

# **CS/DS 541: Class 2**

Jacob Whitehill

# Linear regression (aka 2-layer NN)

# Linear regression

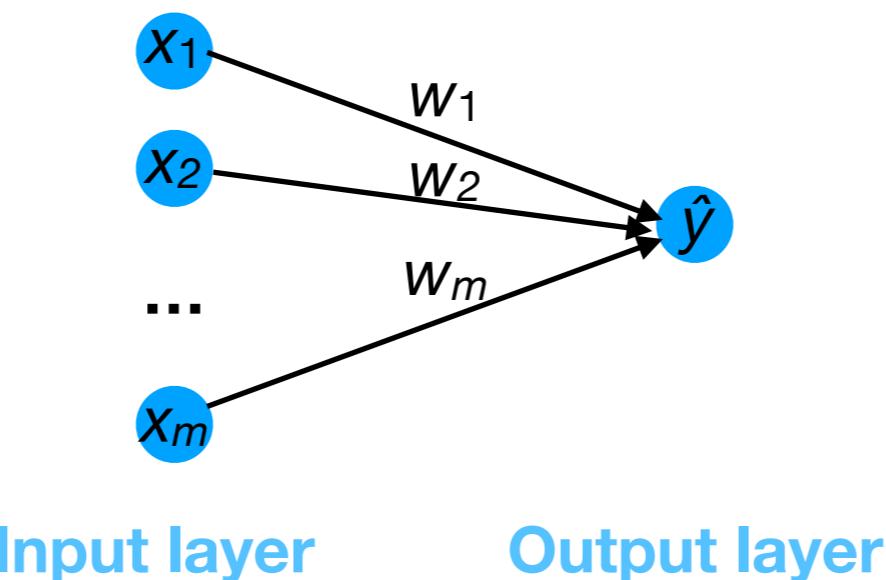
- Let dataset  $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n$
- Want to create a neural network — which we can also treat as a “machine” — to estimate each  $y^{(i)}$  with high accuracy.
- Let us define the machine by a function  $g$  (with parameters  $\mathbf{w}$ ) whose output  $\hat{y}$  is linear in its inputs:

$$\hat{y} \doteq g(\mathbf{x}; \mathbf{w}) \doteq \sum_{i=1}^m x_i w_i = \mathbf{x}^\top \mathbf{w}$$

# Linear regression

- Note that this function is equivalent to a 2-layer neural network (with no activation function):

$$\hat{y} \doteq g(\mathbf{x}; \mathbf{w}) \doteq \sum_{i=1}^m x_i w_i = \mathbf{x}^\top \mathbf{w}$$



# Solving for $\mathbf{w}$

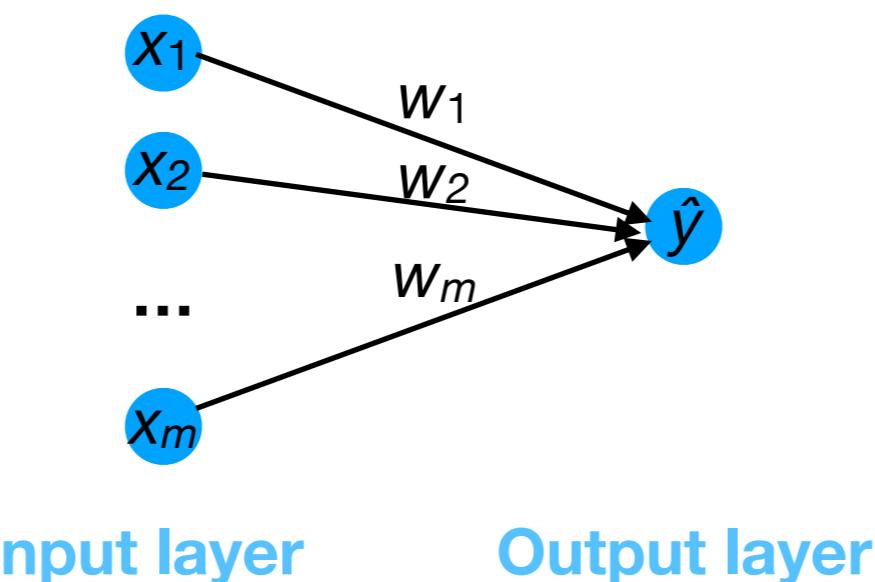
- By setting to 0, splitting the sum apart, and solving, we reach the solution:

$$\begin{aligned}\nabla_{\mathbf{w}} f_{\text{MSE}}(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{w}) &= \frac{1}{n} \sum_{i=1}^n \mathbf{x}^{(i)} \left( \mathbf{x}^{(i)\top} \mathbf{w} - y^{(i)} \right) \\ 0 &= \sum_i \mathbf{x}^{(i)} \mathbf{x}^{(i)\top} \mathbf{w} - \sum_i \mathbf{x}^{(i)} y^{(i)} \\ \sum_i \mathbf{x}^{(i)} \mathbf{x}^{(i)\top} \mathbf{w} &= \sum_i \mathbf{x}^{(i)} y^{(i)} \\ \mathbf{w} &= \left( \sum_i \mathbf{x}^{(i)} \mathbf{x}^{(i)\top} \right)^{-1} \sum_i \mathbf{x}^{(i)} y^{(i)}\end{aligned}$$

# Solving for $\mathbf{w}$

- We have found the optimal  $\mathbf{w}$  to minimize  $f_{\text{MSE}}$ .
- In other words, we can optimize a 2-layer linear NN for the MSE loss using the closed formula:

$$\mathbf{w} = \left( \sum_i \mathbf{x}^{(i)} \mathbf{x}^{(i)\top} \right)^{-1} \sum_i \mathbf{x}^{(i)} y^{(i)}$$



# Matrix notation

- We can also derive the same solution using matrix notation:
- Let's define a matrix  $\mathbf{X}$  to contain all the training images:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \dots & \mathbf{x}^{(n)} \end{bmatrix}$$

- In statistics,  $\mathbf{X}$  is called the **design matrix**.\*
- Let's define vector  $\mathbf{y}$  to contain all the training labels:

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

\* Actually, statistics literature typically defines this as  $\mathbf{X}^T$ .

# Matrix notation

- Using summation notation, we derived:

$$\mathbf{w} = \left( \sum_{i=1}^n \mathbf{x}^{(i)} \mathbf{x}^{(i)\top} \right)^{-1} \left( \sum_{i=1}^n \mathbf{x}^{(i)} y^{(i)} \right)$$

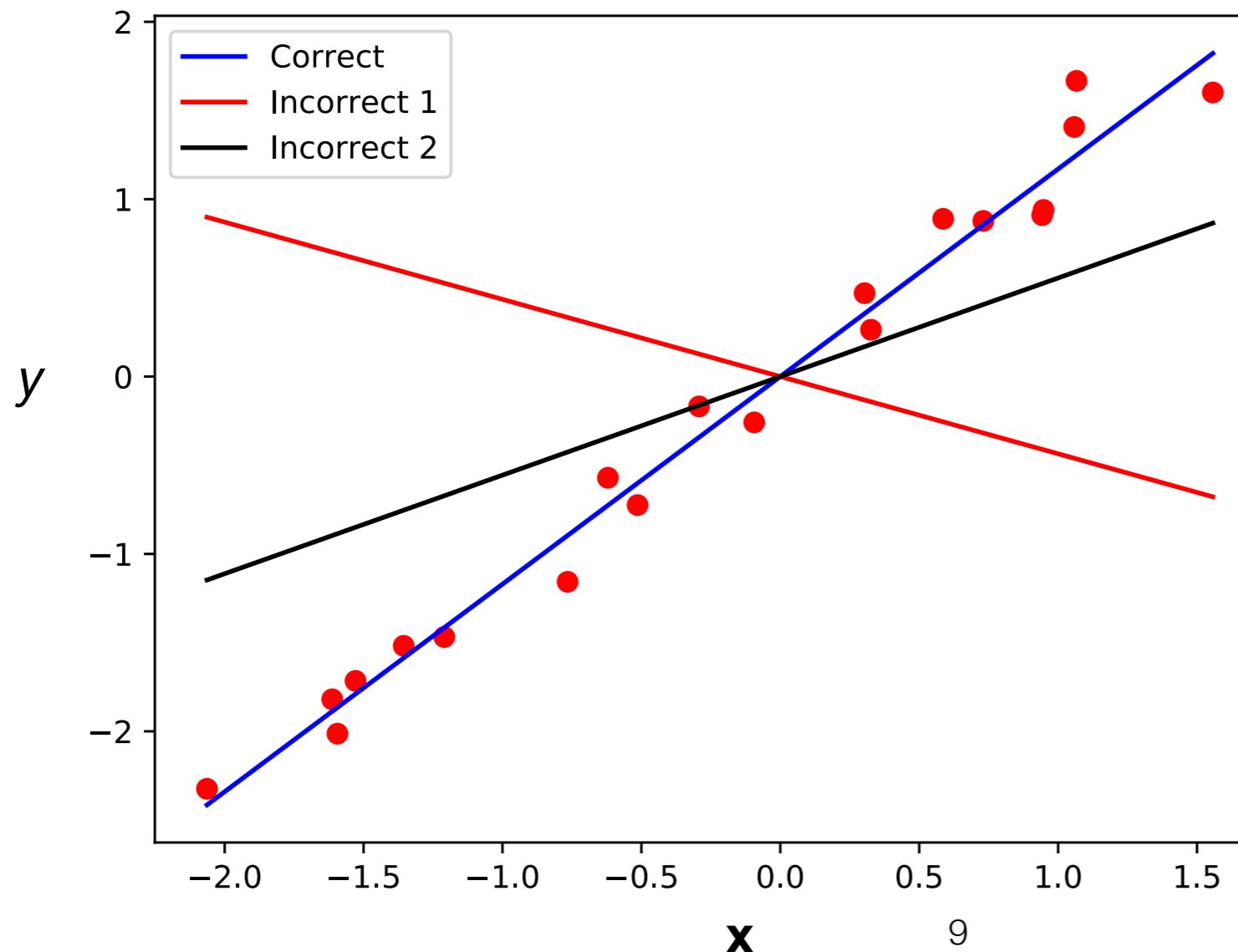
- Using matrix notation, we can write the solution as:

$$\mathbf{w} = (\mathbf{X}\mathbf{X}^\top)^{-1} \mathbf{X}\mathbf{y}$$

where  $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} & \dots & \mathbf{x}^{(n)} \end{bmatrix}$

# 1-d example

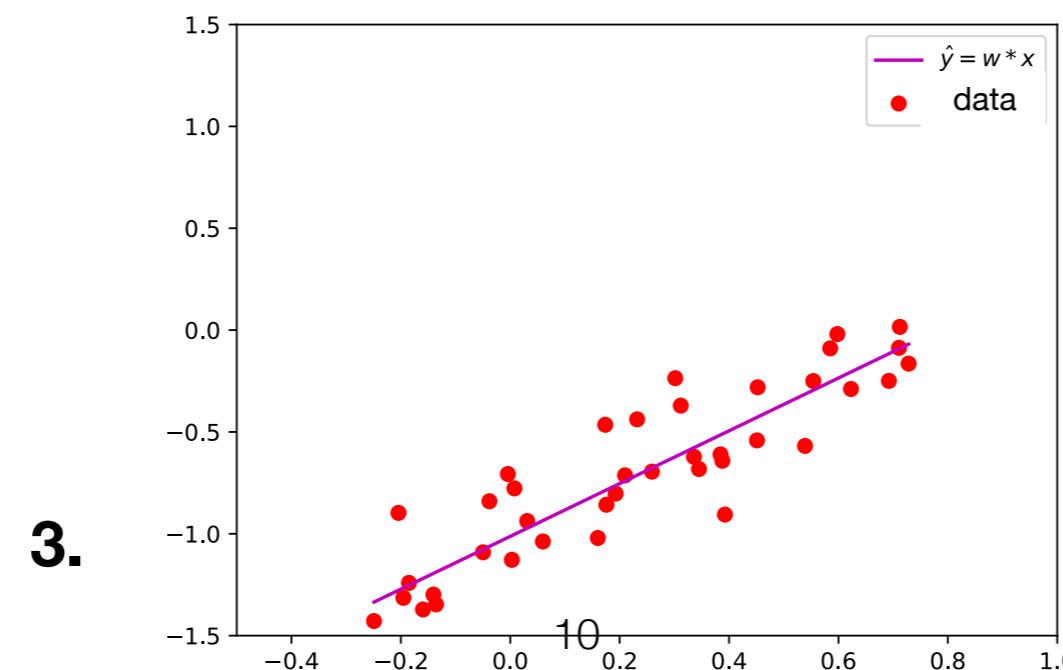
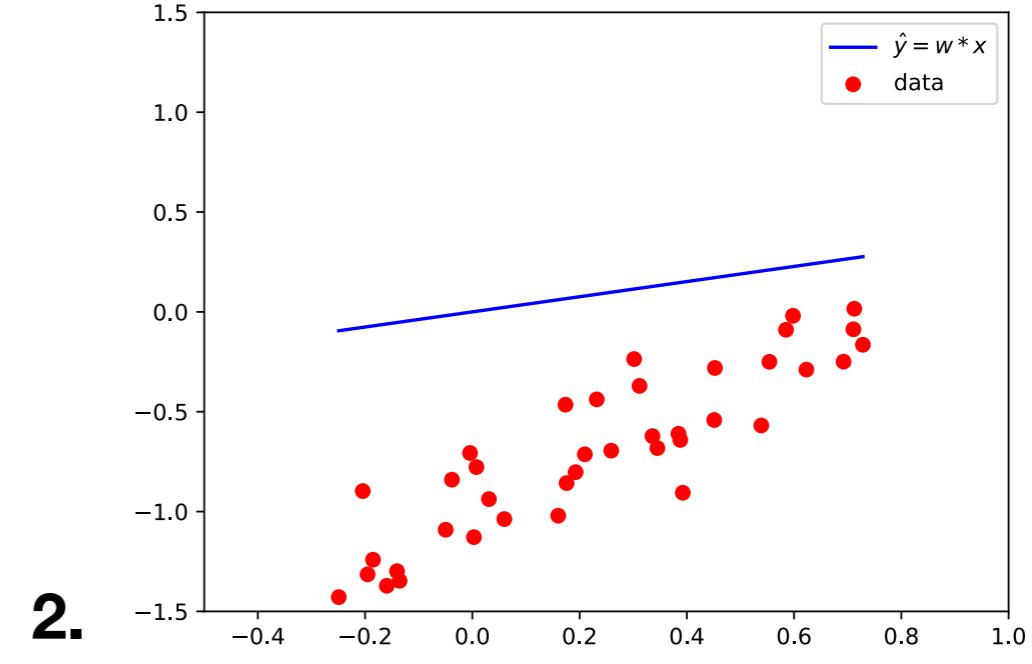
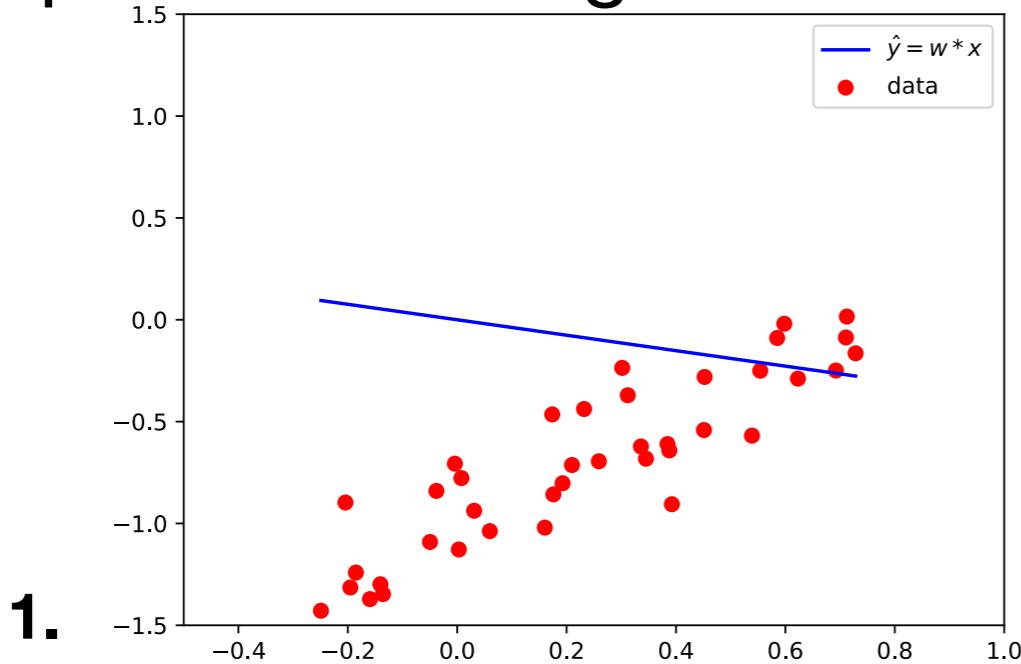
- Linear regression finds the weight vector  $\mathbf{w}$  that minimizes the  $f_{\text{MSE}}$ . Here's an example where each  $\mathbf{x}$  is just 1-d...



The best  $\mathbf{w}$  is the one such that  $f_{\text{MSE}}(\mathbf{y}, \hat{\mathbf{y}})$  is as small as possible, where each  $\hat{\mathbf{y}} = \mathbf{x}^T \mathbf{w}$ .

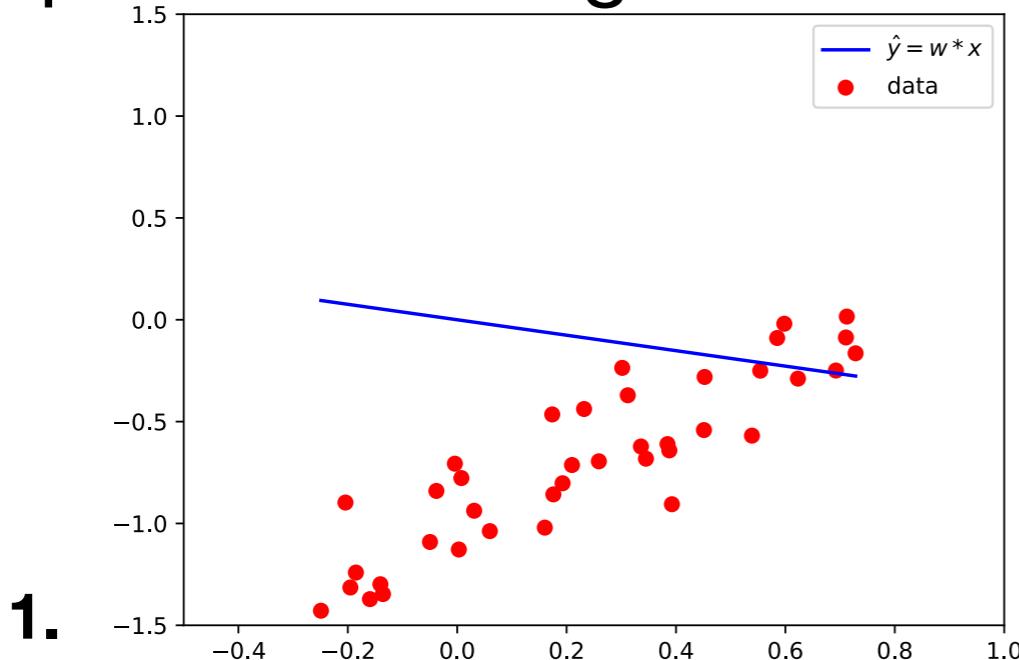
# Exercise

- Which of the following regression lines would be predicted using the model described above?



# Exercise

- Which of the following regression lines would be predicted using the model described above?



Notice that the model enforces that  $(x,y)=(0,0)$  lie in the graph. Because of this constraint, the model learns the wrong slope to minimize the MSE.

# Bias term

- In order to account for target values with non-zero mean, we can add a bias term to our model:

$$\hat{y} = \mathbf{x}^\top \mathbf{w} + b$$

- We can then compute the gradient w.r.t. both  $\mathbf{w}$  and  $b$ , set to 0, and then solve the resulting system of equations.

$$\nabla_{\mathbf{w}} f_{\text{MSE}}(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{w}, b) = \nabla_{\mathbf{w}} \left[ \frac{1}{2n} \sum_{i=1}^n \left( \mathbf{x}^{(i)\top} \mathbf{w} + b - y^{(i)} \right)^2 \right]$$

$$\nabla_b f_{\text{MSE}}(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{w}, b) = \nabla_b \left[ \frac{1}{2n} \sum_{i=1}^n \left( \mathbf{x}^{(i)\top} \mathbf{w} + b - y^{(i)} \right)^2 \right]$$

# Bias term

- Alternatively, we can implicitly include a bias term by augmenting each input vector  $\mathbf{x}$  with a 1 at the end:

$$\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$$

- Correspondingly, our weight vector  $\mathbf{w}$  will have an extra component (bias term) at the end.

$$\tilde{\mathbf{w}} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

# Bias term

- To see why, notice that:

$$\begin{aligned}\hat{y} &= \tilde{\mathbf{x}}^\top \tilde{\mathbf{w}} \\ &= \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{x}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} \\ &= \mathbf{x}^\top \mathbf{w} + b\end{aligned}$$

# Bias term

- We can find the optimal  $\mathbf{w}$  and  $b$  based on all the training data using matrix notation.
- First define an augmented design matrix:

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{x}^{(1)} & \dots & \mathbf{x}^{(n)} \\ 1 & \dots & 1 \end{bmatrix}$$

- Then compute:

$$\tilde{\mathbf{w}} = (\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top)^{-1}\tilde{\mathbf{X}}\mathbf{y}$$

# Regularization

- The larger the coefficients (weights)  $\mathbf{w}$  are allowed to be, the more the neural network can overfit.
- If we “encourage” the weights to be small, we can reduce overfitting.
- This is a form of **regularization** — any practice designed to improve the machine’s ability to **generalize** to new data.

# Regularization

- One of the simplest and oldest regularization techniques is to *penalize* large weights in the cost function.
- The “unregularized”  $f_{\text{MSE}}$  is:

$$f_{\text{MSE}}(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2$$

# Regularization

- One of the simplest and oldest regularization techniques is to *penalize* large weights in the cost function.
- The “unregularized”  $f_{\text{MSE}}$  is:

$$f_{\text{MSE}}(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2$$

- The  $L_2$ -regularized  $f_{\text{MSE}}$  becomes:

$$f_{\text{MSE}}(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2 + \frac{\alpha}{2} \mathbf{w}^\top \mathbf{w}$$

# Gradient descent

# Linear regression

- Linear regression is one of the few ML algorithms that has an analytical solution:

$$\mathbf{w} = (\mathbf{X}\mathbf{X}^\top)^{-1} \mathbf{X}\mathbf{y}$$

- **Analytical solution:** there is a closed formula for the answer.

# Linear regression

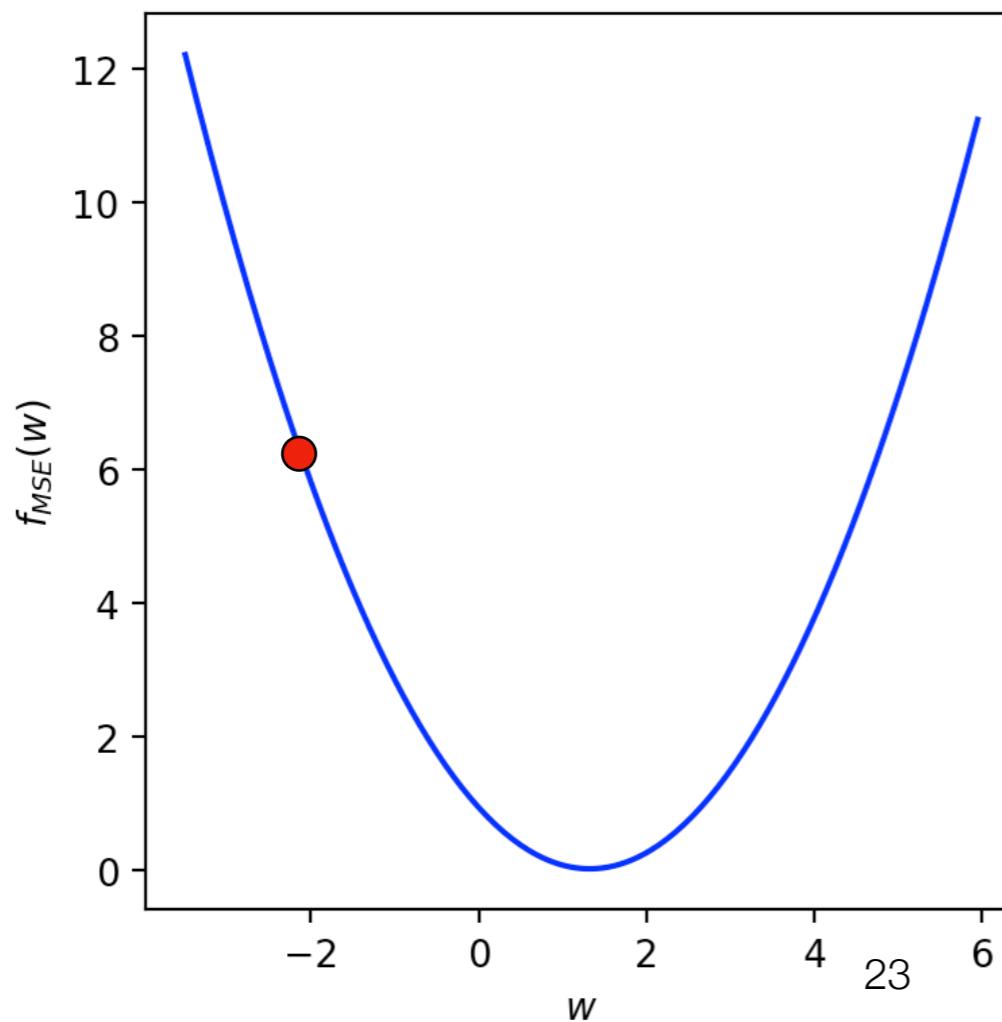
- Alternatively, linear regression can be solved numerically using gradient descent.
- **Numerical solution:** need to iterate (according to some algorithm) many times to *approximate* the optimal value.
- Gradient descent is more laborious to code than the one-shot solution, but it generalizes to a wide variety of ML models.

# Gradient descent

- Gradient descent is a **hill climbing algorithm** that uses the gradient (aka slope) to decide which way to “move”  $w$  to reduce the objective function (e.g.,  $f_{\text{MSE}}$ ).

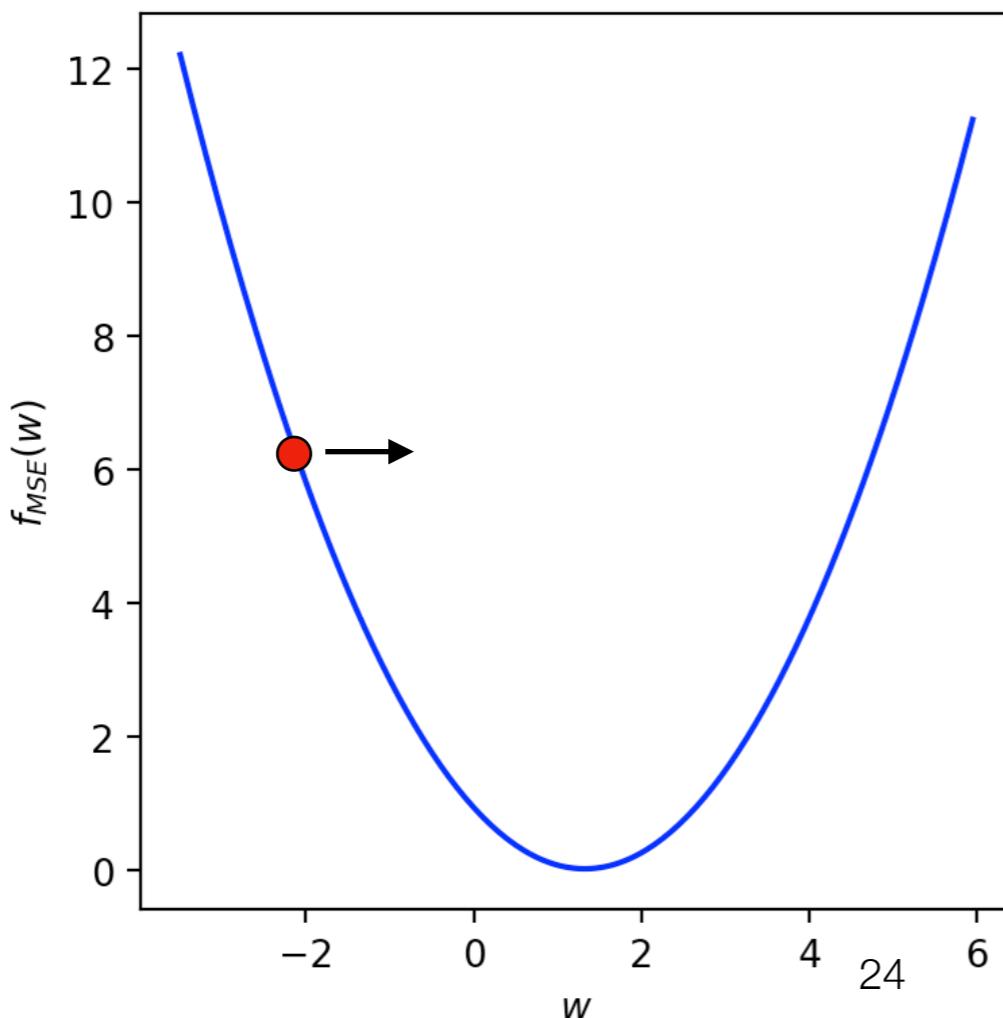
# Gradient descent

- Suppose we just guess an initial value for  $w$  (e.g., -2.1).
- How can we make it better – increase it or decrease it?



# Gradient descent

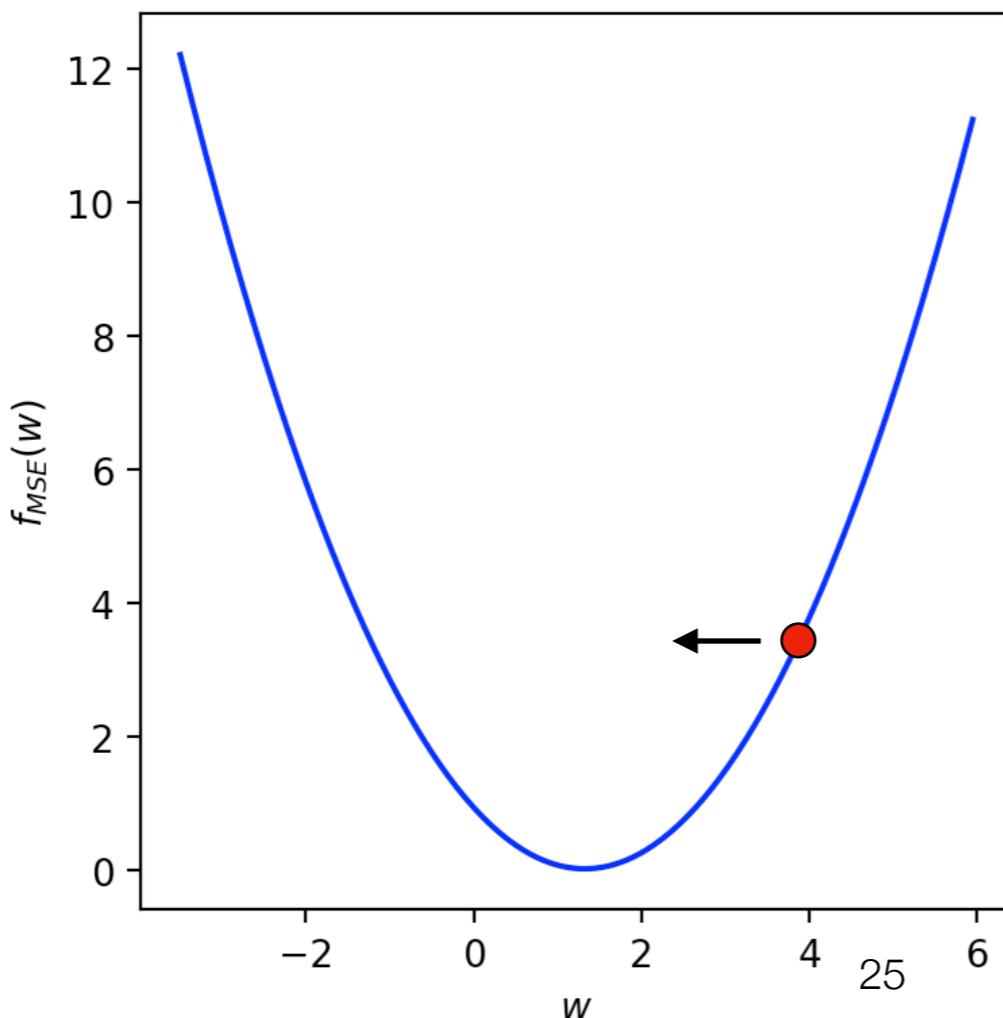
- Suppose we just guess an initial value for  $w$  (e.g., -2.1).
- How can we make it better — **increase** it or decrease it?
  - What does the **slope** of  $f_{\text{MSE}}$  tell us to do?



The slope at  $f_{\text{MSE}}(-2.1)$  is negative, i.e., we can decrease our cost by increasing  $w$ .

# Gradient descent

- Or maybe our initial guess for  $w$  was 3.9.
- How can we make it better – increase it or **decrease** it?
  - What does the **slope** of  $f_{\text{MSE}}$  tell us to do?



The slope at  $f_{\text{MSE}}(3.9)$  is *positive*, i.e., we can *decrease* our cost by *decreasing*  $w$ .

# Gradient descent

- How do we know the slope? Compute the **gradient** of  $f_{\text{MSE}}$  w.r.t.  $\mathbf{w}$ :

$$\begin{aligned}\nabla_{\mathbf{w}} f_{\text{MSE}}(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{w}) &= \nabla_{\mathbf{w}} \left[ \frac{1}{2n} \sum_{i=1}^n \left( \mathbf{x}^{(i)\top} \mathbf{w} - y^{(i)} \right)^2 \right] \\ &= \frac{1}{2n} \sum_{i=1}^n \nabla_{\mathbf{w}} \left[ \left( \mathbf{x}^{(i)\top} \mathbf{w} - y^{(i)} \right)^2 \right] \\ &= \frac{1}{n} \sum_{i=1}^n \mathbf{x}^{(i)} \left( \mathbf{x}^{(i)\top} \mathbf{w} - y^{(i)} \right) \\ &= \frac{1}{n} \mathbf{X} (\mathbf{X}^\top \mathbf{w} - \mathbf{y})\end{aligned}$$

# Gradient descent

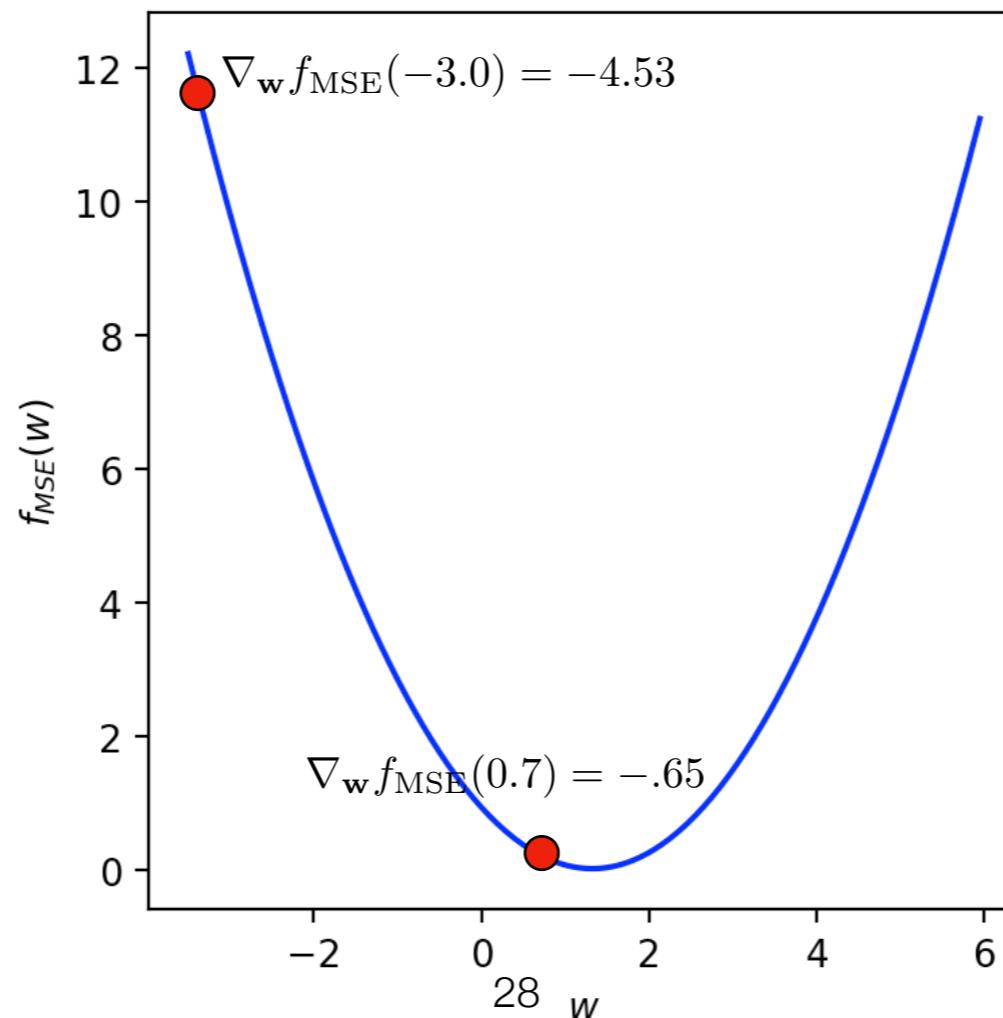
- How do we know the slope? Compute the **gradient** of  $f_{\text{MSE}}$  w.r.t.  $\mathbf{w}$ :

$$\begin{aligned}\nabla_{\mathbf{w}} f_{\text{MSE}}(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{w}) &= \nabla_{\mathbf{w}} \left[ \frac{1}{2n} \sum_{i=1}^n \left( \mathbf{x}^{(i)\top} \mathbf{w} - y^{(i)} \right)^2 \right] \\ &= \frac{1}{2n} \sum_{i=1}^n \nabla_{\mathbf{w}} \left[ \left( \mathbf{x}^{(i)\top} \mathbf{w} - y^{(i)} \right)^2 \right] \\ &= \frac{1}{n} \sum_{i=1}^n \mathbf{x}^{(i)} \left( \mathbf{x}^{(i)\top} \mathbf{w} - y^{(i)} \right) \\ &= \frac{1}{n} \mathbf{X} (\mathbf{X}^\top \mathbf{w} - \mathbf{y})\end{aligned}$$

- Then plug in the current value of  $\mathbf{w}$ .  
(Note that  $\mathbf{X}$  and  $\mathbf{y}$  are computed from the data and are constant.)

# Gradient descent

- How *far* do we “move” left or right?
  - Notice that, in the graph below, the **magnitude** of the slope (aka gradient) gives an indication of how far we need to go to reach the optimal  $\mathbf{w}$ .

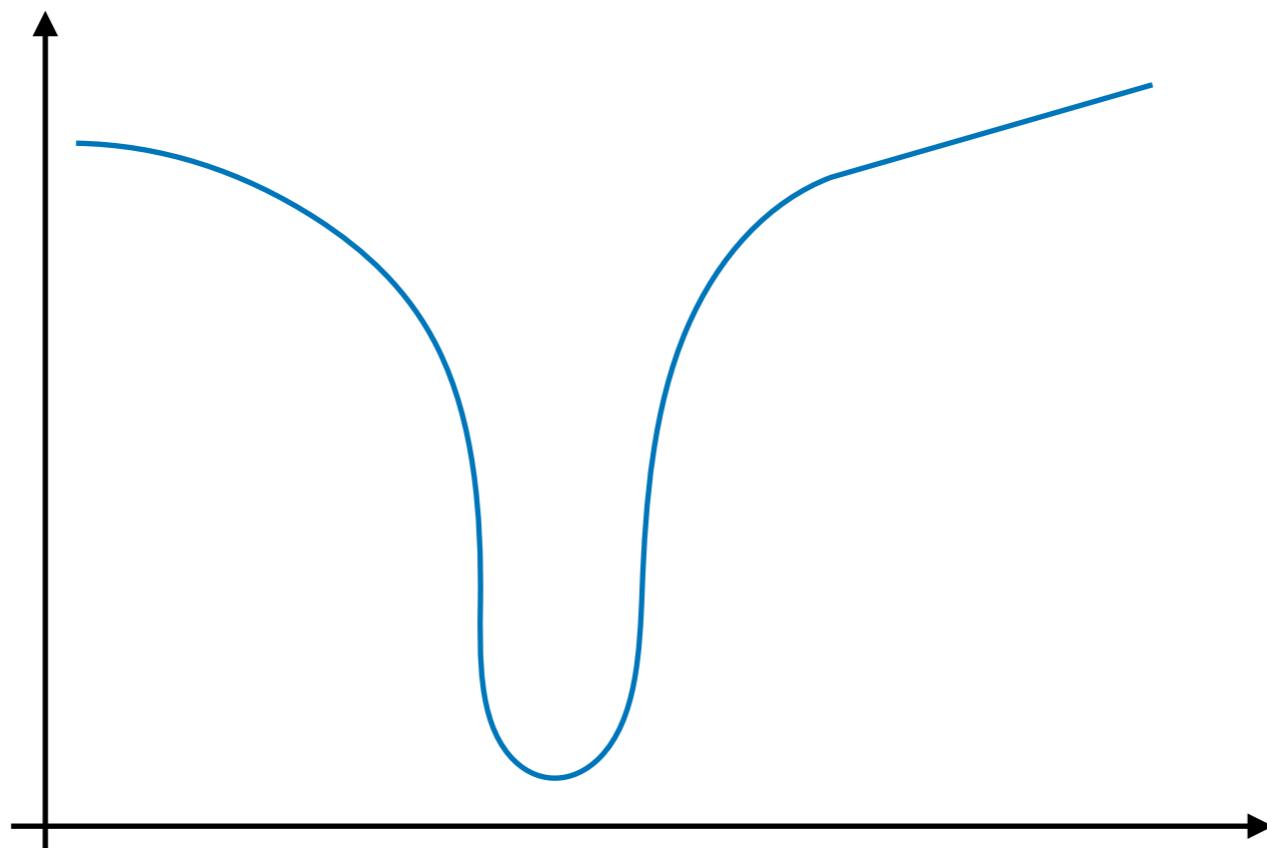


# Exercise

- Draw on paper a function such that this property is false.

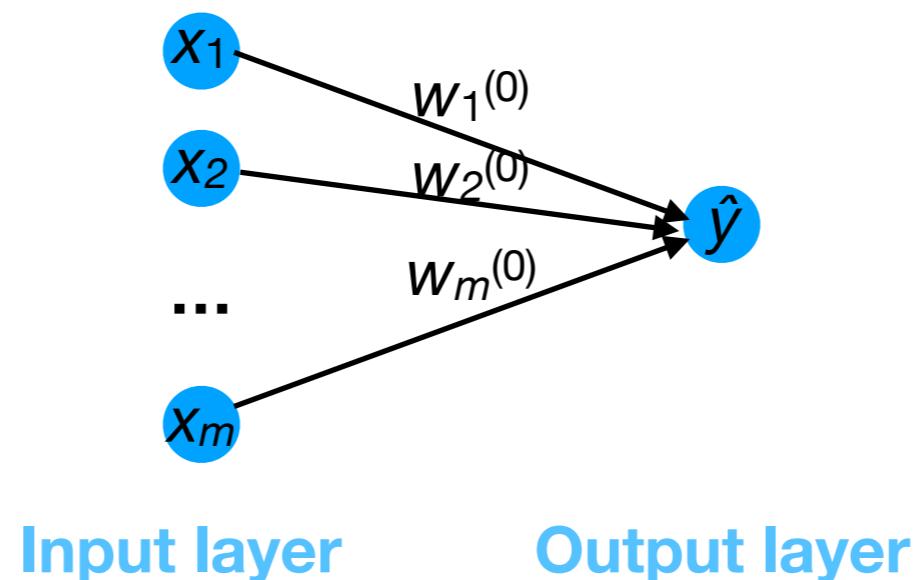
# Exercise

- Draw on paper a function such that this property is false.



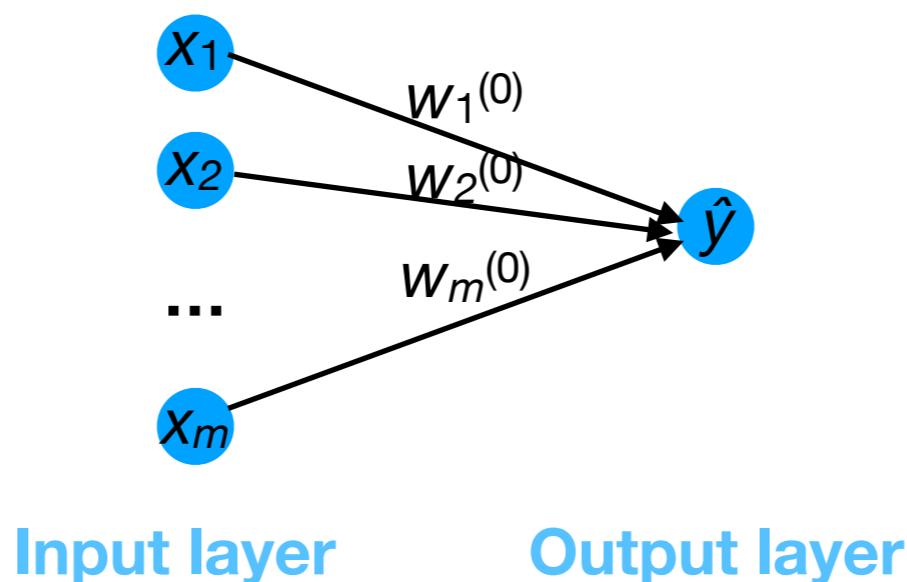
# Gradient descent algorithm

- Set  $\mathbf{w}$  to random values; call this initial choice  $\mathbf{w}^{(0)}$ .



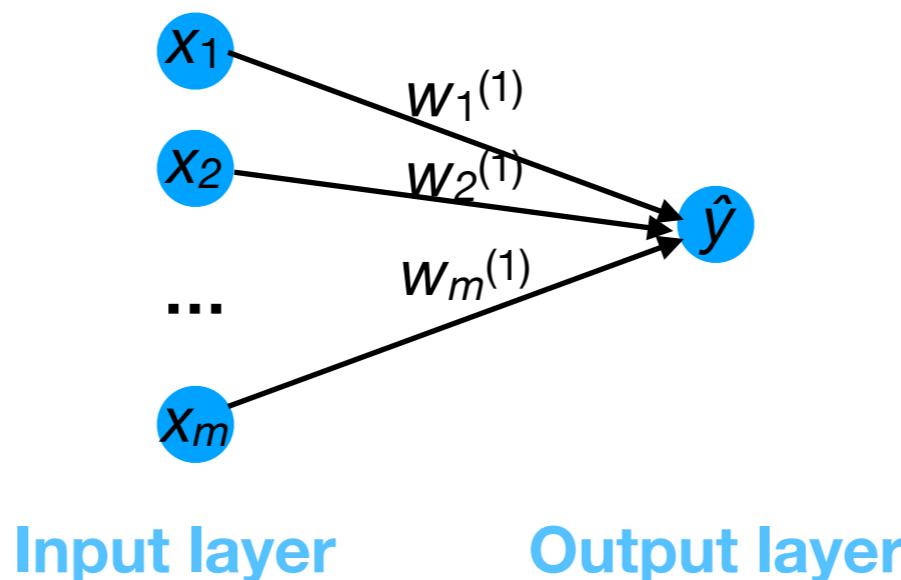
# Gradient descent algorithm

- Set  $\mathbf{w}$  to random values; call this initial choice  $\mathbf{w}^{(0)}$ .
- Compute the gradient:  $\nabla_{\mathbf{w}} f(\mathbf{w}^{(0)})$



# Gradient descent algorithm

- Set  $\mathbf{w}$  to random values; call this initial choice  $\mathbf{w}^{(0)}$ .
- Compute the gradient:  $\nabla_{\mathbf{w}} f(\mathbf{w}^{(0)})$
- Update  $\mathbf{w}$  by moving opposite the gradient, multiplied by a **learning rate**  $\epsilon$ .  $\mathbf{w}^{(1)} \leftarrow \mathbf{w}^{(0)} - \epsilon \nabla_{\mathbf{w}} f(\mathbf{w}^{(0)})$



# Gradient descent algorithm

- Set  $\mathbf{w}$  to random values; call this initial choice  $\mathbf{w}^{(0)}$ .
- Compute the gradient:  $\nabla_{\mathbf{w}} f(\mathbf{w}^{(0)})$
- Update  $\mathbf{w}$  by moving opposite the gradient, multiplied by a **learning rate**  $\epsilon$ .  
$$\mathbf{w}^{(1)} \leftarrow \mathbf{w}^{(0)} - \epsilon \nabla_{\mathbf{w}} f(\mathbf{w}^{(0)})$$
- Repeat...  
$$\mathbf{w}^{(2)} \leftarrow \mathbf{w}^{(1)} - \epsilon \nabla_{\mathbf{w}} f(\mathbf{w}^{(1)})$$
$$\mathbf{w}^{(3)} \leftarrow \mathbf{w}^{(2)} - \epsilon \nabla_{\mathbf{w}} f(\mathbf{w}^{(2)})$$
  
...
- $\mathbf{w}^{(t)} \leftarrow \mathbf{w}^{(t-1)} - \epsilon \nabla_{\mathbf{w}} f(\mathbf{w}^{(t-1)})$

# Gradient descent algorithm

- Set  $\mathbf{w}$  to random values; call this initial choice  $\mathbf{w}^{(0)}$ .
- Compute the gradient:  $\nabla_{\mathbf{w}} f(\mathbf{w}^{(0)})$
- Update  $\mathbf{w}$  by moving opposite the gradient, multiplied by a **learning rate**  $\epsilon$ .  
$$\mathbf{w}^{(1)} \leftarrow \mathbf{w}^{(0)} - \epsilon \nabla_{\mathbf{w}} f(\mathbf{w}^{(0)})$$
- Repeat...  
$$\mathbf{w}^{(2)} \leftarrow \mathbf{w}^{(1)} - \epsilon \nabla_{\mathbf{w}} f(\mathbf{w}^{(1)})$$
$$\mathbf{w}^{(3)} \leftarrow \mathbf{w}^{(2)} - \epsilon \nabla_{\mathbf{w}} f(\mathbf{w}^{(2)})$$
  
...  
$$\mathbf{w}^{(t)} \leftarrow \mathbf{w}^{(t-1)} - \epsilon \nabla_{\mathbf{w}} f(\mathbf{w}^{(t-1)})$$
- ...until some convergence condition (e.g., #iterations, tolerance in function value diff, tolerance in weight diff, etc.).

# Gradient descent demos

- Demo video.

# **Stochastic gradient descent**

# Gradient descent

- With gradient descent, we only update the weights after scanning the *entire* training set.
  - This is slow.
  - If the training set contains 20K examples, then the gradient is an *average* over 20K images.
  - How much would the gradient really change if we just used, say, 10K images? 5K images? 128 images?

$$\nabla_{\mathbf{w}} f_{\text{MSE}}(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{w}) = \frac{1}{n} \mathbf{X} (\mathbf{X}^\top \mathbf{w} - \mathbf{y})$$

Average over entire training set.

# Stochastic gradient descent

- This is the idea behind **stochastic gradient descent (SGD)**:
  - Randomly sample a small ( $\ll n$ ) **mini-batch** (or sometimes just **batch**) of training examples.
  - Estimate the gradient on just the mini-batch.
  - Update weights based on *mini-batch* gradient estimate.
  - Repeat.

# Stochastic gradient descent

- In practice, SGD is usually conducted over multiple epochs.
  - An **epoch** is a single pass through the entire training set.
- Procedure:
  1. Let  $\tilde{n} \ll n$  equal the size of the mini-batch.

# Stochastic gradient descent

- In practice, SGD is usually conducted over multiple epochs.
  - An **epoch** is a single pass through the entire training set.
- Procedure:
  1. Let  $\tilde{n} \ll n$  equal the size of the mini-batch.
  2. Randomize the order of the examples in the training set.

# Stochastic gradient descent

- In practice, SGD is usually conducted over multiple epochs.
  - An **epoch** is a single pass through the entire training set.
- Procedure:
  1. Let  $\tilde{n} \ll n$  equal the size of the mini-batch.
  2. Randomize the order of the examples in the training set.
  3. For  $e = 0$  to  $\text{numEpochs}$ :

# Stochastic gradient descent

- In practice, SGD is usually conducted over multiple epochs.
  - An **epoch** is a single pass through the entire training set.
- Procedure:
  1. Let  $\tilde{n} \ll n$  equal the size of the mini-batch.
  2. Randomize the order of the examples in the training set.
  3. For  $e = 0$  to  $\text{numEpochs}$ :
    - I. For  $i = 0$  to  $(\lceil n/\tilde{n} \rceil - 1)$  (one epoch):

# Stochastic gradient descent

- In practice, SGD is usually conducted over multiple epochs.
  - An **epoch** is a single pass through the entire training set.
- Procedure:
  1. Let  $\tilde{n} \ll n$  equal the size of the mini-batch.
  2. Randomize the order of the examples in the training set.
  3. For  $e = 0$  to  $\text{numEpochs}$ :
    - I. For  $i = 0$  to  $(\lceil n/\tilde{n} \rceil - 1)$  (one epoch):
      - A. Select a mini-batch  $\mathcal{J}$  containing the next  $\tilde{n}$  examples.

# Stochastic gradient descent

- In practice, SGD is usually conducted over multiple epochs.
  - An **epoch** is a single pass through the entire training set.
- Procedure:
  1. Let  $\tilde{n} \ll n$  equal the size of the mini-batch.
  2. Randomize the order of the examples in the training set.
  3. For  $e = 0$  to  $\text{numEpochs}$ :
    - I. For  $i = 0$  to  $(\lceil n/\tilde{n} \rceil - 1)$  (one epoch):
      - A. Select a mini-batch  $\mathcal{J}$  containing the next  $\tilde{n}$  examples.
      - B. Compute the gradient on this mini-batch:  $\frac{1}{\tilde{n}} \sum_{i \in \mathcal{J}} \nabla_{\mathbf{W}} f(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)}; \mathbf{W})$

# Stochastic gradient descent

- In practice, SGD is usually conducted over multiple epochs.
  - An **epoch** is a single pass through the entire training set.
- Procedure:
  1. Let  $\tilde{n} \ll n$  equal the size of the mini-batch.
  2. Randomize the order of the examples in the training set.
  3. For  $e = 0$  to  $\text{numEpochs}$ :
    - I. For  $i = 0$  to  $(\lceil n/\tilde{n} \rceil - 1)$  (one epoch):
      - A. Select a mini-batch  $\mathcal{J}$  containing the next  $\tilde{n}$  examples.
      - B. Compute the gradient on this mini-batch:  $\frac{1}{\tilde{n}} \sum_{i \in \mathcal{J}} \nabla_{\mathbf{W}} f(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)}; \mathbf{W})$
      - C. Update the weights based on the current mini-batch gradient.

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples.
- Here is how *regular* gradient descent would proceed:
  - **Initialize weights  $w^{(0)}$  to random values.**

Training  
examples

1
2
3
4
5
6
7
8

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples.
- Here is how *regular* gradient descent would proceed:
  - Initialize weights  $\mathbf{w}^{(0)}$  to random values.
  - For each round:
    - **Compute gradient on all  $n$  examples.**

Training  
examples

1
2
3
4
5
6
7
8

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples.
- Here is how *regular* gradient descent would proceed:
  - Initialize weights  $\mathbf{w}^{(0)}$  to random values.
  - For each round:
    - Compute gradient on all  $n$  examples.
    - **Update weights:**  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \epsilon \nabla_{\mathbf{w}} f$

Training examples
1
2
3
4
5
6
7
8

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples.
- Here is how *regular* gradient descent would proceed:
  - Initialize weights  $\mathbf{w}^{(0)}$  to random values.
  - For each round:
    - **Compute gradient on all  $n$  examples.**
    - Update weights:  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \epsilon \nabla_{\mathbf{w}} f$

Training examples
1
2
3
4
5
6
7
8

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples.
- Here is how *regular* gradient descent would proceed:
  - Initialize weights  $\mathbf{w}^{(0)}$  to random values.
  - For each round:
    - Compute gradient on all  $n$  examples.
    - **Update weights:**  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \epsilon \nabla_{\mathbf{w}} f$

Training examples
1
2
3
4
5
6
7
8

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples with  $\tilde{n} = 2$ .
- Here is how *stochastic* gradient descent would proceed:
  - **Initialize weights  $w^{(0)}$  to random values.**

**Training  
examples**

1
2
3
4
5
6
7
8

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples with  $\tilde{n} = 2$ .
- Here is how *stochastic* gradient descent would proceed:
  - Initialize weights  $\mathbf{w}^{(0)}$  to random values.
  - **Randomize the order of the training data.**

Training  
examples

4
1
3
5
7
6
8
2

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples with  $\tilde{n} = 2$ .
- Here is how *stochastic* gradient descent would proceed:
  - Initialize weights  $\mathbf{w}^{(0)}$  to random values.
  - Randomize the order of the training data.
  - For each epoch ( $e=1, \dots, E$ ): **e=1**
    - For each round ( $r=1, \dots, \lceil n/\tilde{n} \rceil$ ):
      - **Compute gradient on next  $\tilde{n}$  examples.**

Training examples
4
1
3
5
7
6
8
2

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples with  $\tilde{n} = 2$ .
- Here is how *stochastic* gradient descent would proceed:
  - Initialize weights  $\mathbf{w}^{(0)}$  to random values.
  - Randomize the order of the training data.
  - For each epoch ( $e=1, \dots, E$ ): **e=1**
    - For each round ( $r=1, \dots, \lceil n/\tilde{n} \rceil$ ):
      - Compute gradient on next  $\tilde{n}$  examples.
      - **Update weights:**  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \epsilon \nabla_{\mathbf{w}} f$

Training examples
4
1
3
5
7
6
8
2

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples with  $\tilde{n} = 2$ .
- Here is how *stochastic* gradient descent would proceed:
  - Initialize weights  $\mathbf{w}^{(0)}$  to random values.
  - Randomize the order of the training data.
  - For each epoch ( $e=1, \dots, E$ ): **e=1**
    - For each round ( $r=1, \dots, \lceil n/\tilde{n} \rceil$ ):
      - **Compute gradient on next  $\tilde{n}$  examples.**
      - Update weights:  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \epsilon \nabla_{\mathbf{w}} f$

Training examples
4
1
3
5
7
6
8
2

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples with  $\tilde{n} = 2$ .
- Here is how *stochastic* gradient descent would proceed:
  - Initialize weights  $\mathbf{w}^{(0)}$  to random values.
  - Randomize the order of the training data.
  - For each epoch ( $e=1, \dots, E$ ): **e=1**
    - For each round ( $r=1, \dots, \lceil n/\tilde{n} \rceil$ ):
      - Compute gradient on next  $\tilde{n}$  examples.
      - **Update weights:**  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \epsilon \nabla_{\mathbf{w}} f$

Training examples
4
1
3
5
7
6
8
2

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples with  $\tilde{n} = 2$ .
- Here is how *stochastic* gradient descent would proceed:
  - Initialize weights  $\mathbf{w}^{(0)}$  to random values.
  - Randomize the order of the training data.
  - For each epoch ( $e=1, \dots, E$ ): **e=1**
    - For each round ( $r=1, \dots, \lceil n/\tilde{n} \rceil$ ):
      - **Compute gradient on next  $\tilde{n}$  examples.**
      - Update weights:  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \epsilon \nabla_{\mathbf{w}} f$

Training examples
4
1
3
5
7
6
8
2

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples with  $\tilde{n} = 2$ .
- Here is how *stochastic* gradient descent would proceed:
  - Initialize weights  $\mathbf{w}^{(0)}$  to random values.
  - Randomize the order of the training data.
  - For each epoch ( $e=1, \dots, E$ ): **e=1**
    - For each round ( $r=1, \dots, \lceil n/\tilde{n} \rceil$ ):
      - Compute gradient on next  $\tilde{n}$  examples.
      - **Update weights:**  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \epsilon \nabla_{\mathbf{w}} f$

Training examples
4
1
3
5
7
6
8
2

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples with  $\tilde{n} = 2$ .
- Here is how *stochastic* gradient descent would proceed:
  - Initialize weights  $\mathbf{w}^{(0)}$  to random values.
  - Randomize the order of the training data.
  - For each epoch ( $e=1, \dots, E$ ): **e=1**
    - For each round ( $r=1, \dots, \lceil n/\tilde{n} \rceil$ ):
      - **Compute gradient on next  $\tilde{n}$  examples.**
      - Update weights:  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \epsilon \nabla_{\mathbf{w}} f$

Training examples
4
1
3
5
7
6
8
2

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples with  $\tilde{n} = 2$ .
- Here is how *stochastic* gradient descent would proceed:
  - Initialize weights  $\mathbf{w}^{(0)}$  to random values.
  - Randomize the order of the training data.
  - For each epoch ( $e=1, \dots, E$ ): **e=1**
    - For each round ( $r=1, \dots, \lceil n/\tilde{n} \rceil$ ):
      - Compute gradient on next  $\tilde{n}$  examples.
      - **Update weights:**  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \epsilon \nabla_{\mathbf{w}} f$

Training examples
4
1
3
5
7
6
8
2

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples with  $\tilde{n} = 2$ .
- Here is how *stochastic* gradient descent would proceed:
  - Initialize weights  $\mathbf{w}^{(0)}$  to random values.
  - Randomize the order of the training data.
  - For each epoch ( $e=1, \dots, E$ ): **e=2**
    - For each round ( $r=1, \dots, \lceil n/\tilde{n} \rceil$ ):
      - **Compute gradient on next  $\tilde{n}$  examples.**
      - Update weights:  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \epsilon \nabla_{\mathbf{w}} f$

Training examples
4
1
3
5
7
6
8
2

# SGD versus GD: example

- Suppose our training set contains  $n=8$  examples with  $\tilde{n} = 2$ .
  - Here is how *stochastic* gradient descent would proceed:
    - Initialize weights  $\mathbf{w}^{(0)}$  to random values.
    - Randomize the order of the training data.
    - For each epoch ( $e=1, \dots, E$ ): **e=2**
      - For each round ( $r=1, \dots, \lceil n/\tilde{n} \rceil$ ):
        - Compute gradient on next  $\tilde{n}$  examples.
        - Update weights:  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \epsilon \tilde{\nabla}_{\mathbf{w}} f$
- ...

Training examples
4
1
3
5
7
6
8
2

# Stochastic gradient descent

- Despite “noise” (statistical inaccuracy) in the mini-batch gradient estimates, we will still converge to local minimum.
- Training can be much faster than regular gradient descent because we adjust the weights *many times* per epoch.

# SGD: learning rates

- With SGD, our learning rate  $\epsilon$  needs to be **annealed** (reduced slowly over time) to guarantee convergence.
  - Otherwise we might just oscillate forever in weight space.
- Necessary conditions:

$$\lim_{T \rightarrow \infty} \sum_{t=1}^T |\epsilon_t|^2 < \infty$$

**Not too big: sum of squared learning rates converges.**

# SGD: learning rates

- With SGD, our learning rate  $\epsilon$  needs to be **annealed** (reduced slowly over time) to guarantee convergence.
  - Otherwise we might just oscillate forever in weight space.
- Necessary conditions:

$$\lim_{T \rightarrow \infty} \sum_{t=1}^T |\epsilon_t|^2 < \infty$$

$$\lim_{T \rightarrow \infty} \sum_{t=1}^T |\epsilon_t| = \infty$$

**Not too small: sum of absolute learning rates grows to infinity.**

# SGD: learning rates

- One common learning rate “schedule” is to multiply  $\epsilon$  by  $c \in (0, 1)$  every  $k$  rounds.
  - This is called **exponential decay**.
- Another possibility (which avoids the issue) is to set the number of epochs  $T$  to a finite number.
  - SGD may not fully converge, but the machine might still perform well.
- There are many other strategies.

# Probabilistic machine learning

# Probability theory

- Many machine learning algorithms, and some deep learning methods, are based on probability theory.
- Probabilities provide a natural way of expressing our **uncertainty** about a particular value, e.g.:
  - The ground-truth  $y$  we are trying to estimate.
  - Our estimate  $\hat{y}$  of the ground-truth.

# Quantifying uncertainty

- **Frequentist** probabilities:
  - Ask a large group of randomly selected people to **label** the face as smiling or not.
  - Count the number of labels for “smile” and divide by the total number of labels.
  - The ratio is the probability of “smile” for that face image.

# Quantifying uncertainty

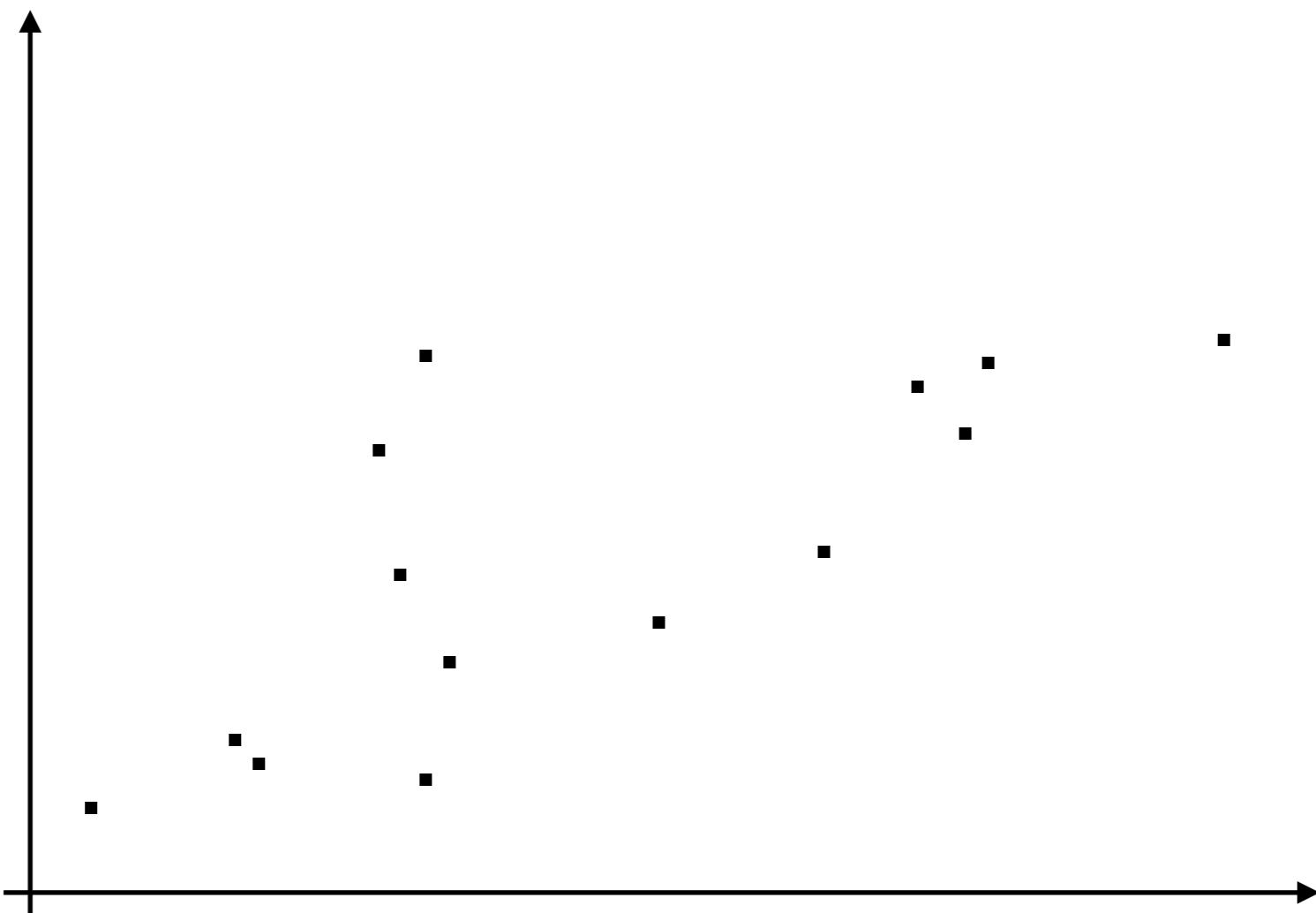
- **Bayesian** probabilities (“beliefs”):
  - Ask one person how much she/he believes the image is smiling, quantified as a number between 0 and 1.
  - The “belief” score is the probability of “smile” for that face image.

# Quantifying uncertainty

- Many a debate has been waged about the validity of Bayesian versus frequentist probabilities.
- For the most part we will steer clear of these debates and use both tools when they are useful.

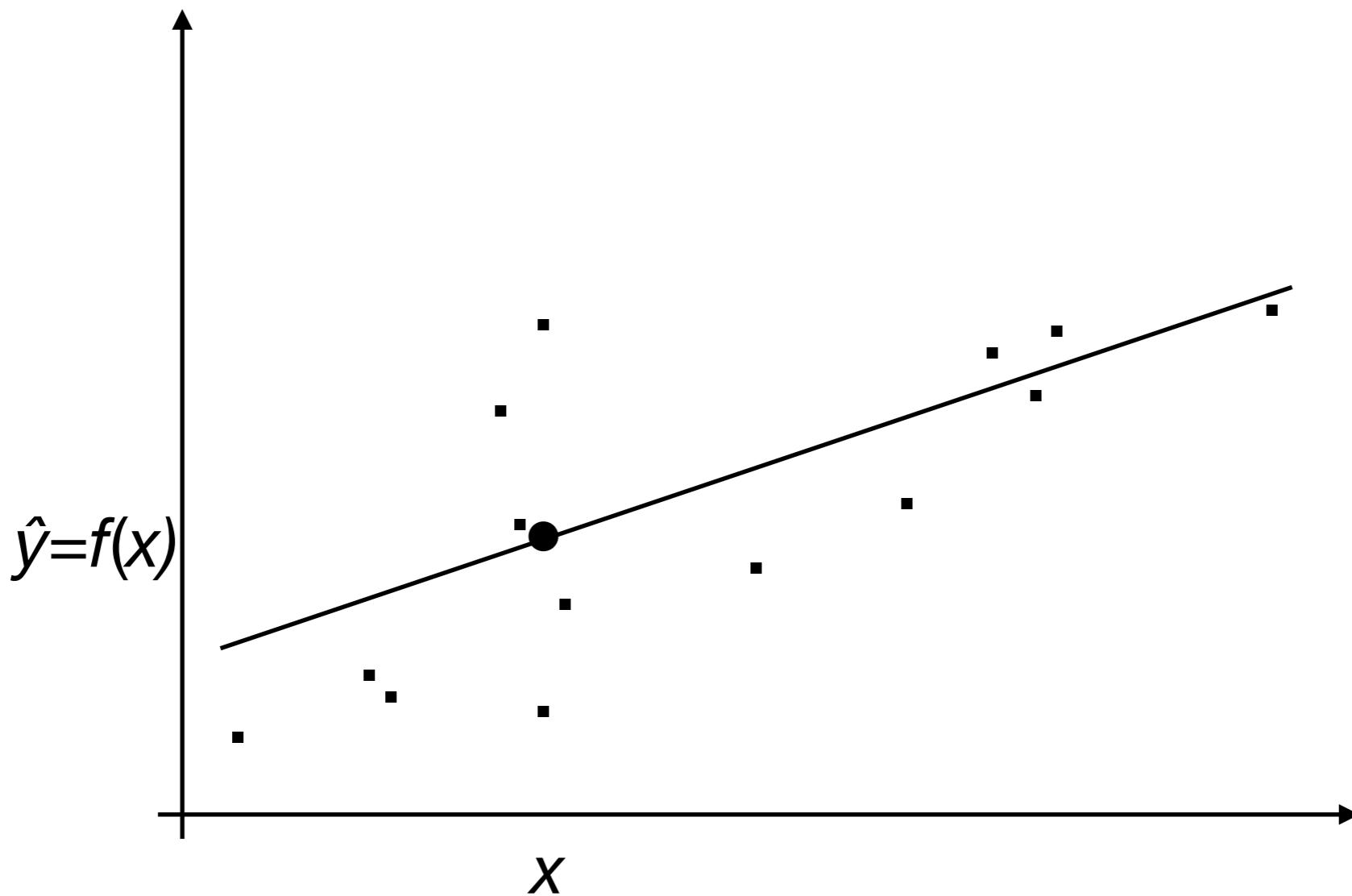
# Uncertainty of estimation

- Sometimes we may be very uncertain about our prediction of the target value  $y$  from the input  $x$ .



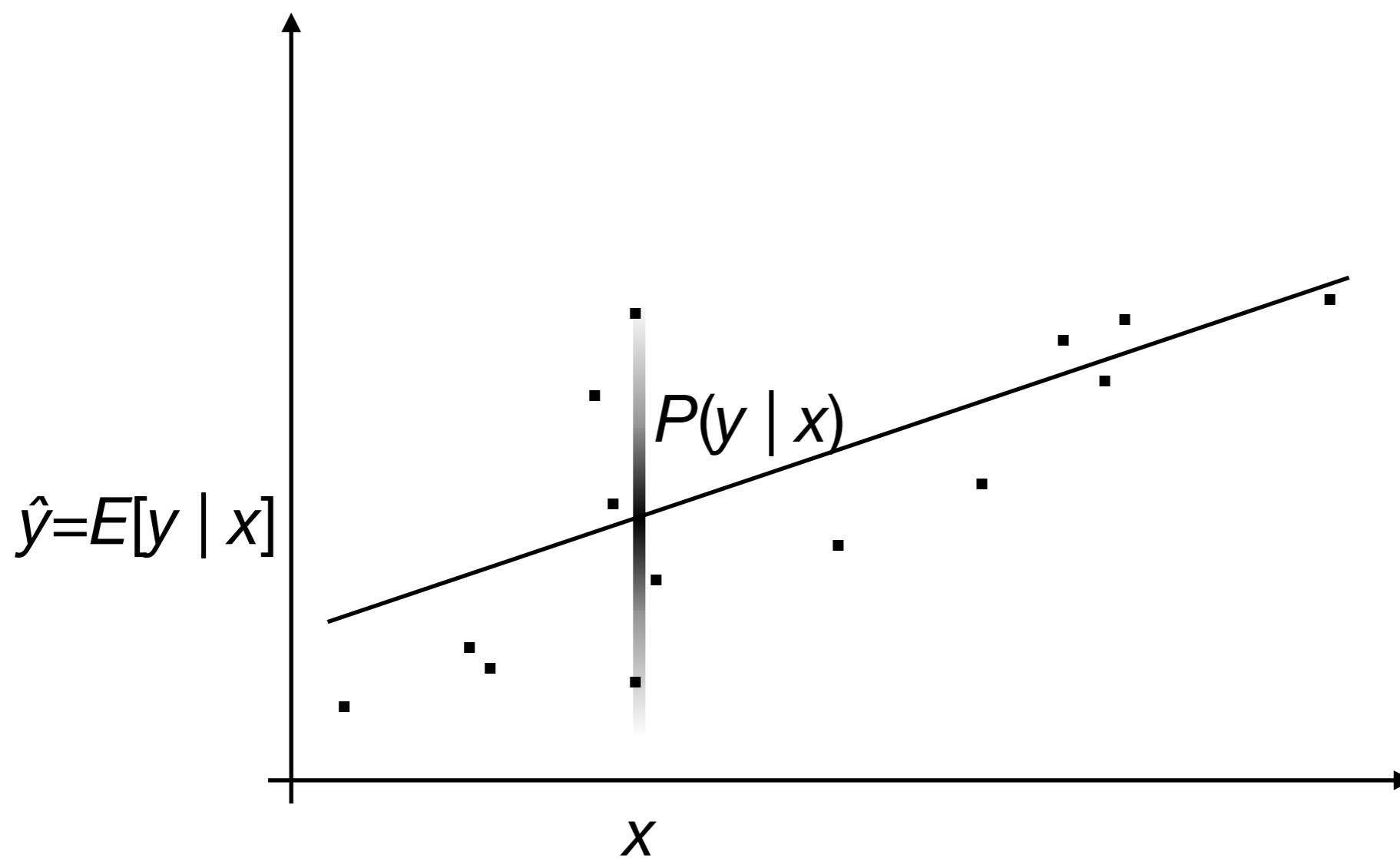
# Uncertainty of estimation

- Rather than just giving a point estimate  $\hat{y}=f(x)\dots$



# Uncertainty of estimation

- ...we can estimate an entire probability distribution  $P(y | x)$  that expresses our (conditional) uncertainty.



# Probabilistic deep learning

- Neural networks can be used in various ways to make probabilistic predictions:
  - For regression, estimate both the expected value and the variance of the prediction.
  - Model a high-dimensional distribution using a probabilistic latent variable model (LVM) – akin to factor analysis but deeper.

# Random variables

\*

- A **random variable (RV)**  $X$  (with some outcome set  $\Omega$ ) has a value we are unsure about, maybe because (a) it is decided by some random process or (b) it is hidden from us.
- Types of outcome sets  $\Omega$ :
  - Finite, e.g.,  $\{ 0, 1 \}$ ,  $\{ \text{red, blue, green} \}$ .
  - Countable, e.g.,  $\mathbb{Z}_{\geq 0}$
  - Uncountable, e.g.,  $\mathbb{R}$

\* This is a practical definition for the purposes of this course, not a formal definition based on measure theory.

# Random variables

- RVs are typically written as capital letters, e.g.,  $X$ ,  $Y$ .
- Once the value of RV  $X$  has been “sampled”, “selected”, “instantiated”, or “realized” (by a random number generator, generative process, God, etc.), it takes a specific value from the outcome set.
- The values the RV can take are typically written as lower-case letters, e.g.,  $x$ ,  $y$ .

# Random variables

- The probability that a random variable  $X$  takes a particular value is determined by a:
  - **Probability mass function (PMF)** for finite or countable outcome sets.
  - **Probability density function (PDF)** for uncountable outcome sets.

# PMF

- Example 1 (finite):



- Let RV  $X$  be the outcome of rolling a 6-sided die.
- If  $X$  is fair, then:  $P(X = i) = \frac{1}{6} \quad \forall i \in \{1, \dots, 6\}$

# PMF

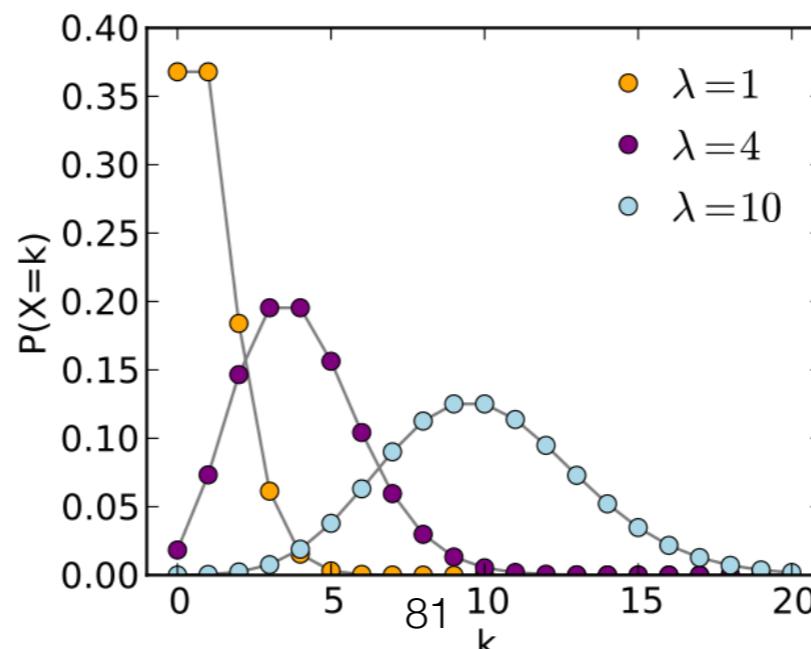


- Example 2 (countable):

- Let RV  $X$  be the number of TCP/IP packets that arrive in 1 second.
- We can model the count of packets with a Poisson distribution:

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

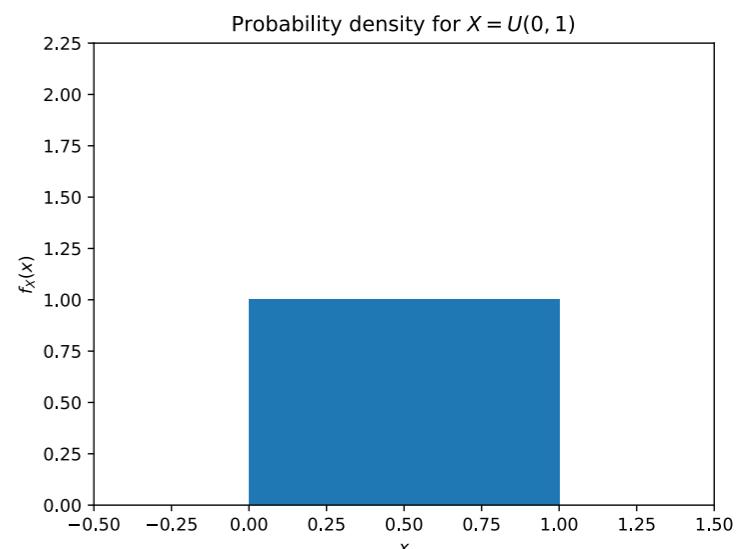
where parameter  $\lambda$  specifies the *rate* of the packet arrivals.



# PDF

- Example 1:

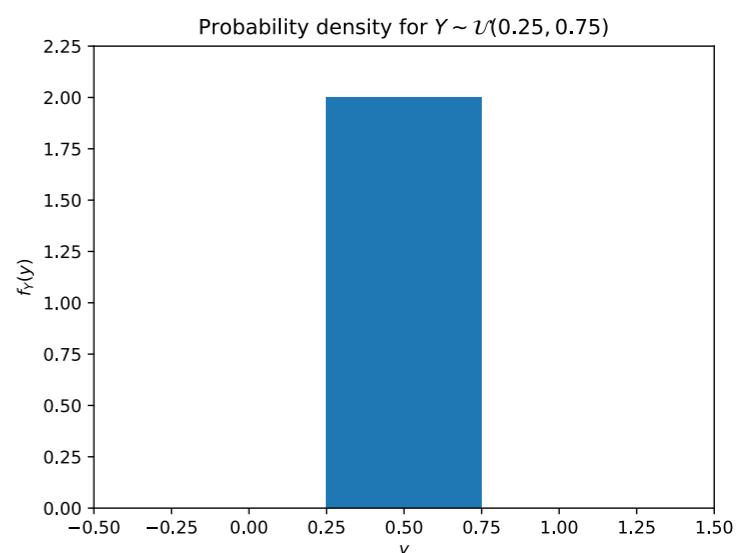
- Let  $X$  be a uniformly-distributed RV over  $\Omega=[0,1]$ .
- Then  $f_X(x) = 1 \quad \forall x \in \Omega$



- Example 2:

- Let  $Y$  be a uniformly-distributed RV over  $\Omega=[1/4, 3/4]$ .

- Then  $f_Y(y) = 2 \quad \forall y \in \Omega$
- Note that the PDF can exceed 1.**



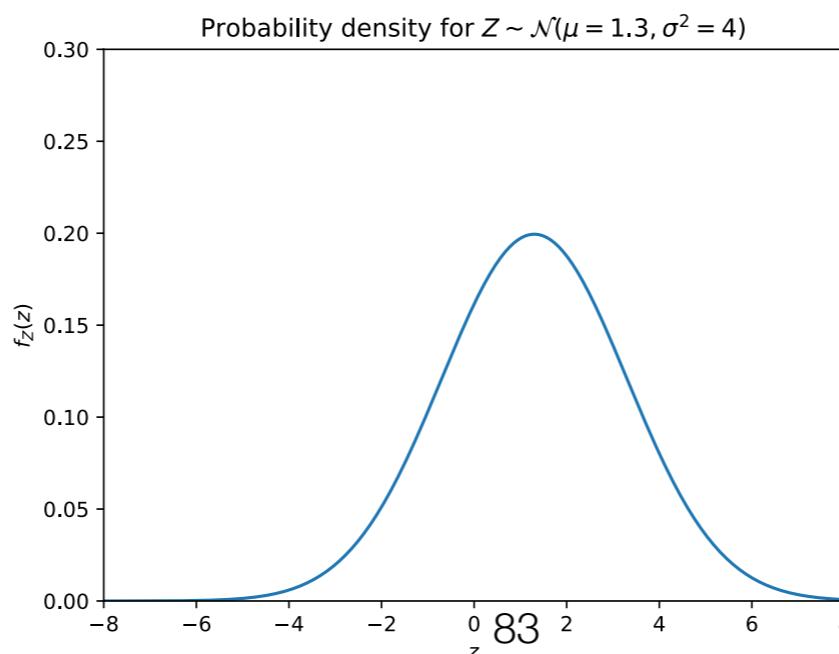
# PDF

- Example 3:

- Let  $Z$  be a **normally** (aka **Gaussian**) distributed RV with mean 1.5 and variance 4, i.e.,  $Z \sim \mathcal{N}(z; \mu = 1.5, \sigma^2 = 4)$ .
- Then

$$f_Z(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(z - \mu)^2}{2\sigma^2}\right)$$

where  $\mu$  is the mean and  $\sigma^2$  is the variance.



# Probability distributions

- In this course, we will relax notation and use “probability distribution” to mean either the PDF or PMF of a RV (as appropriate).
- As a notational shortcut, we use  $P(x)$  to mean  $P(X=x)$  or  $f_X(x)$ .

# Joint probability distributions

- For multiple random variables  $X, Y, \dots$ , we can construct the joint probability distribution  $P(x, y, \dots)$  to mean the probability that  $(X=x) \wedge (Y=y) \wedge \dots$
- Note that  $P$  must still sum to 1 over all possible joint values  $(x, y, \dots)$ .

# Joint probability distributions

- Example in 2-D – crayons:



- Let  $X$  be the color (red, blue, green, white).
- Let  $Y$  be the intensity (low, medium, high).

	Red	Blue	Green	White
Low	0.1	0.05	0.025	0.2
Med	0.075	0.05	0.1	0
High	0.25	0.05	0.075	0.025

# Joint probability distributions

- Exercise:
  - What is the overall probability of obtaining a white crayon?



	Red	Blue	Green	White
Low	0.1	0.05	0.025	0.2
Med	0.075	0.05	0.1	0
High	0.25	0.05	0.075	0.025

# Joint probability distributions

- Example in 2-D – crayons:
  - From the joint distribution we can compute the **marginal distributions**  $P(x)$  and  $P(y)$ .



$$P(x) = \sum_y P(x, y) \quad P(y) = \sum_x P(x, y)$$

	Red	Blue	Green	White	$P(y)$
Low	0.1	0.05	0.025	0.2	<b>0.375</b>
Med	0.075	0.05	0.1	0	<b>0.225</b>
High	0.25	0.05	0.075	0.025	<b>0.4</b>
$P(x)$	<b>0.425</b>	<b>0.15</b>	<b>0.2</b>	<b>0.225</b>	

# Law of total probability

- This is also called the **law of total probability**:
  - For any RVs  $X$  and  $Y$ :

$$P(x) = \sum_y P(x, y)$$

# Joint probability distributions

- In machine learning, we often use joint distributions of many variables that are part of a collection, e.g.:
  - Sequence  $(W_1, W_2, \dots, W_T)$  of words in a sentence.
    - $W_t$  is the  $t^{\text{th}}$  RV in the sequence.
  - Grid  $(I_{11}, \dots, I_{1M}, \dots, I_{N1}, \dots, I_{NM})$  of the pixels in an  $N \times M$  image.
    - $I_{ij}$  is the RV corresponding to location  $(i, j)$ .

# Conditional probability distributions

- Sometimes the value of one RV is predictive of the value of another RV.
- Examples:
  - If I know a person's height  $H$ , then I have some information about their weight  $W$ .
  - If I know how much cholesterol  $C$  a person eats, then I have some information about their chance of coronary heart disease  $D$ .

# Conditional probability distributions

- We can form a **conditional probability distribution** of RV  $X$  **given** the value of RV  $Y$ :

$$P(x \mid y)$$

- Examples:
  - Height given weight:  $P(h \mid w)$
  - Heart disease given cholesterol:  $P(d \mid c)$

# Conditional probability distributions

- We can form a **conditional probability distribution** of RV  $X$  **given** the value of RV  $Y$ :

$$P(x \mid y)$$

“given”

- Examples:
  - Height given weight:  $P(h \mid w)$
  - Heart disease given cholesterol:  $P(d \mid c)$

# Conditional probability distributions

- More generally, we can form a conditional probability distribution of  $X_1, \dots, X_n$  given the values of  $Y_1, \dots, Y_m$ :

$$P(x_1, \dots, x_n \mid y_1, \dots, y_m)$$

# Conditional probability distributions

- A conditional probability distribution is related to the joint probability distribution as follows:

$$P(x \mid y)P(y) = P(x, y)$$

- It follows that:

$$P(x \mid y, z)P(y \mid z) = P(x, y \mid z)$$

# Conditional probability distributions

- More generally:

$$P(x_1, \dots, x_n \mid y_1, \dots, y_m)P(y_1, \dots, y_m) = P(x_1, \dots, x_n, y_1, \dots, y_m)$$

- And also:

$$\begin{aligned} P(x_1, \dots, x_n \mid y_1, \dots, y_m, z_1, \dots, z_p)P(y_1, \dots, y_m \mid z_1, \dots, z_p) \\ = P(x_1, \dots, x_n, y_1, \dots, y_m \mid z_1, \dots, z_p) \end{aligned}$$

# Conditional probability distributions

- Note that the same joint probability can be factored in different ways, e.g.:

$$\begin{aligned} P(x, y, z) &= P(x, y \mid z)P(z) \\ &= P(x \mid y, z)P(y, z) \end{aligned}$$

# Conditional probability distributions

- Exercises:

1.  $P(a, b, c, d) = P(a, c) * ?$
2.  $P(w_1, w_2, w_3) = P(w_3 | w_1) * ? * ?$
3.  $P(x_1, x_2, x_3) = P(x_1) * ? * P(x_3 | x_1, x_2)$
4.  $P(x_1, \dots, x_n) = P(x_1) * ? * ? * ? * \dots * ?$   
 **$n-1$  terms**

# Conditional probability distributions

- Exercises:

$$1. \quad P(a, b, c, d) = P(a, c) * P(b, d | a, c)$$

$$2. \quad P(w_1, w_2, w_3) = P(w_3 | w_1) * P(w_1) * P(w_2 | w_1, w_3)$$

$$3. \quad P(x_1, x_2, x_3) = P(x_1) * P(x_2 | x_1) * P(x_3 | x_1, x_2)$$

$$4. \quad P(x_1, \dots, x_n) = P(x_1) * \prod_{i=2}^n P(x_i | x_1, \dots, x_{i-1})$$

# Independence

- RVs  $X$  and  $Y$  are independent iff  $P(x, y) = P(x)P(y)$   $\forall x, y$ , i.e., the joint distribution equals the product of the marginal distributions.
- Note that this implies that  $P(x | y) = P(x)$  and  $P(y | x) = P(y)$  since  $P(x, y) = P(x | y)P(y) = P(y | x)P(x)$  by definition of conditional probability.

# Conditional independence

- RVs  $X$  and  $Y$  are conditionally independent given RV  $Z$  iff:

$$P(x, y \mid z) = P(x \mid z)P(y \mid z) \quad \forall x, y, z$$

- Note that this implies:

$$P(x \mid y, z) = P(x \mid z)$$

- In words: “If I already know the value of  $Z$ , then knowing  $Y$  tells me nothing *further* about  $X$ ”.

# Conditional independence

- More generally:  $X_1, \dots, X_n$  and  $Y_1, \dots, Y_m$  are conditionally independent given  $Z_1, \dots, Z_p$  iff:

$$P(x_1, \dots, x_n, y_1, \dots, y_m \mid z_1, \dots, z_p) \\ = P(x_1, \dots, x_n \mid z_1, \dots, z_p)P(y_1, \dots, y_m \mid z_1, \dots, z_p)$$

# Bayes' rule

- It is often useful to compute  $P(x | y)$  in terms of  $P(y | x)$ .
  - For example, if  $X$  represents a student's skill level, and  $Y$  is their test score, it's often easier to compute  $P(y | x)$ . But given a student's test score  $Y$ , we really want to know  $P(x | y)$ .
- Bayes' rule:

$$P(x | y) = \frac{P(x, y)}{P(y)} = \frac{P(y|x)P(x)}{P(y)}$$

# Bayes' rule

- We can also generalize Bayes' rule to cases where we always condition on a tertiary variable  $Z$ :

$$P(x \mid y, z) = \frac{P(y \mid x, z)P(x \mid z)}{P(y \mid z)}$$

# Bayes' rule

- It is sometimes possible – and more convenient – to work with **unnormalized** probabilities.
- For instance, it might suffice to know that  $[P(y | x^{(1)}), P(y | x^{(2)}), P(y | x^{(3)})] \propto [3.5, 7, 0.04]$  rather than their exact (normalized) values.

# Bayes' rule

- In this case, instead of needing to compute the denominator of...

$$\begin{aligned} P(x \mid y) &= \frac{P(y \mid x)P(x)}{P(y)} \\ &= \frac{P(y \mid x)P(x)}{\int P(x, y)dx} \\ &= \frac{P(y \mid x)P(x)}{\int P(y \mid x)P(x)dx} \end{aligned}$$

# Bayes' rule

- In this case, instead of needing to compute the denominator of...

$$\begin{aligned} P(x \mid y) &= \frac{P(y \mid x)P(x)}{P(y)} \\ &= \frac{P(y \mid x)P(x)}{\int P(x, y)dx} \\ &= \frac{P(y \mid x)P(x)}{\int P(y \mid x)P(x)dx} \end{aligned}$$

Normalizing constant

# Bayes' rule

- In this case, instead of needing to compute the denominator of...

$$\begin{aligned} P(x \mid y) &= \frac{P(y \mid x)P(x)}{P(y)} \\ &= \frac{P(y \mid x)P(x)}{\int P(x, y)dx} \\ \text{... we might only need to compute the numerator: } &= \frac{P(y \mid x)P(x)}{\int P(y \mid x)P(x)dx} \\ P(x \mid y) &\propto P(y \mid x)P(x) \end{aligned}$$

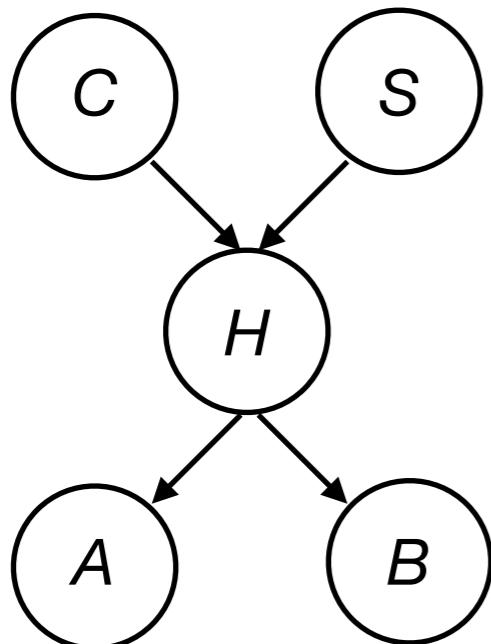
# Probabilistic inference

# Probabilistic graphical models

- To express the conditional independence relationships between multiple RVs, it is useful to represent their dependencies in a graph.
- A formal theory of probabilistic graphical models (Pearl 1998) has been devised.
  - Conditional independence can be determined via the principle of **d-separation** (beyond the scope of this course).

# Probabilistic graphical models

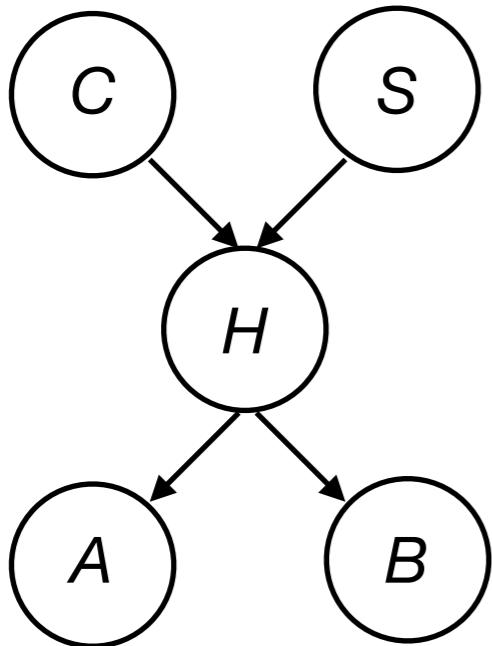
- Example 1 – medical diagnosis:



- $C$ : whether the patient eats **caviar**.
- $S$ : the patient's **sex**
- $H$ : whether the patient has **high cholesterol**
- $A$ : whether the patient will have a heart **attack**.
- $B$ : whether the patient has shortness of **breath**.

# Probabilistic graphical models

- Example 1 – medical diagnosis:



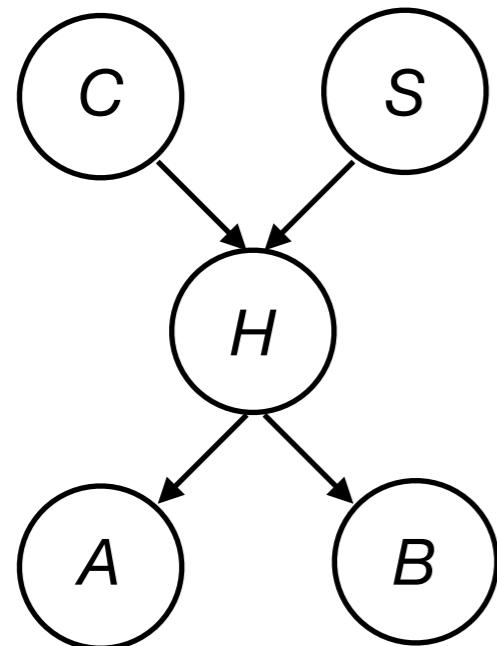
- This model implies that:

$$P(a, b \mid h, c, s) = P(a, b \mid h) \text{ and}$$

$$P(c, s \mid h, a, b) = P(c, s \mid h)$$

# Probabilistic graphical models

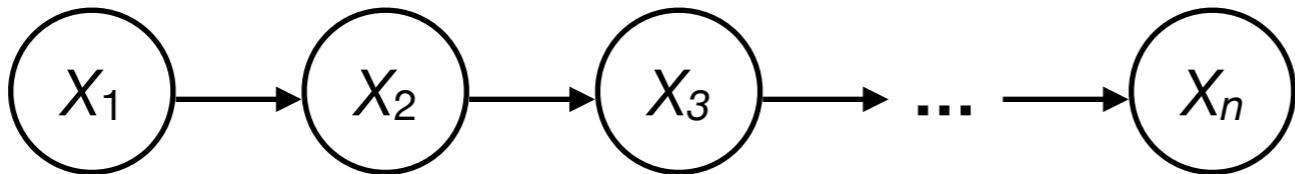
- Example 1 – medical diagnosis:



- In words, “If I want to know the probability the patient will have a heart attack  $A$ , and I already know the patient has high cholesterol  $H$ , then the patient’s sex and whether she/he eats caviar  $C$  is irrelevant.”

# Probabilistic graphical models

- Example 2 – Markov chain:



- This chain-like model of  $X_1, \dots, X_n$  implies that:

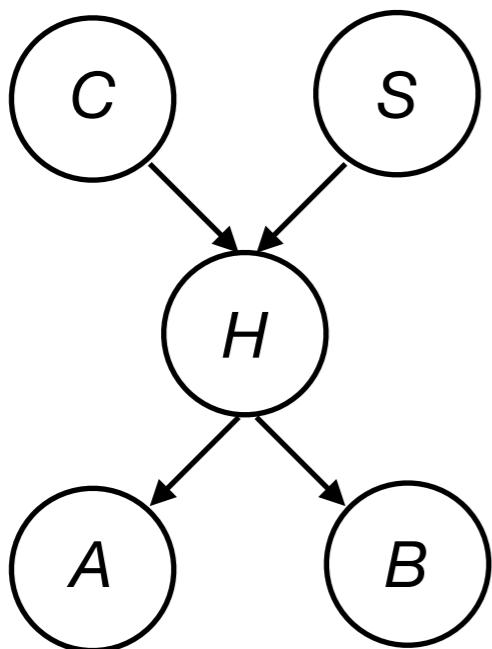
$$P(x_i \mid x_1, \dots, x_{i-1}) = P(x_i \mid x_{i-1}) \text{ and}$$

$$P(x_i \mid x_{i+1}, \dots, x_n) = P(x_i \mid x_{i+1})$$

- In words, “If I want to know the value of  $X_i$  and I already know  $X_{i-1}$ , then the values of any ‘earlier’ or ‘later’  $X$ ’s are irrelevant.”

# Probabilistic inference

- Given a model with multiple RVs and how they are related to each other, we can **infer** the values of other RVs.
- For the medical diagnosis example, suppose we knew the conditional probability distributions:

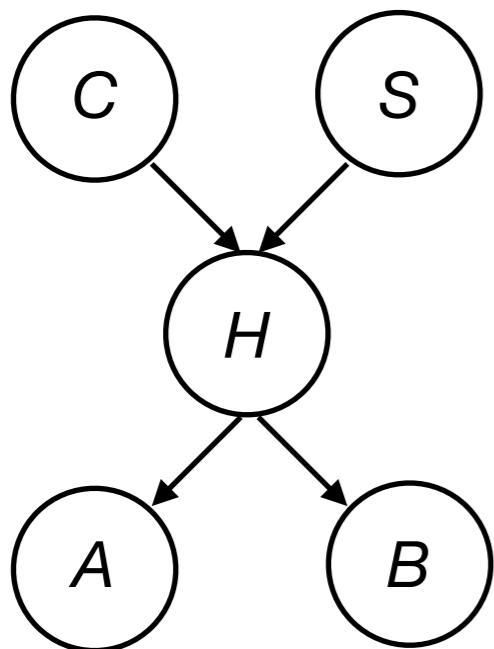


	$P(H=0)$	
	C=0	C=1
S=Ma		
S=Fe		

	$P(H=1)$	
	C=0	C=1
S=Ma	0.3	0.6
S=Fe	0.2	0.25

# Probabilistic inference

- Given a model with multiple RVs and how they are related to each other, we can **infer** the values of other RVs.
- For the medical diagnosis example, suppose we knew the conditional probability distributions:

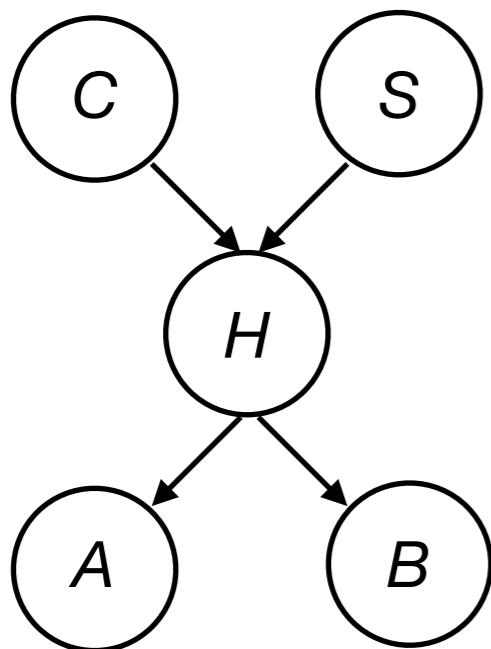


	$P(H=0)$	
	C=0	C=1
S=Ma	0.7	0.4
S=Fe	0.8	0.75

	$P(H=1)$	
	C=0	C=1
S=Ma	0.3	0.6
S=Fe	0.2	0.25

# Probabilistic inference

- Given a model with multiple RVs and how they are related to each other, we can **infer** the values of other RVs.
- For the medical diagnosis example, suppose we knew the conditional probability distributions:

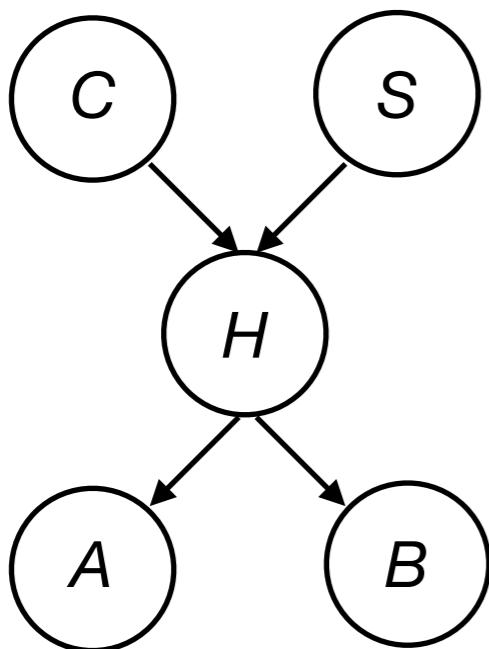


$P(A=0)$	
$H=0$	0.75
$H=1$	0.2

$P(A=1)$	
$H=0$	0.25
$H=1$	0.8

# Probabilistic inference

- Given a model with multiple RVs and how they are related to each other, we can **infer** the values of other RVs.
- For the medical diagnosis example, suppose we knew the conditional probability distributions:

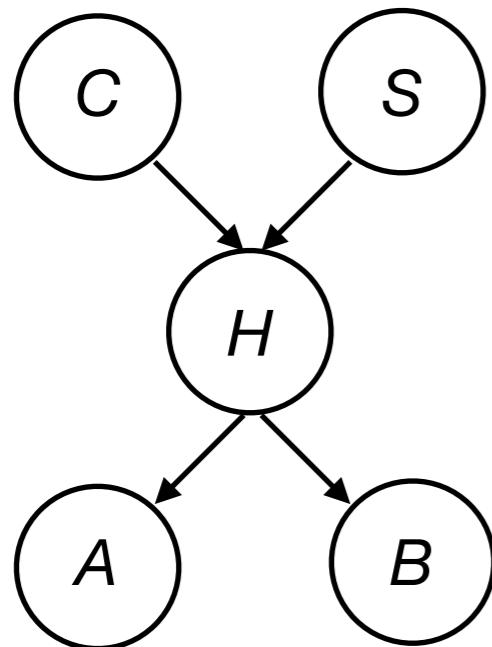


$P(B=0)$	
$H=0$	0.9
$H=1$	0.1

$P(B=1)$	
$H=0$	0.1
$H=1$	0.9

# Probabilistic inference

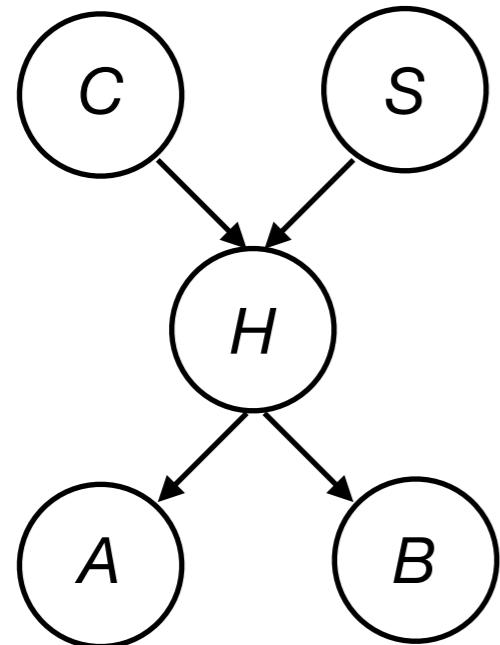
- Suppose we meet a male patient who eats caviar.
- What is the **posterior probability** that  $H=1$ , i.e.,  $P(H=1 | C=1, S=\text{Ma})$ ? (*Posterior* means after observing  $C, S$ .)



		$P(H=0)$		$P(H=1)$	
		$C=0$	$C=1$	$C=0$	$C=1$
$S=\text{Ma}$	$C=0$	0.7	0.4	$S=\text{Ma}$	0.3
	$C=1$	0.8	0.75		0.2

# Probabilistic inference

- Suppose we meet a male patient who eats caviar.
- What is the **posterior probability** that  $H=1$ , i.e.,  $P(H=1 | C=1, S=\text{Ma})$ ? (*Posterior* means after observing  $C, S$ .)

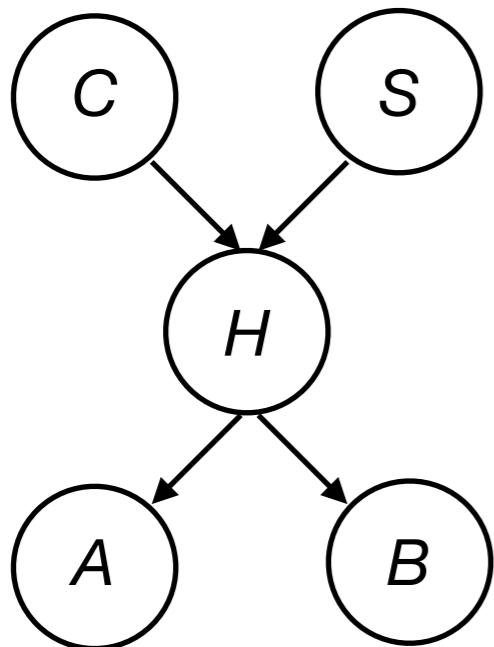


		$P(H=0)$		$P(H=1)$	
		$C=0$	$C=1$	$C=0$	$C=1$
$S=\text{Ma}$	$C=0$	0.7	0.4	$S=\text{Ma}$	0.3
	$C=1$	0.8	0.75		0.6
$S=\text{Fe}$		0.2	0.25		

Just consult the conditional probability distribution.

# Probabilistic inference

- What if we also know that the patient is short of breath?



		$P(H=0)$	
		C=0	C=1
S=Ma	0.7	0.4	
	0.8	0.75	

		$P(H=1)$	
		C=0	C=1
S=Ma	0.3	0.6	
	0.2	0.25	

		$P(B=0)$	
H=0	H=1	0.9	0.1
H=1			

		$P(B=1)$	
H=0	H=1	0.1	0.9
H=1			

**Conditional independence from the graphical model:**  
 $P(b | h, c, s) = P(b | h)$

# Probabilistic inference

$$P(H = 1 \mid C = 1, S = \text{Ma}, B = 1)$$

$$= \frac{P(B = 1 \mid H = 1, C = 1, S = \text{Ma})P(H = 1 \mid C = 1, S = \text{Ma})}{P(B = 1 \mid C = 1, S = \text{Ma})}$$

Bayes' rule

		$P(H=0)$	
		C=0	C=1
S=Ma	C=0	0.7	0.4
	S=Fe	0.8	0.75

		$P(H=1)$	
		C=0	C=1
S=Ma	C=0	0.3	<b>0.6</b>
	S=Fe	0.2	0.25

		$P(B=0)$	
		H=0	0.9
H=1	H=0	0.1	
	H=1		0.1

		$P(B=1)$	
		H=0	0.1
H=1	H=0	0.9	
	H=1		0.9

Conditional independence from the graphical model:

$$P(b \mid h, c, s) = P(b \mid h)$$

# Probabilistic inference

$$P(H = 1 \mid C = 1, S = \text{Ma}, B = 1)$$

$$= \frac{P(B = 1 \mid H = 1, C = 1, S = \text{Ma})P(H = 1 \mid C = 1, S = \text{Ma})}{P(B = 1 \mid C = 1, S = \text{Ma})}$$

Bayes' rule

$$= \frac{P(B = 1 \mid H = 1)P(H = 1 \mid C = 1, S = \text{Ma})}{P(B = 1 \mid C = 1, S = \text{Ma})}$$

Conditional independence

$P(H=0)$		
	C=0	C=1
S=Ma	0.7	0.4
S=Fe	0.8	0.75

$P(H=1)$		
	C=0	C=1
S=Ma	0.3	<b>0.6</b>
S=Fe	0.2	0.25

$P(B=0)$	
H=0	0.9
H=1	0.1

$P(B=1)$	
H=0	0.1
H=1	0.9

Conditional independence from the graphical model:

$$P(b \mid h, c, s) = P(b \mid h)$$

# Probabilistic inference

$$P(H = 1 \mid C = 1, S = \text{Ma}, B = 1)$$

$$= \frac{P(B = 1 \mid H = 1, C = 1, S = \text{Ma})P(H = 1 \mid C = 1, S = \text{Ma})}{P(B = 1 \mid C = 1, S = \text{Ma})}$$

**Bayes' rule**

$$= \frac{P(B = 1 \mid H = 1)P(H = 1 \mid C = 1, S = \text{Ma})}{P(B = 1 \mid C = 1, S = \text{Ma})}$$

**Conditional independence**

$$= \frac{0.9 * 0.6}{\sum_{h=0}^1 P(B = 1, H = h \mid C = 1, S = \text{Ma})}$$

**Law of total probability**

$P(H=0)$		
	C=0	C=1
S=Ma	0.7	0.4
S=Fe	0.8	0.75

$P(H=1)$		
	C=0	C=1
S=Ma	0.3	<b>0.6</b>
S=Fe	0.2	0.25

$P(B=0)$	
H=0	0.9
H=1	0.1

$P(B=1)$	
H=0	0.1
H=1	0.9

**Conditional independence from the graphical model:**

$$P(b \mid h, c, s) = P(b \mid h)$$

# Probabilistic inference

$$P(H = 1 \mid C = 1, S = \text{Ma}, B = 1)$$

$$= \frac{P(B = 1 \mid H = 1, C = 1, S = \text{Ma})P(H = 1 \mid C = 1, S = \text{Ma})}{P(B = 1 \mid C = 1, S = \text{Ma})}$$

Bayes' rule

$$= \frac{P(B = 1 \mid H = 1)P(H = 1 \mid C = 1, S = \text{Ma})}{P(B = 1 \mid C = 1, S = \text{Ma})}$$

Conditional independence

$$= \frac{0.9 * 0.6}{\sum_{h=0}^1 P(B = 1, H = h \mid C = 1, S = \text{Ma})}$$

Law of total probability

$$= \frac{0.54}{\sum_{h=0}^1 P(B = 1 \mid H = h, C = 1, S = \text{Ma})P(H = h \mid C = 1, S = \text{Ma})}$$

Def. of  
cond. prob.

# Probabilistic inference

$$P(H = 1 \mid C = 1, S = \text{Ma}, B = 1)$$

$$= \frac{P(B = 1 \mid H = 1, C = 1, S = \text{Ma})P(H = 1 \mid C = 1, S = \text{Ma})}{P(B = 1 \mid C = 1, S = \text{Ma})}$$

Bayes' rule

$$= \frac{P(B = 1 \mid H = 1)P(H = 1 \mid C = 1, S = \text{Ma})}{P(B = 1 \mid C = 1, S = \text{Ma})}$$

Conditional independence

$$= \frac{0.9 * 0.6}{\sum_{h=0}^1 P(B = 1, H = h \mid C = 1, S = \text{Ma})}$$

Law of total probability

$$= \frac{0.54}{\sum_{h=0}^1 P(B = 1 \mid H = h, C = 1, S = \text{Ma})P(H = h \mid C = 1, S = \text{Ma})}$$

Def. of  
cond. prob.

$$= \frac{0.54}{\sum_{h=0}^1 P(B = 1 \mid H = h)P(H = h \mid C = 1, S = \text{Ma})}$$

Conditional independence

# Probabilistic inference

$$P(H = 1 \mid C = 1, S = \text{Ma}, B = 1)$$

$$= \frac{P(B = 1 \mid H = 1, C = 1, S = \text{Ma})P(H = 1 \mid C = 1, S = \text{Ma})}{P(B = 1 \mid C = 1, S = \text{Ma})}$$

Bayes' rule

$$= \frac{P(B = 1 \mid H = 1)P(H = 1 \mid C = 1, S = \text{Ma})}{P(B = 1 \mid C = 1, S = \text{Ma})}$$

Conditional independence

$$= \frac{0.9 * 0.6}{\sum_{h=0}^1 P(B = 1, H = h \mid C = 1, S = \text{Ma})}$$

Law of total probability

$$= \frac{0.54}{\sum_{h=0}^1 P(B = 1 \mid H = h, C = 1, S = \text{Ma})P(H = h \mid C = 1, S = \text{Ma})}$$

Def. of  
cond. prob.

$$= \frac{0.54}{\sum_{h=0}^1 P(B = 1 \mid H = h)P(H = h \mid C = 1, S = \text{Ma})}$$

Conditional independence

$$= \frac{0.54}{0.1 * 0.4 + 0.9 * 0.6}$$

# Probabilistic inference

$$P(H = 1 \mid C = 1, S = \text{Ma}, B = 1)$$

$$= \frac{P(B = 1 \mid H = 1, C = 1, S = \text{Ma})P(H = 1 \mid C = 1, S = \text{Ma})}{P(B = 1 \mid C = 1, S = \text{Ma})}$$

Bayes' rule

$$= \frac{P(B = 1 \mid H = 1)P(H = 1 \mid C = 1, S = \text{Ma})}{P(B = 1 \mid C = 1, S = \text{Ma})}$$

Conditional independence

$$= \frac{0.9 * 0.6}{\sum_{h=0}^1 P(B = 1, H = h \mid C = 1, S = \text{Ma})}$$

Law of total probability

$$= \frac{0.54}{\sum_{h=0}^1 P(B = 1 \mid H = h, C = 1, S = \text{Ma})P(H = h \mid C = 1, S = \text{Ma})}$$

Def. of  
cond. prob.

$$= \frac{0.54}{\sum_{h=0}^1 P(B = 1 \mid H = h)P(H = h \mid C = 1, S = \text{Ma})}$$

Conditional independence

$$= \frac{0.54}{0.1 * 0.4 + 0.9 * 0.6}$$

$$= \frac{0.54}{0.04 + 0.54}$$

$$\approx 0.93$$

# Probabilistic inference

- Alternatively, it is often easier to work with **unnormalized probabilities**, i.e., values *proportional* to the probabilities.

$$P(H = h \mid C = 1, S = \text{Ma}, B = 1)$$

$$\propto P(B = 1 \mid H = h, C = 1, S = \text{Ma}) P(H = h \mid C = 1, S = \text{Ma}) \quad \text{Bayes' rule}$$

$$= P(B = 1 \mid H = h) P(H = h \mid C = 1, S = \text{Ma}) \quad \text{Conditional independence}$$

# Probabilistic inference

- Alternatively, it is often easier to work with **unnormalized probabilities**, i.e., values *proportional* to the probabilities.

$$\begin{aligned} & P(H = 1 \mid C = 1, S = \text{Ma}, B = 1) \\ & \propto P(B = 1 \mid H = 1)P(H = 1 \mid C = 1, S = \text{Ma}) \\ & = 0.9 * 0.6 \end{aligned}$$

# Probabilistic inference

- Alternatively, it is often easier to work with **unnormalized probabilities**, i.e., values *proportional* to the probabilities.

$$P(H = 1 \mid C = 1, S = \text{Ma}, B = 1)$$

$$\propto P(B = 1 \mid H = 1)P(H = 1 \mid C = 1, S = \text{Ma})$$

$$= 0.9 * 0.6$$

$$P(H = 0 \mid C = 1, S = \text{Ma}, B = 1)$$

$$\propto P(B = 1 \mid H = 0)P(H = 0 \mid C = 1, S = \text{Ma})$$

$$= 0.1 * 0.4$$

⇒

# Probabilistic inference

- Alternatively, it is often easier to work with **unnormalized probabilities**, i.e., values *proportional* to the probabilities.

$$P(H = 1 \mid C = 1, S = \text{Ma}, B = 1)$$

$$\propto P(B = 1 \mid H = 1)P(H = 1 \mid C = 1, S = \text{Ma})$$

$$= 0.9 * 0.6$$

$$P(H = 0 \mid C = 1, S = \text{Ma}, B = 1)$$

$$\propto P(B = 1 \mid H = 0)P(H = 0 \mid C = 1, S = \text{Ma})$$

$$= 0.1 * 0.4$$

⇒

$$P(H = 1 \mid C = 1, S = \text{Ma}, B = 1) = \frac{0.9 * 0.6}{0.9 * 0.6 + 0.1 * 0.4} = 0.93$$

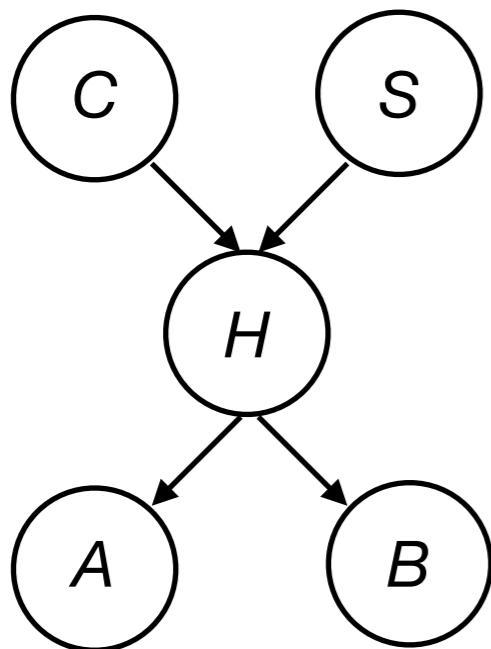
$$P(H = 0 \mid C = 1, S = \text{Ma}, B = 1) = \frac{0.1 * 0.4}{0.9 * 0.6 + 0.1 * 0.4} = 0.07$$

**since the probabilities must sum to 1.**

# **Maximum likelihood estimation (MLE)**

# Parameters in probability distributions

- Most probabilistic models have parameters we want to estimate.
- For example, the conditional probabilities for medical diagnosis are all parameters that must be learned.



$P(H=0)$		
	C=0	C=1
S=Ma	0.7	0.4
S=Fe	0.8	0.75

$P(H=1)$		
	C=0	C=1
S=Ma	0.3	0.6
S=Fe	0.2	0.25

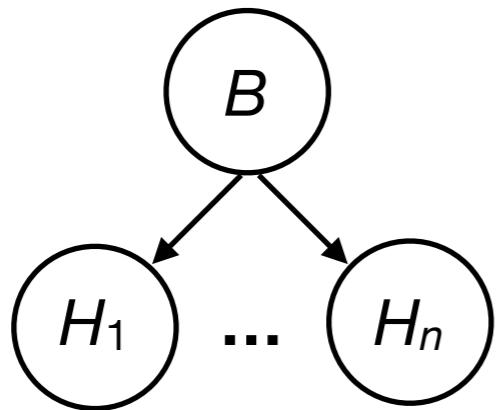
$P(B=0)$	
H=0	0.9
H=1	0.1

$P(B=1)$	
H=0	0.1
H=1	0.9

# Parameters in probability distributions

- Most probabilistic models have parameters we want to estimate.
- As another example, we might want to estimate the bias  $B$  of a coin after observing  $n$  coin flips  $H_1, \dots, H_n$ :



$$P(H_i = 1 | b) = b$$

**Conditional independence:**  
 $P(h_i | b, h_1, \dots, h_{i-1}, h_{i+1}, \dots, h_n) = P(h_i | b)$

- What is a principled approach to estimating  $B$ ?

# Maximum likelihood estimation (MLE)

- Maximum likelihood estimation (MLE):
  - The value of a latent variable is estimated as the one that makes the observed data as likely (probable) as possible.
- The **likelihood** of  $H_1, \dots, H_n$  given  $B$  is:

$$P(h_1, \dots, h_n \mid b) = P(h_1 \mid b) \prod_{i=2}^n P(h_i \mid b, h_1, \dots, h_{i-1})$$

# Maximum likelihood estimation (MLE)

- Maximum likelihood estimation (MLE):
  - The value of a latent variable is estimated as the one that makes the observed data as likely (probable) as possible.
- The **likelihood** of  $H_1, \dots, H_n$  given  $B$  is:

$$\begin{aligned} P(h_1, \dots, h_n \mid b) &= P(h_1 \mid b) \prod_{i=2}^n P(h_i \mid b, h_1, \dots, h_{i-1}) \\ &= P(h_1 \mid b) \prod_{i=2}^n P(h_i \mid b) \quad \text{Conditional independence} \end{aligned}$$

# Maximum likelihood estimation (MLE)

- Maximum likelihood estimation (MLE):
  - The value of a latent variable is estimated as the one that makes the observed data as likely (probable) as possible.
- The **likelihood** of  $H_1, \dots, H_n$  given  $B$  is:

$$\begin{aligned} P(h_1, \dots, h_n \mid b) &= P(h_1 \mid b) \prod_{i=2}^n P(h_i \mid b, h_1, \dots, h_{i-1}) \\ &= P(h_1 \mid b) \prod_{i=2}^n P(h_i \mid b) \quad \text{Conditional independence} \\ &= \prod_{i=1}^n P(h_i \mid b) \end{aligned}$$

# Maximum likelihood estimation (MLE)

- We can express the probability of each  $h_i$  given  $b$  as:

$$P(h_i \mid b) = b^{h_i} (1 - b)^{1-h_i}$$

# Maximum likelihood estimation (MLE)

- We can express the probability of each  $h_i$  given  $b$  as:

$$\begin{aligned} P(h_i \mid b) &= b^{h_i} (1 - b)^{1-h_i} \\ &= b \text{ if } h_i = 1 \\ &= (1 - b) \text{ if } h_i = 0 \end{aligned}$$

The exponent “chooses”  
the correct probability  
for  $H_i=1$  or  $H_i=0$ .

# Maximum likelihood estimation (MLE)

- We seek to maximize the probability of  $h_1, \dots, h_n$  by optimizing  $b$ .
- It's often easier instead to optimize the **log-likelihood**.

$$\arg \max_b P(h_1, \dots, h_n \mid b) = \arg \max_b \log P(h_1, \dots, h_n \mid b)$$

# Maximum likelihood estimation (MLE)

- We seek to maximize the probability of  $h_1, \dots, h_n$  by optimizing  $b$ .
- It's often easier instead to optimize the **log-likelihood**.

$$\arg \max_b P(h_1, \dots, h_n \mid b) = \arg \max_b \log P(h_1, \dots, h_n \mid b)$$

where

$$\log P(h_1, \dots, h_n \mid b) = \log \prod_{i=1}^n P(h_i \mid b)$$

# Maximum likelihood estimation (MLE)

- We seek to maximize the probability of  $h_1, \dots, h_n$  by optimizing  $b$ .
- It's often easier instead to optimize the **log-likelihood**.

$$\arg \max_b P(h_1, \dots, h_n \mid b) = \arg \max_b \log P(h_1, \dots, h_n \mid b)$$

where

$$\begin{aligned}\log P(h_1, \dots, h_n \mid b) &= \log \prod_{i=1}^n P(h_i \mid b) \\ &= \sum_{i=1}^n \log P(h_i \mid b)\end{aligned}$$

# Maximum likelihood estimation (MLE)

- We seek to maximize the probability of  $h_1, \dots, h_n$  by optimizing  $b$ .
- It's often easier instead to optimize the **log-likelihood**.

$$\arg \max_b P(h_1, \dots, h_n \mid b) = \arg \max_b \log P(h_1, \dots, h_n \mid b)$$

where

$$\begin{aligned}\log P(h_1, \dots, h_n \mid b) &= \log \prod_{i=1}^n P(h_i \mid b) \\ &= \sum_{i=1}^n \log P(h_i \mid b) \\ &= \sum_{i=1}^n \log b^{h_i} (1 - b)^{1-h_i}\end{aligned}$$

# Maximum likelihood estimation (MLE)

- We seek to maximize the probability of  $h_1, \dots, h_n$  by optimizing  $b$ .
- It's often easier instead to optimize the **log-likelihood**.

$$\arg \max_b P(h_1, \dots, h_n \mid b) = \arg \max_b \log P(h_1, \dots, h_n \mid b)$$

where

$$\begin{aligned}\log P(h_1, \dots, h_n \mid b) &= \log \prod_{i=1}^n P(h_i \mid b) \\ &= \sum_{i=1}^n \log P(h_i \mid b) \\ &= \sum_{i=1}^n \log b^{h_i} (1 - b)^{1-h_i} \\ &= \sum_{i=1}^n h_i \log b + (1 - h_i) \log(1 - b)\end{aligned}$$

# Maximum likelihood estimation (MLE)

- We seek to maximize the probability of  $h_1, \dots, h_n$  by optimizing  $b$ .
- It's often easier instead to optimize the **log-likelihood**.

$$\arg \max_b P(h_1, \dots, h_n \mid b) = \arg \max_b \log P(h_1, \dots, h_n \mid b)$$

where

$$\begin{aligned}\log P(h_1, \dots, h_n \mid b) &= \log \prod_{i=1}^n P(h_i \mid b) \\ &= \sum_{i=1}^n \log P(h_i \mid b) \\ &= \sum_{i=1}^n \log b^{h_i} (1 - b)^{1-h_i} \\ &= \sum_{i=1}^n h_i \log b + (1 - h_i) \log(1 - b) \\ &= n_1 \log b + (n - n_1) \log(1 - b)\end{aligned}$$

***n<sub>1</sub>* is number of heads.**

# Maximum likelihood estimation (MLE)

- We can now differentiate w.r.t.  $b$ , set to 0, and solve to obtain the MLE of  $B$ :

$$\nabla_b [n_1 \log b + (n - n_1) \log(1 - b)] = \frac{n_1}{b} - \frac{(n - n_1)}{1 - b}$$

$$b = ?$$

# Maximum likelihood estimation (MLE)

- We can now differentiate w.r.t.  $b$ , set to 0, and solve to obtain the MLE of  $B$ :

$$\nabla_b [n_1 \log b + (n - n_1) \log(1 - b)] = \frac{n_1}{b} - \frac{(n - n_1)}{1 - b}$$

$$(1 - b)n_1 - b(n - n_1) = 0$$

$$n_1 - bn_1 - bn + bn_1 = 0$$

$$n_1 = bn$$

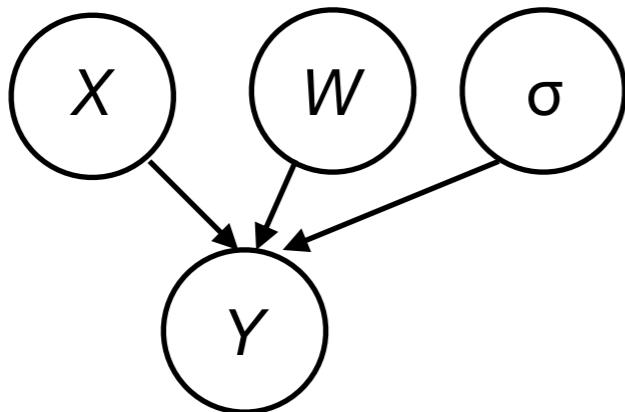
$$b = \frac{n_1}{n}$$

The MLE for  $B$  is the fraction of coin flips that are heads.

# **Linear-Gaussian models**

# Linear-Gaussian model

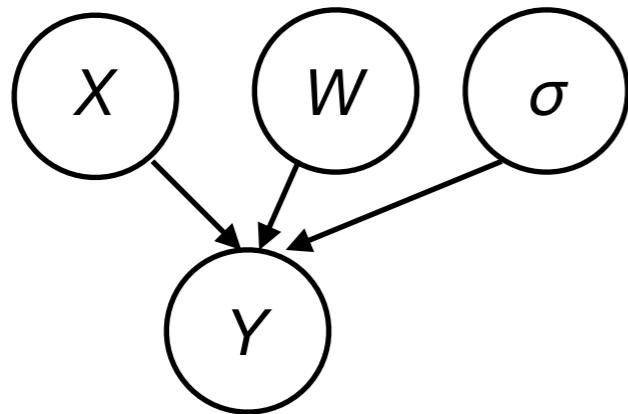
- Let's consider a different model that contains **real-valued** RVs (not just from a finite outcome set).



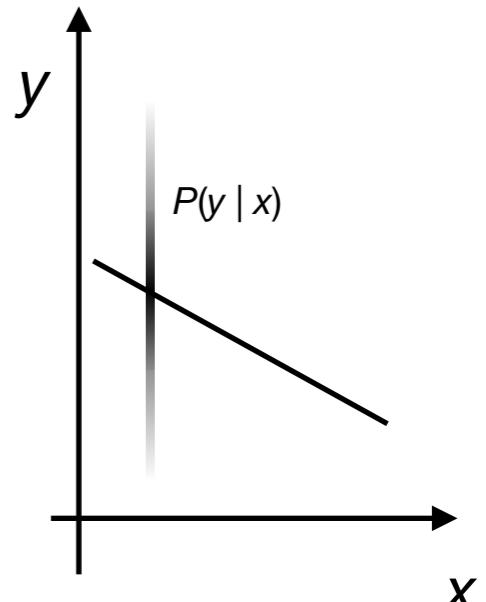
- $X$  is some feature vector (e.g., face image).
- $Y$  is some outcome variable (e.g., age).
- $W$  is a vector of weights that characterize how  $Y$  is related to  $X$ .
- $\sigma$  expresses how uncertain we are about  $Y$  after seeing  $X$ .

# Linear-Gaussian model

- Suppose we model the relationship between  $X$ ,  $W$ ,  $\sigma$ , and  $Y$  such that:

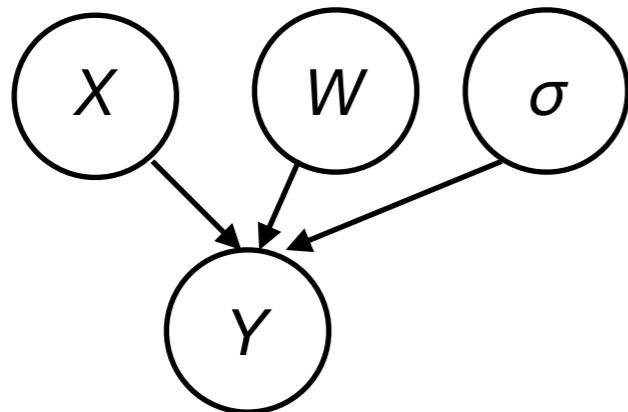


- $Y$  is a normal/Gaussian random variable.
- The expected value of  $Y$  is  $X^T W$ .
- The variance of  $Y$  is constant ( $\sigma^2$ ) for all possible  $X$ .



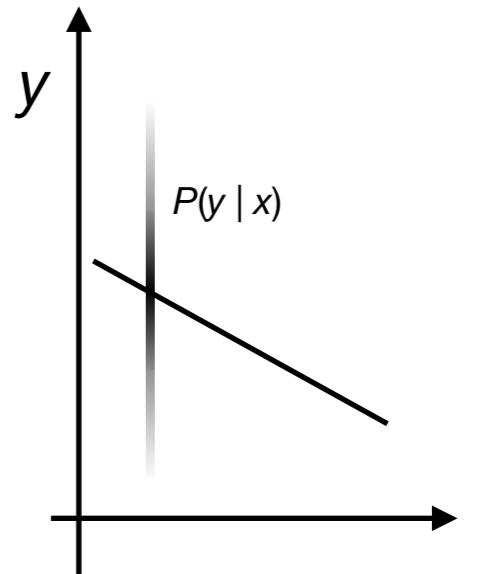
# Linear-Gaussian model

- Suppose we model the relationship between  $X$ ,  $W$ ,  $\sigma$ , and  $Y$  such that:



$$P(y \mid \mathbf{w}, \mathbf{x}) = \mathcal{N}(y; \mathbf{x}^\top \mathbf{w}, \sigma^2)$$

- If we collect a dataset  $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n$ , what is the MLE for  $W$  and  $\sigma$ ?



# Linear-Gaussian model

$$P(y \mid \mathbf{x}, \mathbf{w}, \sigma^2) = \mathcal{N}(y; \mathbf{x}^\top \mathbf{w}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \mathbf{x}^\top \mathbf{w})^2}{2\sigma^2}\right)$$

# Linear-Gaussian model

$$P(y \mid \mathbf{x}, \mathbf{w}, \sigma^2) = \mathcal{N}(y; \mathbf{x}^\top \mathbf{w}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \mathbf{x}^\top \mathbf{w})^2}{2\sigma^2}\right)$$

$$P(\mathcal{D} \mid \mathbf{w}, \sigma^2) = \prod_{i=1}^n P(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w}) \quad \text{Conditional independence}$$

# Linear-Gaussian model

$$P(y \mid \mathbf{x}, \mathbf{w}, \sigma^2) = \mathcal{N}(y; \mathbf{x}^\top \mathbf{w}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \mathbf{x}^\top \mathbf{w})^2}{2\sigma^2}\right)$$

$$P(\mathcal{D} \mid \mathbf{w}, \sigma^2) = \prod_{i=1}^n P(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w}) \quad \text{Conditional independence}$$

$$\log P(\mathcal{D} \mid \mathbf{w}, \sigma^2) = \log \prod_{i=1}^n P(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w})$$

# Linear-Gaussian model

$$P(y \mid \mathbf{x}, \mathbf{w}, \sigma^2) = \mathcal{N}(y; \mathbf{x}^\top \mathbf{w}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \mathbf{x}^\top \mathbf{w})^2}{2\sigma^2}\right)$$

$$P(\mathcal{D} \mid \mathbf{w}, \sigma^2) = \prod_{i=1}^n P(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w}) \quad \text{Conditional independence}$$

$$\log P(\mathcal{D} \mid \mathbf{w}, \sigma^2) = \log \prod_{i=1}^n P(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w})$$

$$= \sum_{i=1}^n \log P(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w})$$

...

# Linear-Gaussian model

- MLE for  $\mathbf{w}$ :

$$\mathbf{w} = \left( \sum_{i=1}^n \mathbf{x}^{(i)} \mathbf{x}^{(i)\top} \right)^{-1} \left( \sum_{i=1}^n \mathbf{x}^{(i)} y^{(i)} \right)$$

This is the same solution as for linear regression, but derived as the MLE of a probabilistic model (instead of the minimum MSE).

# Linear-Gaussian model

- MLE for  $\sigma^2$ :

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}^{(i)^\top} \mathbf{w} - y^{(i)})^2$$

This is the sum of squared residuals of the predictions w.r.t. ground-truth.