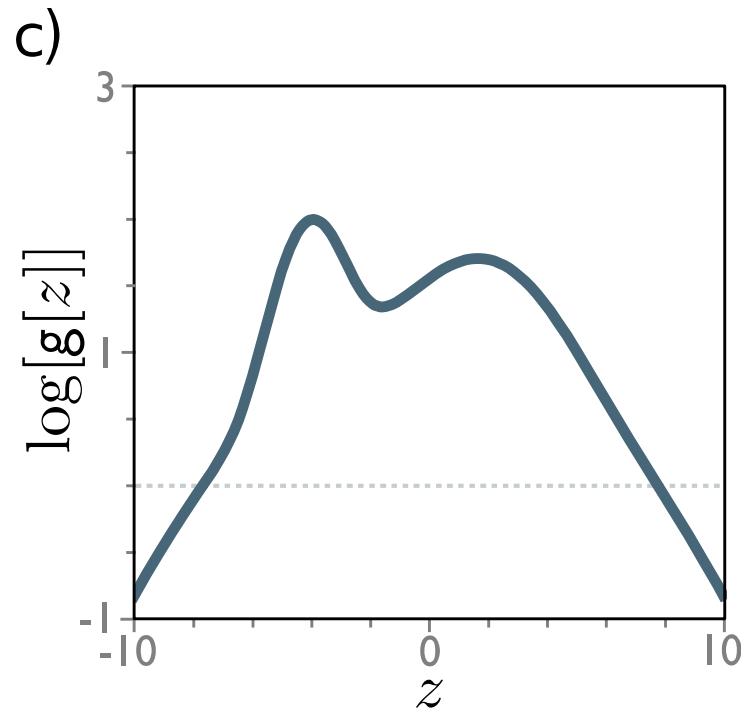
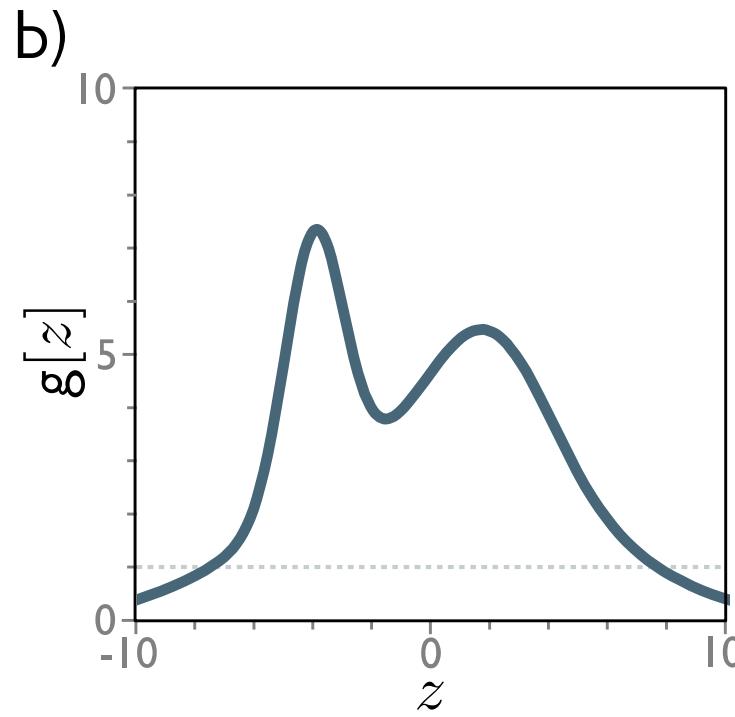
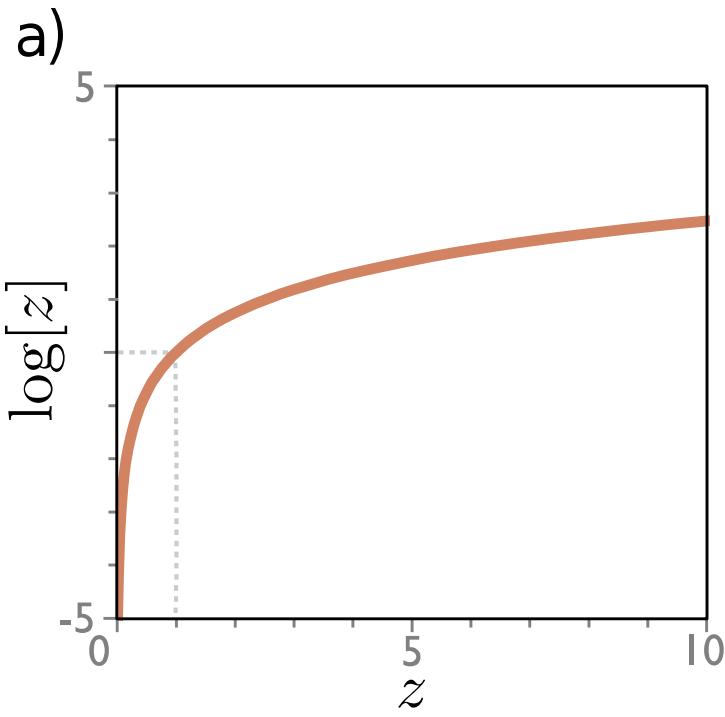
When we consider this probability as a function of the parameters ϕ , we call it a **likelihood**.

Problem:

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

- 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

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

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

Minimizing negative log likelihood

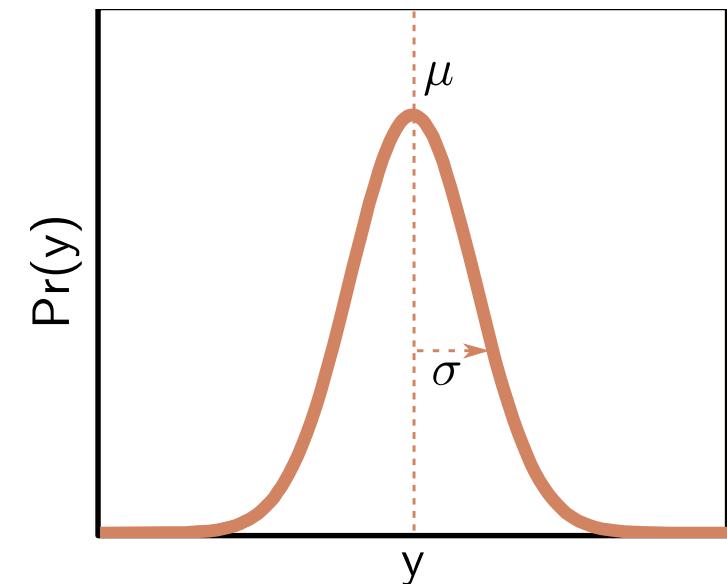
- By convention, we minimize things (i.e., a loss)

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

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)

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



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

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

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

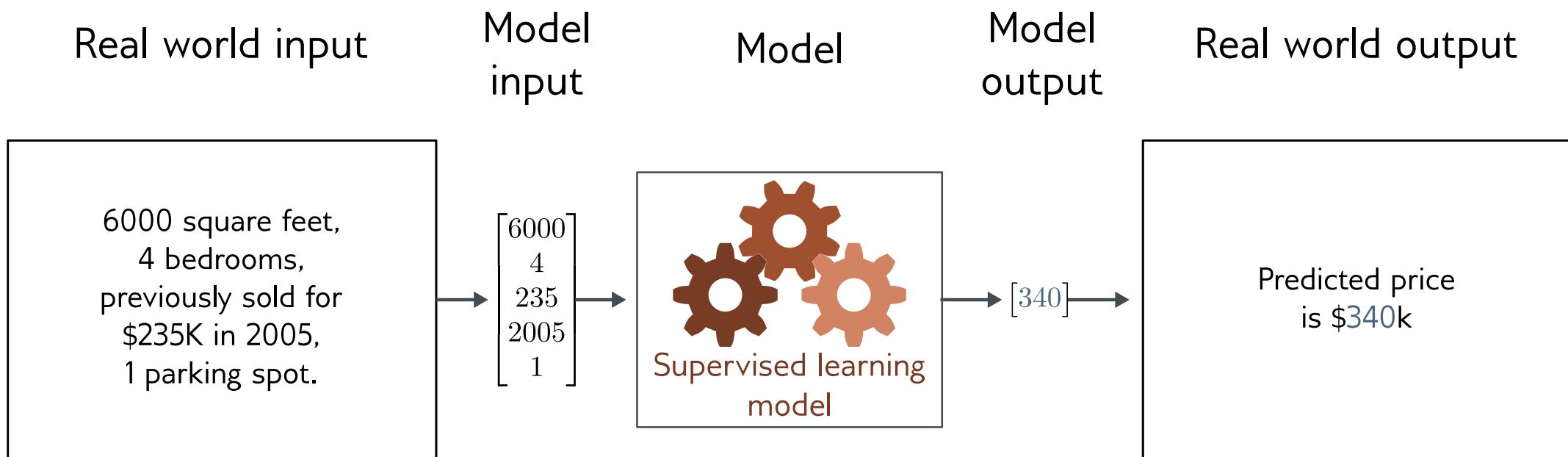
$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} [L[\boldsymbol{\phi}]] = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[- \sum_{i=1}^I \log \left[Pr(\mathbf{y}_i | \mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]) \right] \right]. \quad (5.7)$$

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

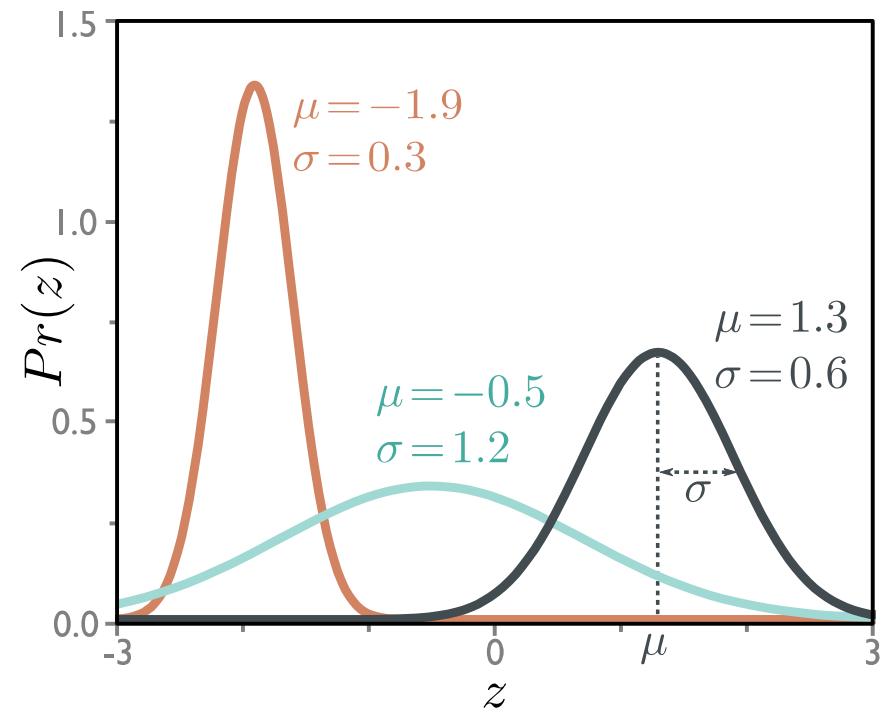


Example 1: univariate regression

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

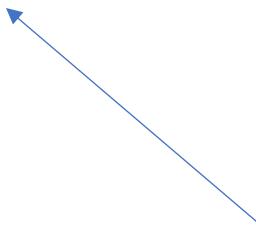
3. To train the model, find the network parameters $\hat{\phi}$ that minimize the negative log-likelihood loss function over the training dataset pairs $\{\mathbf{x}_i, \mathbf{y}_i\}$:

$$\begin{aligned} L[\phi] &= -\sum_{i=1}^I \log [Pr(y_i | f[\mathbf{x}_i, \phi], \sigma^2)] \\ &= -\sum_{i=1}^I \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(y_i - f[\mathbf{x}_i, \phi])^2}{2\sigma^2} \right] \right] \end{aligned}$$

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

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

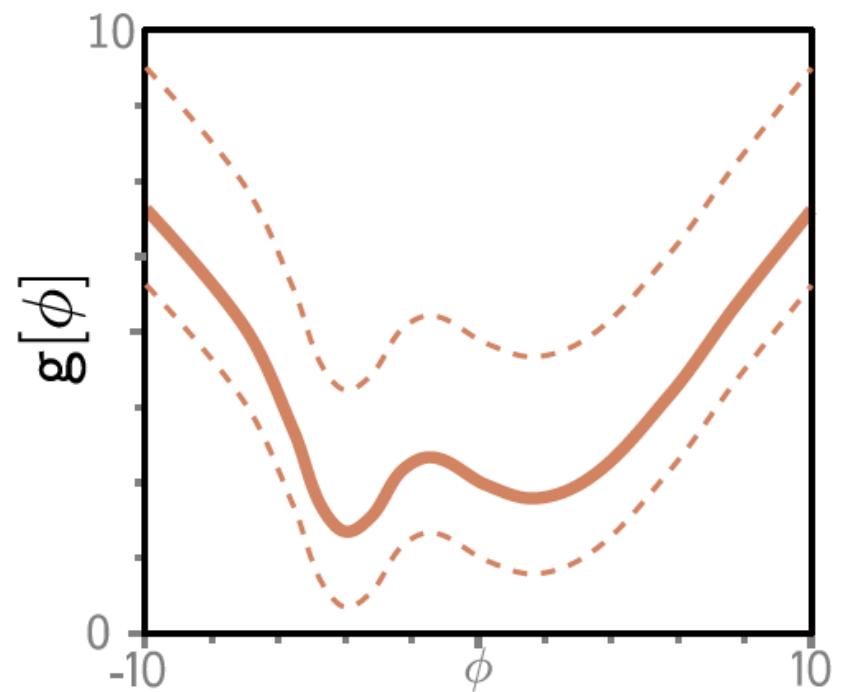
$$= \operatorname{argmin}_{\phi} \left[- \sum_{i=1}^I \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \right] + \log \left[\exp \left[-\frac{(y_i - f[\mathbf{x}_i, \phi])^2}{2\sigma^2} \right] \right] \right]$$



$$\log[a \cdot b] = \log[a] + \log[b]$$

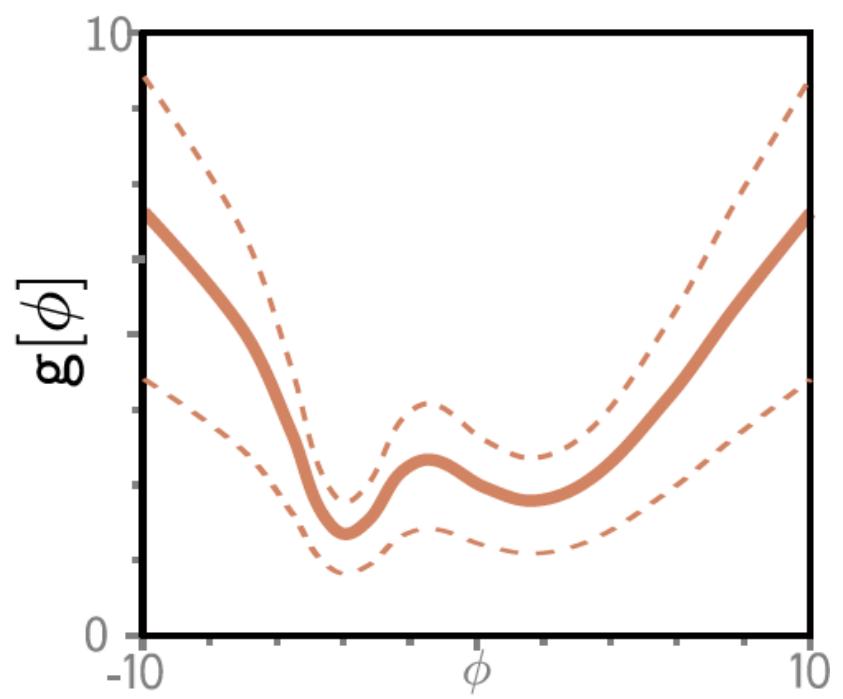
$$\begin{aligned}
\hat{\phi} &= \operatorname{argmin}_{\phi} \left[- \sum_{i=1}^I \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(y_i - f[\mathbf{x}_i, \phi])^2}{2\sigma^2} \right] \right] \right] \\
&= \operatorname{argmin}_{\phi} \left[- \sum_{i=1}^I \log \left[\frac{1}{\sqrt{2\pi\sigma^2}}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

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



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

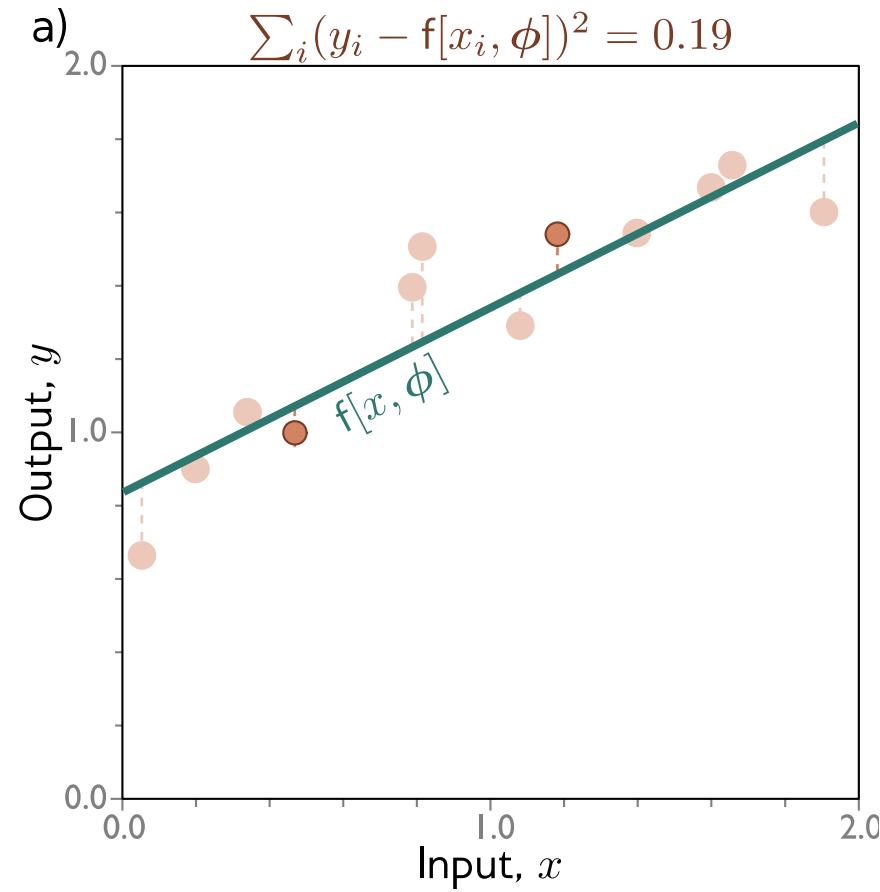
$$\begin{aligned}
\hat{\phi} &= \operatorname{argmin}_{\phi} \left[- \sum_{i=1}^I \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(y_i - f[\mathbf{x}_i, \phi])^2}{2\sigma^2} \right] \right] \right] \\
&= \operatorname{argmin}_{\phi} \left[- \sum_{i=1}^I \log \left[\frac{1}{\sqrt{2\pi\sigma^2}}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



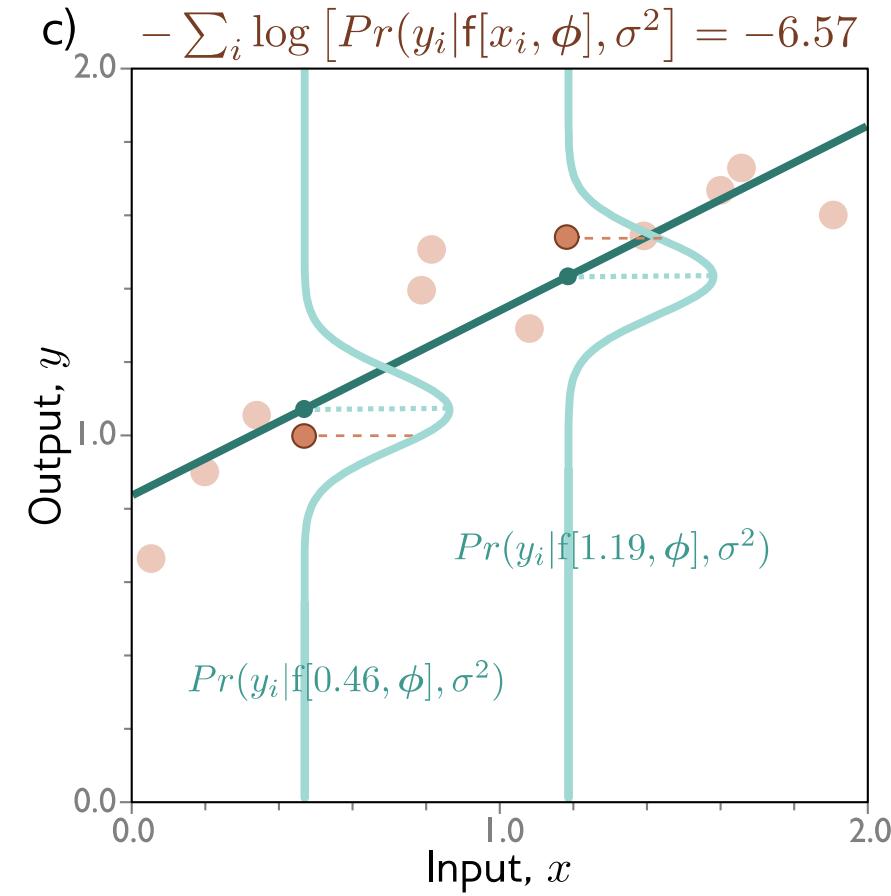
$$\begin{aligned}
\hat{\phi} &= \operatorname{argmin}_{\phi} \left[- \sum_{i=1}^I \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(y_i - f[\mathbf{x}_i, \phi])^2}{2\sigma^2} \right] \right] \right] \\
&= \operatorname{argmin}_{\phi} \left[- \sum_{i=1}^I \log \left[\frac{1}{\sqrt{2\pi\sigma^2}}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

Least squares!

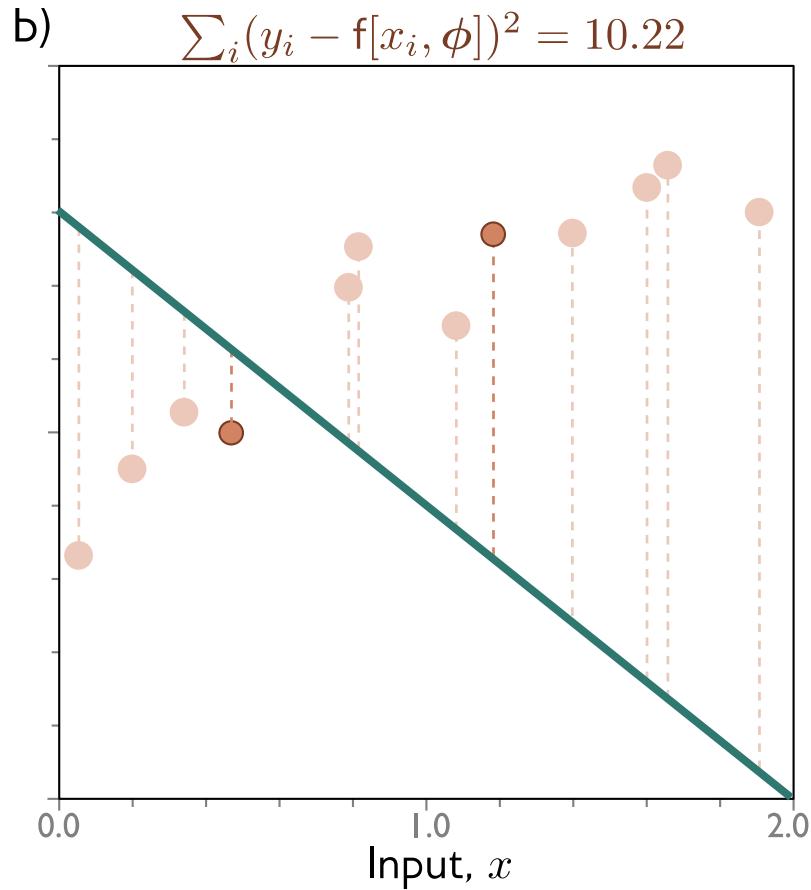
Least squares



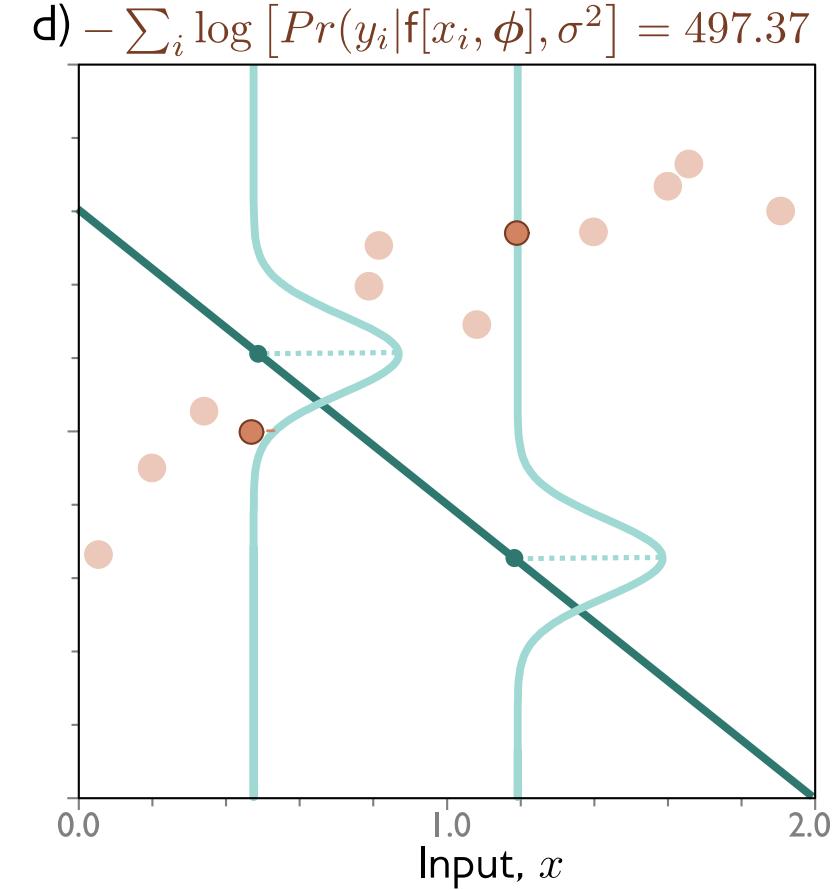
Maximum likelihood



Least squares



Maximum likelihood

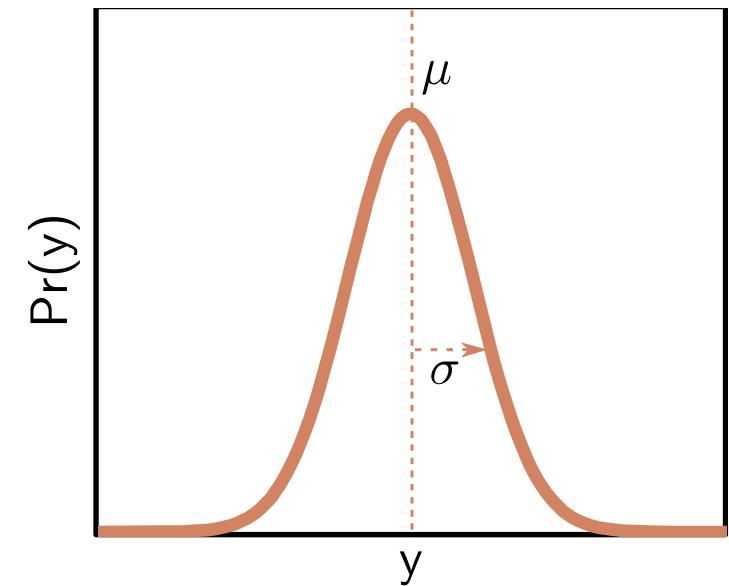


Example 1: univariate regression

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

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

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



Estimating variance

- Perhaps surprisingly, the variance term disappeared:

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

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

- But we could learn it:

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

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:

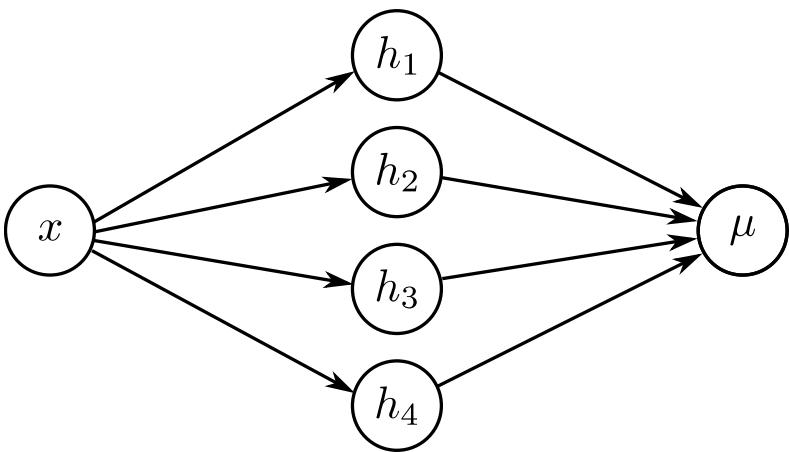
$$\mu = f_1[\mathbf{x}, \phi]$$

$$\sigma^2 = f_2[\mathbf{x}, \phi]^2$$

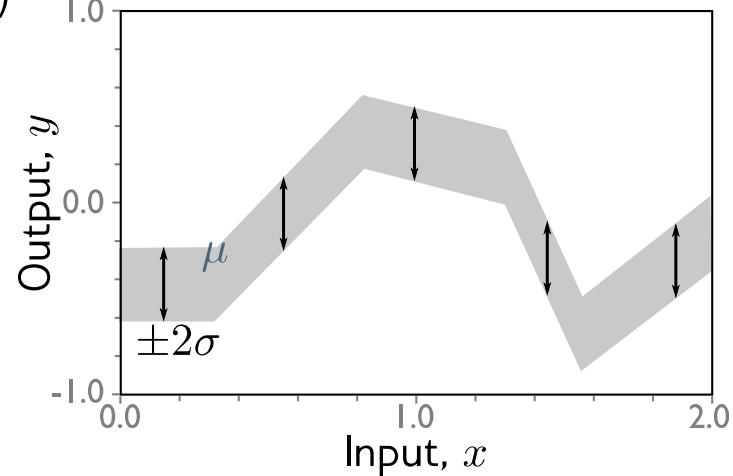
$$\hat{\phi} = \operatorname{argmin}_{\phi} \left[- \sum_{i=1}^I \log \left[\frac{1}{\sqrt{2\pi f_2[\mathbf{x}_i, \phi]^2}} \right] - \frac{(y_i - f_1[\mathbf{x}_i, \phi])^2}{2f_2[\mathbf{x}_i, \phi]^2} \right]$$

Heteroscedastic regression

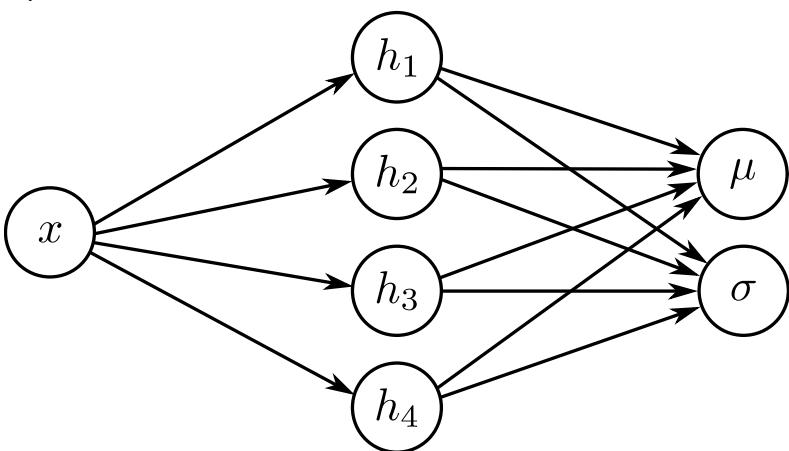
a)



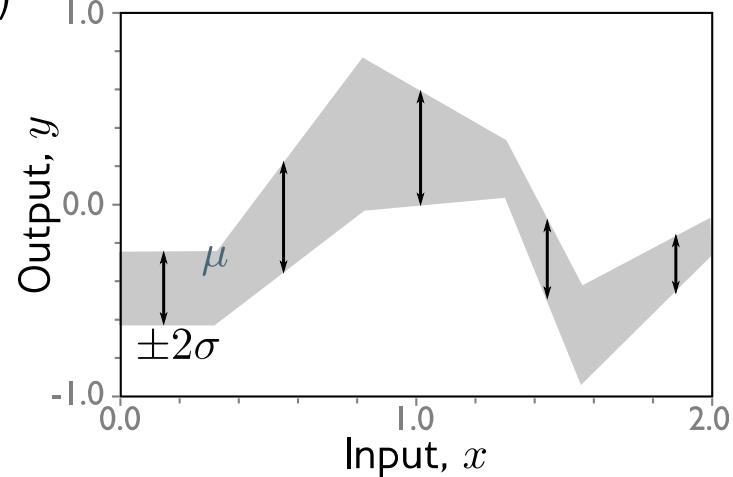
b)



c)



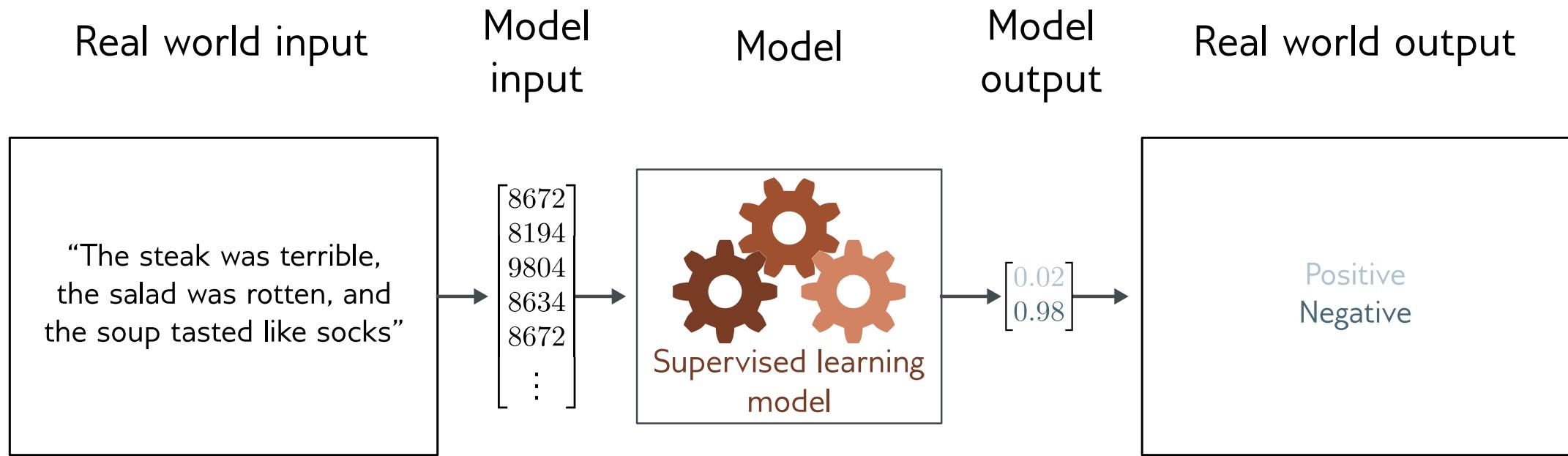
d)



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

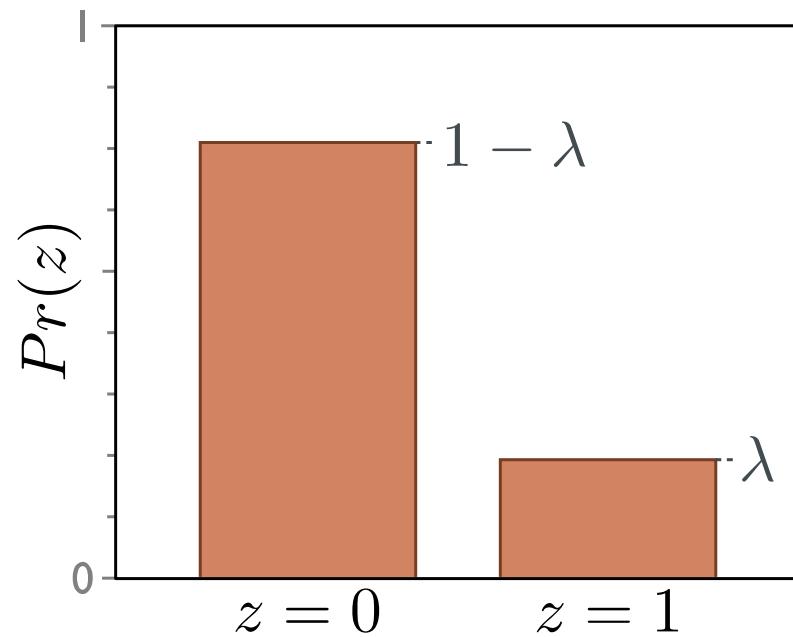
Example 2: binary classification

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

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

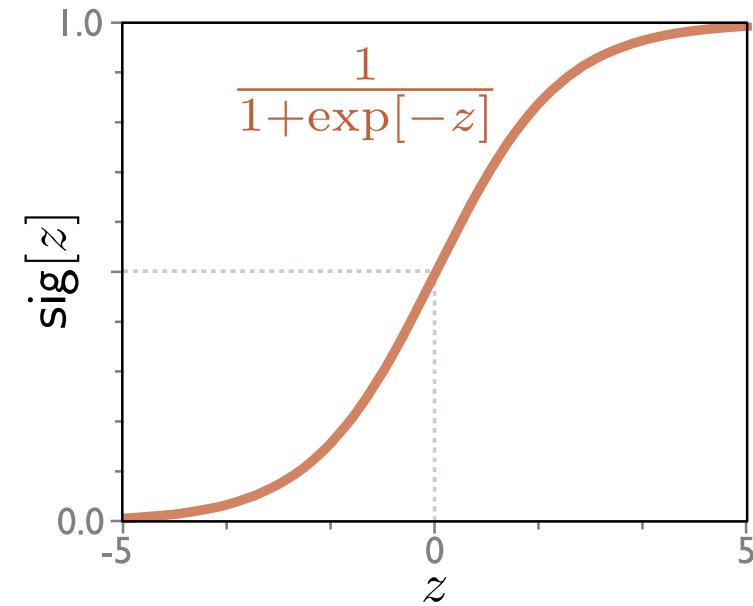
Problem:

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

Solution:

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

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



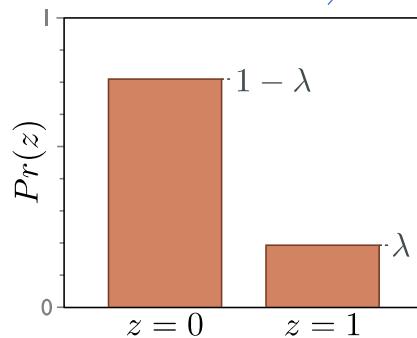
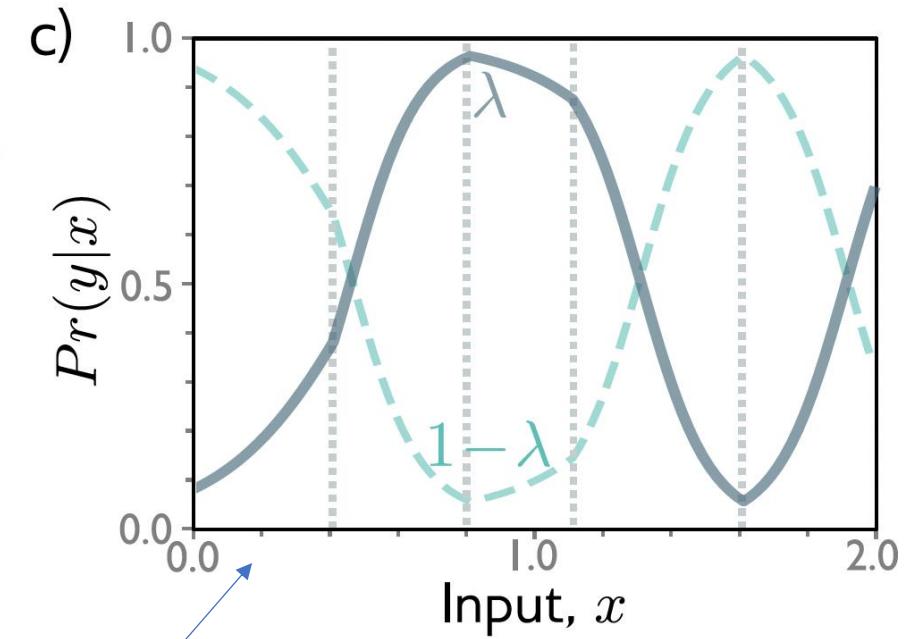
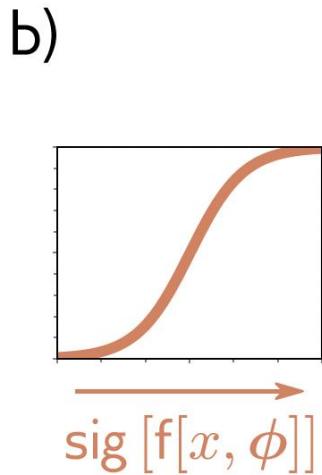
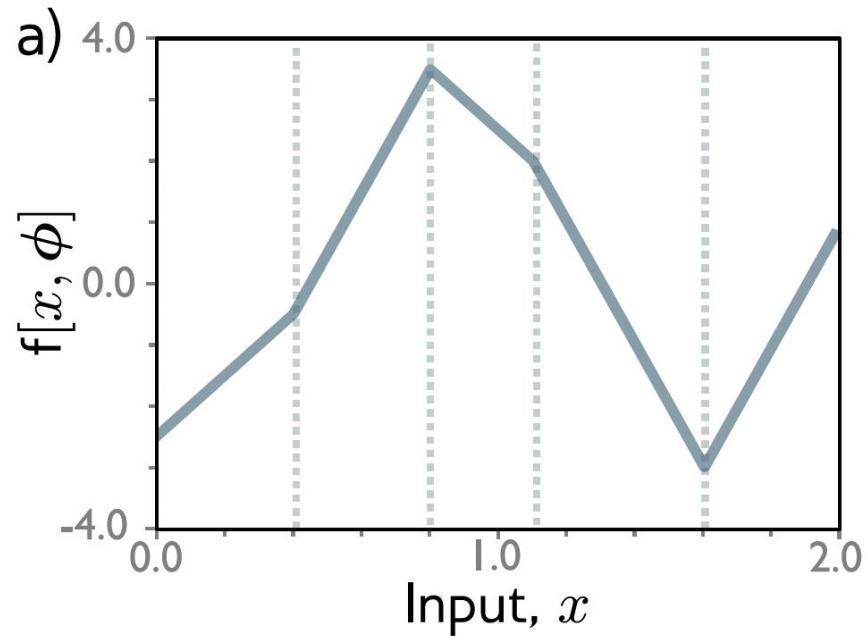
Example 2: binary classification

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

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

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

Example 2: binary classification



Example 2: binary classification

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

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

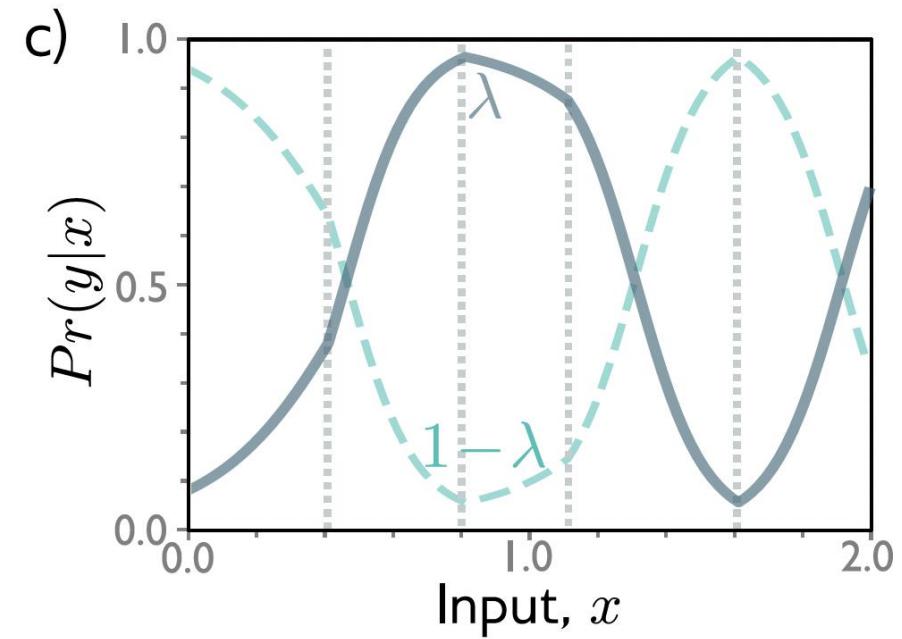
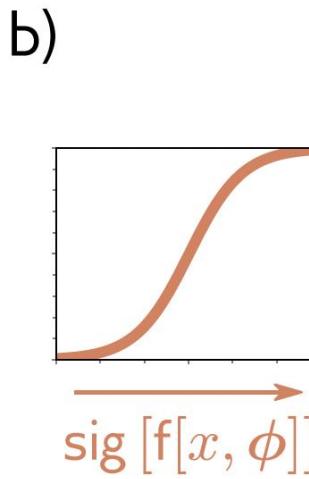
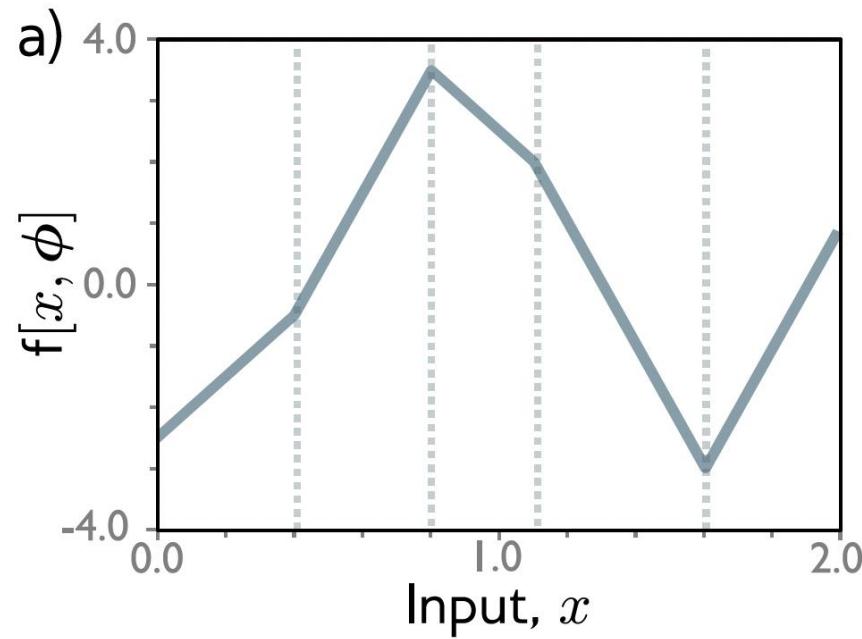
$$Pr(y|\mathbf{x}) = (1 - \operatorname{sig}[\mathbf{f}[\mathbf{x}|\phi]])^{1-y} \cdot \operatorname{sig}[\mathbf{f}[\mathbf{x}|\phi]]^y$$

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

Binary cross-entropy loss

Example 2: binary classification

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

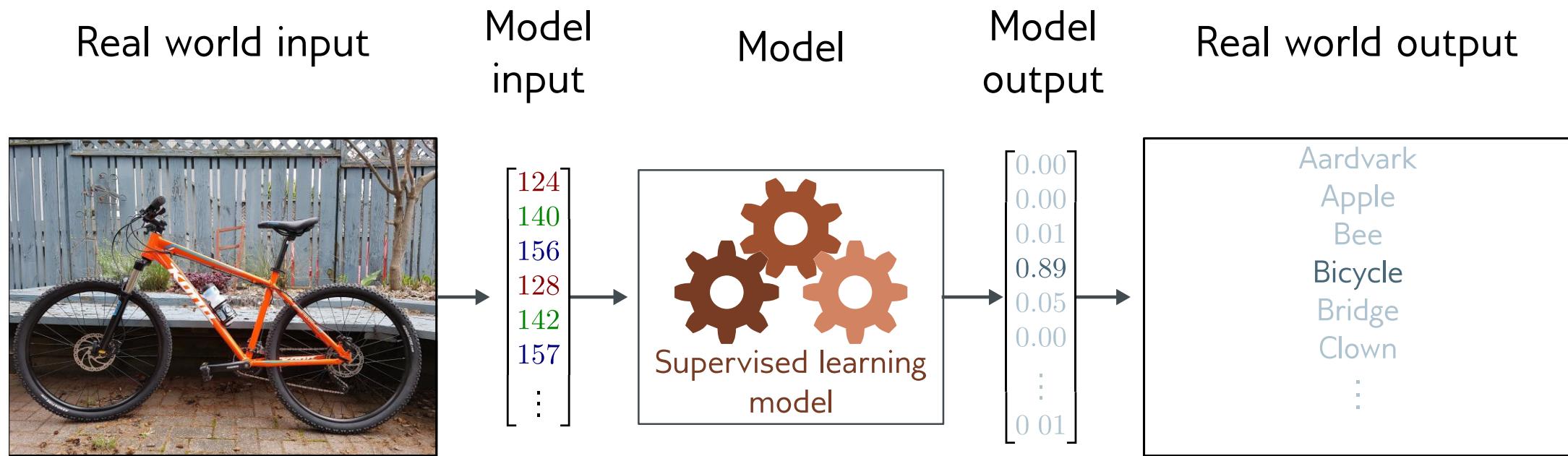


Choose $y=1$ where λ is greater than 0.5, otherwise 0

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



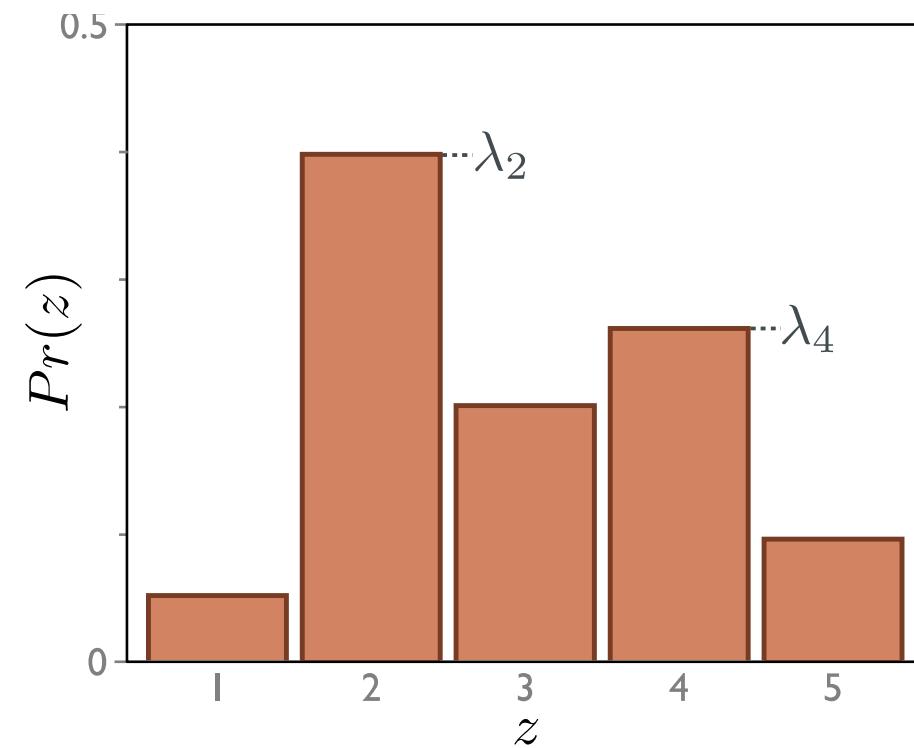
Goal: predict which of K classes $y \in \{1, 2, \dots, K\}$ the input x belongs to

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: $y \in \{1, 2, \dots, K\}$
- Categorical distribution
- K parameters $\lambda_k \in [0, 1]$
- Sum of all parameters = 1

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



Example 3: multiclass classification

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

Problem:

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

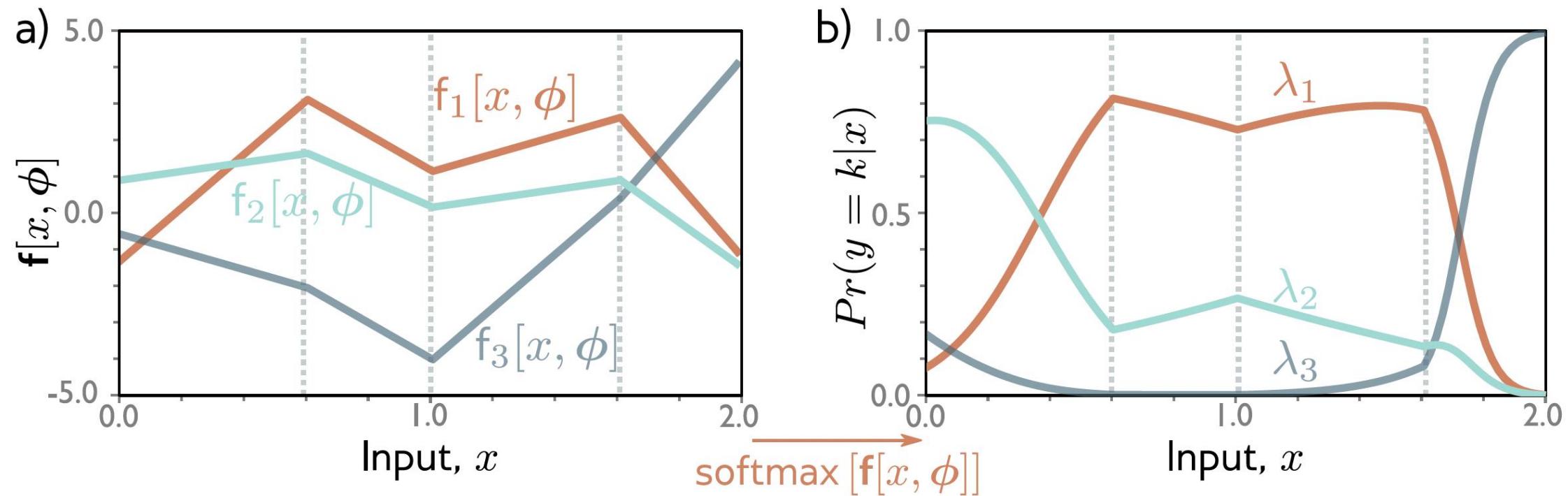
Solution:

- Pass through function that maps “anything” to $[0,1]$, sum to one

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

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

Example 3: multiclass classification



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

Example 3: multiclass classification

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

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

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

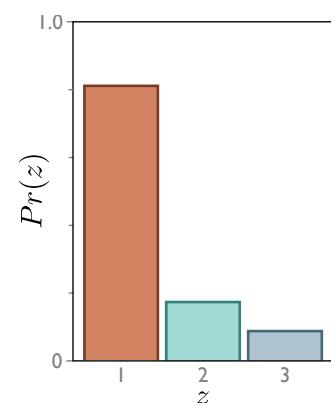
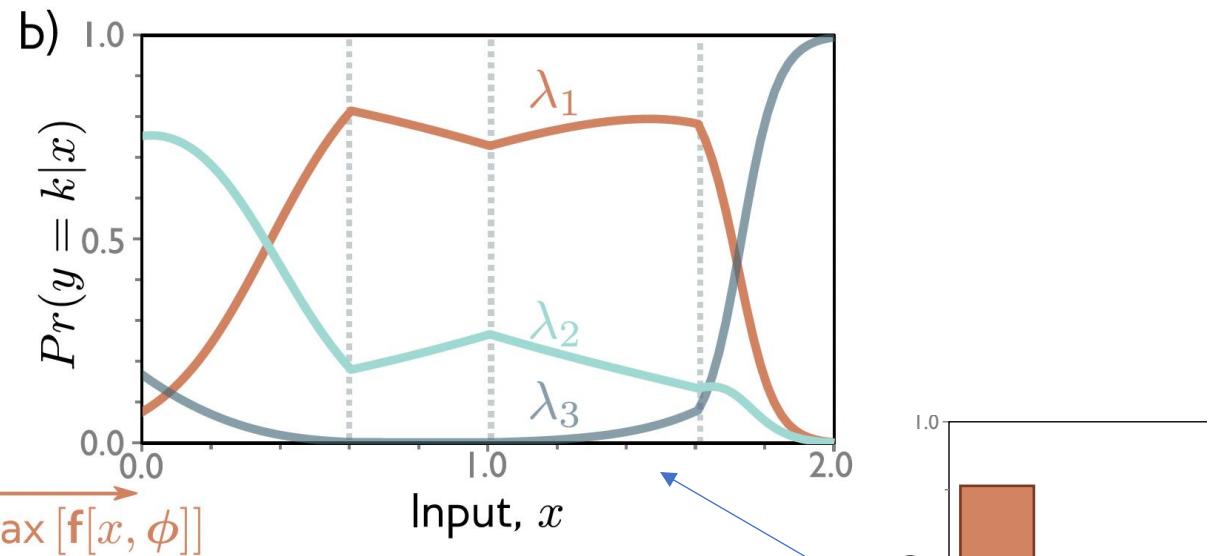
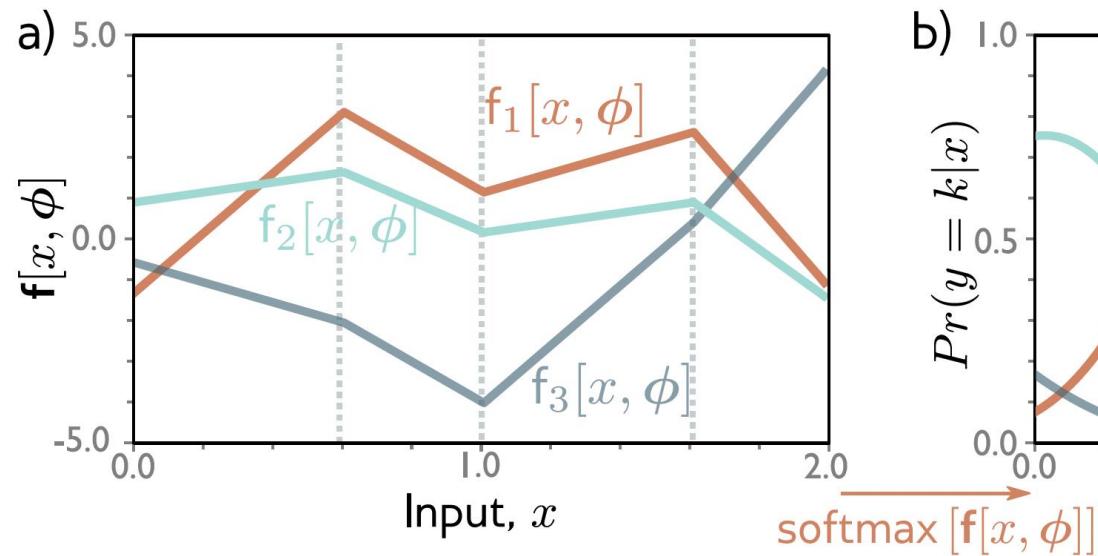
$$\text{softmax}_k[\mathbf{z}] = \frac{\exp[z_k]}{\sum_{k'=1}^K \exp[z_{k'}]}$$

$$= - \sum_{i=1}^I f_{y_i} [\mathbf{x}_i, \phi] - \log \left[\sum_{k=1}^K \exp [f_k [\mathbf{x}_i, \phi]] \right]$$

Multiclass cross-entropy loss

Example 3: multiclass classification

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



Choose the class with the largest probability

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	$y \in \{1, 2, \dots, K\}$	categorical	multiclass classification
univariate, discrete, bounded below	$y \in [0, 1, 2, 3, \dots]$	Poisson	predicting event counts
multivariate, discrete, permutation	$\mathbf{y} \in \text{Perm}[1, 2, \dots, K]$	Plackett-Luce	ranking

Figure 5.11 Distributions for loss functions for different prediction types.

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

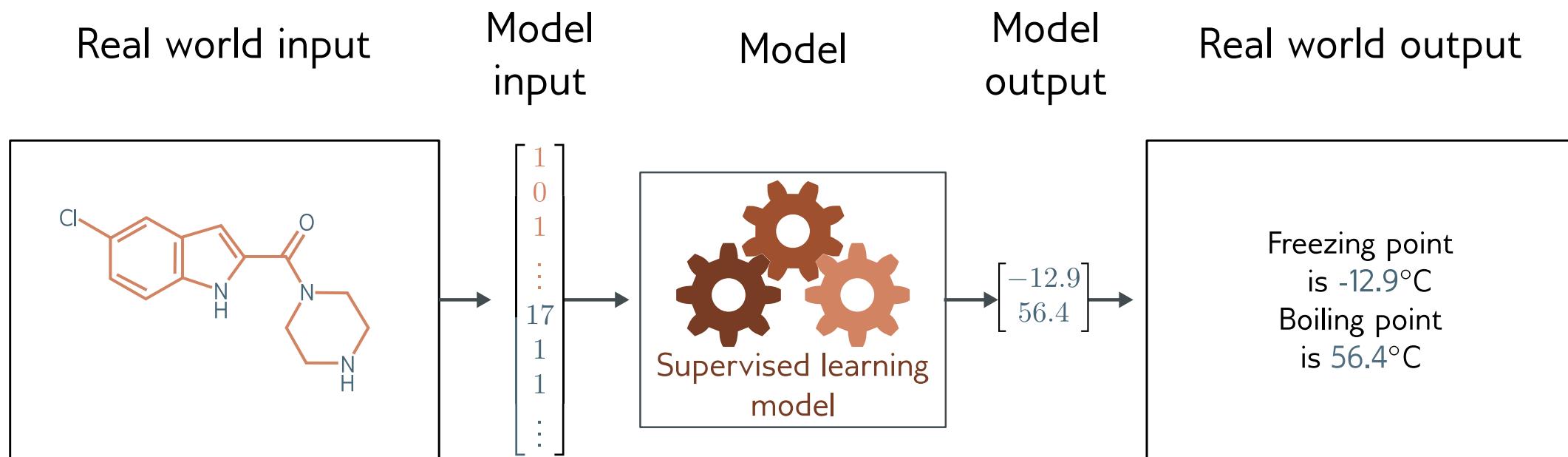
- Treat each output y_d as independent:

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

- Negative log likelihood becomes sum of terms:

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

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

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

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

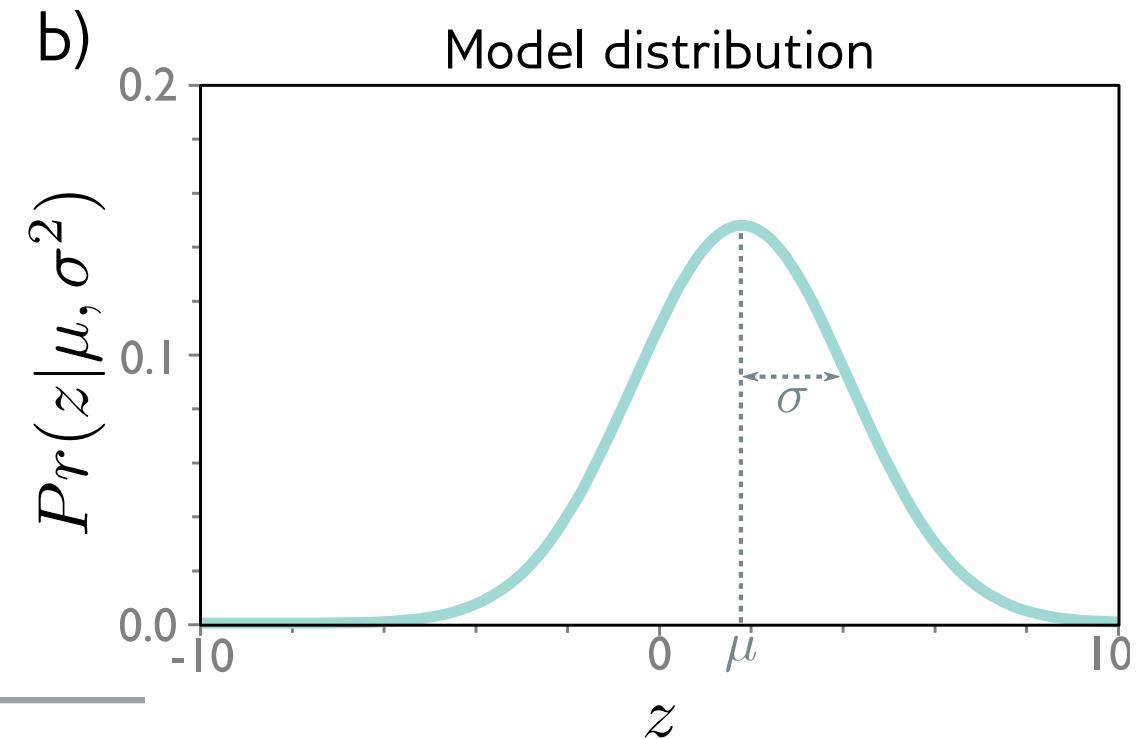
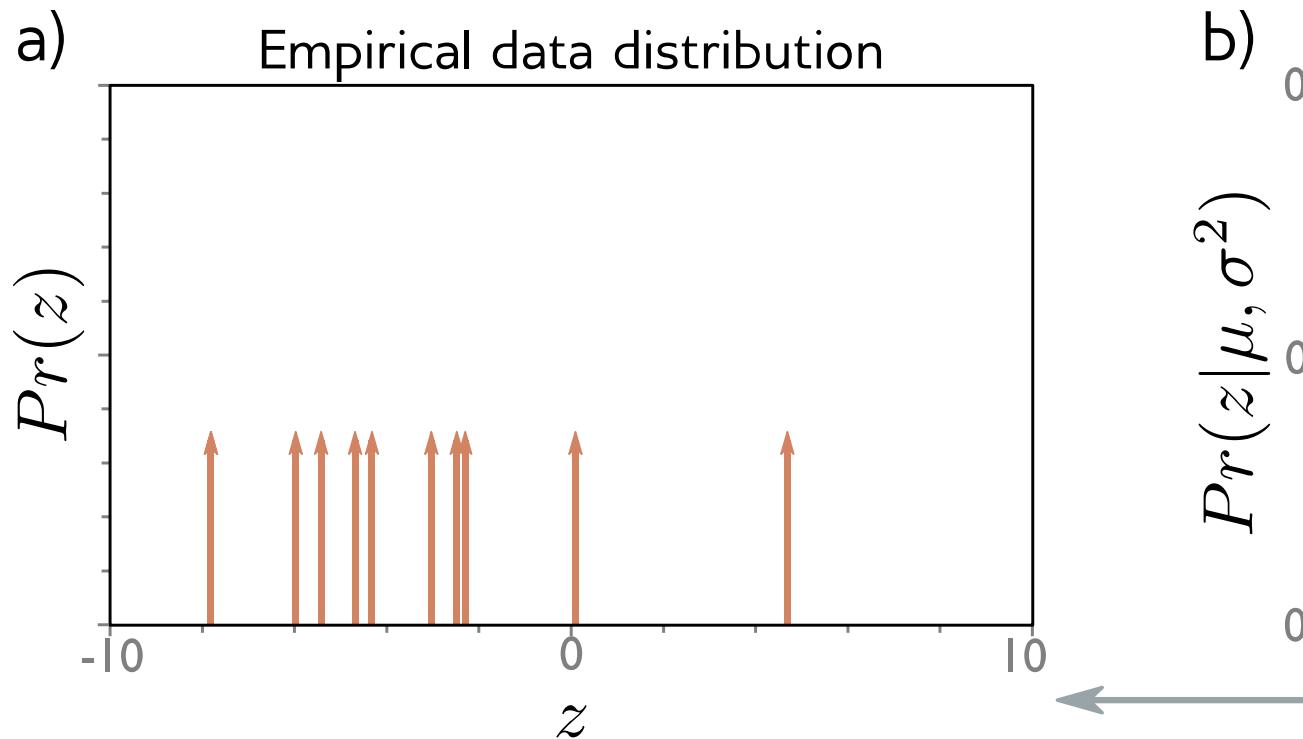
Cross Entropy

- In **regression problems**, the output is a continuous value, and we use **Mean Squared Error (MSE)**
- In **classification problems**, the output is a **probability distribution** over classes, and we need a different approach:
- We want the model to **assign high probability to the correct class**.
- We use **cross-entropy loss**, which measures how different the predicted distribution is from the true distribution.

KL Divergence Loss

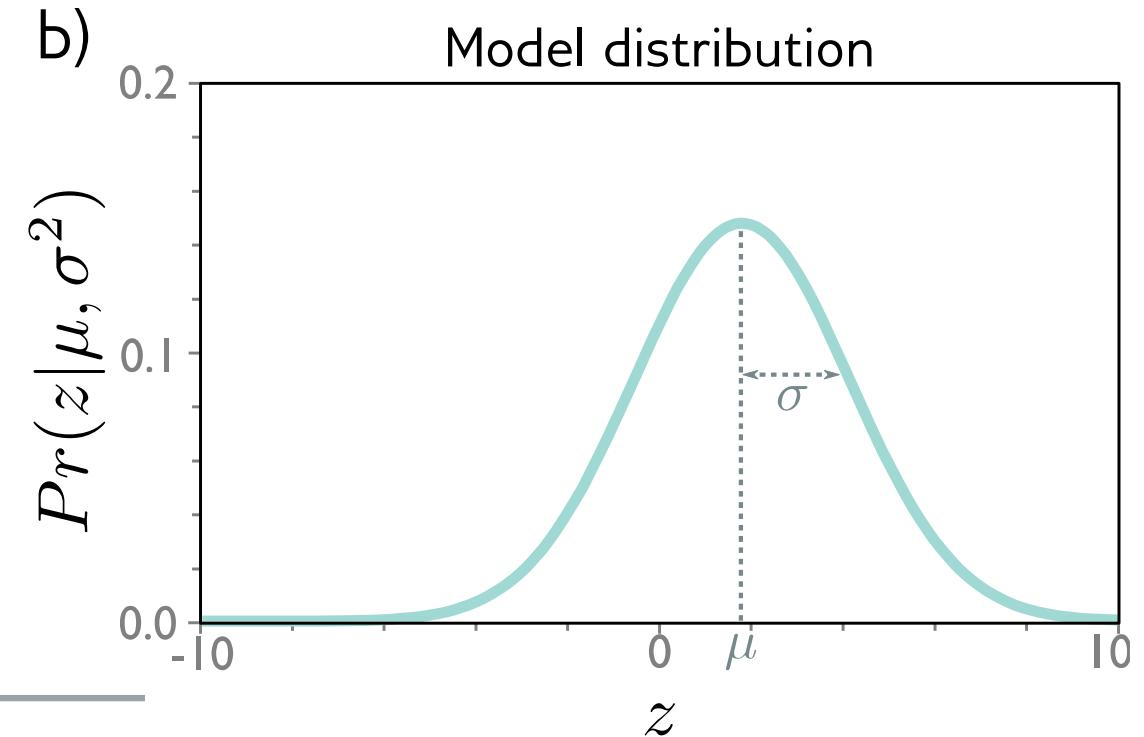
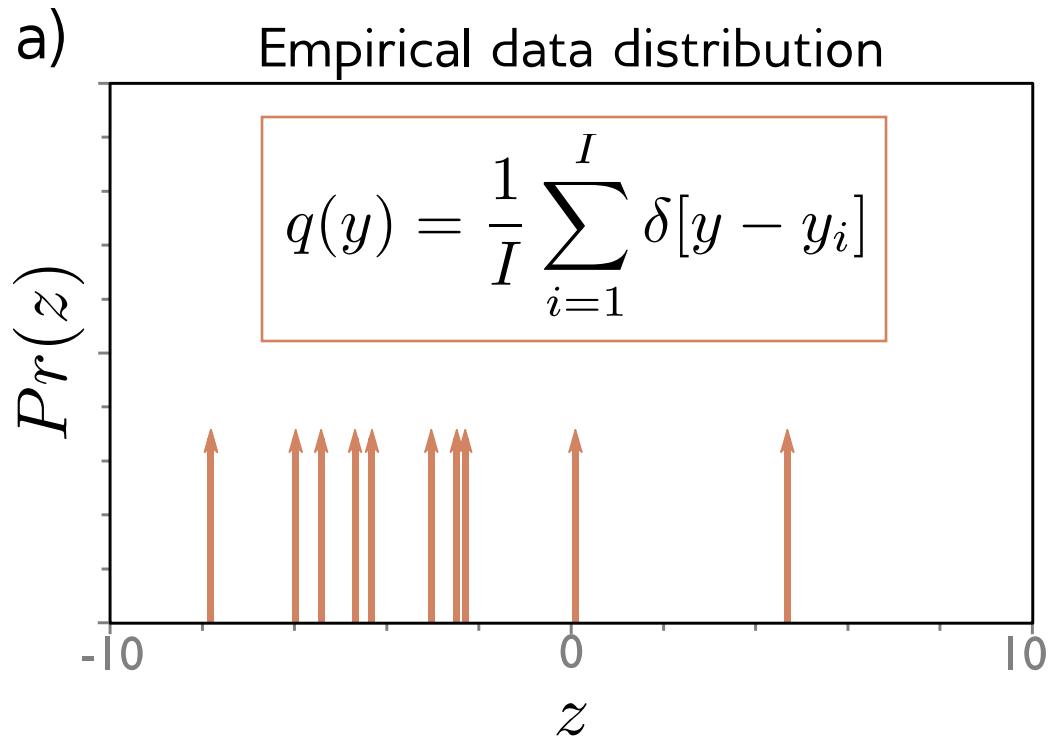
- The KL divergence measures how different the empirical distribution $q(y)$ (observed data) is from the model's predicted distribution $P(y|\theta)$
- Minimizing DKL ensures the model's predicted probabilities are as close as possible to the true data distribution.

Cross Entropy



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

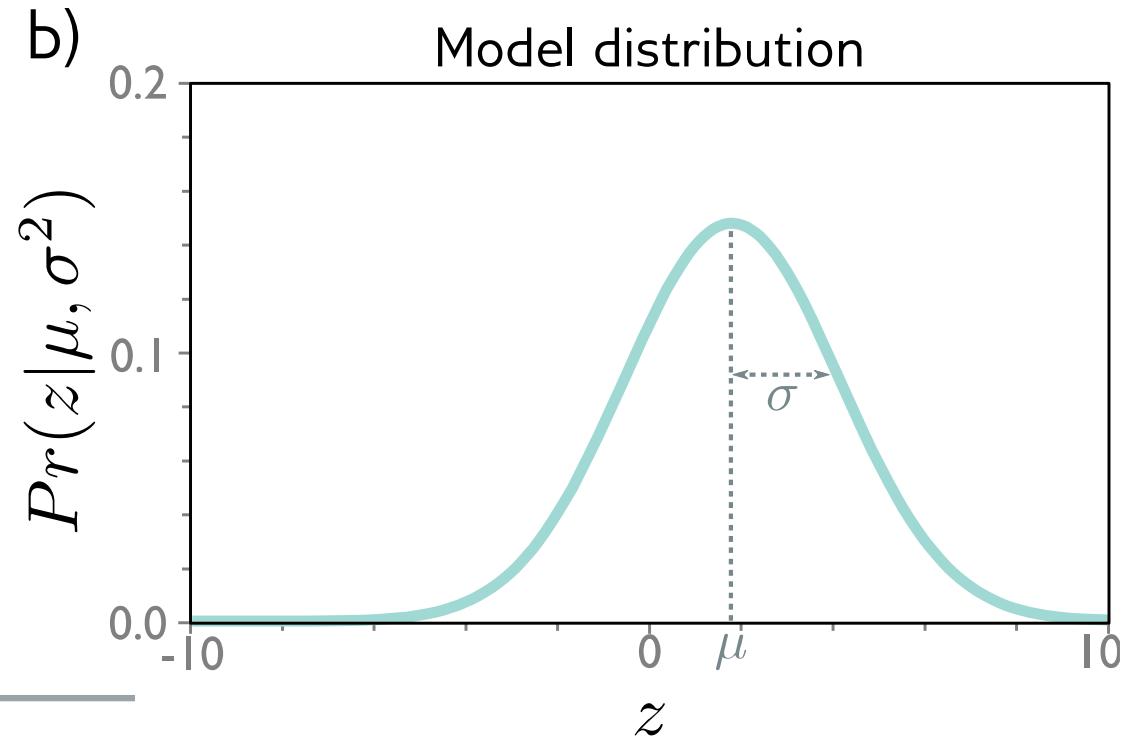
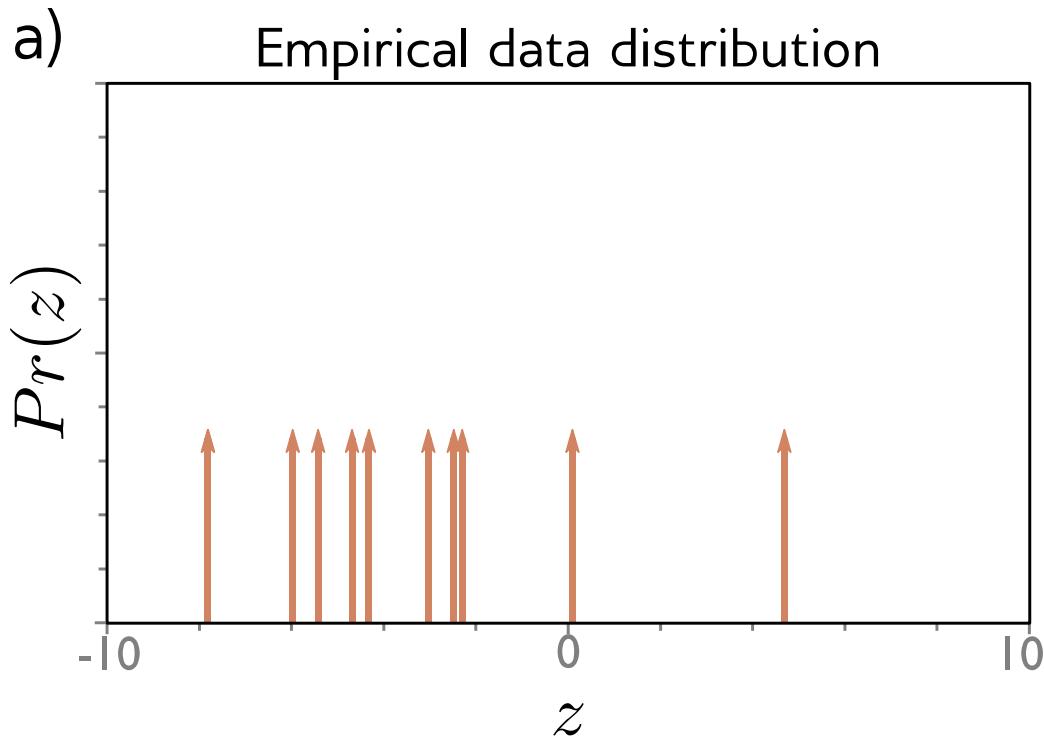
Cross Entropy



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

Kullback-Leibler Divergence -- a measure between probability distributions

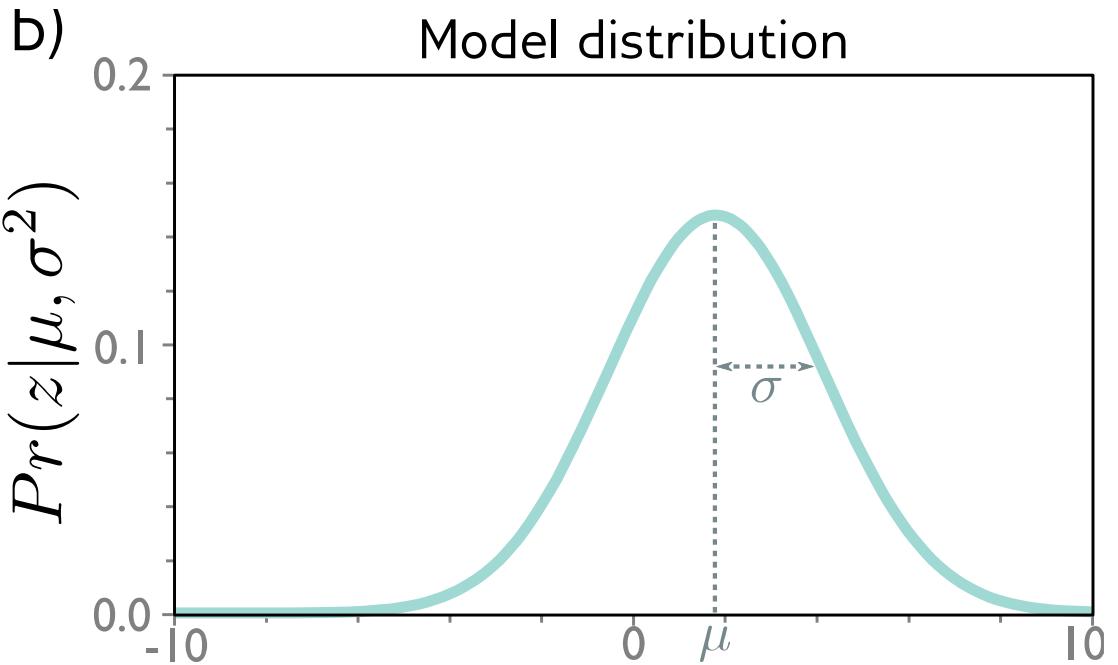
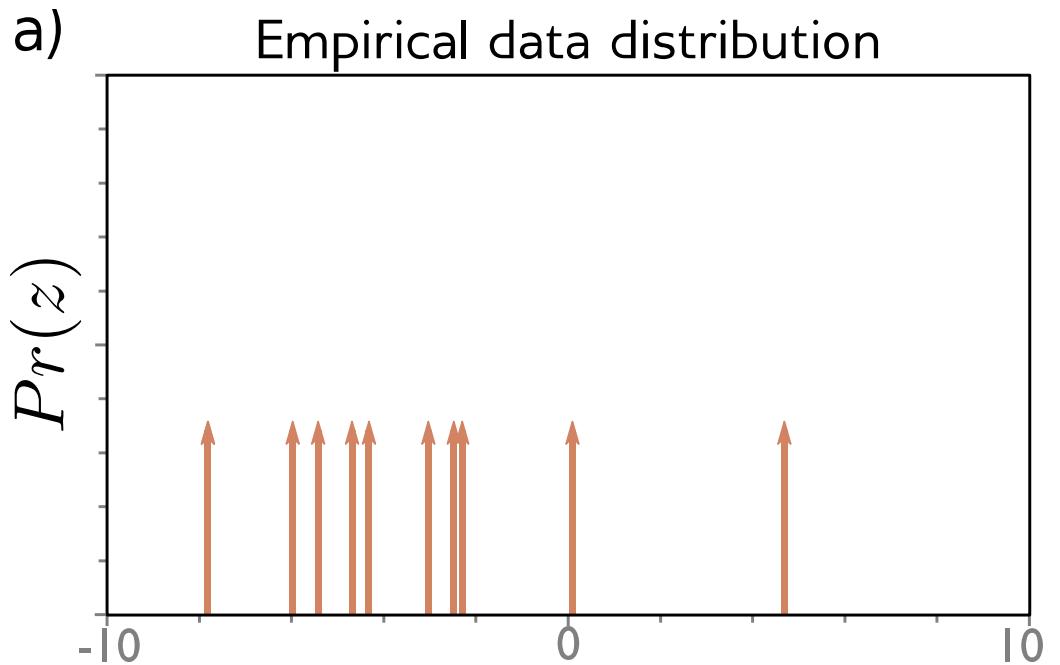
Cross Entropy



First term does not
depend on theta

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

Cross Entropy



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

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

Minimum
negative log
likelihood

Cross entropy in machine learning

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

Minimum
negative log
likelihood

In machine learning:

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

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



Feedback