

Deep Learning for Data Science

DS 542

<https://dl4ds.github.io/fa2025/>

Loss Functions



Recap

- Last time we talked about supervised learning with the example of linear regression.
- Models have parameters, ϕ , that we want to choose for a best possible mapping between input and output training data
- A loss function or cost function, $L[\phi]$, returns a single number that describes a mismatch between $f[x_i, \phi]$ and the ground truth outputs, y_i .

Plan for Today

- Use cases for loss functions
- Maximum likelihood approach
- Deriving common loss functions
 - Real-valued univariate regression
 - Binary classification
 - Multiclass classification
 - Multiple outputs (if extra time)
- Connections to cross entropy (if extra time)

How do we choose a loss function?

sensitivity to outliers

- least squares penalizes big errors



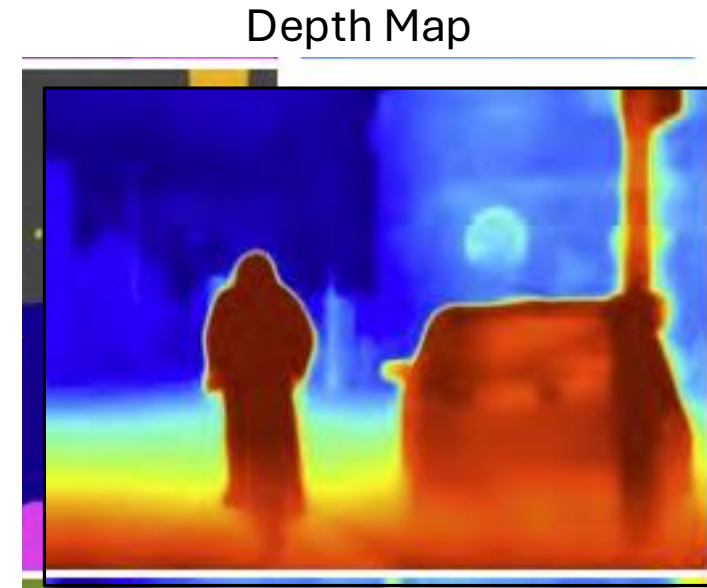
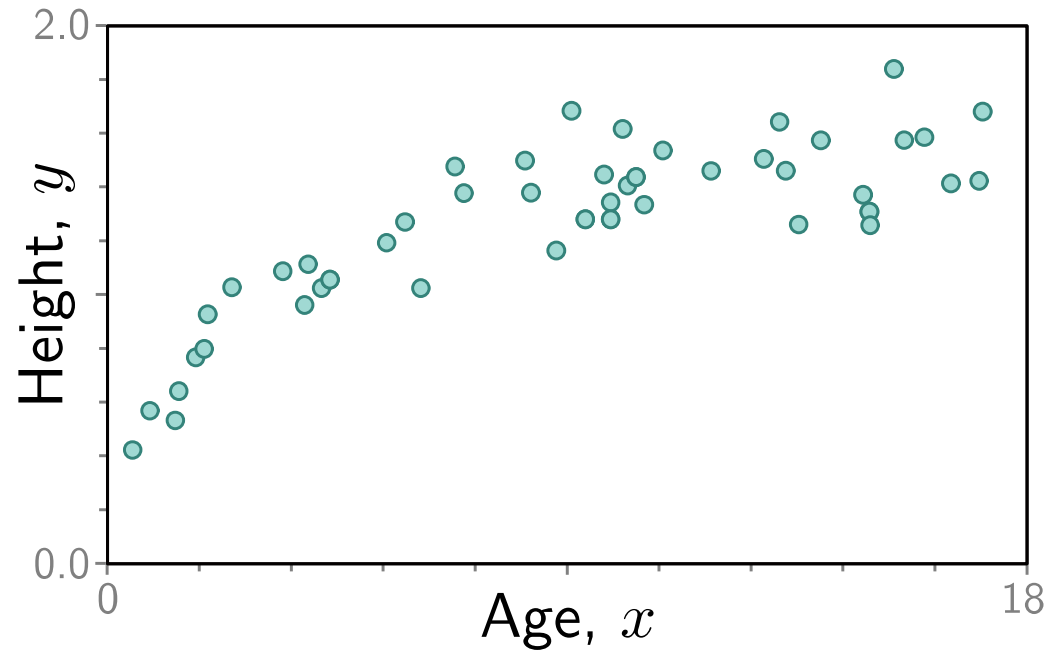
help drive predictions

+ actual results together

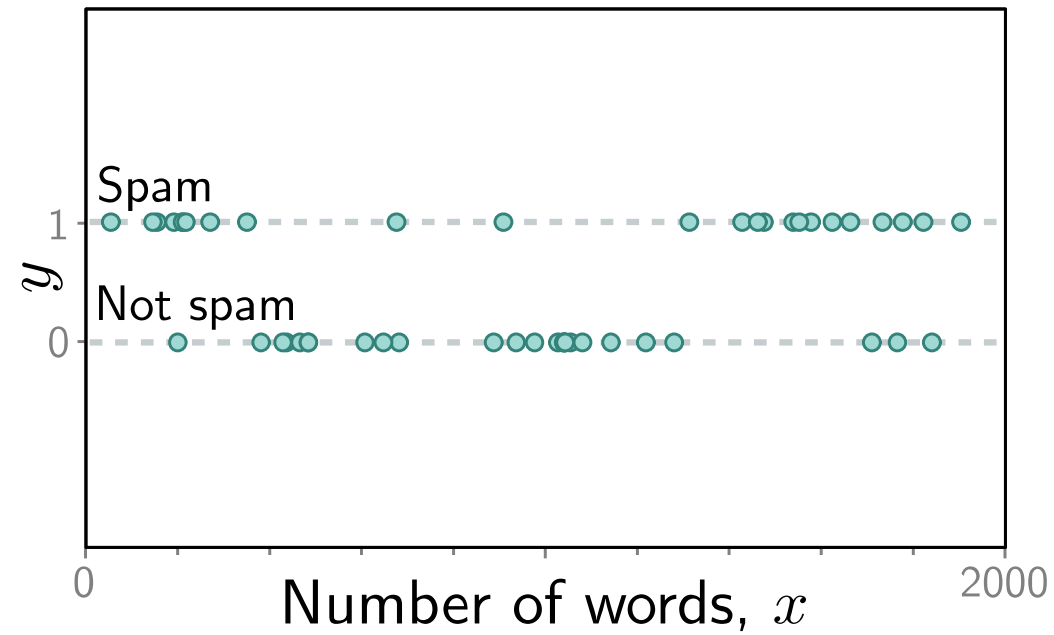
want to take derivative

want to compare train + test
to assess generalization

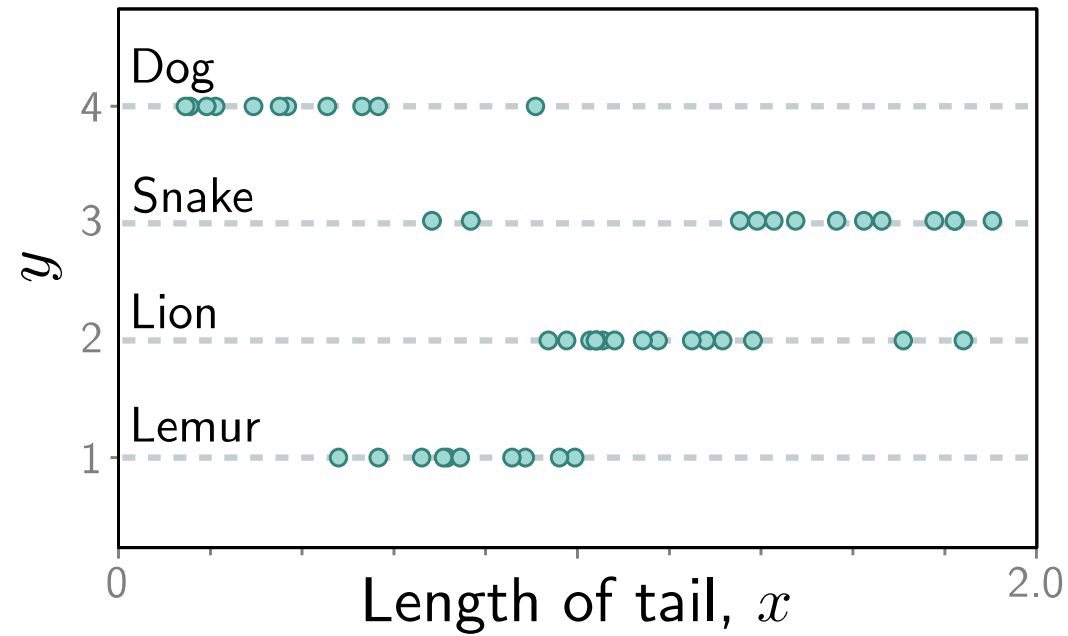
Univariate and Multivariate Regression

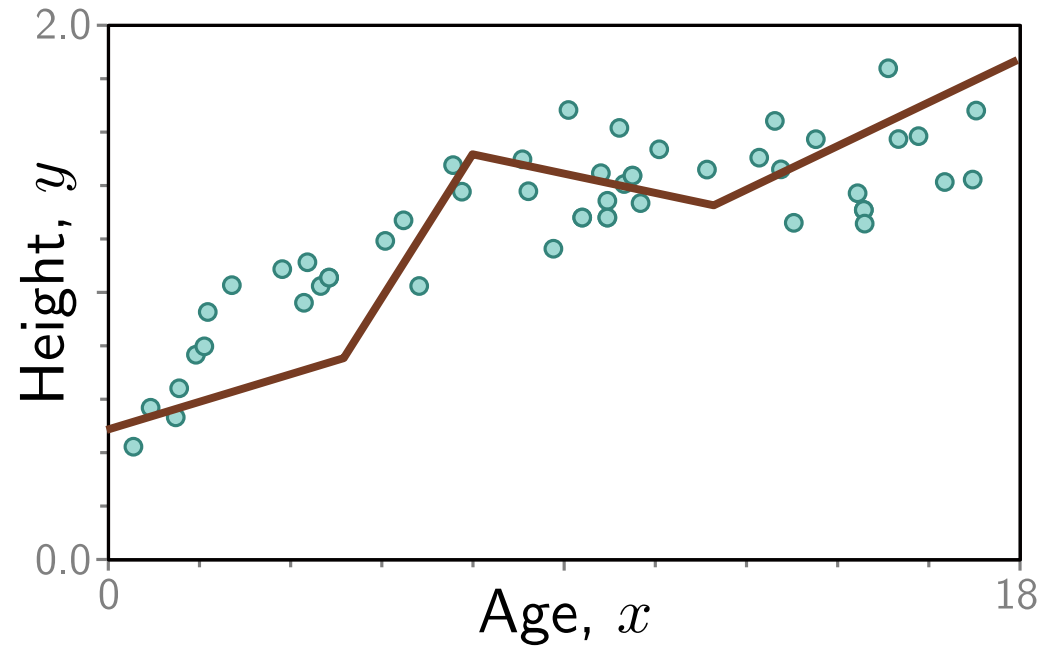


Binary Classification



Multiclass Classification





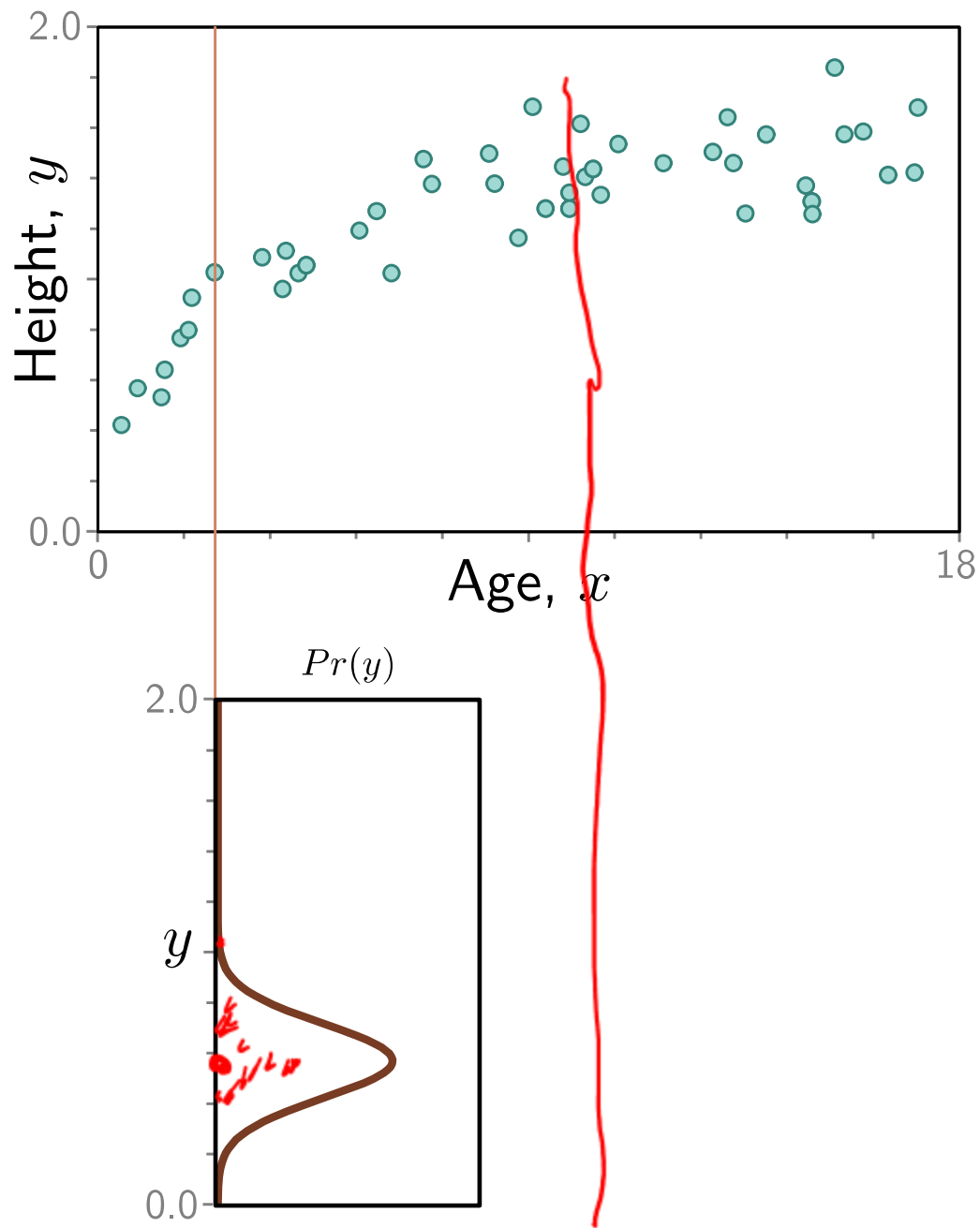
So far, we thought about fitting a model to the data...

Plan for Today

- Use cases for loss functions
- Maximum likelihood approach
- Deriving common loss functions
 - Real-valued univariate regression
 - Binary classification
 - Multiclass classification
 - Multiple outputs (if extra time)
- Connections to cross entropy (if extra time)

Competing Takes on Loss Functions

1. How bad are my model estimates on average?
 - Model predicts a specific value.
 - Loss function compares that value to the ground truth.
2. How likely did my model think the actual result was?
 - Model predicts a probability distribution.
 - Loss function checks likelihood of ground truth from that distribution.



Suppose we fit a *probability model* to this data, and outputs conditional probability distribution

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

Isn't this a better fit for the reality?

Probability Approach Suggests Maximum Likelihood Estimation

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

training data (inputs)
model parameters $\rightarrow P(\text{output} \mid \text{training input, m.p.})$

- This will directly suggest choices of loss functions.

derive
which loss
matches?

minimize
loss



can change
model parameters
to maximize

How do we do this?

- Model predicts a conditional probability distribution:

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

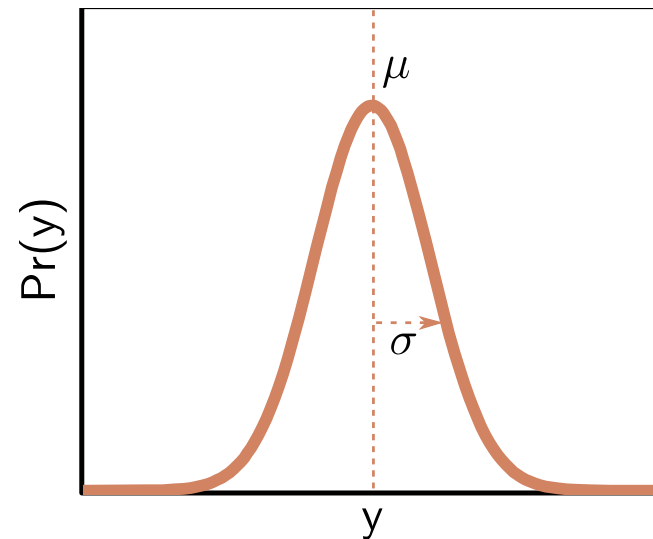
over outputs $\mathbf{y}$ given inputs $\mathbf{x}$.

- Define and minimize a loss function that makes the outputs have high probability.

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

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

e.g., the normal distribution $\theta = \{\mu, \sigma^2\}$ *why? because many/most error distributions look approximately normal.*



also mathematically convenient

2. Use model to predict parameters θ of probability distribution.

Maximize the joint, conditional probability

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

$$\Pr(y_1, y_2, \dots, y_I | x_1, x_2, \dots, x_I)$$

Two simplifying assumptions

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

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

simplifying assumption

Independent

Independent and identically distributed (i.i.d)

Maximum likelihood criterion

Pick model parameters ϕ to maximize the likelihood

last slide

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

θ_i are the parameters of the probability distribution
 $\hat{\theta}_i$ parameters predicted for $\mathbf{x}_i$

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

best model parameters that do maximize likelihood

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

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

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

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

Practical Problem:

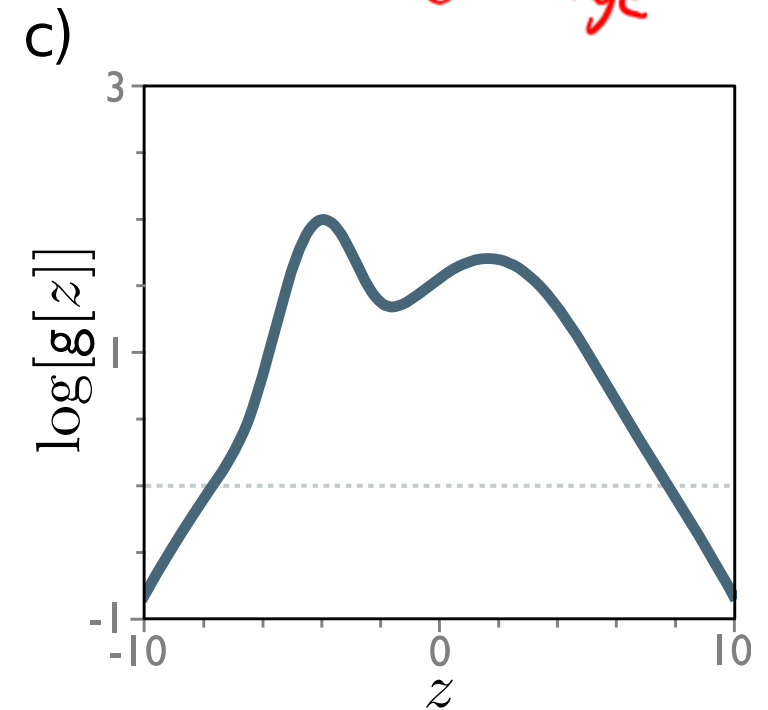
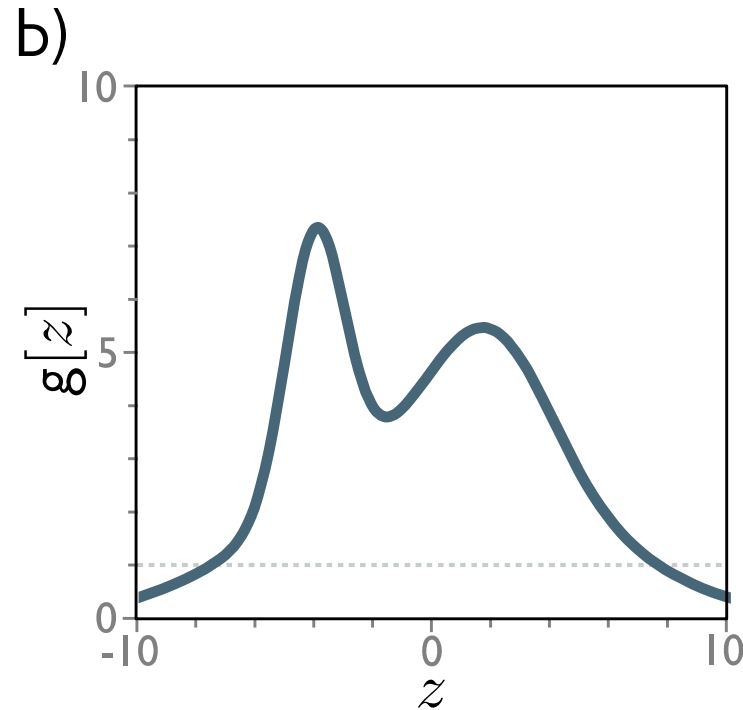
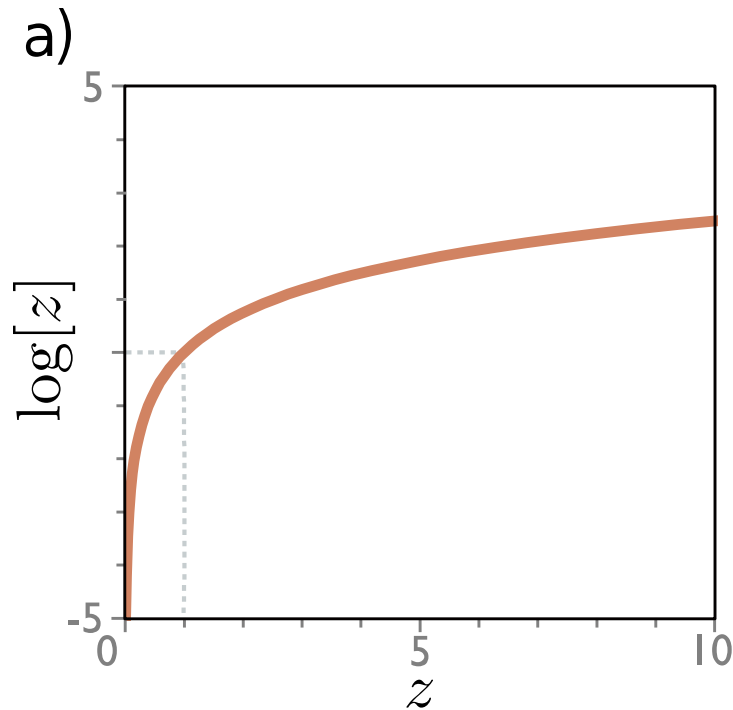
$$\hat{\phi} = \operatorname{argmax}_{\phi} \left[\prod_{i=1}^I \underbrace{Pr(\mathbf{y}_i | \mathbf{f}[\mathbf{x}_i, \phi])}_{\text{probabilities, so in } [0,1]} \right]$$

gets too small for floating point
↓
becomes $\emptyset$

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

The log function is monotonic

$\log(\pi(P_h))$
has smaller range



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

Maximum log likelihood

$$\begin{aligned}\hat{\phi} &= \operatorname{argmax}_{\phi} \left[\prod_{i=1}^I \operatorname{Pr}(\mathbf{y}_i | \mathbf{f}[\mathbf{x}_i, \phi]) \right] && \text{previous formula} \\ &= \operatorname{argmax}_{\phi} \left[\log \left[\prod_{i=1}^I \operatorname{Pr}(\mathbf{y}_i | \mathbf{f}[\mathbf{x}_i, \phi]) \right] \right] && \text{w/ underflow issue} \\ &= \operatorname{argmax}_{\phi} \left[\sum_{i=1}^I \log \left[\operatorname{Pr}(\mathbf{y}_i | \mathbf{f}[\mathbf{x}_i, \phi]) \right] \right] && \text{same choice} \\ &&& \text{b/c log is monotonic} \\ &&& \text{same answer,} \\ &&& \text{minimally faster,} \\ &&& \text{conveniences like} \\ &&& \text{mean functions}\end{aligned}$$

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)

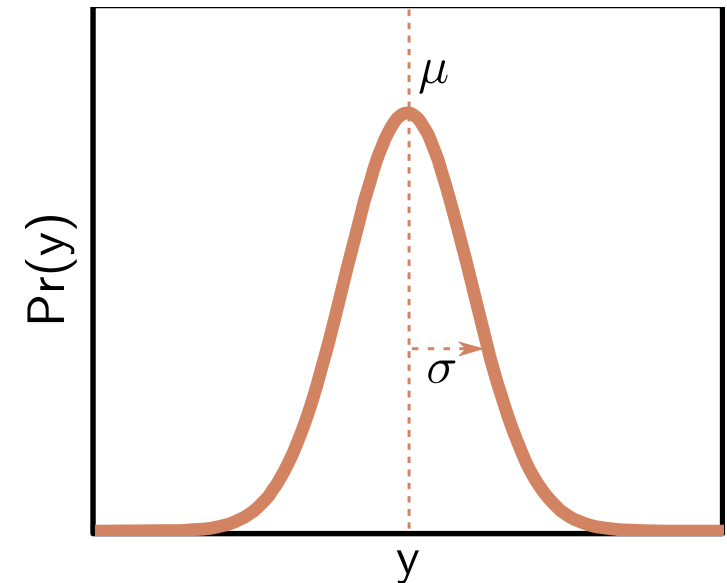
$$\begin{aligned}\hat{\phi} &= \operatorname{argmax}_{\phi} \left[\sum_{i=1}^I \log \left[\operatorname{Pr}(\mathbf{y}_i | \mathbf{f}[\mathbf{x}_i, \phi]) \right] \right] \\ &= \operatorname{argmin}_{\phi} \left[- \sum_{i=1}^I \log \left[\operatorname{Pr}(\mathbf{y}_i | \mathbf{f}[\mathbf{x}_i, \phi])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

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)

Predict?
max vs mean
Same if predicting
normal distribution

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



Recipe for loss functions

If you use this model w/
empty $\mathbf{x}$, then you get a
generative model for outputs.

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

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

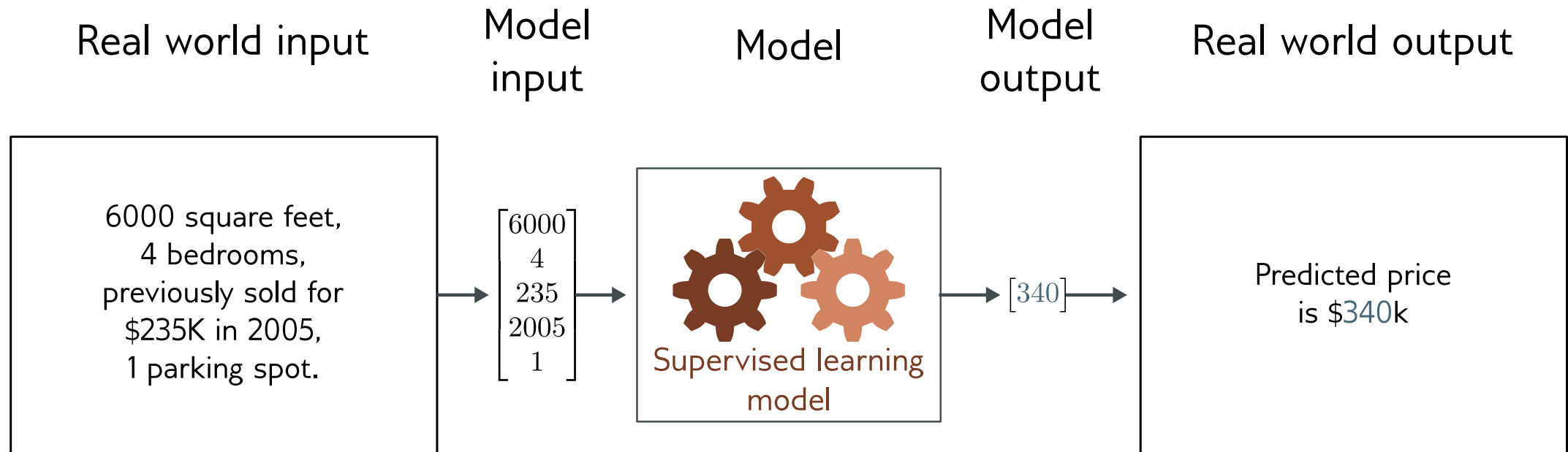
= max likelihood

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.

Plan for Today

- Use cases for loss functions
- Maximum likelihood approach
- Deriving common loss functions
 - Real-valued univariate regression
 - Binary classification
 - Multiclass classification
 - Multiple outputs (if extra time)
- Connections to cross entropy (if extra time)

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

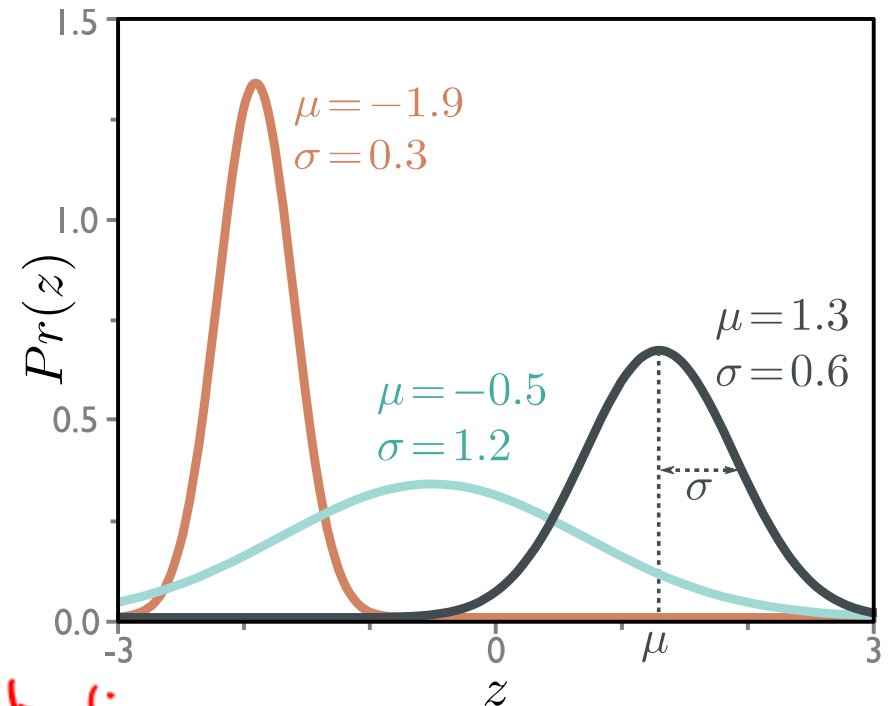
$$Pr(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(y - \mu)^2}{2\sigma^2} \right]$$

↑
mean

↑
variance

$\sigma = \text{std. dev.}$

usual formula for
PDF of normal distribution



Example 1: univariate regression

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

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

In this case,
just the mean

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

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

simpler model $\Rightarrow \mu = \mathbf{f}[\mathbf{x}, \phi]$

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

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

*generic formula
plugging in $f[\mathbf{x}, \phi]$
for μ*

*plugging in
conditional
probability*

Loss function

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

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

Handwritten red annotations: 'a' above the first fraction, 'b' above the exponential term, and a red bracket under the product of the two terms with arrows pointing to the separated terms in the second line.

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

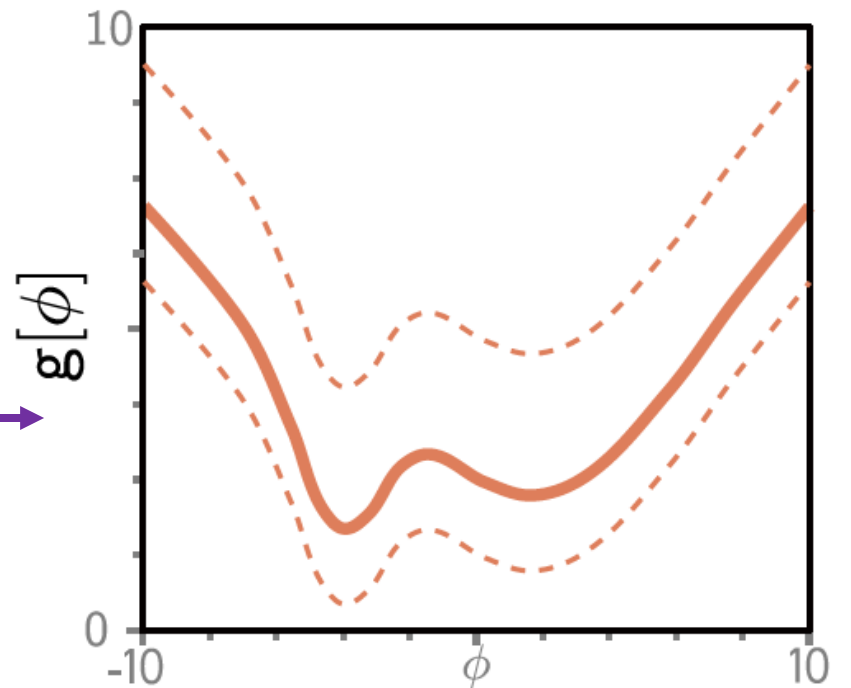
A blue arrow points from this equation to the log term in the second line of the derivation above.

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

$$\log[\exp[x]] = x$$

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

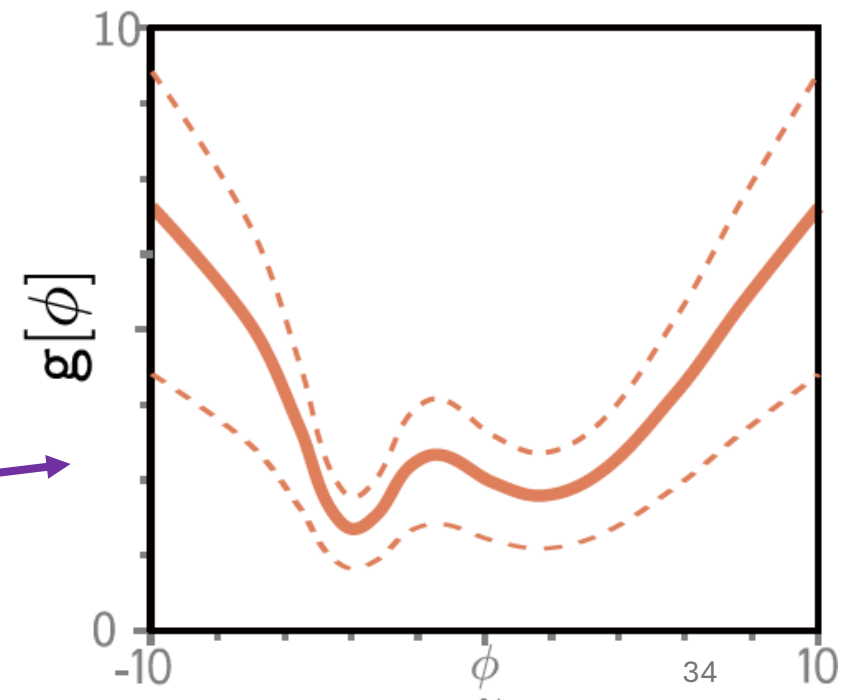
Just a constant
offset



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

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

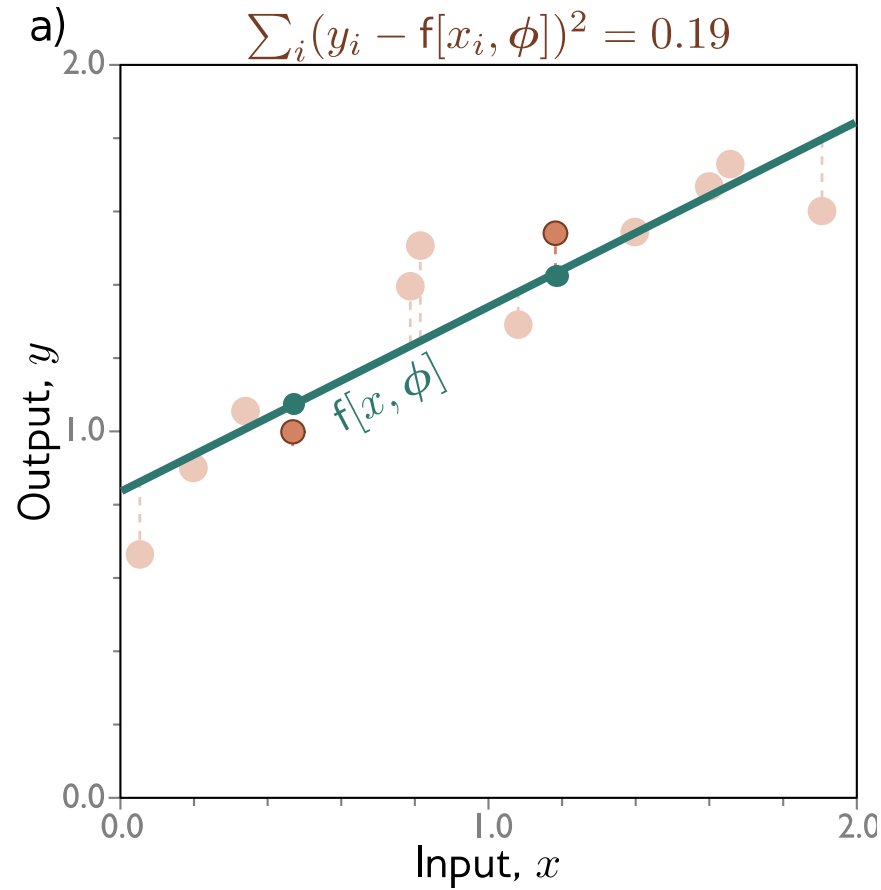
Just dividing by a
positive constant



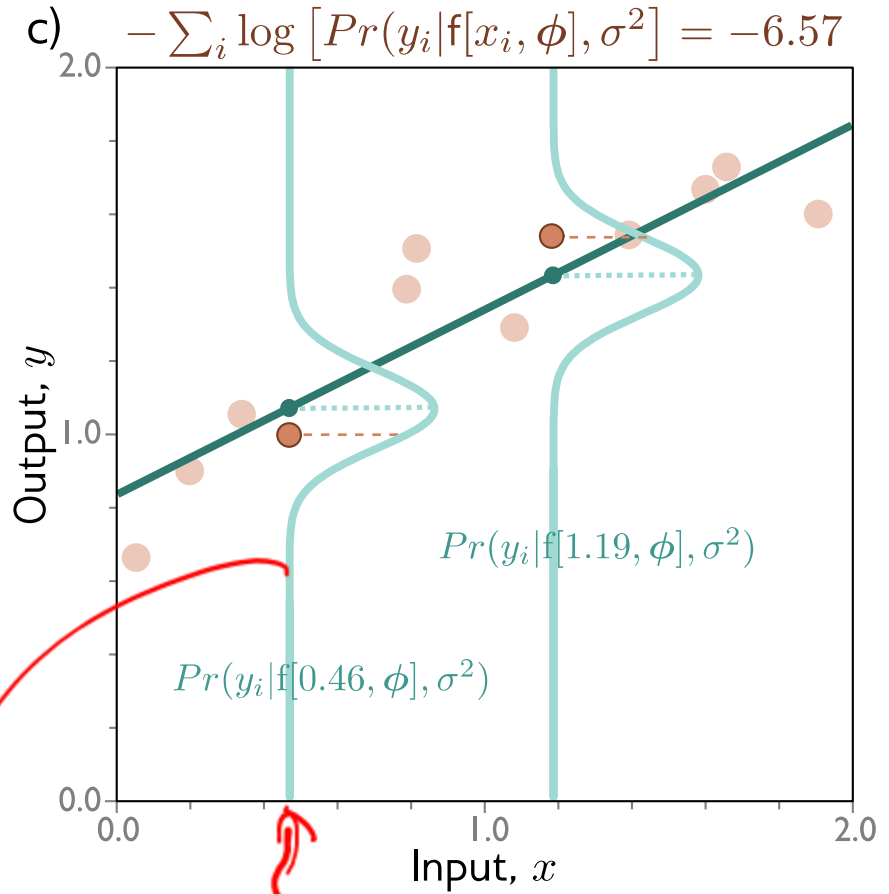
$$\begin{aligned}
\hat{\phi} &= \operatorname{argmin}_{\phi} \left[- \sum_{i=1}^I \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(y_i - f[\mathbf{x}_i, \phi])^2}{2\sigma^2} \right] \right] \right] \\
&= \operatorname{argmin}_{\phi} \left[- \sum_{i=1}^I \log \left[\frac{1}{\sqrt{2\pi\sigma^2}}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

assumed normally
distributed probabilities,
fixed variance σ^2

Least squares

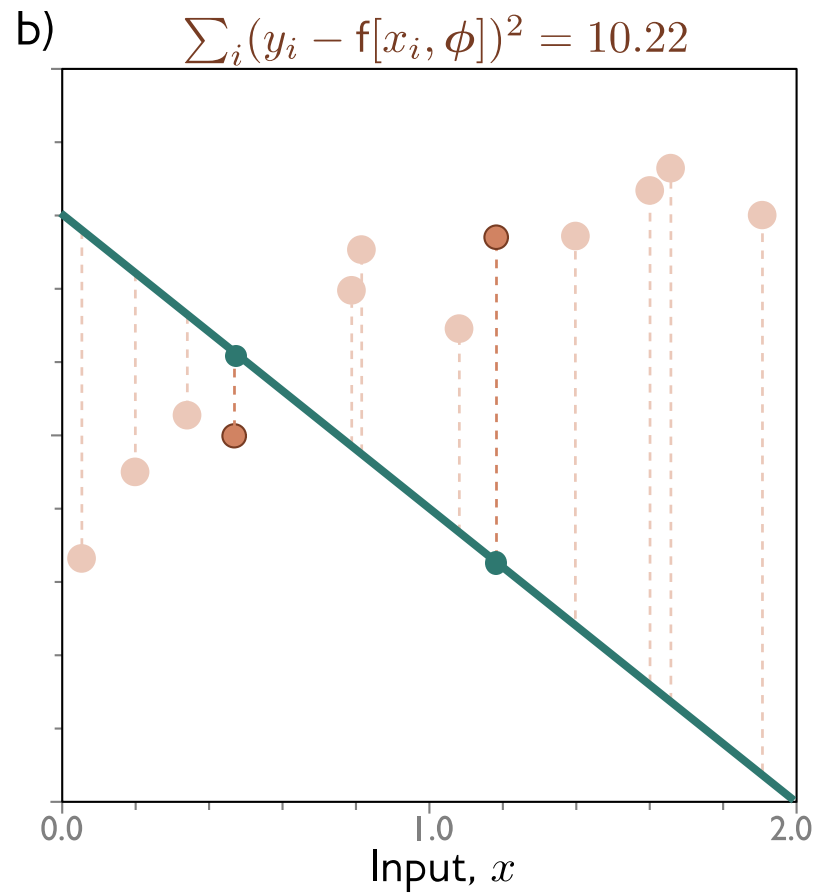


Negative log likelihood

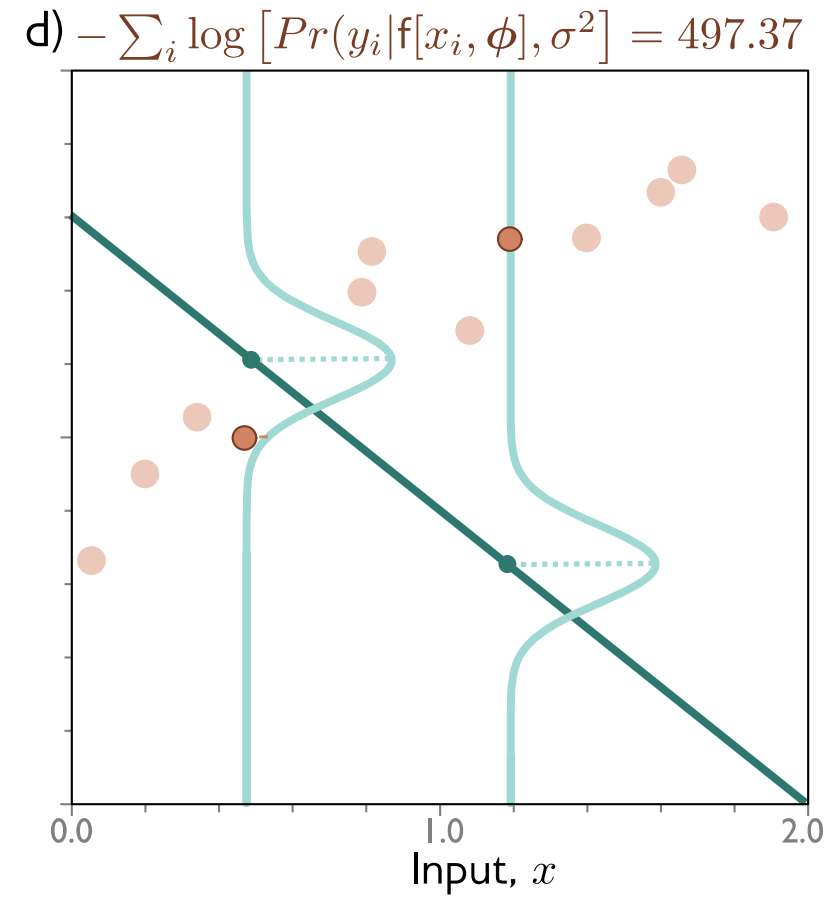


sideways pdf of prediction at this x

Least squares



Maximum likelihood



Example 1: univariate regression

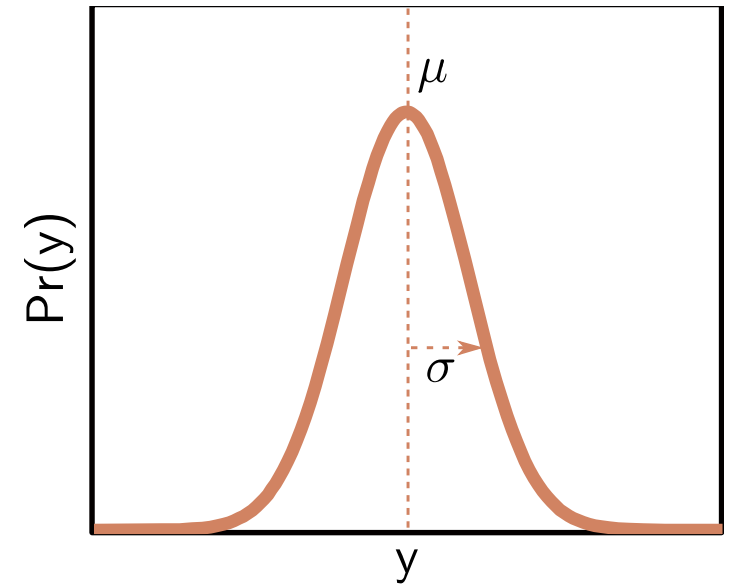
4. To perform inference for a new test example $\mathbf{x}$, return either the full distribution $Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \hat{\phi}])$ or the maximum of this distribution.

Full distribution:

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

Max probability:

$$\hat{y} = \hat{\mu} = \mathbf{f}[\mathbf{x} | \phi]$$



Estimating variance

- Perhaps surprisingly, the variance term disappeared:

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

if fixed variance,
 σ^2 just shifts + rescales
log probs.
Does not change order or choice.

Estimating variance

- But we could learn it during training:

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

- Do gradient descent on both model parameters, ϕ , and the variance, σ^2

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

Heteroscedastic regression

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

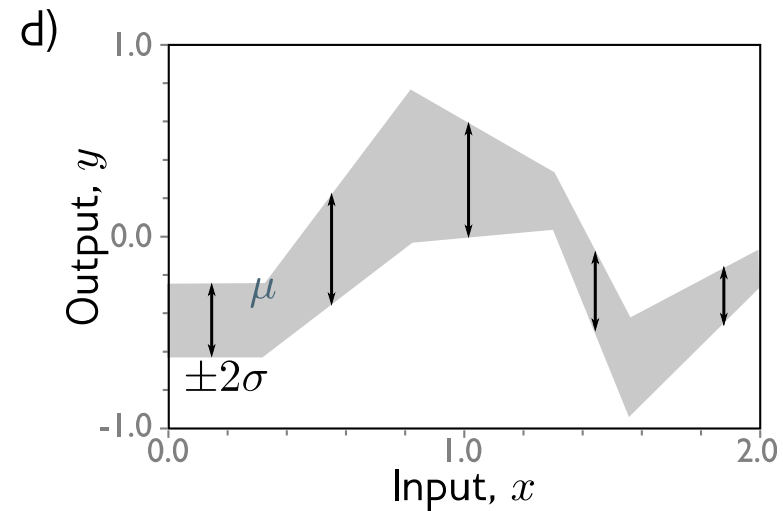
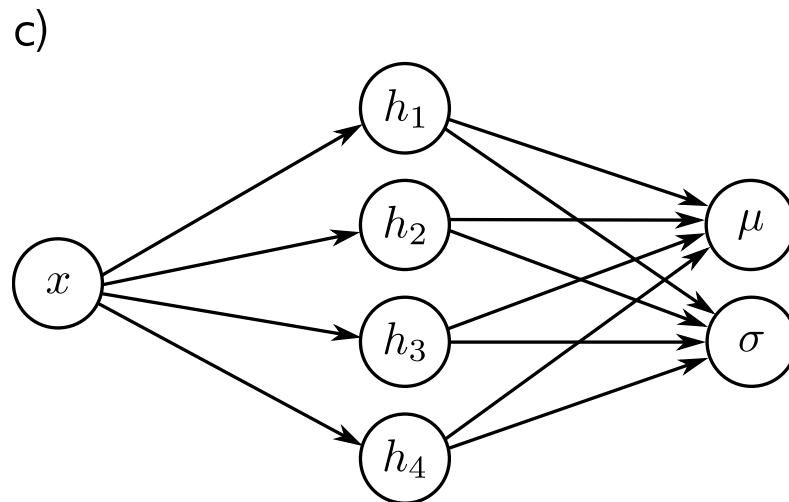
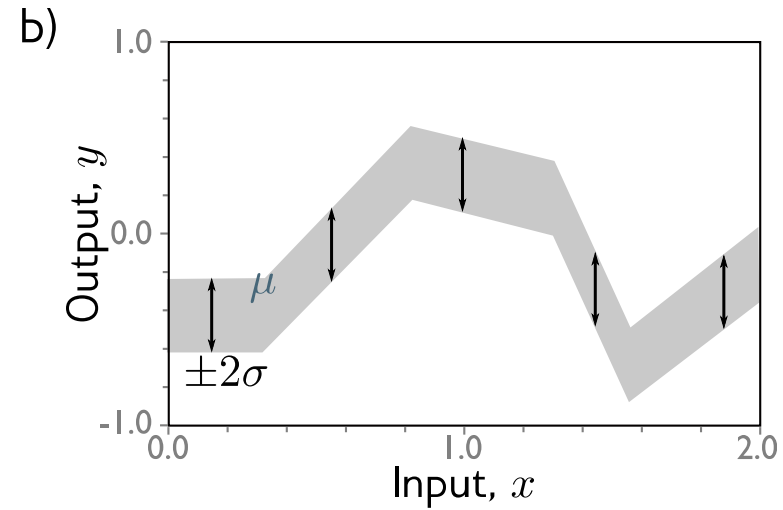
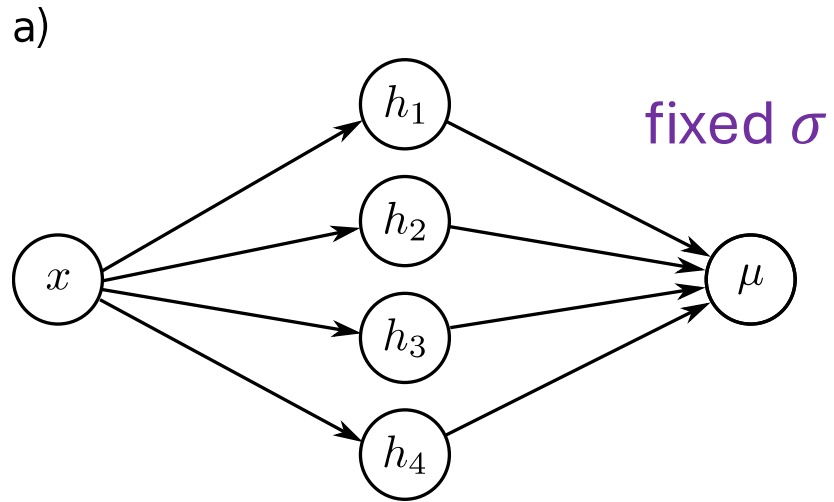
$$\mu = f_1[\mathbf{x}, \phi]$$

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

Squared to ensure it
is positive

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

Heteroscedastic regression



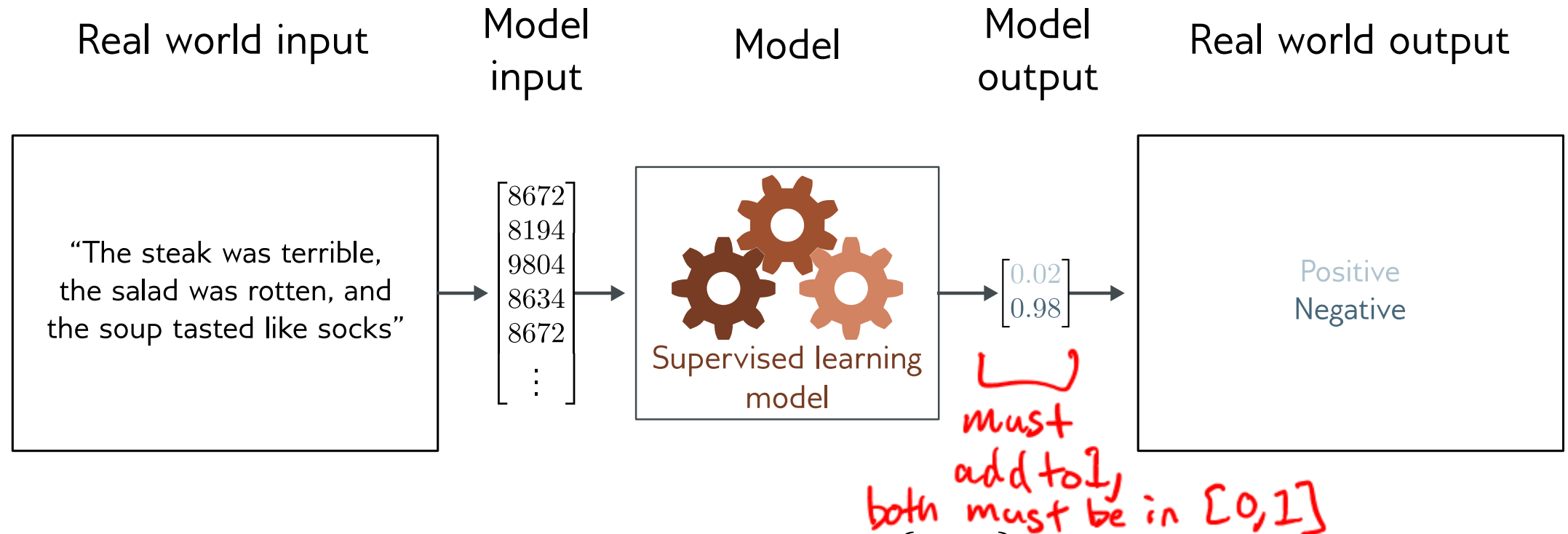
Example 1: Univariate Regression Takeaways

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

Plan for Today

- Use cases for loss functions
- Maximum likelihood approach
- Deriving common loss functions
 - Real-valued univariate regression
 - Binary classification
 - Multiclass classification
 - Multiple outputs (if extra time)
- Connections to cross entropy (if extra time)

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 $\lambda \in [0, 1]$

Definitions:

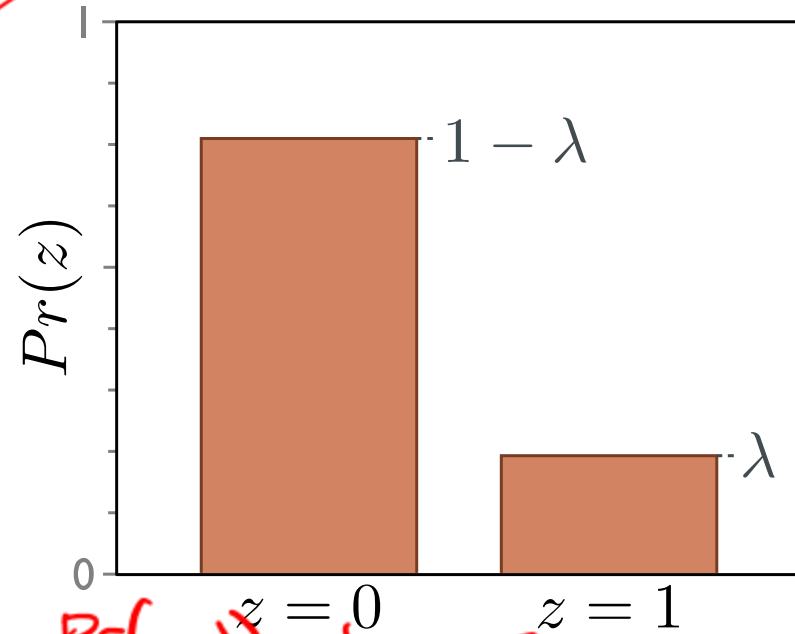
$$\rightarrow Pr(y|\lambda) = \begin{cases} 1 - \lambda & y = 0 \\ \lambda & y = 1 \end{cases}$$

or

Clever $\Rightarrow$ differentiable

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

why?



if $y=1$, $Pr(y, \lambda) = \lambda$
if $y=0$, $Pr(y, \lambda) = 1 - \lambda$ } same as def

Example 2: binary classification

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

Problem:

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

*pis too generic!
range $[-\infty, \infty]$?
want $[0,1]$*

Solution:

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

*also preserve order. ~~pres~~
preserve sense of high vs low.
so monotonic.*

Example 2: binary classification

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

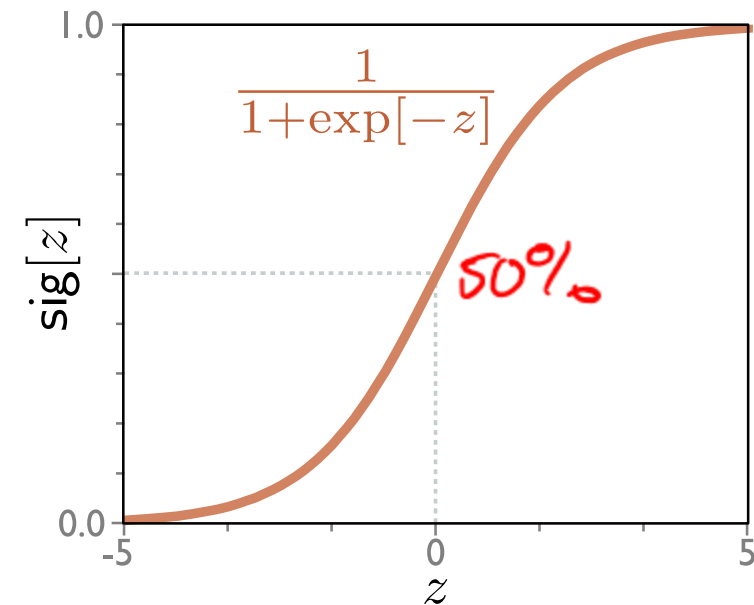
Problem:

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

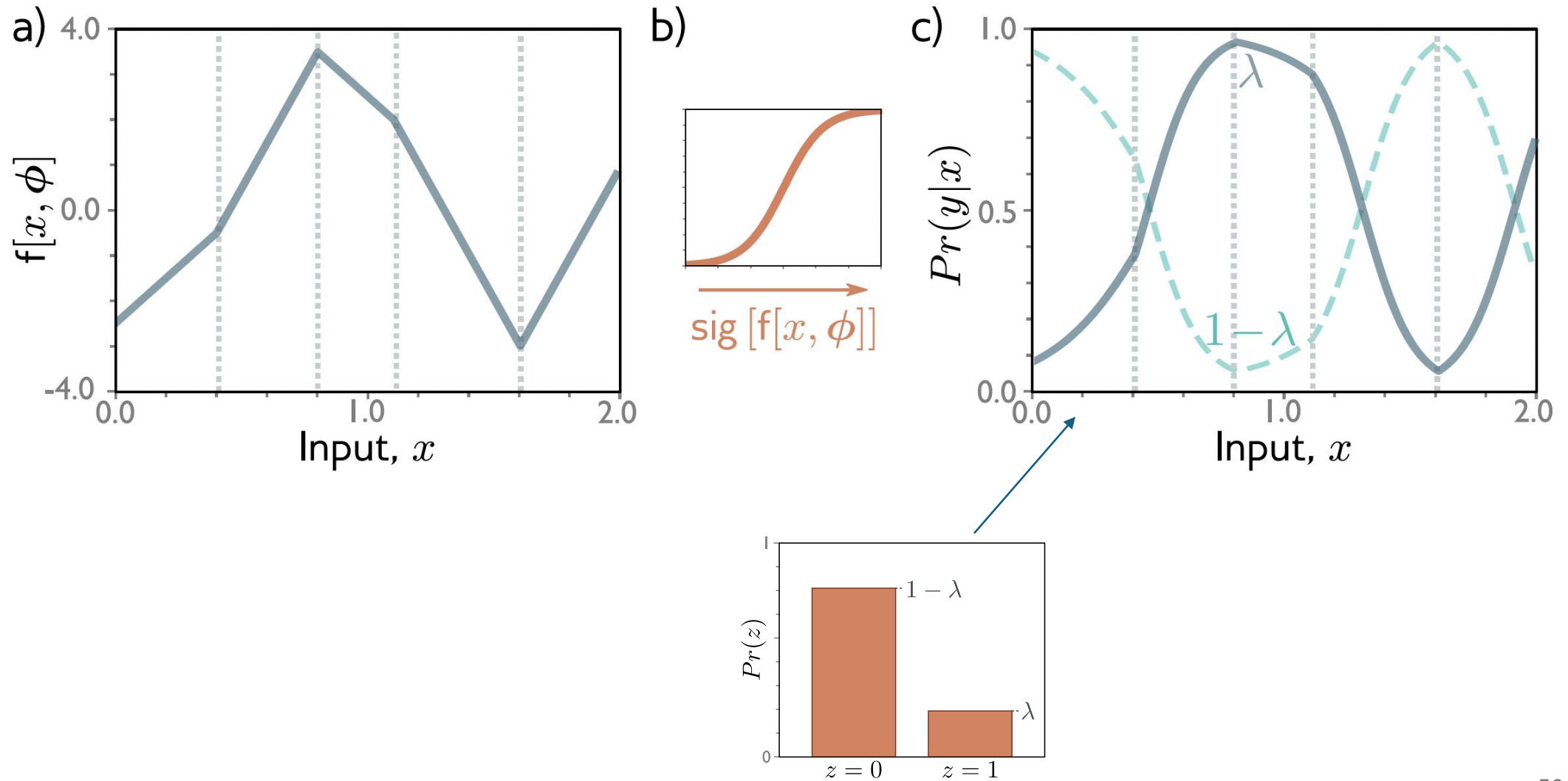
$$Pr(y|\lambda) = (1 - \lambda)^{1-y} \cdot \lambda^y$$

previous clever version

$\lambda = \text{sig}[f[\mathbf{x}, \phi]]$

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

Example 2: binary classification



Example 2: binary classification

As $P(\text{correct}) \rightarrow 1$,
 $\log(P(\text{correct})) \rightarrow 0$
 neg $\rightarrow$
 $-\log(P(\text{correct})) \rightarrow 0$
 pos $\rightarrow$

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

Model training

$P(0.1) \rightarrow P(0.99)$

$\log(0.1) \rightarrow \log 0.99$

$-2 \rightarrow -0.1$

$-\log(0.1) \rightarrow -\log 0.99$

$2 \rightarrow 0.1$

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

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

either y_i or $1-y_i$
is ϕ .

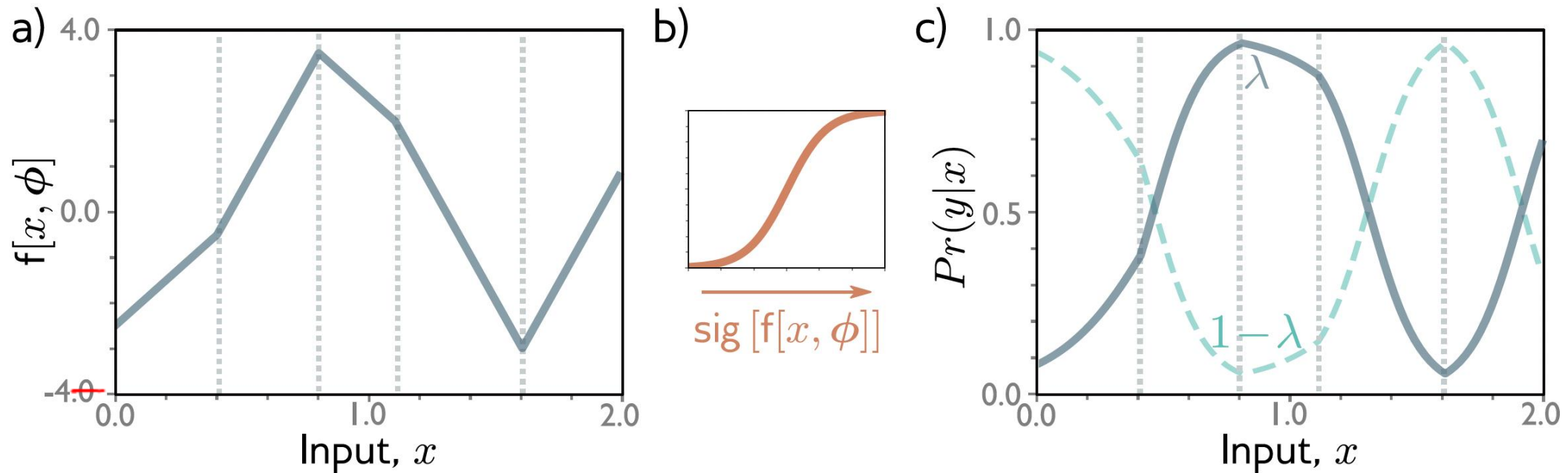
$$L[\phi] = \sum_{i=1}^I -(1 - y_i) \log [1 - \operatorname{sig}[f[\mathbf{x}_i|\phi]]] - y_i \log [\operatorname{sig}[f[\mathbf{x}_i|\phi]]] = \sum -\log(P(\text{correct answer}))$$

$P(y=0)$ $P(y=1)$

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

Example 2: binary classification

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



Choose $y = 1$ where λ is greater than 0.5, otherwise 0
And we get a probability estimate!

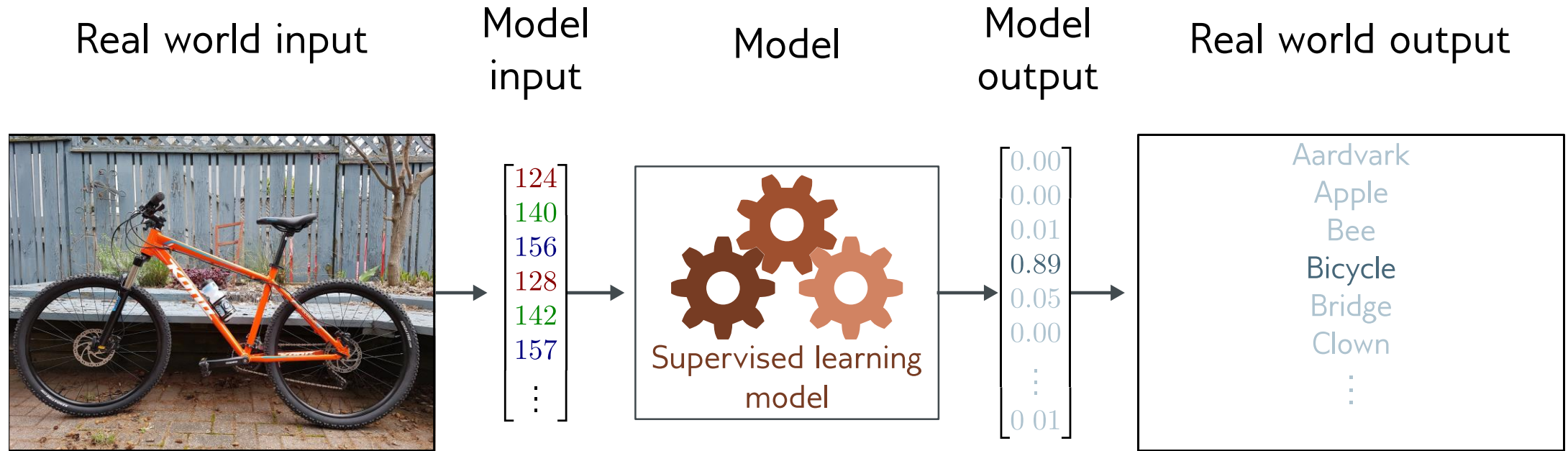
Example 2: Binary Classification Takeaways

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

Plan for Today

- Use cases for loss functions
- Maximum likelihood approach
- Deriving common loss functions
 - Real-valued univariate regression
 - Binary classification
 - Multiclass classification
 - Multiple outputs (if extra time)
- Connections to cross entropy (if extra time)

Example 3: multiclass classification



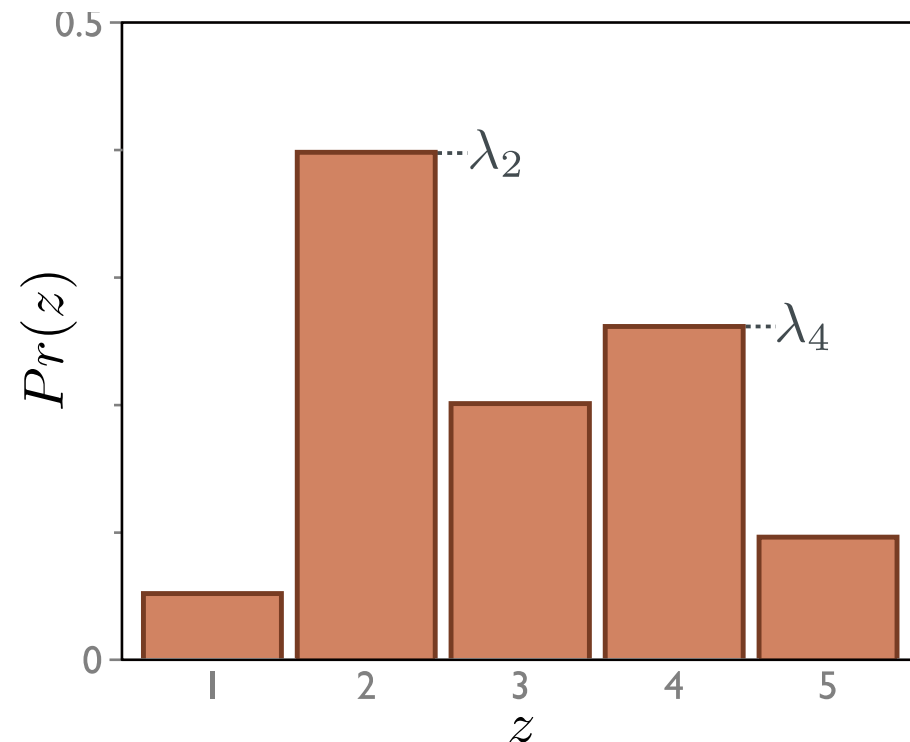
Goal: predict which of K classes $y \in \{1, 2, \dots, K\}$ the input x belongs to.

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 $\lambda_k \in [0, 1]$
- $\sum_k \lambda_k = 1$

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



Example 3: multiclass classification

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

Problem:

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

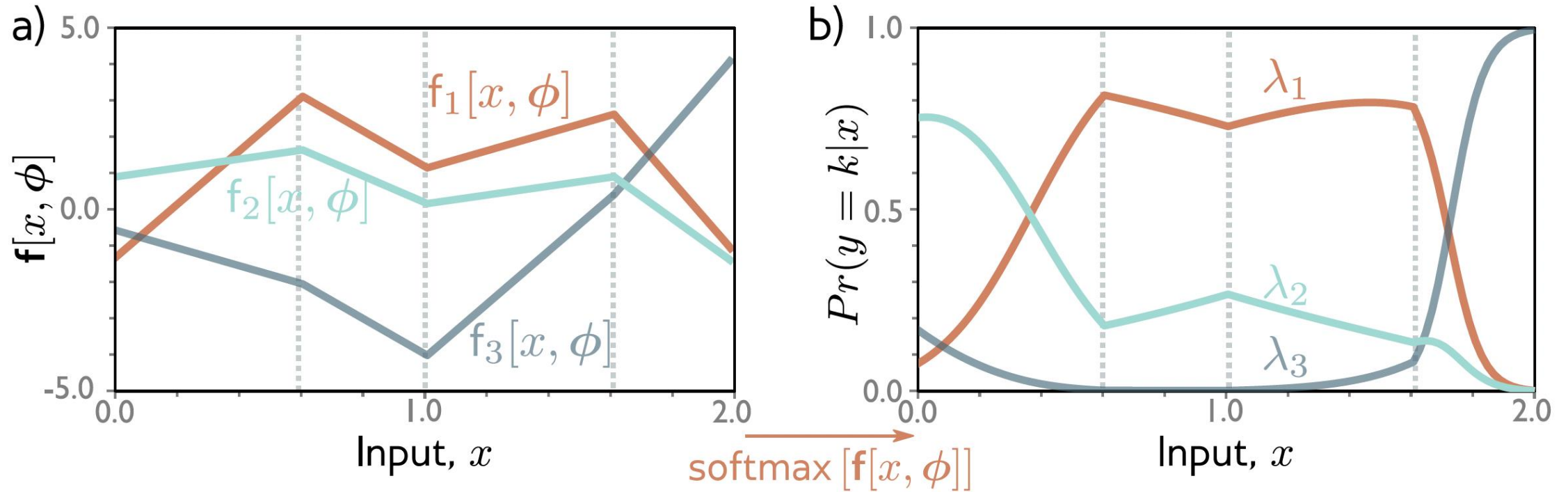
Solution:

- Pass through function that maps “anything” to $[0,1]$ and sums 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 $\{\mathbf{x}_i, \mathbf{y}_i\}$:

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

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

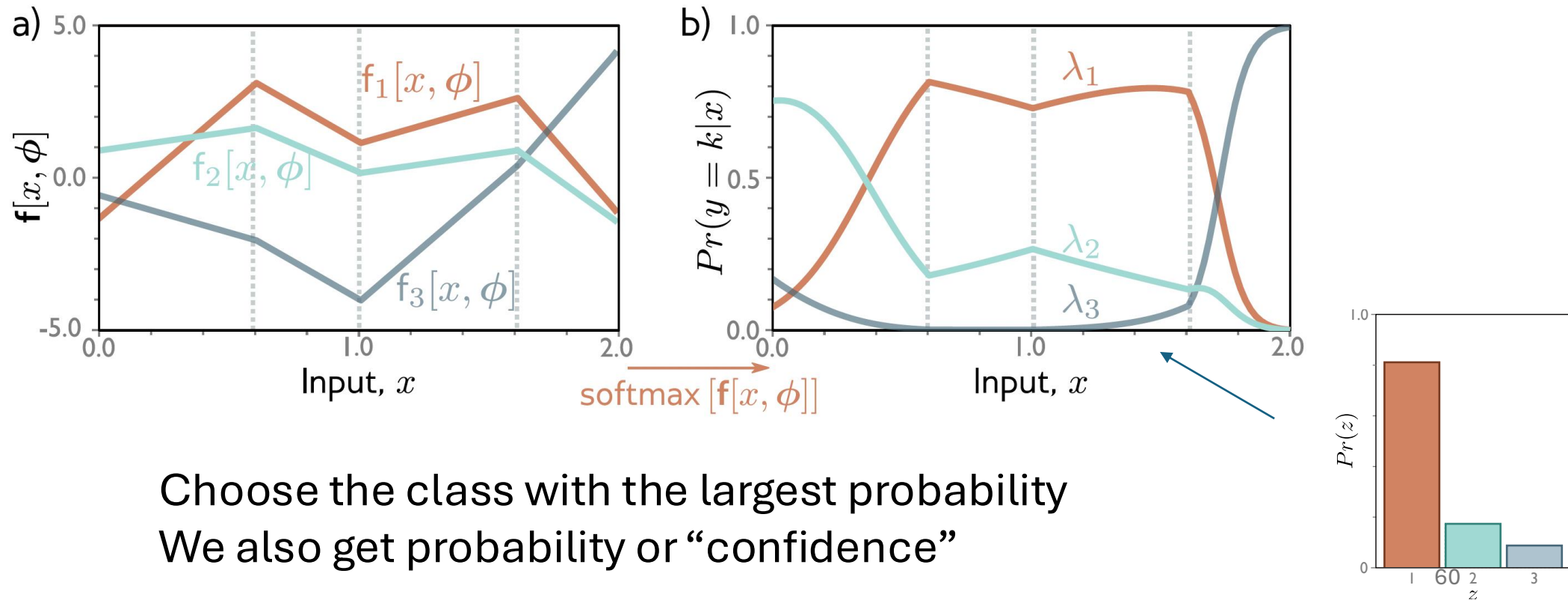
$$= - \sum_{i=1}^I f_{y_i} [\mathbf{x}_i, \phi] - \log \left[\sum_{k=1}^K \exp [f_k [\mathbf{x}_i, \phi]] \right]$$

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

Multiclass cross-entropy loss

Example 3: multiclass classification

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



Plan for Today

- Use cases for loss functions
- Maximum likelihood approach
- Deriving common loss functions
 - Real-valued univariate regression
 - Binary classification
 - Multiclass classification
 - Multiple outputs (if extra time)
- Connections to cross entropy (if extra time)

Multiple outputs

- Treat each output y_d as *independent*:

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

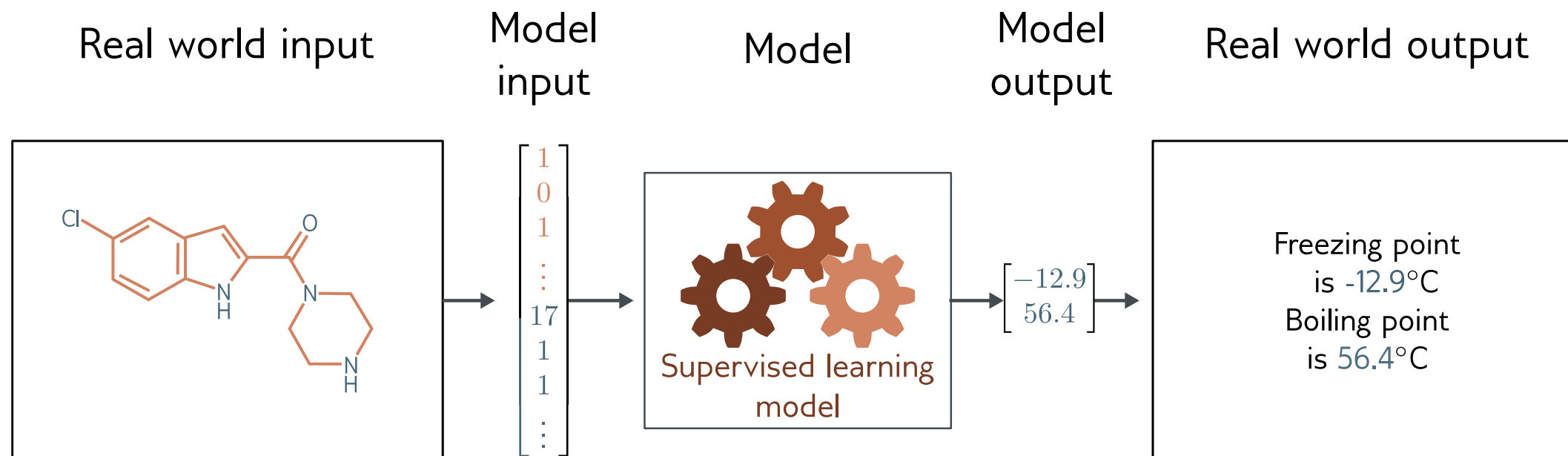
where $\mathbf{f}_d[\mathbf{x}, \phi]$ is the d^{th} set of network outputs

- Negative log likelihood becomes sum of terms:

$$L[\phi] = - \sum_{i=1}^I \log \left[Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}_i, \phi]) \right] = - \sum_{i=1}^I \sum_d \log \left[\underbrace{Pr(y_{id}|\mathbf{f}_d[\mathbf{x}_i, \phi])}_{d^{th} \text{ output of the } i^{th} \text{ training example}} \right]$$

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

Example 4: multivariate regression



Example 4: multivariate regression

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

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

- Make network with D_o outputs to predict means

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

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

Plan for Today

- Use cases for loss functions
- Maximum likelihood approach
- Deriving common loss functions
 - Real-valued univariate regression
 - Binary classification
 - Multiclass classification
 - Multiple outputs (if extra time)
- Connections to cross entropy (if extra time)

Cross-entropy loss

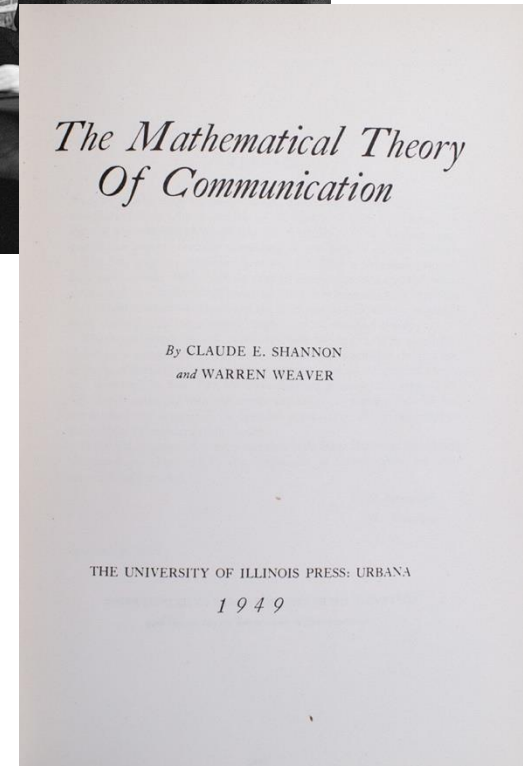
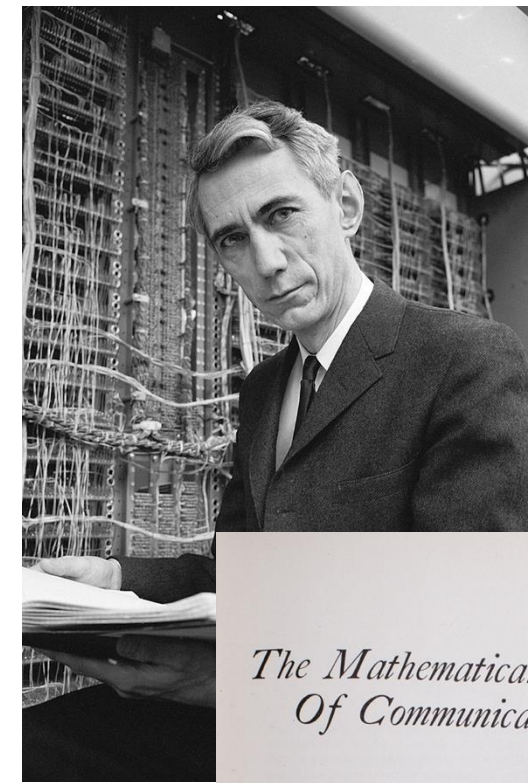
- So far we defined loss functions that minimize negative log-likelihood.
- Cross-entropy loss is common in neural network training.
- We can show that it is equivalent to negative log-likelihood

One can approach problems from different mathematical formulations.

Information Theory and Entropy

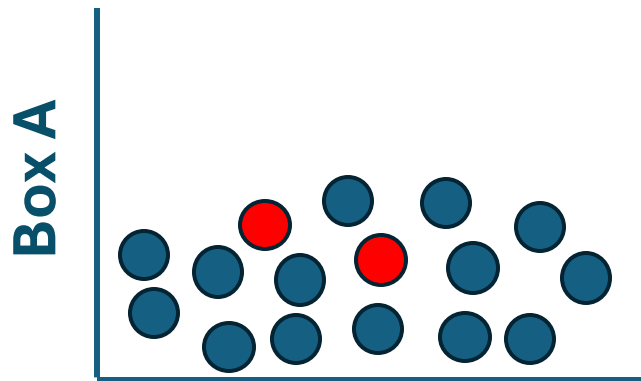
- **Claude Shannon:** the "father of information theory," was an American mathematician, electrical engineer, and cryptographer
- **Theory of Communication:** In his landmark 1948 paper, "A Mathematical Theory of Communication," Shannon introduced a formal framework for the transmission, processing,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

$$H(x) = - \sum_x P(x) \log_2(P(x))$$

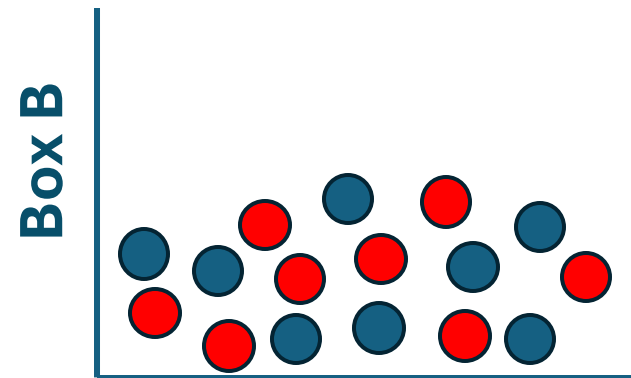


Entropy is a measure of *surprise* or *uncertainty*

Randomly pick a ball from the box



Low or High
Entropy?



Low or High
Entropy?

In class poll: <https://piazza.com/class/m5v834h9pcatx/post/27>

Connection to Deep Learning

Cross-Entropy Loss

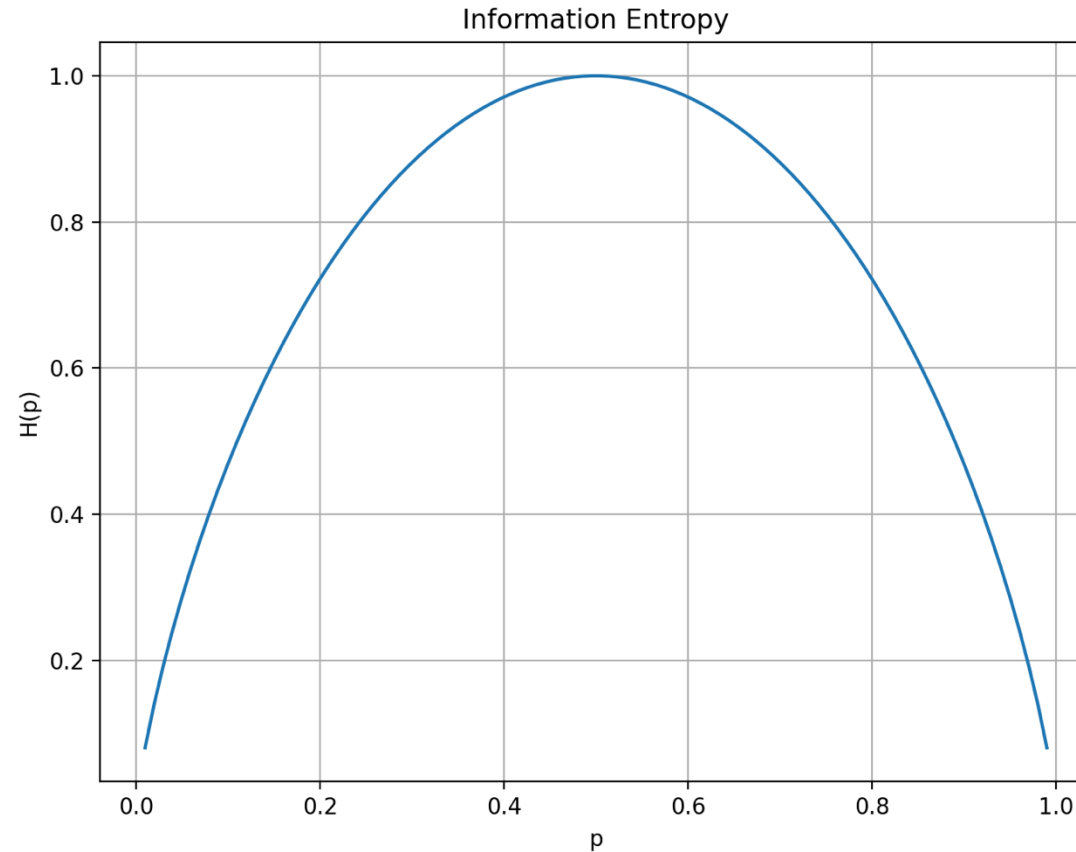
- If a neural network predicts **(0.25, 0.25, 0.25, 0.25)** for four possible classes, **high entropy** → uncertain.
- If it predicts **(0.99, 0.01, 0, 0)**, **low entropy** → confident.

Regularization & Overfitting

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

“Raise the temperature in LLMs.”

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

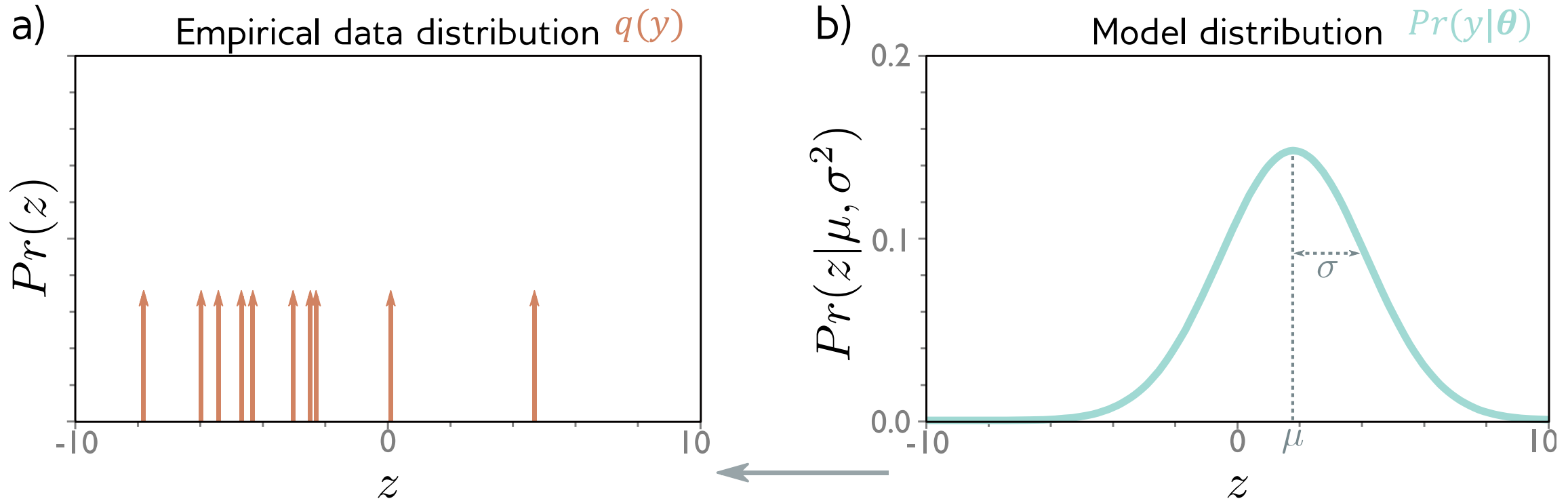


Peaks at 50/50.

$$H(x) = - \sum_x P(x) \log_2(P(x)) = -p \log_2(p) - (1 - p) \log_2(1 - p)$$

Cross Entropy – Concept from Information Theory

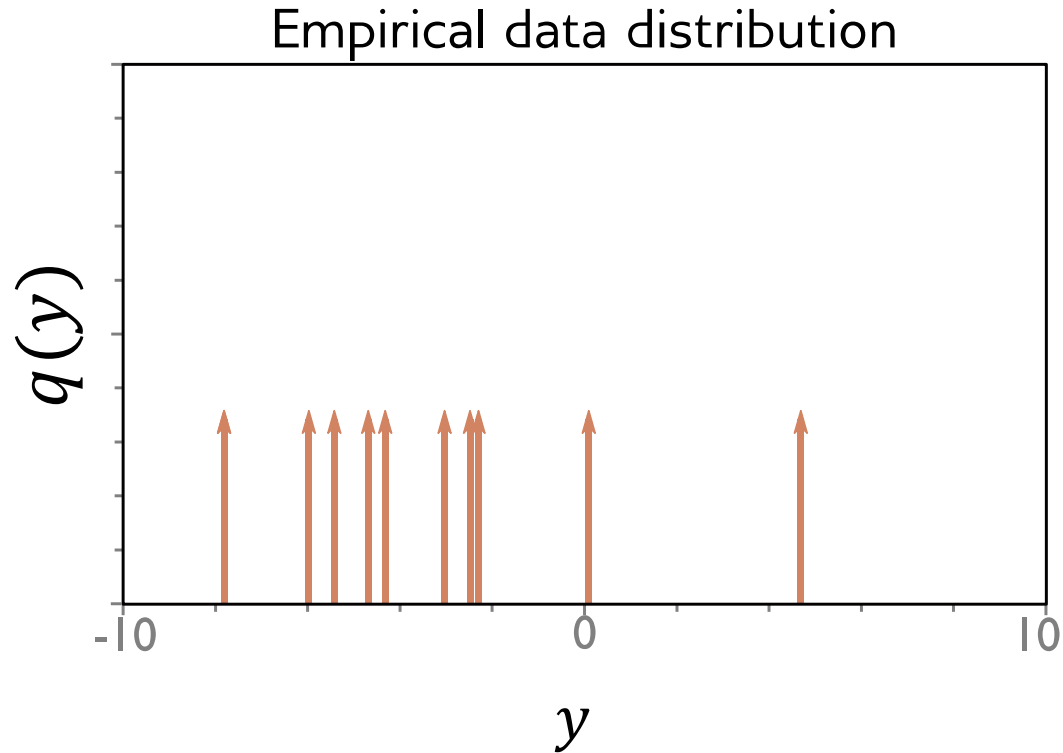
Measures the difference between the empirical distribution, $q(y)$, and a model distribution, $\Pr(y|\theta)$.



Kullback-Leibler Divergence -- a measure between probability distributions

Empirical Distribution – Collection of samples

Each sample represented by a shifted Dirac delta function.



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

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

So, we say empirical distribution is

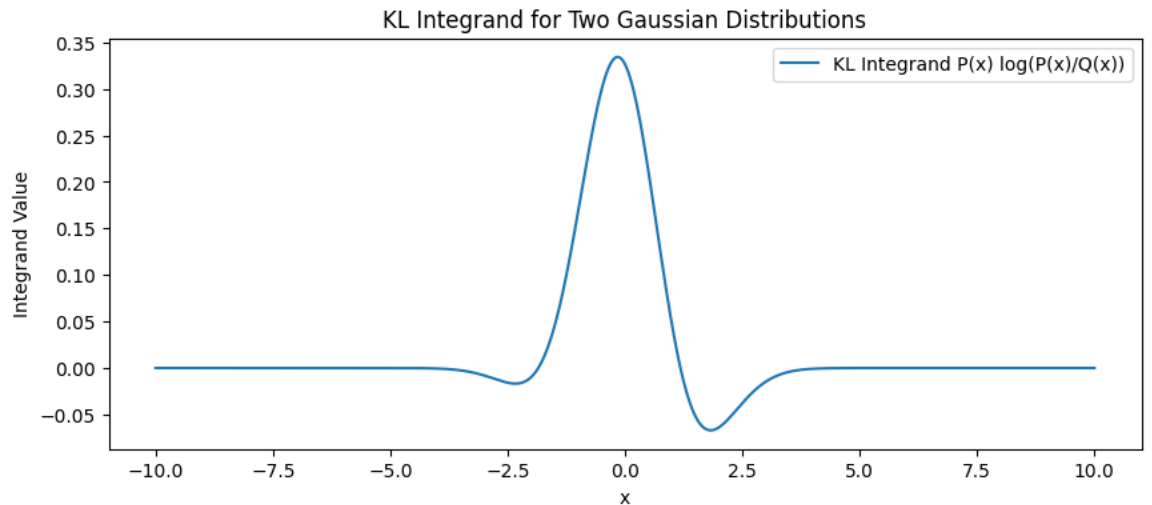
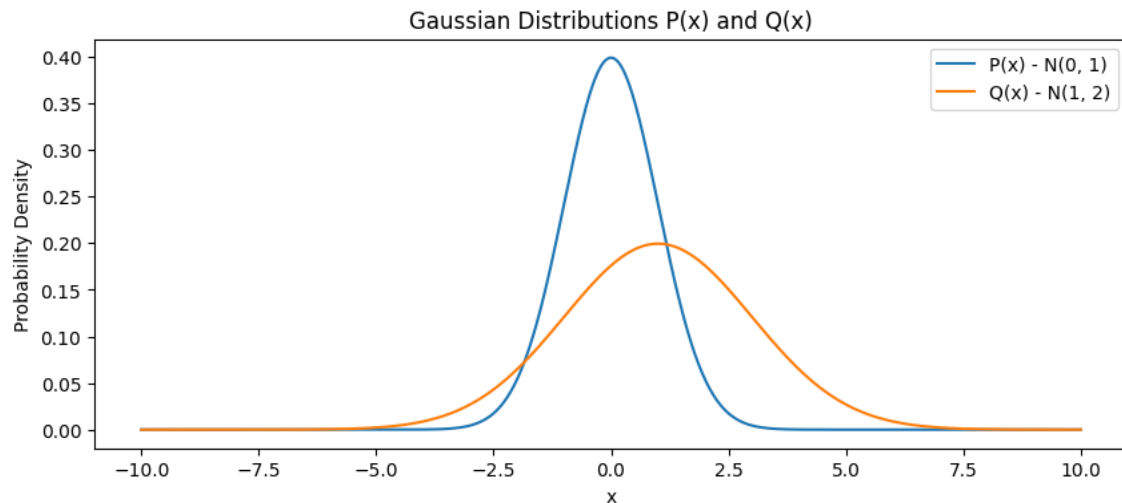
$$q(y) = \frac{1}{I} \sum_{i=1}^I \delta[y - y_i]$$

which will be helpful formulation in a moment.

Kullback-Leibler (KL) divergence

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

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

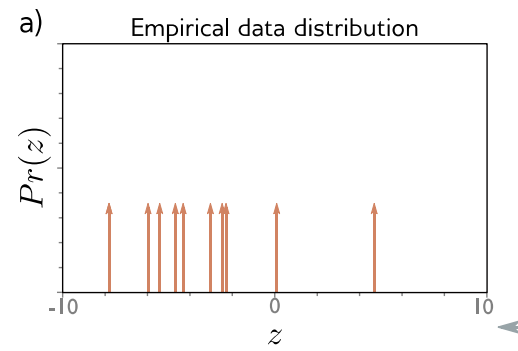


[Interactive Colab Notebook](#)

KL Divergence: 0.4431

Derivation

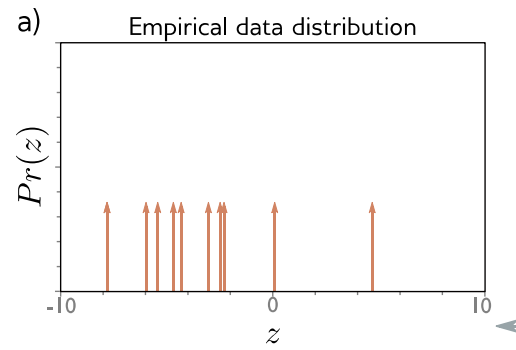
$$q(y) = \frac{1}{I} \sum_{i=1}^I \delta[y - y_i],$$



Training dataset as collection of Dirac delta functions.

Derivation

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



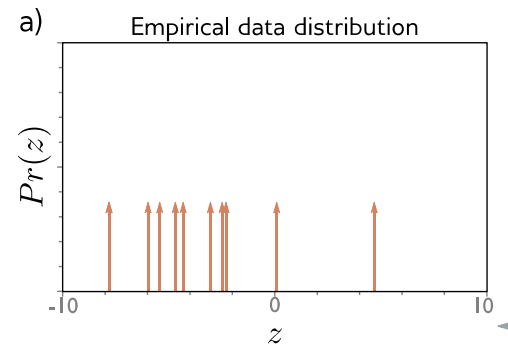
Training dataset as collection of Dirac delta functions.

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

Minimize KL divergence.

Derivation

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

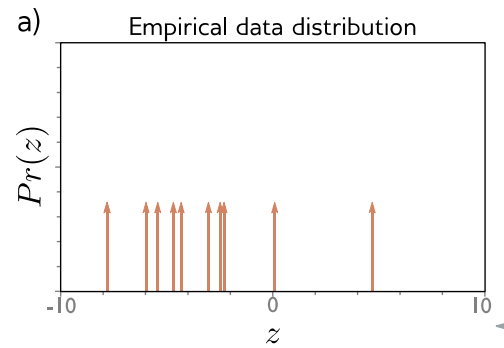


Training dataset as collection of Dirac delta functions.

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= \operatorname{argmin}_{\boldsymbol{\theta}} \left[\int_{-\infty}^{\infty} q(y) \log[q(y)] dy - \int_{-\infty}^{\infty} q(y) \log[Pr(y|\boldsymbol{\theta})] dy \right] && \text{Minimize KL divergence.} \\ &= \operatorname{argmin}_{\boldsymbol{\theta}} \left[- \int_{-\infty}^{\infty} q(y) \log[Pr(y|\boldsymbol{\theta})] dy \right], && \text{1st term not dependent on } \theta. \end{aligned}$$

Derivation

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



Training dataset as collection of Dirac delta functions.

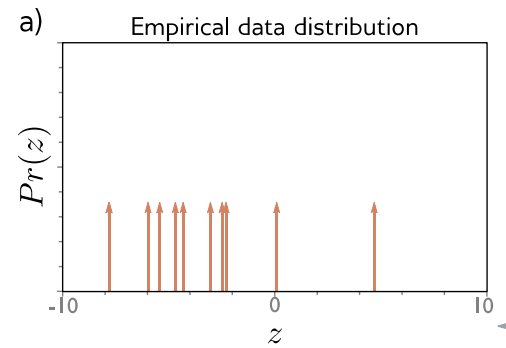
$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left[\int_{-\infty}^{\infty} q(y) \log[q(y)] dy - \int_{-\infty}^{\infty} q(y) \log[Pr(y|\theta)] dy \right] \quad \text{Minimize KL divergence.}$$

$$= \underset{\theta}{\operatorname{argmin}} \left[- \int_{-\infty}^{\infty} q(y) \log[Pr(y|\theta)] dy \right], \quad \text{1st term not dependent on } \theta.$$

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

Derivation

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



Training dataset as collection of Dirac delta functions.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left[\int_{-\infty}^{\infty} q(y) \log[q(y)] dy - \int_{-\infty}^{\infty} q(y) \log[Pr(y|\theta)] dy \right] \quad \text{Minimize KL divergence.}$$

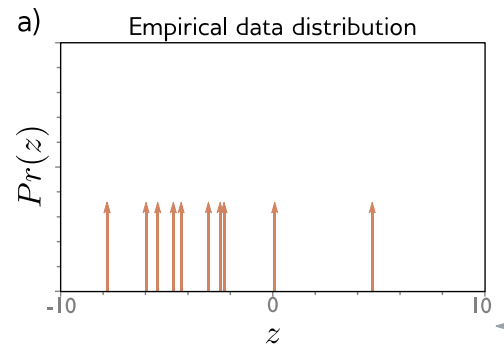
$$= \underset{\theta}{\operatorname{argmin}} \left[- \int_{-\infty}^{\infty} q(y) \log[Pr(y|\theta)] dy \right], \quad \text{1st term not dependent on } \theta.$$

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

$$= \underset{\theta}{\operatorname{argmin}} \left[- \frac{1}{I} \sum_{i=1}^I \log[Pr(y_i|\theta)] \right] \quad \text{Property of the Dirac delta function.}$$

Derivation

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



Training dataset as collection of Dirac delta functions.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left[\int_{-\infty}^{\infty} q(y) \log[q(y)] dy - \int_{-\infty}^{\infty} q(y) \log[Pr(y|\theta)] dy \right] \quad \text{Minimize KL divergence.}$$

$$= \underset{\theta}{\operatorname{argmin}} \left[- \int_{-\infty}^{\infty} q(y) \log[Pr(y|\theta)] dy \right], \quad \text{1st term not dependent on } \theta.$$

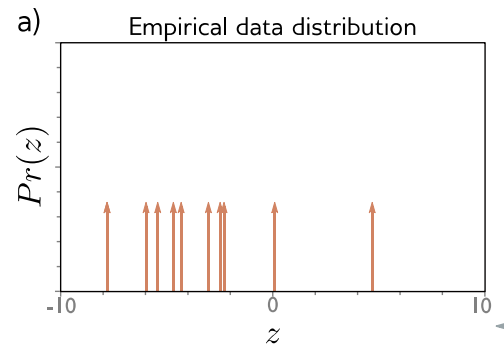
$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left[- \int_{-\infty}^{\infty} \left(\frac{1}{I} \sum_{i=1}^I \delta[y - y_i] \right) \log[Pr(y|\theta)] dy \right] \quad \text{Substituting for } q(y).$$

$$= \underset{\theta}{\operatorname{argmin}} \left[- \frac{1}{I} \sum_{i=1}^I \log[Pr(y_i|\theta)] \right] \quad \text{Property of the Dirac delta function.}$$

$$= \underset{\theta}{\operatorname{argmin}} \left[- \sum_{i=1}^I \log[Pr(y_i|\theta)] \right]. \quad \text{1/I is just a constant, so ignore.}$$

Derivation

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



Training dataset as collection of Dirac delta functions.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left[\int_{-\infty}^{\infty} q(y) \log[q(y)] dy - \int_{-\infty}^{\infty} q(y) \log[Pr(y|\theta)] dy \right] \quad \text{Minimize KL divergence.}$$

$$= \underset{\theta}{\operatorname{argmin}} \left[- \int_{-\infty}^{\infty} q(y) \log[Pr(y|\theta)] dy \right], \quad \text{1st term not dependent on } \theta.$$

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

$$= \underset{\theta}{\operatorname{argmin}} \left[- \frac{1}{I} \sum_{i=1}^I \log[Pr(y_i|\theta)] \right] \quad \text{Property of the Dirac delta function.}$$

$$= \underset{\theta}{\operatorname{argmin}} \left[- \sum_{i=1}^I \log[Pr(y_i|\theta)] \right]. \quad \text{1/I is just a constant, so ignore.}$$

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \left[- \sum_{i=1}^I \log[Pr(y_i|\mathbf{f}[\mathbf{x}_i, \phi])] \right] \quad \text{Model is predicting } \theta \rightarrow \text{Negative Log Likelihood!!}$$

Minimizing Negative Log Likelihood (or equivalently KL Divergence)

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