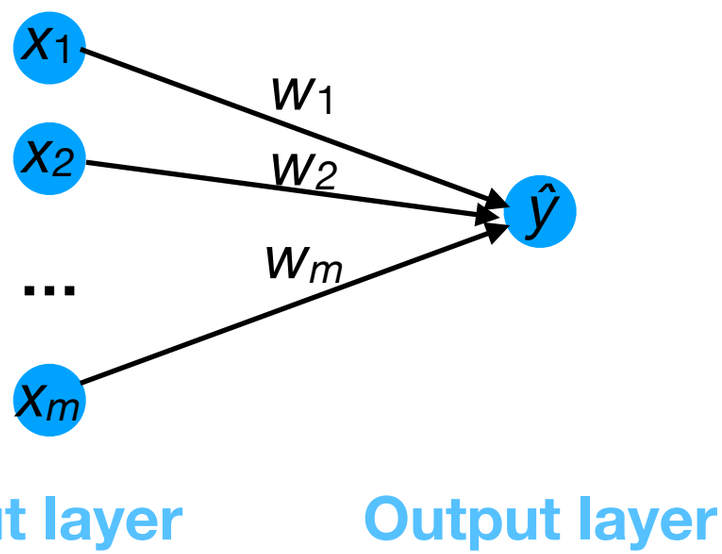
CS/DS 541: Class 3

Jacob Whitehill

Exercise

Gradient descent

- For the 2-layer NN below, let m=2 and $\mathbf{w}^{(0)}$=[1 0]^T.
- Compute the updated weight vector $\mathbf{w}^{(1)}$ after one iteration of gradient descent using MSE loss, a single training example ($\mathbf{x}$,y)=([2, 3]^T, 4), and learning rate $\boldsymbol{\epsilon}$=0.1.

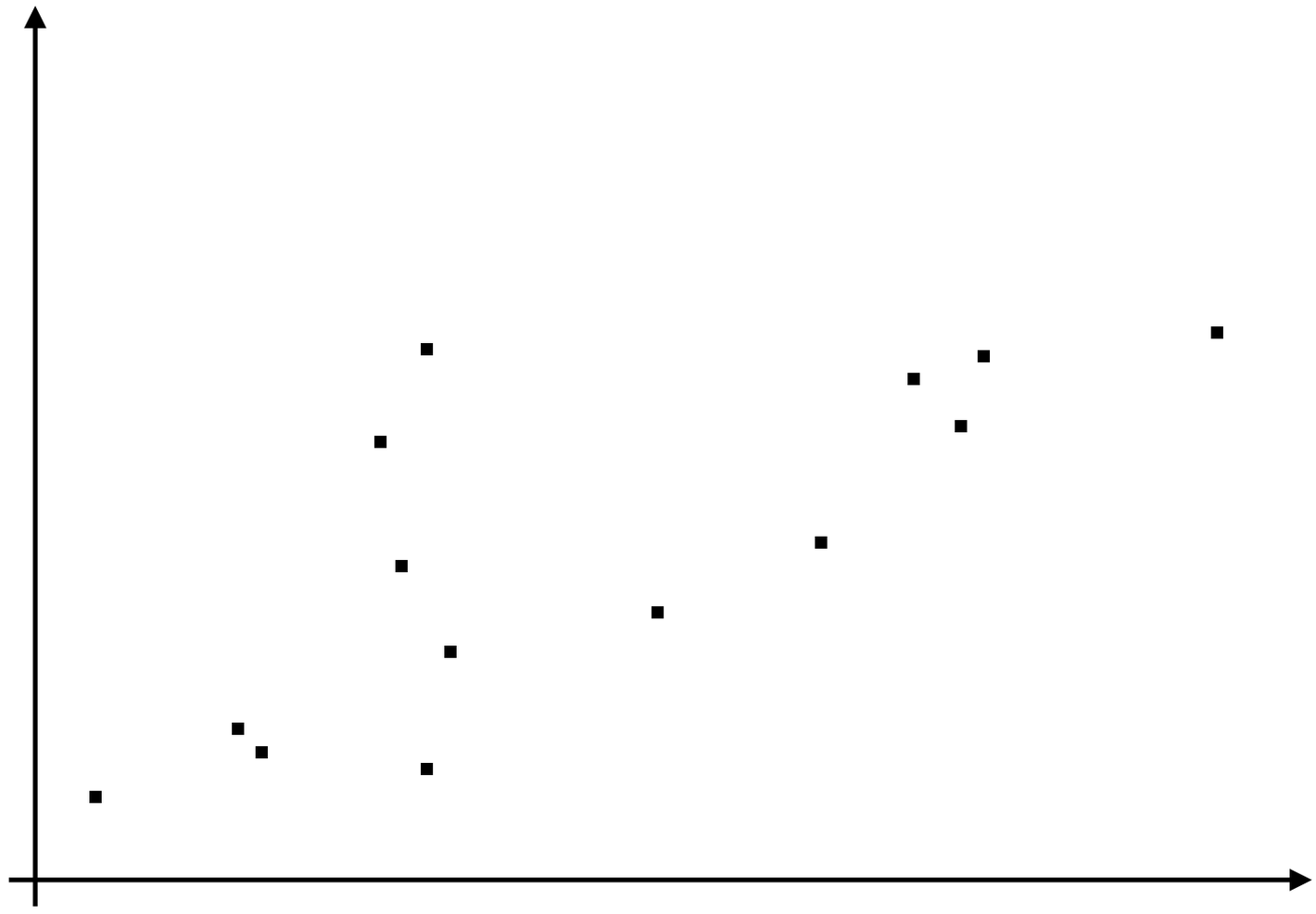


Solution

Probabilistic machine learning

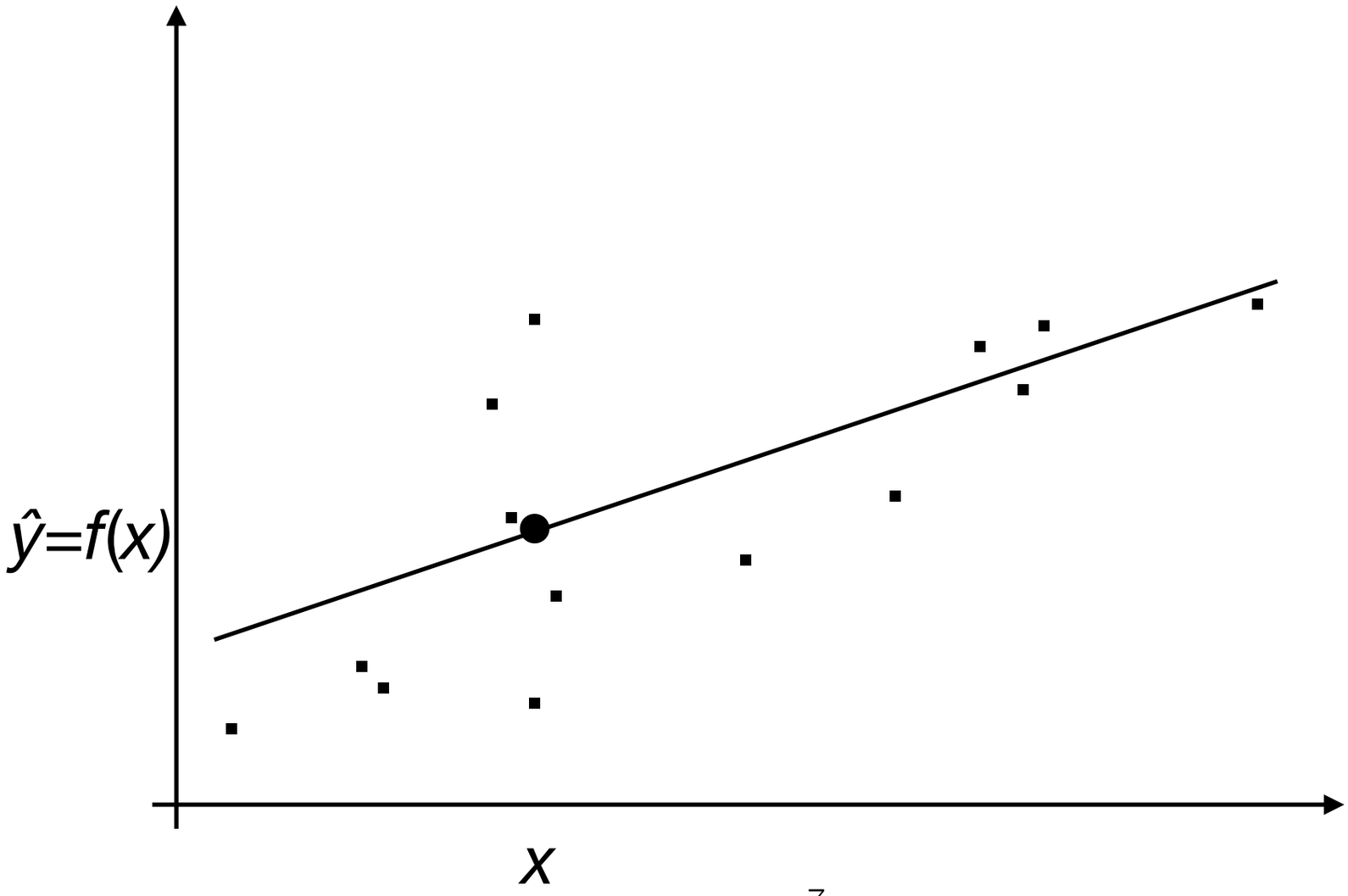
Uncertainty of estimation

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



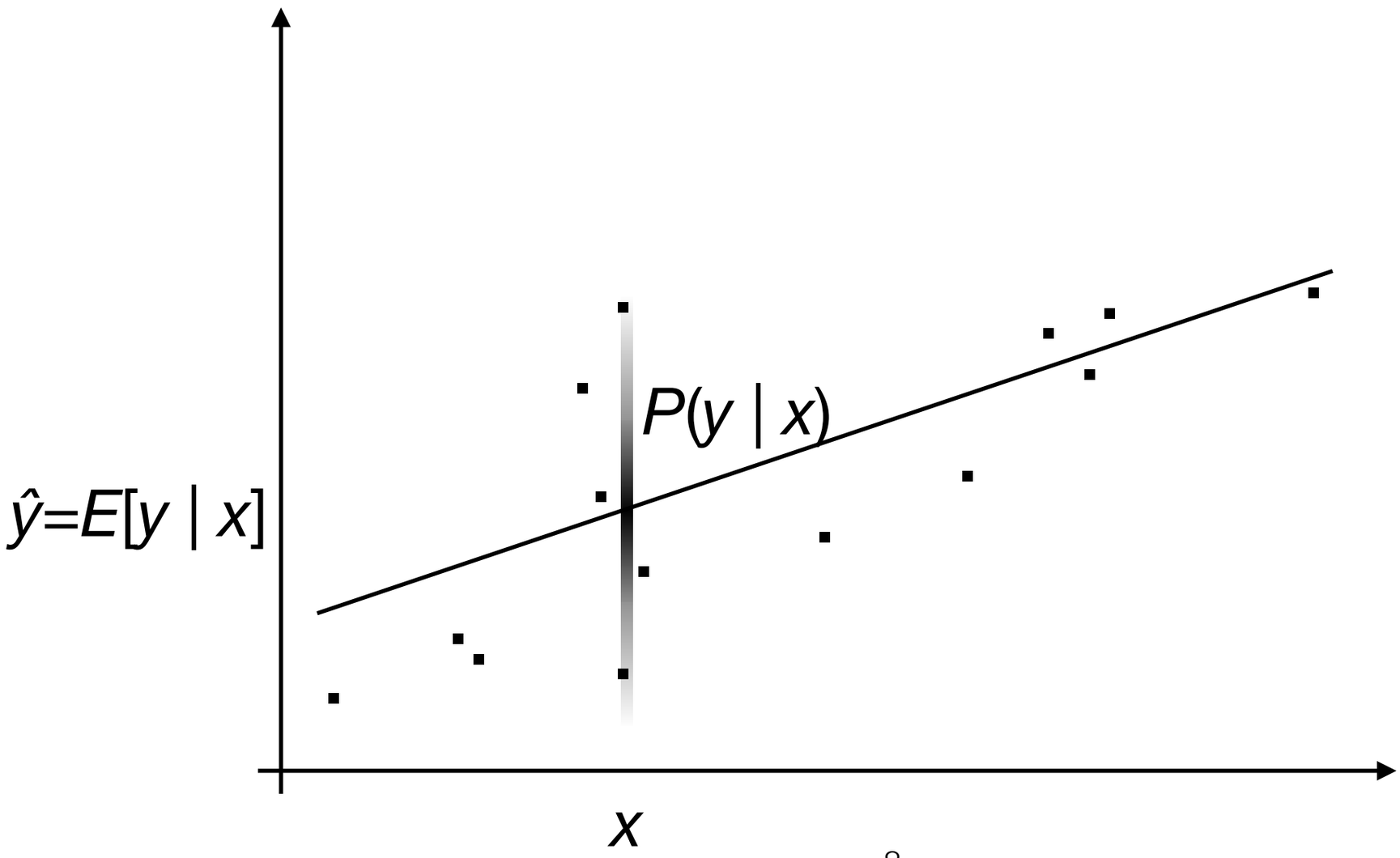
Uncertainty of estimation

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



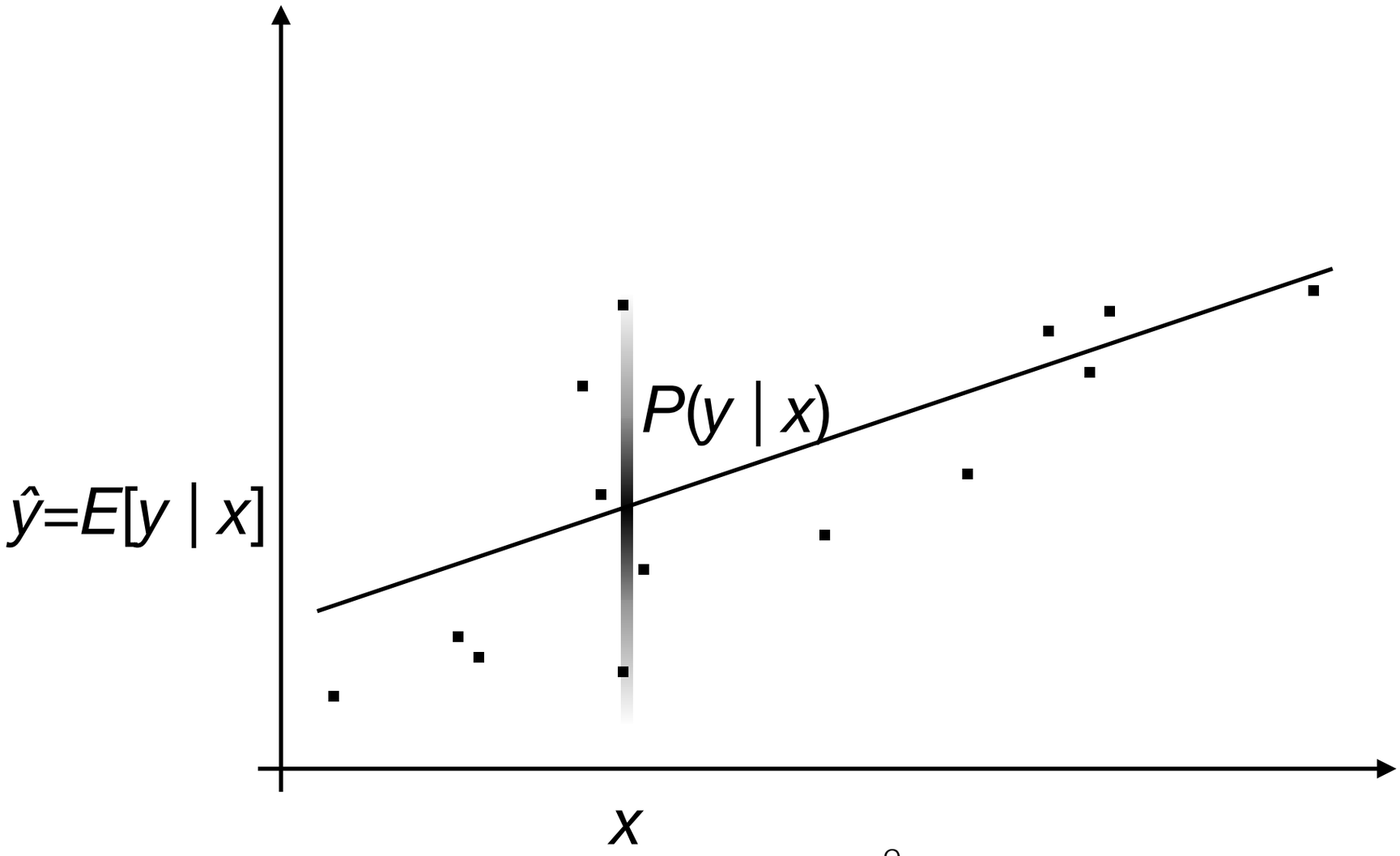
Uncertainty of estimation

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



Uncertainty of estimation

- Indeed, it turns out that the optimal parameters for a conditional Gaussian probability model are **exactly the same** as for linear regression with minimal MSE (i.e., 2-layer NN).



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 sample space Ω) 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 sample spaces Ω:
  - Finite, e.g., { 0, 1 }, { 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 sample space.
- 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 sample spaces.

- **Probability density function** (**PDF**) for uncountable sample spaces.

PMF

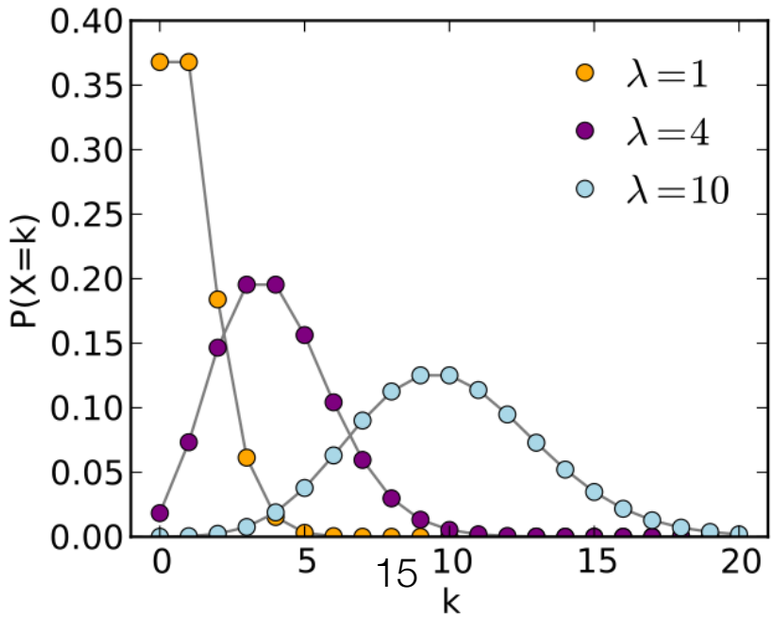
- Example 1 (finite): 
- Let RV *X* be the outcome of rolling a 6-sided die.
- If *X* is fair, then: $P(X = i) = \frac{1}{6} \quad \forall i \in \{1, \ldots, 6\}$

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:

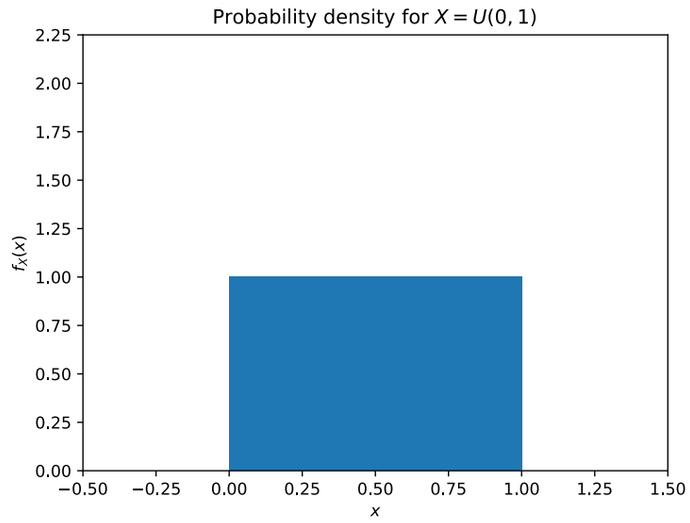
$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

where parameter λ specifies the *rate* of the packet arrivals.



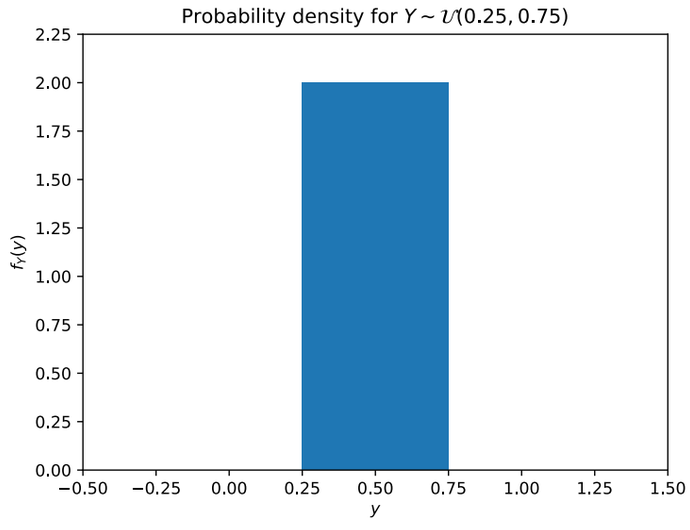
PDF

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



- Example 2:
  - Let *Y* be a uniformly-distributed RV over Ω=[1/4, 3/4].
    - Then $f_Y(y) = 2 \quad \forall y \in \Omega$

Note that the PDF can exceed 1.

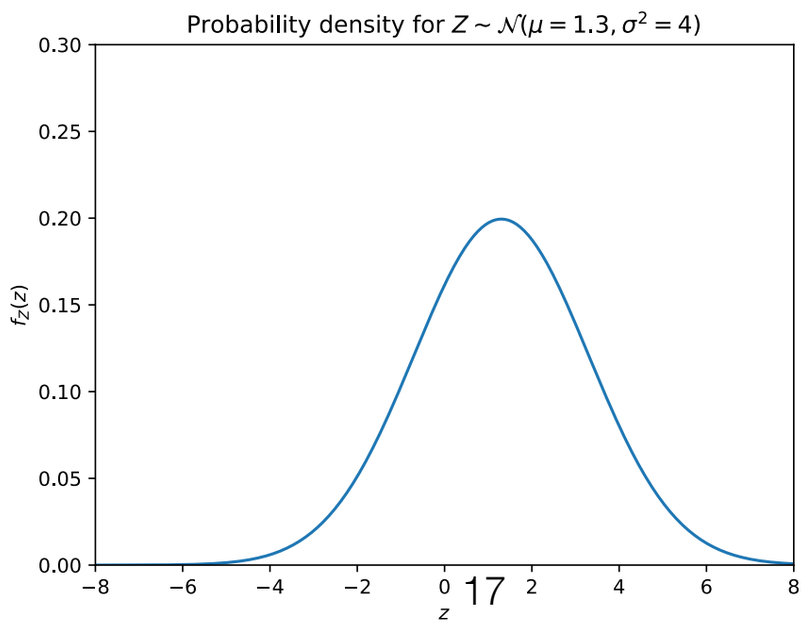


PDF

- 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 μ is the mean and σ^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*, …, we can construct the joint probability distribution *P*(*x*, *y*, …) to mean the probability that (*X*=*x*) ^ (*Y*=*y*) ^ ….

- Note that *P* must still sum to 1 over all possible joint values (*x*, *y*, …).

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

https://www.nytimes.com/2017/03/31/business/crayon-dandelion-crayola.html

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

https://www.nytimes.com/2017/03/31/business/crayon-dandelion-crayola.html

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	**0.375**
Med	0.075	0.05	0.1	0	**0.225**
High	0.25	0.05	0.075	0.025	**0.4**
P(x)	**0.425**	**0.15**	**0.2**	**0.225**	

https://www.nytimes.com/2017/03/31/business/crayon-dandelion-crayola.html

Law of total probability

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

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

https://www.nytimes.com/2017/03/31/business/crayon-dandelion-crayola.html

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, …, W_T) of words in a sentence.
    - W_t is the t^{th} RV in the sequence.
  - Grid (I_{11}, …, I_{1M}, …, I_{N1}, …, I_{NM}) of the pixels in an N x 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* | *w*)
  - Heart disease given cholesterol: *P*(*d* | *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, …, X_n given the values of Y_1, …, Y_m:

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

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:

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

- And also:

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

Conditional probability distributions

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

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

Conditional probability distributions

- Exercises:

1. $P(a, b, c, d) = P(a, c) * ?$
2. $P(w_1, w_2, w_3) = P(w_3 \mid w_1) * ? * ?$
3. $P(x_1, x_2, x_3) = P(x_1) * ? * P(x_3 \mid x_1, x_2)$
4. $P(x_1, \ldots, x_n) = P(x_1) * ? * ? * \ldots * ?$

n-1 terms

Conditional probability distributions

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 $P(x \mid y) = P(x)$ and $P(y \mid x) = P(y)$ since $P(x, y) = P(x \mid y)P(y) = P(y \mid x)P(x)$ by definition of conditional probability.

Conditional independence

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

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

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

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

Bayes' rule

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

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

Bayes' rule

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

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

Bayes' rule

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

Bayes' rule

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

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

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

$$= \frac{P(y \mid x)P(x)}{\int P(y \mid x)P(x)dx}$$

Bayes' rule

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

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

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

$$= \frac{P(y \mid x)P(x)}{\int P(y \mid x)P(x)dx}$$

Normalizing constant

Bayes' rule

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

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

… we might only need to compute the numerator:

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

$$= \frac{P(y \mid x)P(x)}{\int P(y \mid x)P(x)dx}$$

$$P(x \mid y) \propto P(y \mid x)P(x)$$

Probabilistic inference

Probabilistic graphical models

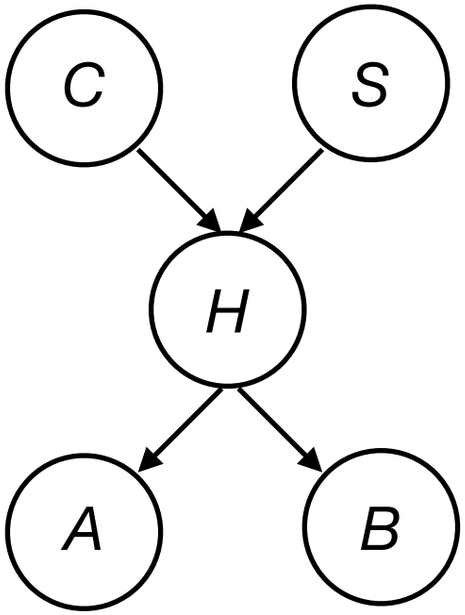
- To express the conditional independence relationships between multiple RVs, it is useful to represent their dependencies in a graph.

- 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

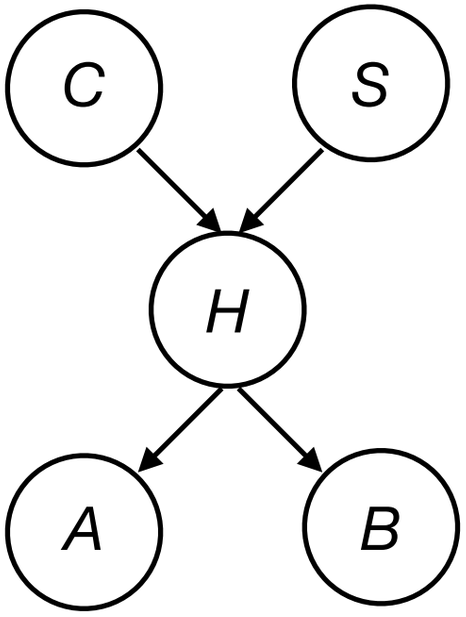
- Example 1 — medical diagnosis:



- *C*: whether the patient eats **c**aviar.
- *S*: the patient's **s**ex
- *H*: whether the patient has **h**igh cholesterol
- *A*: whether the patient will have a heart **a**ttack.
- *B*: whether the patient has shortness of **b**reath.

Probabilistic graphical models

- Example 1 — medical diagnosis:



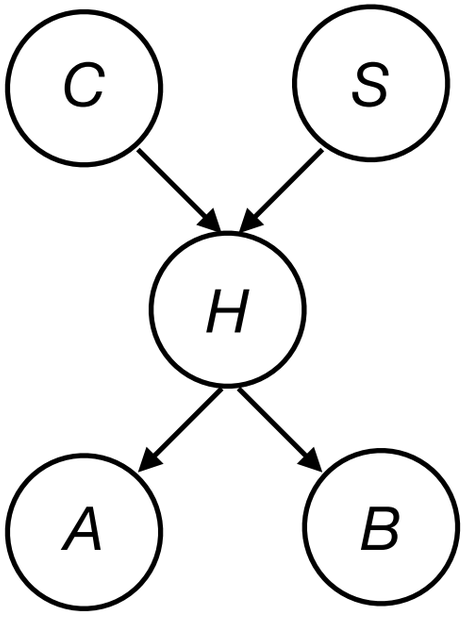
- This model implies that:

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

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

Probabilistic graphical models

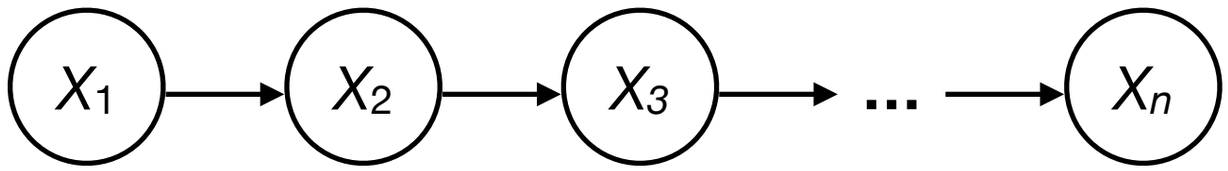
- Example 1 — medical diagnosis:



- 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, …, X_n implies that:

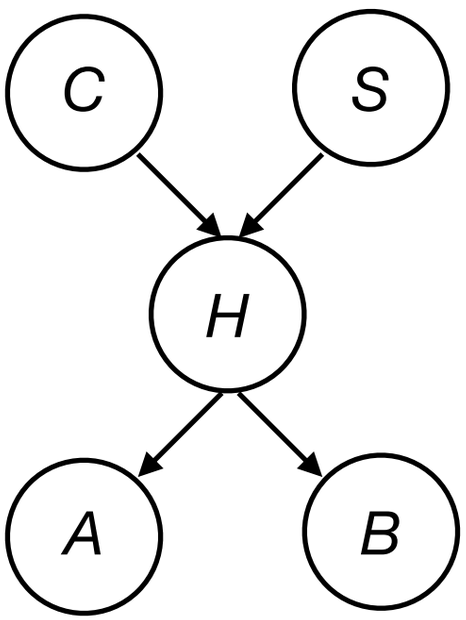
$$P(x_i \mid x_1, \ldots, x_{i-1}) = P(x_i \mid x_{i-1}) \text{ and}$$
$$P(x_i \mid x_{i+1}, \ldots, x_n) = P(x_i \mid x_{i+1})$$

- In words, “If I want to know the value of X_i and I already know $X_{i\text{-}1}$, then the values of any ‘earlier’ 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:



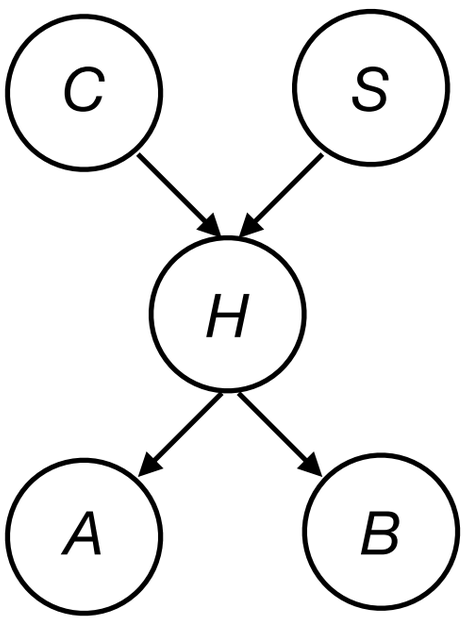
$P(H=0)$		
	$C=0$	$C=1$
S=Ma		
S=Fe		

$P(H=1)$		
	$C=0$	$C=1$
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:



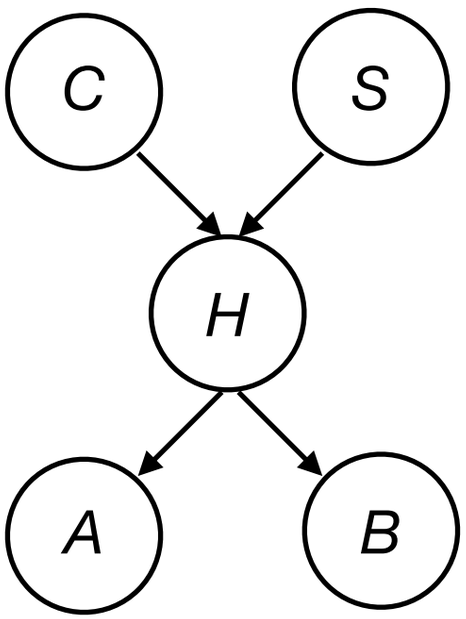
$P(H=0)$	$C=0$	$C=1$
S=Ma	0.7	0.4
S=Fe	0.8	0.75

$P(H=1)$	$C=0$	$C=1$
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:



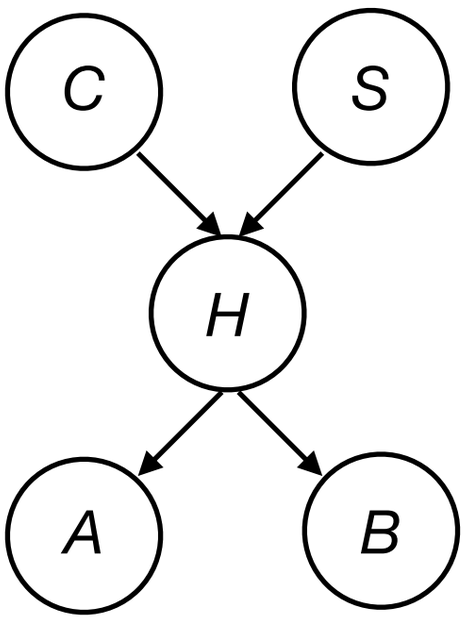
$P(A=0)$	
$H=0$	0.75
$H=1$	0.2

$P(A=1)$	
$H=0$	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:



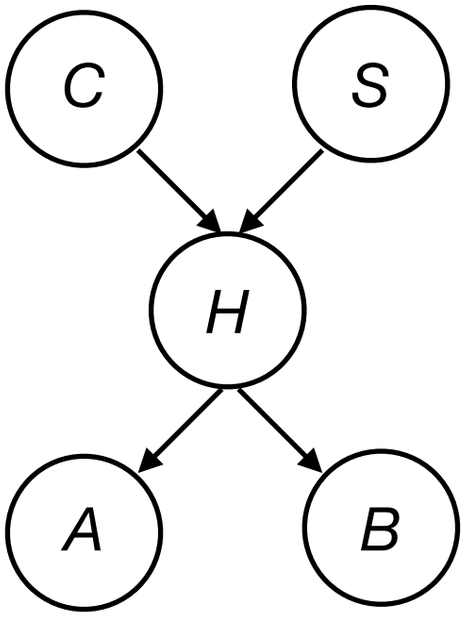
$P(B=0)$	
$H=0$	0.9
$H=1$	0.1

$P(B=1)$	
$H=0$	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., $P(H=1 \mid C=1, S=\text{Ma})$? (*Posterior* means after observing C, S.)



$P(H=0)$

	C=0	C=1
S=Ma	0.7	0.4
S=Fe	0.8	0.75

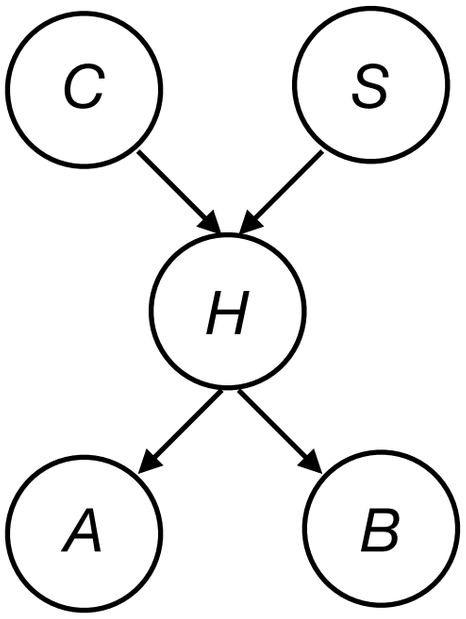
$P(H=1)$

	C=0	C=1
S=Ma	0.3	0.6
S=Fe	0.2	0.25

Probabilistic inference

- Suppose we meet a male patient who eats caviar.

- What is the **posterior probability** that H=1, i.e., $P(H=1 \mid C=1, S=\text{Ma})$? (*Posterior* means after observing C, S.)



$P(H=0)$

	C=0	C=1
S=Ma	0.7	0.4
S=Fe	0.8	0.75

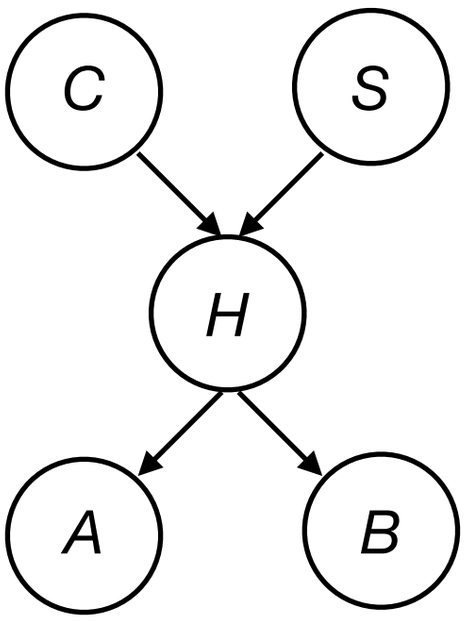
$P(H=1)$

	C=0	C=1
S=Ma	0.3	**0.6**
S=Fe	0.2	0.25

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
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

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

$P(H=0)$	$C=0$	$C=1$
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

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	**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

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	**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

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.

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

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

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

Probabilistic inference

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

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

Probabilistic inference

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

$$
\begin{aligned}
&P(H = 1 \mid C = 1, S = \mathrm{Ma}, B = 1) \\
&\propto P(B = 1 \mid H = 1) P(H = 1 \mid C = 1, S = \mathrm{Ma}) \\
&= 0.9 * 0.6 \\
&P(H = 0 \mid C = 1, S = \mathrm{Ma}, B = 1) \\
&\propto P(B = 1 \mid H = 0) P(H = 0 \mid C = 1, S = \mathrm{Ma}) \\
&= 0.1 * 0.4 \\
&\Longrightarrow
\end{aligned}
$$

Probabilistic inference

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

$$
\begin{aligned}
&P(H = 1 \mid C = 1, S = \text{Ma}, B = 1) \\
&\propto P(B = 1 \mid H = 1) P(H = 1 \mid C = 1, S = \text{Ma}) \\
&= 0.9 * 0.6 \\
&P(H = 0 \mid C = 1, S = \text{Ma}, B = 1) \\
&\propto P(B = 1 \mid H = 0) P(H = 0 \mid C = 1, S = \text{Ma}) \\
&= 0.1 * 0.4
\end{aligned}
$$

$$\Longrightarrow$$

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

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

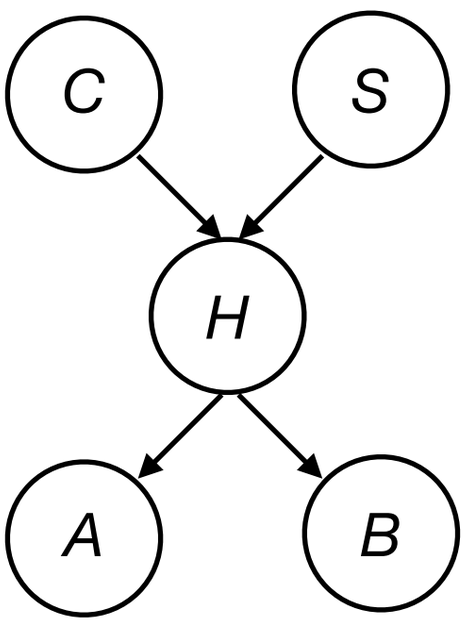
since the probabilities must sum to 1.

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

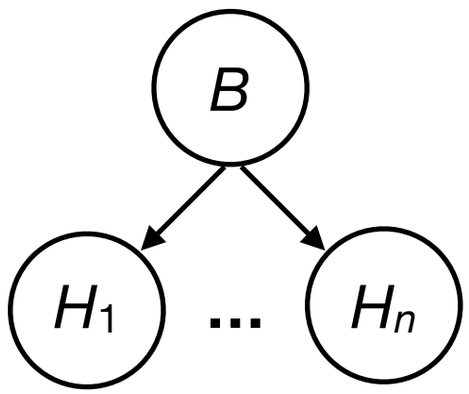
$P(B=1)$

$H=0$	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 $H_1, \ldots, H_n$:



$$P(H_i = 1 \mid b) = b$$

Conditional independence:

$$P(h_i \mid b, h_1, \ldots, h_{i-1}, h_{i+1}, \ldots, h_n) = \mathrm{P}(h_i \mid b)$$

- 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, …, H_n given B is:

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

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 $H_1, \ldots, H_n$ given B is:

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

$$= P(h_1 \mid b) \prod_{i=2}^{n} P(h_i \mid b)$$

Conditional independence

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 $H_1, \ldots, H_n$ given B is:

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

$$= P(h_1 \mid b) \prod_{i=2}^{n} P(h_i \mid b) \quad \textbf{Conditional independence}$$

$$= \prod_{i=1}^{n} P(h_i \mid b)$$

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:

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

$$= b \text{ if } h_i = 1 \quad \text{or}$$

$$(1-b) \text{ if } h_i = 0$$

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 $h_1, \ldots, h_n$ by optimizing b.

- It's often easier instead to optimize the **log-likelihood**.

$$\arg\max_b P(h_1, \ldots, h_n \mid b) = \arg\max_b \log P(h_1, \ldots, h_n \mid b)^*$$

Maximum likelihood estimation (MLE)

- We seek to maximize the probability of h_1, …, h_n by optimizing b.

- It's often easier instead to optimize the **log-likelihood**.

$$\arg\max_b P(h_1, \ldots, h_n \mid b) = \arg\max_b \log P(h_1, \ldots, h_n \mid b)^*$$

* assuming the probability is never exactly 0.

Maximum likelihood estimation (MLE)

- We seek to maximize the probability of h_1, …, h_n by optimizing b.

- It's often easier instead to optimize the **log-likelihood**.

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

$$\log P(h_1, \ldots, h_n \mid b) = \log \prod_{i=1}^{n} P(h_i \mid b)$$ due to conditional independence

Maximum likelihood estimation (MLE)

- We seek to maximize the probability of h_1, …, h_n by optimizing b.

- It's often easier instead to optimize the **log-likelihood**.

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

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

$$= \sum_{i=1}^{n} \log P(h_i \mid b)$$

Maximum likelihood estimation (MLE)

- We seek to maximize the probability of h_1, …, h_n by optimizing b.

- It's often easier instead to optimize the **log-likelihood**.

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

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

Maximum likelihood estimation (MLE)

- We seek to maximize the probability of $h_1, \ldots, h_n$ by optimizing b.

- It's often easier instead to optimize the **log-likelihood**.

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

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

Maximum likelihood estimation (MLE)

- We seek to maximize the probability of $h_1, \ldots, h_n$ by optimizing b.

- It's often easier instead to optimize the **log-likelihood**.

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

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

n_1 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*:

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

$b = ?$

Maximum likelihood estimation (MLE)

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

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

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

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

$$n_1 = bn$$

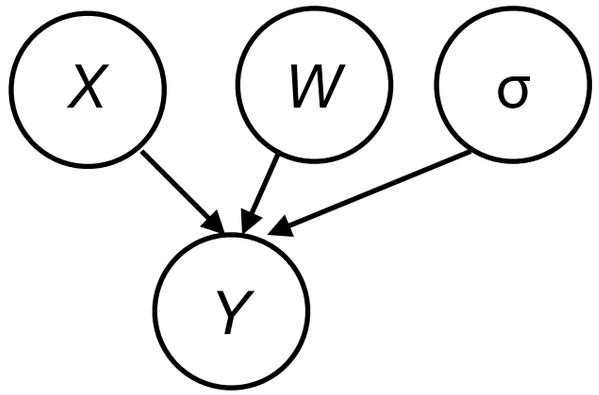
$$b = \frac{n_1}{n}$$

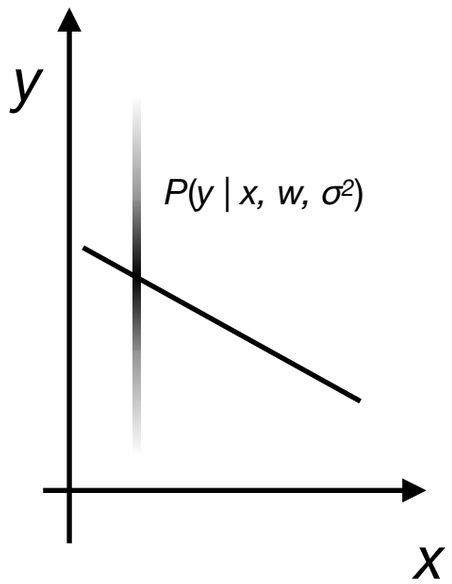
The MLE for *B* is the fraction of coin flips that are heads.

Linear-Gaussian models

Linear-Gaussian model

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

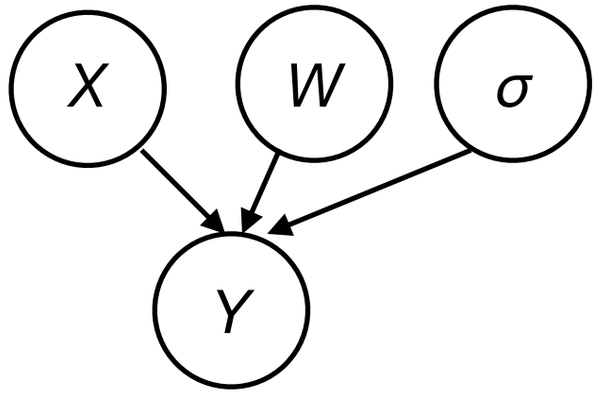


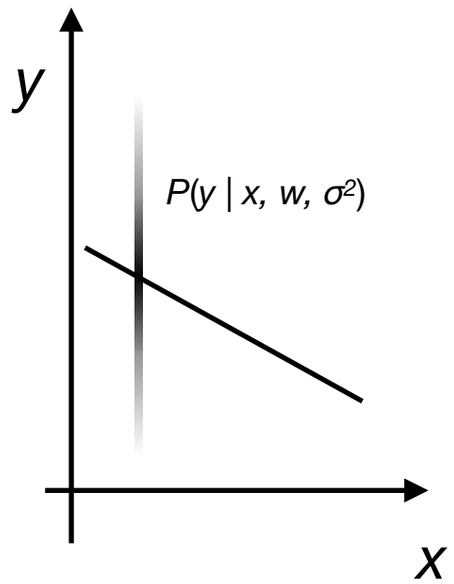


- *X* is some feature vector (e.g., face image).
- *Y* is some outcome variable (e.g., age).
- *W* is a vector of weights that characterize how *Y* is related to *X*.
- σ expresses how uncertain we are about *Y* after seeing *X*.

Linear-Gaussian model

- Suppose we model the relationship between X, W, σ, and Y such that:

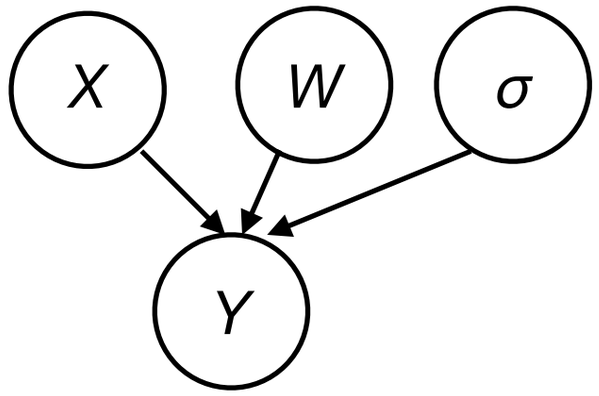




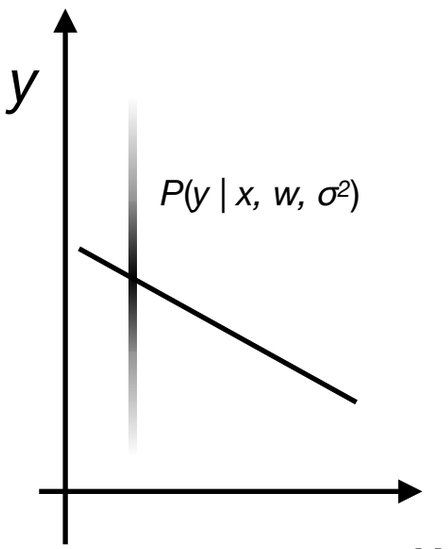
- Y is a normal/Gaussian random variable.
- The expected value of Y is $x^{\mathsf{T}}w$.
- The variance of Y is constant (σ^2) for all possible x.

Linear-Gaussian model

- Suppose we model the relationship between X, W, σ, 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 σ?

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

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

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

Conditional independence

Linear-Gaussian model

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

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

Conditional independence

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

Linear-Gaussian model

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

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

Conditional independence

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

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

Linear-Gaussian model

- MLE for **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 σ^2:

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

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

L_2 Regularization

Regularization

- The larger the coefficients (weights) **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_{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_{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_{MSE} becomes:

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

Hyperparameter tuning

Hyperparameter tuning

- The values we **optimize** when training a machine learning model — e.g., **w** and b for linear regression — are the **parameters** of the model.

- There are also values related to the training process itself — e.g., learning rate ε, batch size $\tilde{n}$, regularization strength α — which are the **hyperparameters** of training.

Hyperparameter tuning

- Both the parameters and hyperparameters can have a huge impact on model performance on test data.
- When estimating the performance of a trained model, it is important to tune both kinds of parameters in a principled way:
  - Training/validation/testing sets
  - Double cross-validation

Training/validation/testing sets

- In an application domain with a large dataset (e.g., 100K examples), it is common to partition it into three subsets:
  - Training (typically 70-80%): optimization of parameters
  - Validation (typically 5-10%): tuning of hyperparameters
  - Testing (typically 5-10%): evaluation of the final model

- For comparison with other researchers' methods, this partition should be fixed.

Training/validation/testing sets

- Hyperparameter tuning works as follows:

  1.For each hyperparameter configuration *h*:

    - Train the parameters on the **training** set using *h*.
    - Evaluate the model on the **validation** set.
    - If performance is better than what we got with the best *h* so far (h^*), then save *h* as h^*.

  2.Train a model with h^*, and evaluate its accuracy *A* on the **testing** set. (You can train either on training data, or on training+validation data).

Training/validation/testing sets

To what machine does the reported accuracy *A* correspond?

- Hyperparameter tuning works as follows:

  1. For each hyperparameter configuration *h*:
     - Train the parameters on the **training** set using *h*.
     - Evaluate the model on the **validation** set.
     - If performance is better than what we got with the best *h* so far (h^*), then save *h* as h^*.
  2. Train a model with h^*, and evaluate its accuracy *A* on the **testing** set. (You can train either on training data, or on training+validation data).

Training/validation/testing sets

To what machine does the reported accuracy *A* correspond?

- Hyperparameter tuning works as follows:

  1. For each hyperparameter configuration *h*:
     - Train the parameters on the **training** set using *h*.
     - Evaluate the model on the **validation** set.
     - If performance is better than what we got with the best *h* so far (h^*), then save *h* as h^*.
  2. Train a model with h^*, and evaluate its accuracy *A* on the **testing** set. (You can train either on training data, or on training+validation data).

Cross-validation

- When working with smaller datasets, cross-validation is commonly used so that we can use **all** data for training.

- Suppose we already know the best hyperparameters h^*.

- We partition the data into k folds of equal sizes.

- Over k iterations, we train on (k-1) folds and test on the remaining fold.

- We then compute the average accuracy over the k testing folds.

Cross-validation

- # *D*=dataset, *k*=# folds, *h*=hyperparameter configuration.
  CrossValidation (*D*, *k*, *h*):
      Partition *D* into *k* folds F_1, ..., F_k
  For *i* = 1, ..., *k*:
      test = F_i
      train = $D \setminus F_i$
      Train the model on *train* using *h*
      acc[*i*] = Evaluate NN on test
  A = Avg[*acc*]
  return *A*

Cross-validation

To what machine does the reported accuracy *A* correspond?

- # *D*=dataset, *k*=# folds, *h*=hyperparameter configuration.
  CrossValidation (*D*, *k*, *h*):
    Partition *D* into *k* folds F_1, ..., F_k
    For *i* = 1, ..., *k*:
      test = F_i
      train = $D \setminus F_i$
      Train the model on *train* using *h*
      acc[*i*] = Evaluate NN on test
  A = Avg[*acc*]
  return *A*

Cross-validation

To what machine does the reported accuracy *A* correspond?

- # *D*=dataset, *k*=# folds, *h*=hyperparameter configuration.
  CrossValidation (*D*, *k*, *h*):
    Partition *D* into *k* folds F_1, ..., F_k
  For *i* = 1, ..., *k*:
    test = F_i
    train = $D \setminus F_i$
    Train the model on *train* using *h*
    acc[*i*] = Evaluate NN on test
  A = Avg[*acc*]
  return *A*

None of them!

Training/validation/testing sets

- Cross-validation does not measure the accuracy of any *single* machine.
- Instead, cross-validation gives the *expected* accuracy of a classifier that is trained on (*k*-1)/*k* of the data.

Training/validation/testing sets

- Cross-validation does not measure the accuracy of any *single* machine.

- Instead, cross-validation gives the *expected* accuracy of a classifier that is trained on (*k*-1)/*k* of the data.

- However, we can train another model *M* using h^* on the entire dataset, and then report *A* as its accuracy.

- Since *M* is trained on more data than any of the cross-validation models, its *expected* accuracy should be >= *A*.

Cross-validation

- But how do we find the best hyperparameters h^* for each fold?
- The typical approach is to use **double cross-validation**, i.e.:
    - For each of the k "outer" folds, run cross-validation in an "inner" loop to determine the best hyperparameter configuration h^* for the kth fold.

Double cross-validation

- # *D*=dataset, *k*=# folds, *H*=set of hyperparameter configurations.
  DoubleCrossValidation (*D*, *k*, *H*):
    Partition *D* into *k* folds F_1, ..., F_k
    For *i* = 1, ..., *k*:
      test = F_i
      train = $D \setminus F_i$
      $A^* = -\infty$
      For *h* in *H*:
        A = CrossValidation(*train*, *k*, *h*)
        if $A > A^*$:
          $A^* = A$
          $h^* = h$
      Train the model on *train* using h^*
      accs[*i*] = Evaluate the model on *test*
  A = Avg[accs]
  return *A*

For your reference…

```
CrossValidation (D, k, h):
    Partition D into k folds F_1, ..., F_k
    For i = 1, ..., k:
        test = F_i
        train = D \ F_i
        Train the model on train using h
        acc[i] = Evaluate NN on test
    A = Avg[acc]
    return A
```

Double cross-validation

To what machine does the reported accuracy *A* correspond?

- # *D*=dataset, *k*=# folds, *H*=set of hyperparameter configurations.
  DoubleCrossValidation (*D*, *k*, *H*):
    Partition *D* into *k* folds F_1, ..., F_k
    For *i* = 1, ..., *k*:
      test = F_i
      train = $D \setminus F_i$
      $A^* = -\infty$
      For *h* in *H*:
        A = CrossValidation(*train*, *k*, *h*)
        if $A > A^*$:
          $A^* = A$
          $h^* = h$
      Train the model on *train* using h^*
      accs[*i*] = Evaluate the model on *test*
  A = Avg[accs]
  return *A*

For your reference…

```
CrossValidation (D, k, h):
    Partition D into k folds F_1, ..., F_k
    For i = 1, ..., k:
        test = F_i
        train = D \ F_i
        Train the model on train using h
        acc[i] = Evaluate NN on test
    A = Avg[acc]
    return A
```

Double cross-validation

To what machine does the reported accuracy *A* correspond?

- # *D*=dataset, *k*=# folds, *H*=set of hyperparameter configurations.
  DoubleCrossValidation (*D*, *k*, *H*):
    Partition *D* into *k* folds F_1, ..., F_k
    For *i* = 1, ..., *k*:
      test = F_i
      train = $D \setminus F_i$
      $A^* = -\infty$
      For *h* in *H*:
        A = CrossValidation(*train*, *k*, *h*)
        if $A > A^*$:
          $A^* = A$
          $h^* = h$
      Train the model on *train* using h^*
      accs[*i*] = Evaluate the model on *test*
  A = Avg[accs]
  return *A*

None of them!

For your reference…

CrossValidation (*D*, *k*, *h*):
  Partition *D* into *k* folds F_1, ..., F_k
  For *i* = 1, ..., *k*:
    test = F_i
    train = $D \setminus F_i$
    Train the model on *train* using *h*
    acc[*i*] = Evaluate NN on test
  A = Avg[*acc*]
  return *A*

Training/validation/testing sets

- In contrast to (single) cross-validation, it's not obvious how to train a model *M* with accuracy >= *A*.

- One strategy: return an **ensemble** model whose output is the average of the *k* models' predictions…but this is rarely done.

Subject independence

- In many machine learning settings, the data are not completely independent from each other — they are *linked* in some way.
- Example:
    - Predict multiple grades for each student based on their Canvas clickstream features (# logins, # forum posts, etc.).

Subject independence

- We could partition the data into folds in different ways:
  - We could randomize across all the data.
  - However, if grades are correlated within each student, then one (or more) training folds can leak information about the testing fold.

	Quiz 1	Quiz 2	Quiz 3
Student 1	45	48	42
Student 2	96	93	93
Student 3	86	86	87
Student 4	10	30	50

Subject independence

- We could partition the data into folds in different ways:
  - Alternatively, we can **stratify** across students, i.e., no student appears in more than 1 fold.
- With this partition, the cross-validation accuracy estimates the model's performance on a subject *not* used for training.

	Quiz 1	Quiz 2	Quiz 3
Student 1	45	48	42
Student 2	96	93	93
Student 3	86	86	87
Student 4	10	30	50

Optimization of ML models

Optimization of ML models

- Gradient descent is guaranteed to converge to a local minimum (eventually) if the learning rate is small enough relative to the steepness of *f*.
- A function $f : \mathbb{R}^m \to \mathbb{R}$ is Lipschitz-continuous if:

$$\exists L : \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^m : \|f(\mathbf{x}) - f(\mathbf{y})\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2$$

- *L* is essentially an upper bound on the absolute slope of *f*.

Optimization of ML models

- Gradient descent is guaranteed to converge to a local minimum (eventually) if the learning rate is small enough relative to the steepness of *f*.

- A function $f : \mathbb{R}^m \to \mathbb{R}$ is Lipschitz-continuous if:

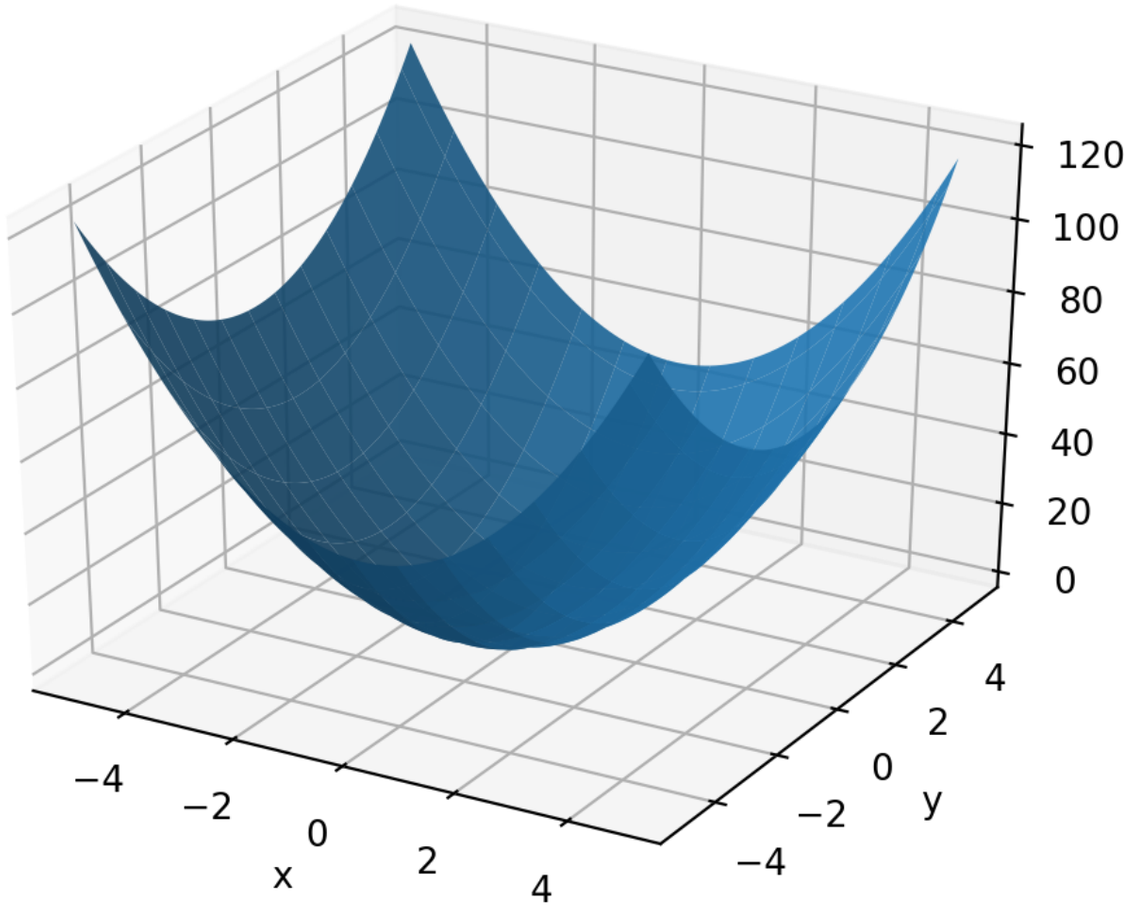
$$\exists L : \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^m : \|f(\mathbf{x}) - f(\mathbf{y})\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2$$

- *L* is essentially an upper bound on the absolute slope of *f*.

- For learning rate $\epsilon \leq \frac{1}{L}$, gradient descent will converge to a local minimum linearly, i.e., the error is *O*(1/*k*) in the iterations *k*.

Optimization of ML models

- With linear regression, the cost function f_{MSE} has a single local minimum w.r.t. the weights **w**:



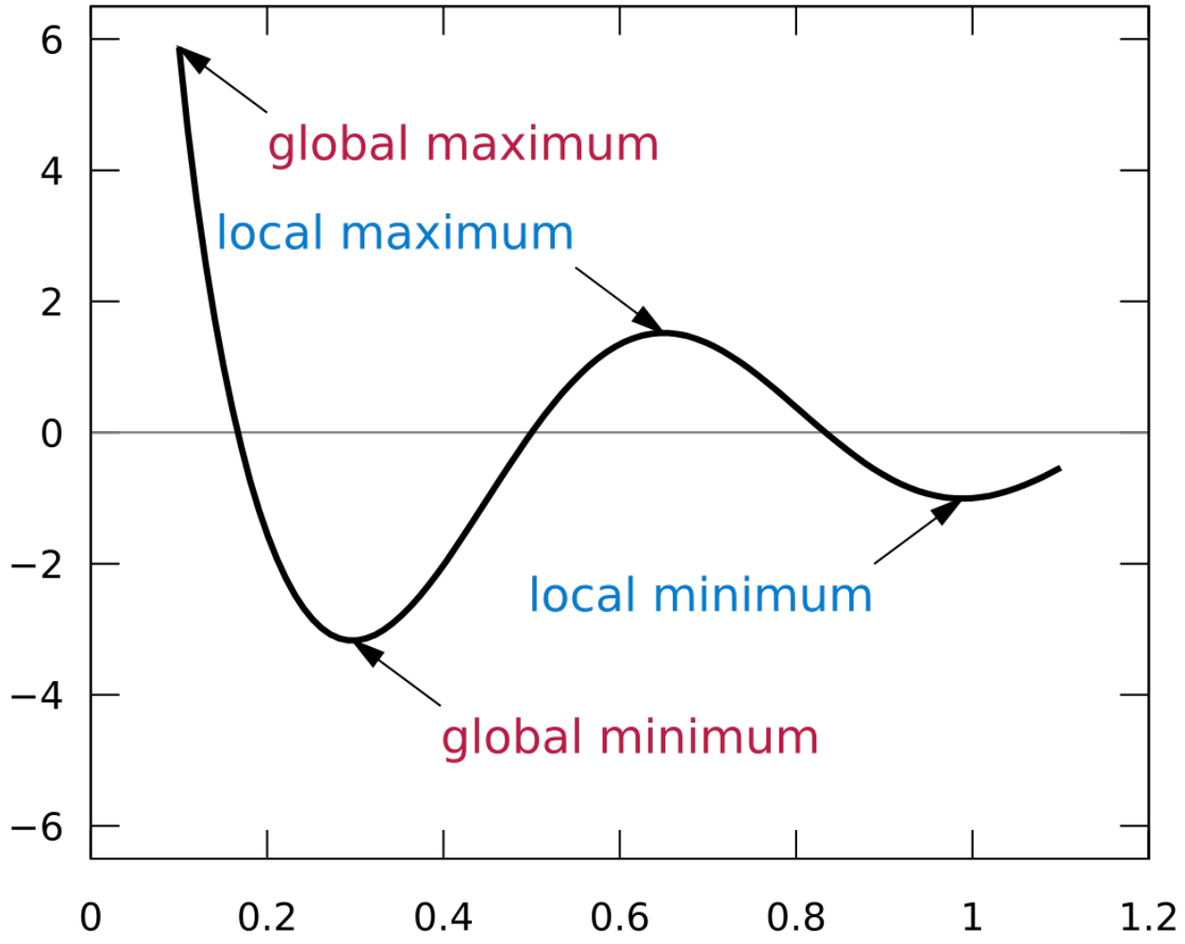
- As long as our learning rate is small enough, we will find **the optimal w**.

Optimization: what can go wrong?

- In general ML and DL models, optimization is usually not so simple, due to:

Optimization: what can go wrong?

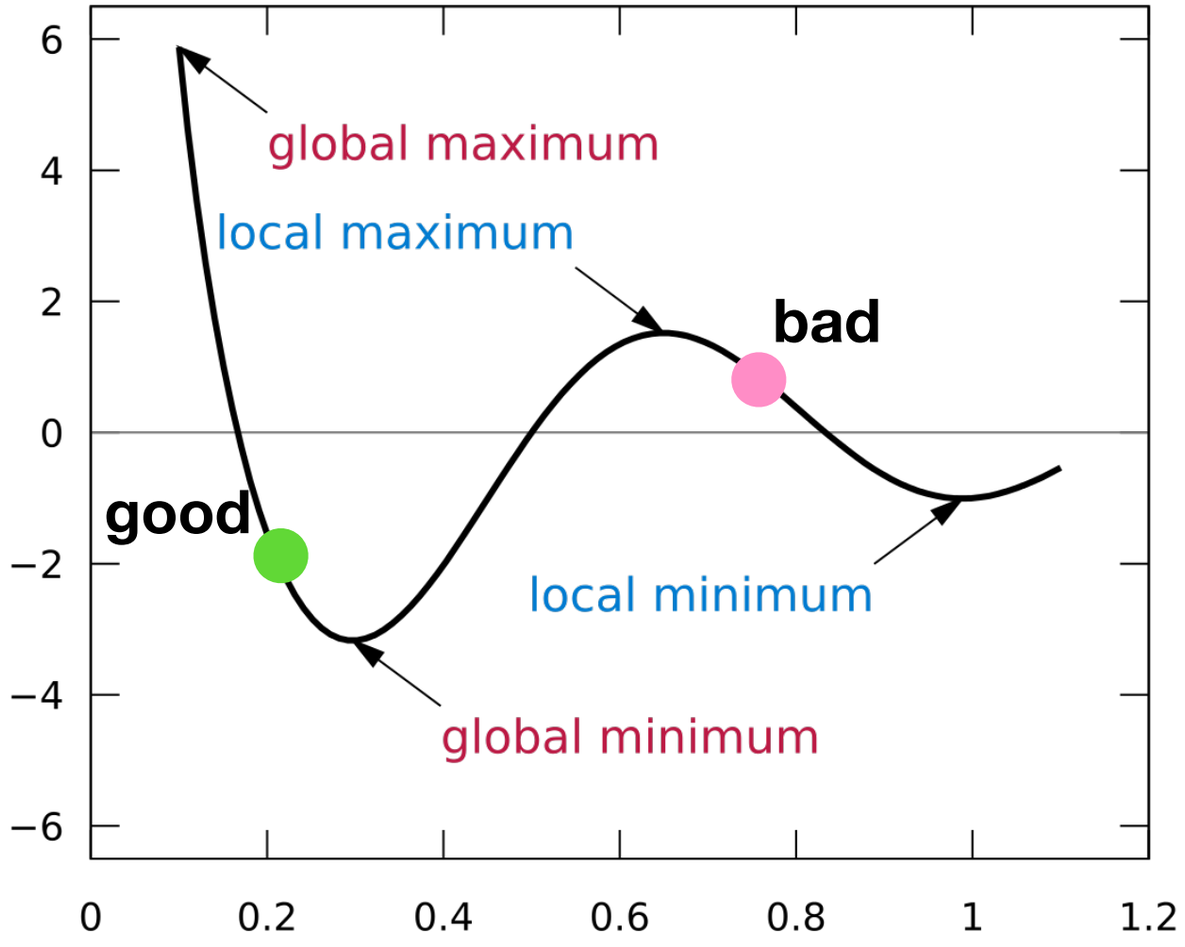
- In general ML and DL models, optimization is usually not so simple, due to:
    1. Presence of multiple local minima



Optimization: what can go wrong?

- In general ML and DL models, optimization is usually not so simple, due to:

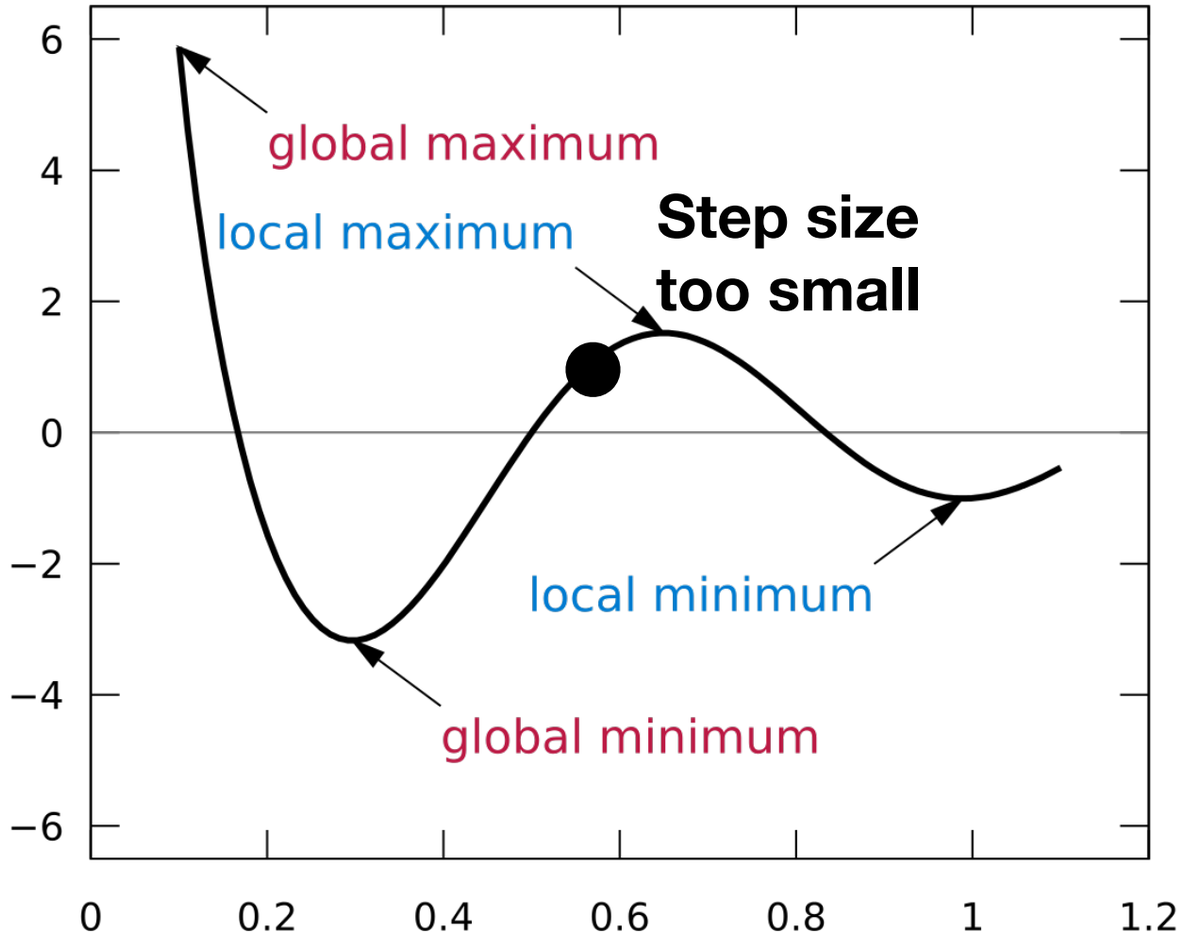
  2. Bad initialization of the weights **w**.



Optimization: what can go wrong?

- In general ML and DL models, optimization is usually not so simple, due to:

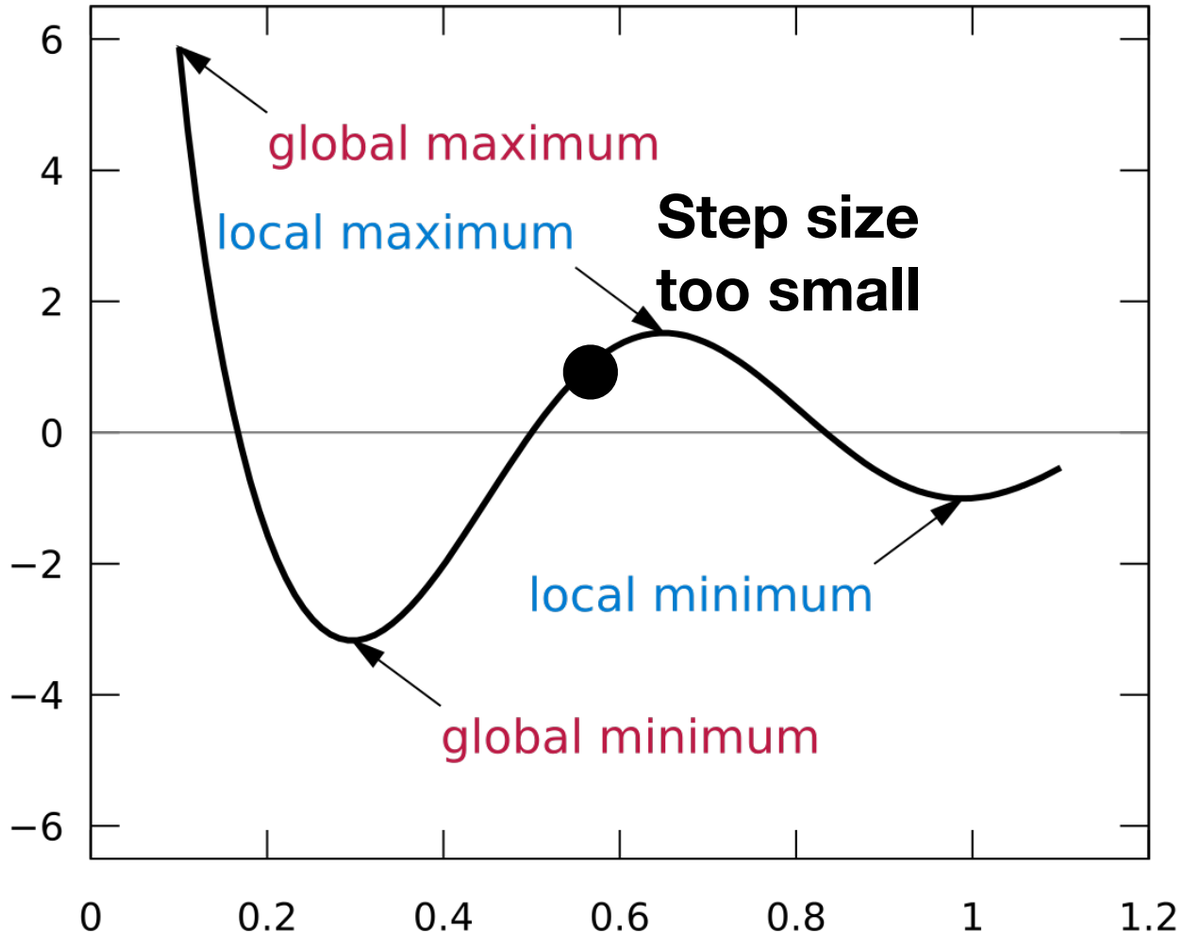
3. Learning rate is too small.



Optimization: what can go wrong?

- In general ML and DL models, optimization is usually not so simple, due to:

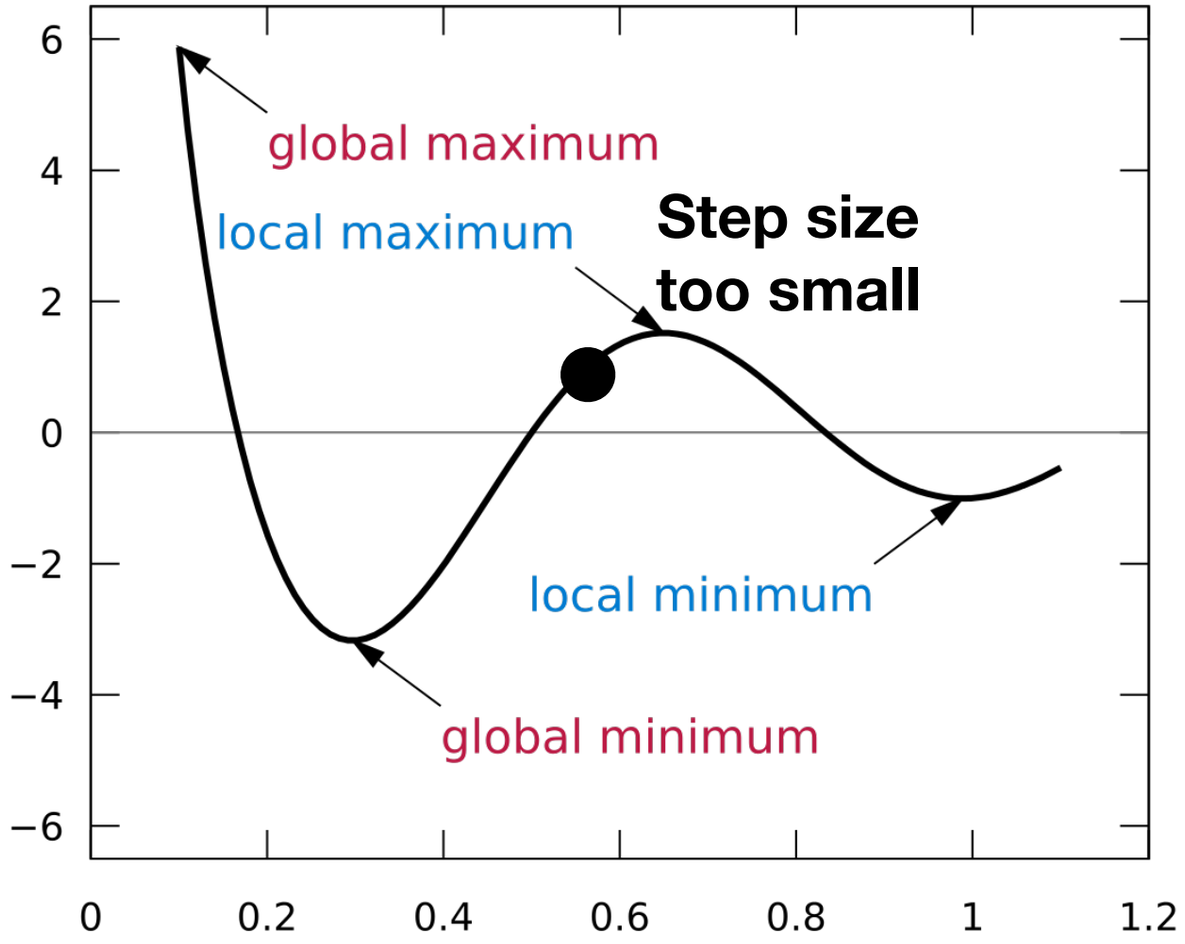
3. Learning rate is too small.



Optimization: what can go wrong?

- In general ML and DL models, optimization is usually not so simple, due to:

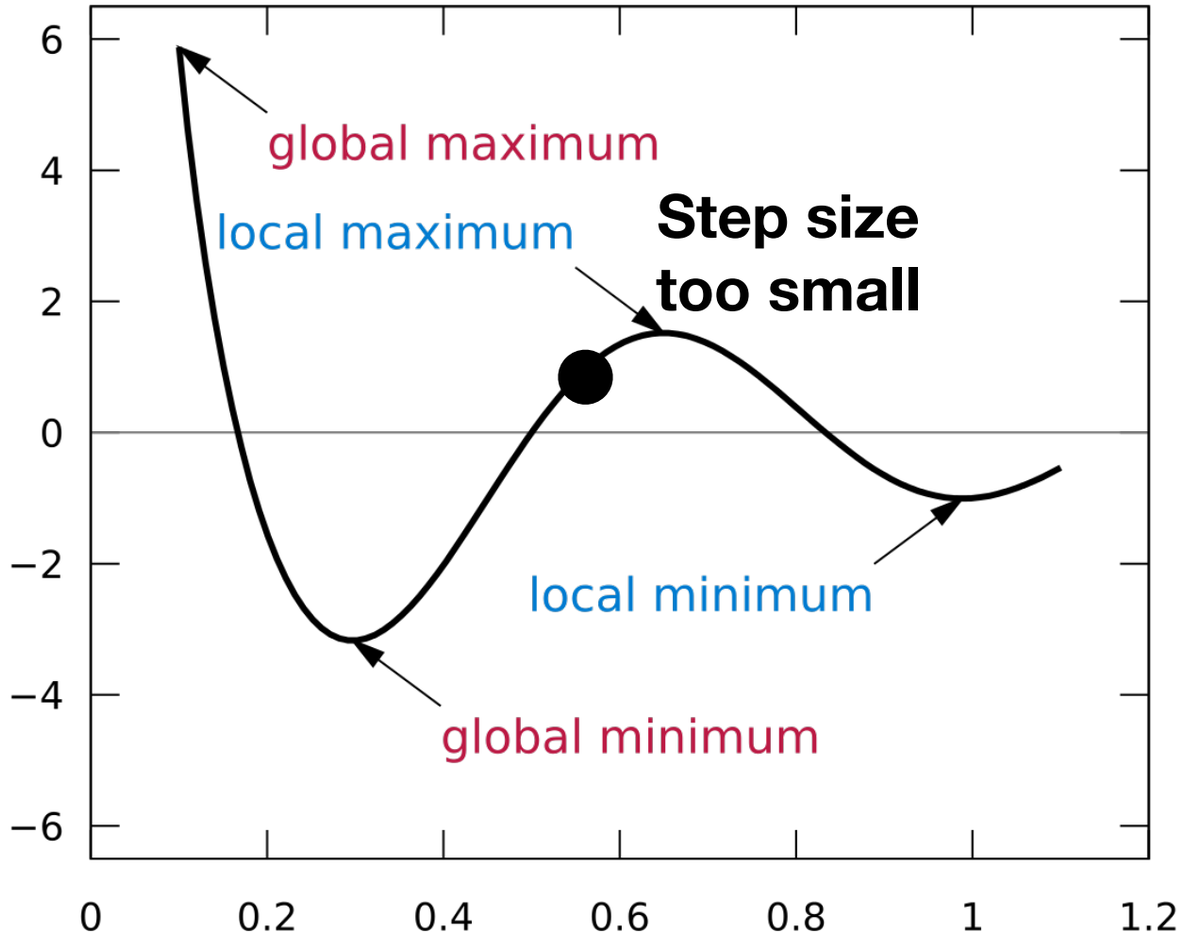
3. Learning rate is too small.



Optimization: what can go wrong?

- In general ML and DL models, optimization is usually not so simple, due to:

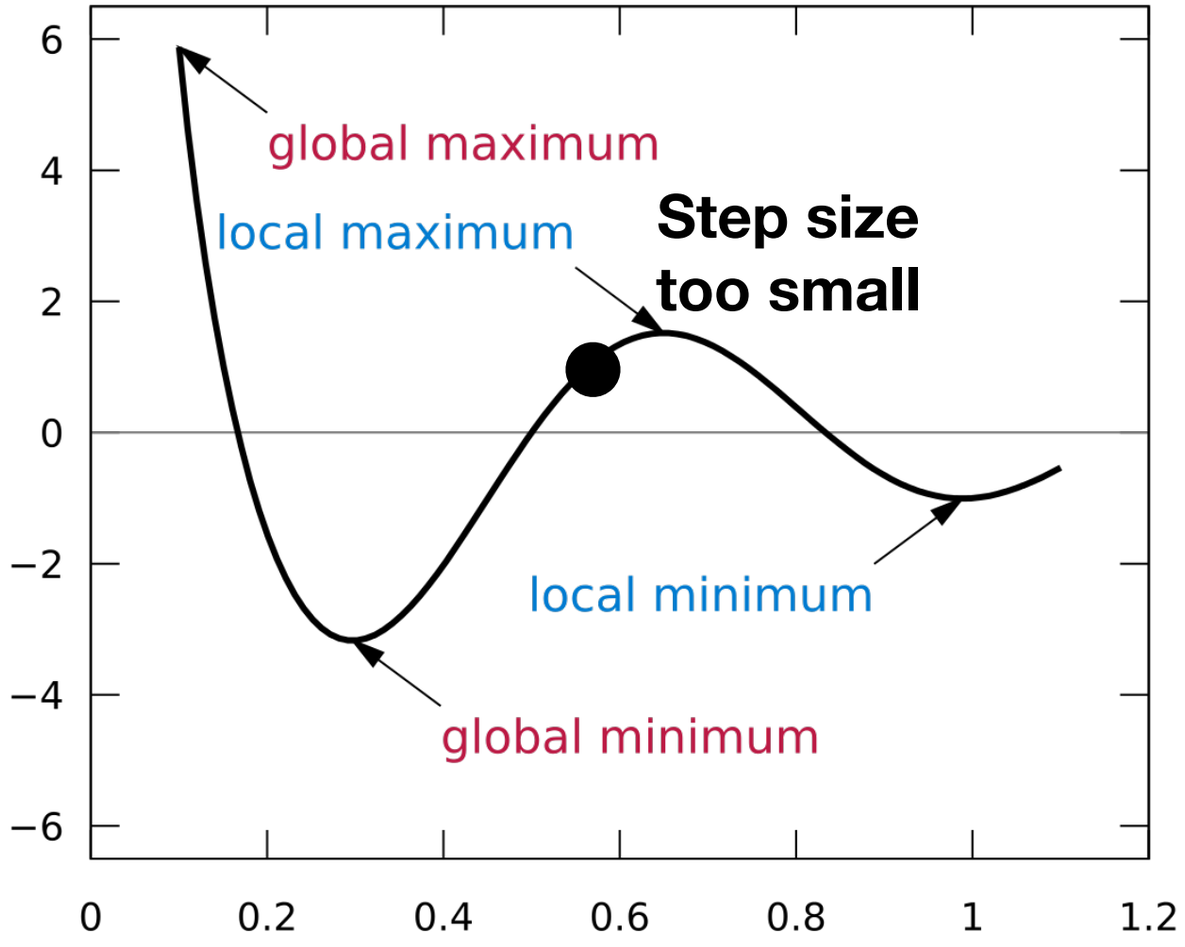
3. Learning rate is too small.



Optimization: what can go wrong?

- In general ML and DL models, optimization is usually not so simple, due to:

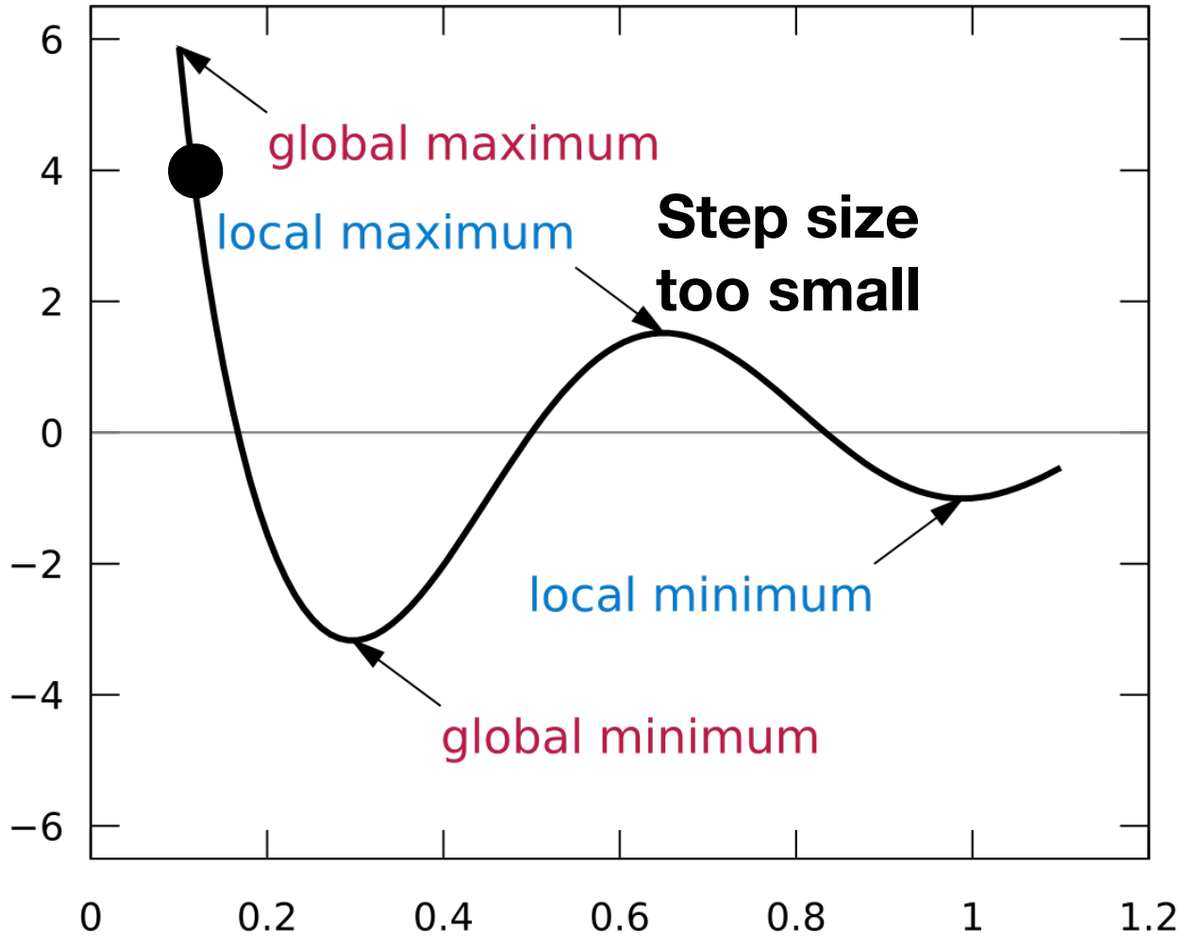
    4. Learning rate is too big.



Optimization: what can go wrong?

- In general ML and DL models, optimization is usually not so simple, due to:

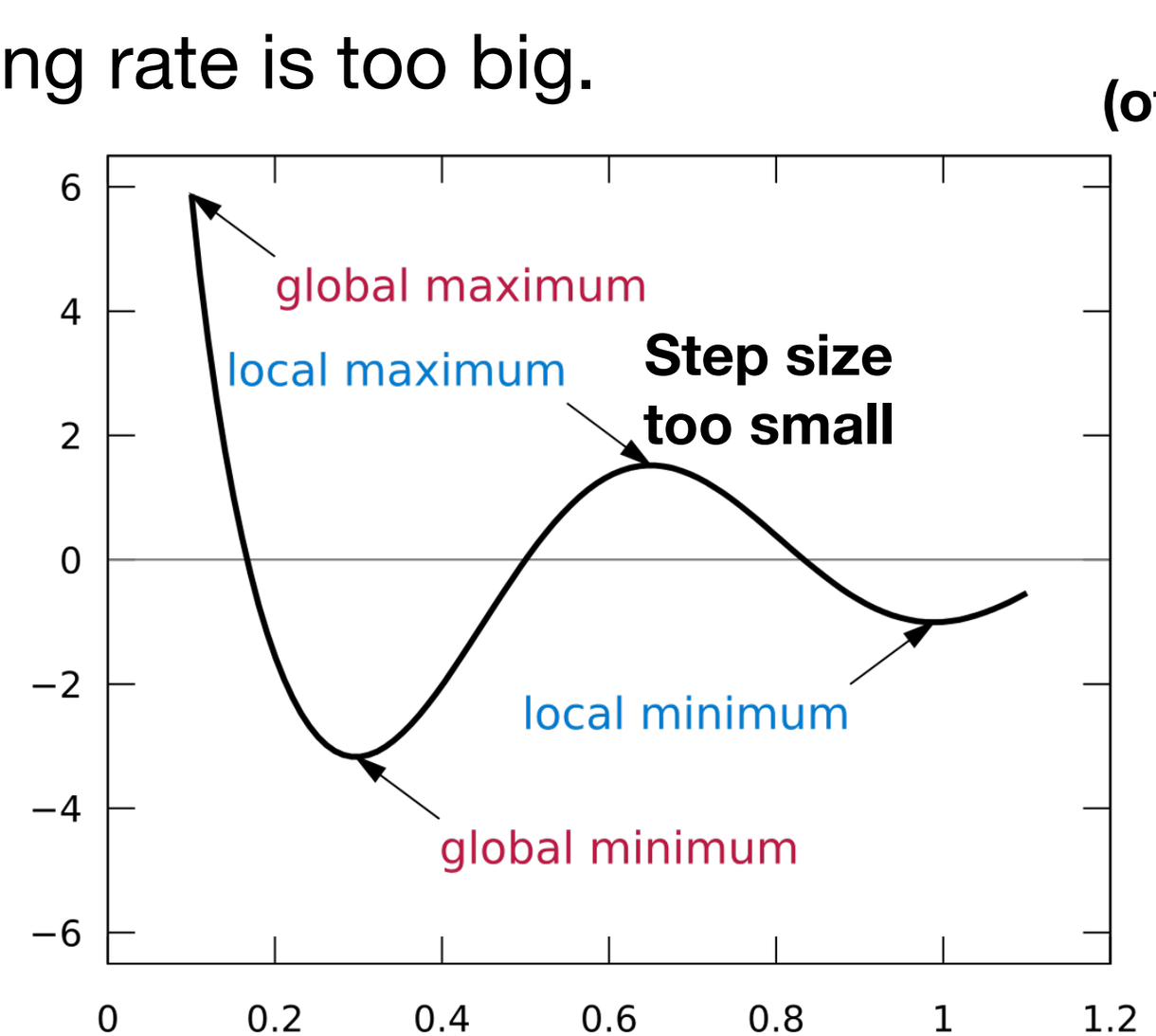
4. Learning rate is too big.



Optimization: what can go wrong?

- In general ML and DL models, optimization is usually not so simple, due to:

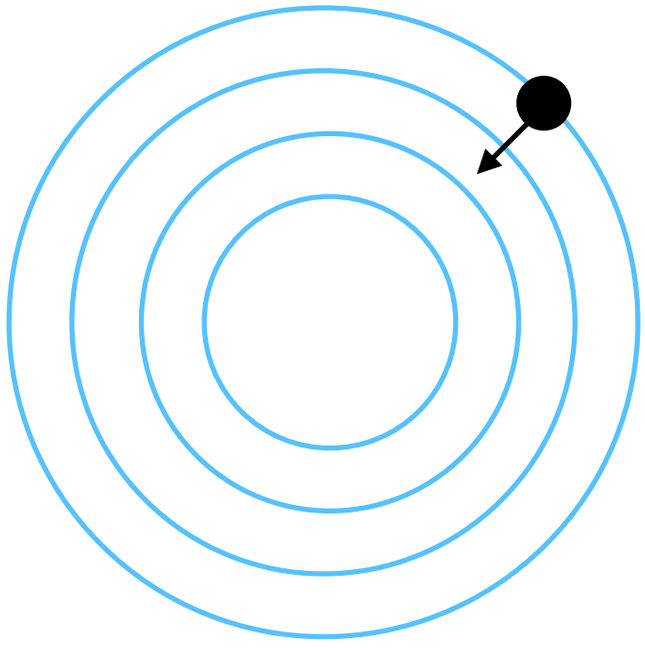
4. Learning rate is too big.



Optimization: what can go wrong?

- With multidimensional weight vectors, badly chosen learning rates can cause more subtle problems.

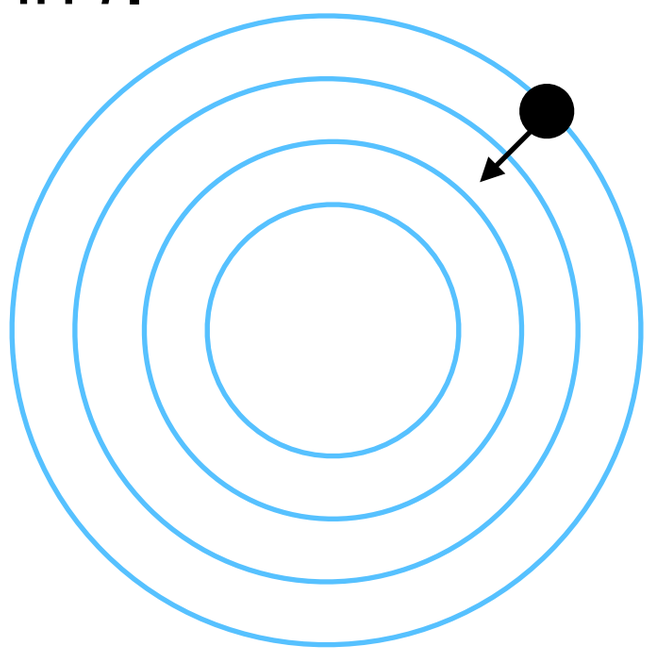
- Consider the cost *f* whose level sets are shown below:



Optimization: what can go wrong?

- With multidimensional weight vectors, badly chosen learning rates can cause more subtle problems.

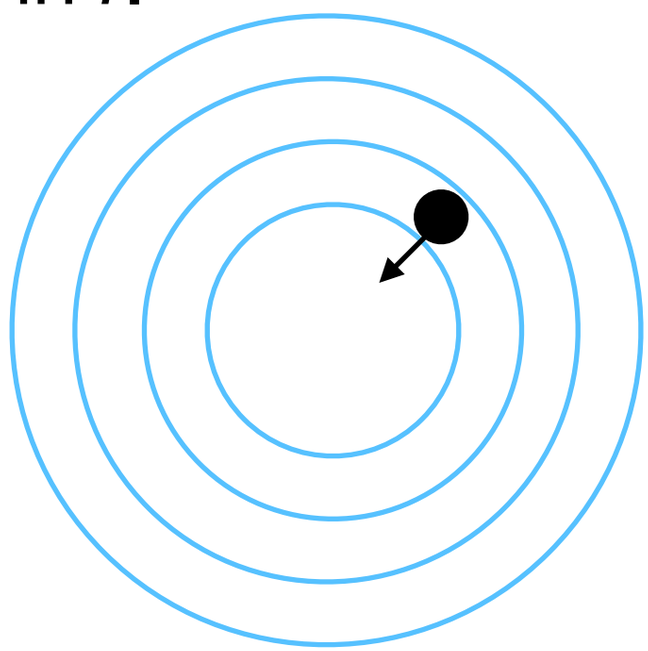
- Gradient descent guides the search along the direction of steepest decrease in *f*.



Optimization: what can go wrong?

- With multidimensional weight vectors, badly chosen learning rates can cause more subtle problems.

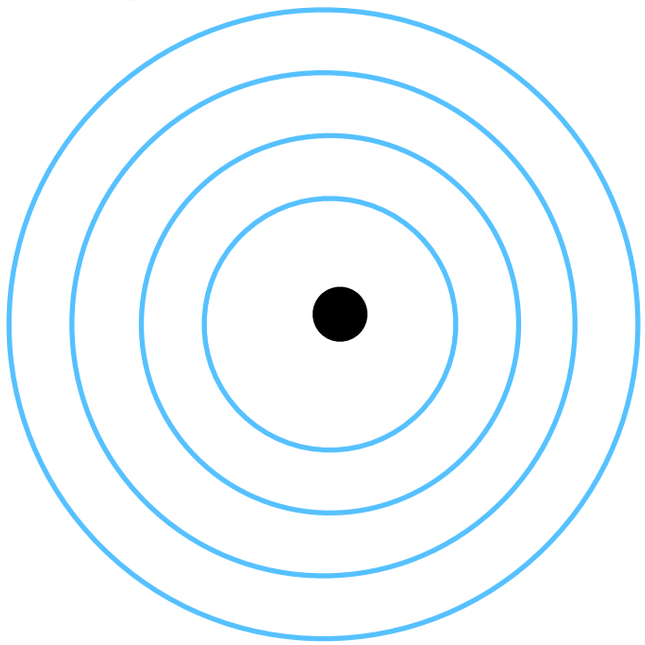
- Gradient descent guides the search along the direction of steepest decrease in *f*.



Optimization: what can go wrong?

- With multidimensional weight vectors, badly chosen learning rates can cause more subtle problems.

- Gradient descent guides the search along the direction of steepest decrease in *f*.



Optimization: what can go wrong?

- With multidimensional weight vectors, badly chosen learning rates can cause more subtle problems.

- But what if the level sets are ellipsoids instead of spheres?

Optimization: what can go wrong?

- With multidimensional weight vectors, badly chosen learning rates can cause more subtle problems.

- But what if the level sets are ellipsoids instead of spheres?

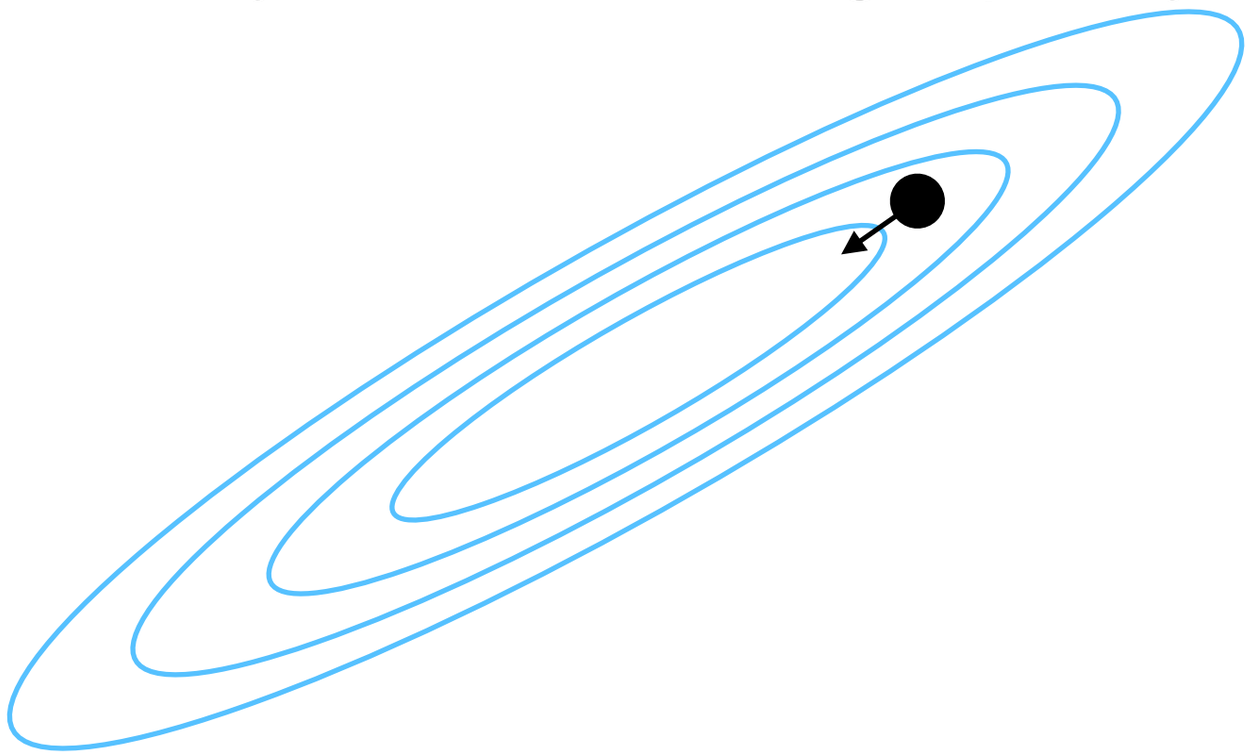
- If we are lucky, we still converge quickly.

Optimization: what can go wrong?

- With multidimensional weight vectors, badly chosen learning rates can cause more subtle problems.

- But what if the level sets are ellipsoids instead of spheres?

- If we are lucky, we still converge quickly.

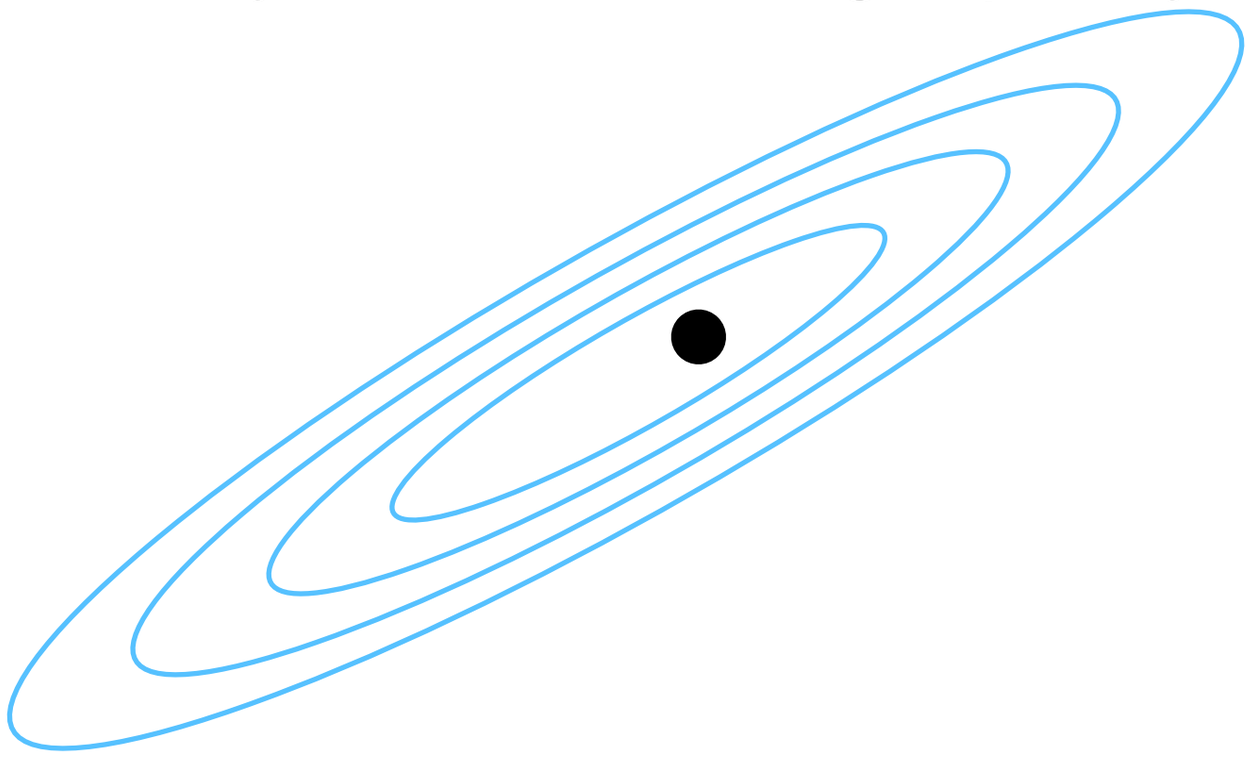


Optimization: what can go wrong?

- With multidimensional weight vectors, badly chosen learning rates can cause more subtle problems.

- But what if the level sets are ellipsoids instead of spheres?

- If we are lucky, we still converge quickly.

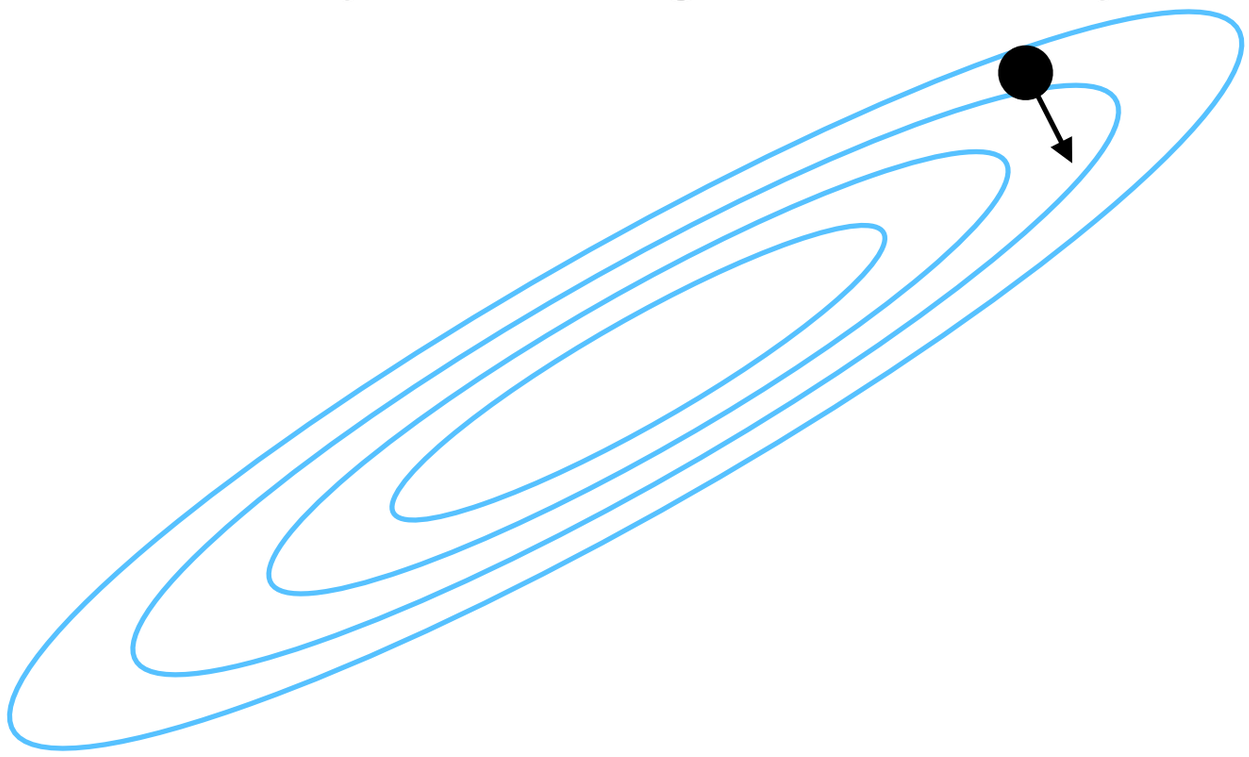


Optimization: what can go wrong?

- With multidimensional weight vectors, badly chosen learning rates can cause more subtle problems.

- But what if the level sets are ellipsoids instead of spheres?

- If we are unlucky, convergence is very slow.



Optimization: what can go wrong?

- With multidimensional weight vectors, badly chosen learning rates can cause more subtle problems.

- But what if the level sets are ellipsoids instead of spheres?

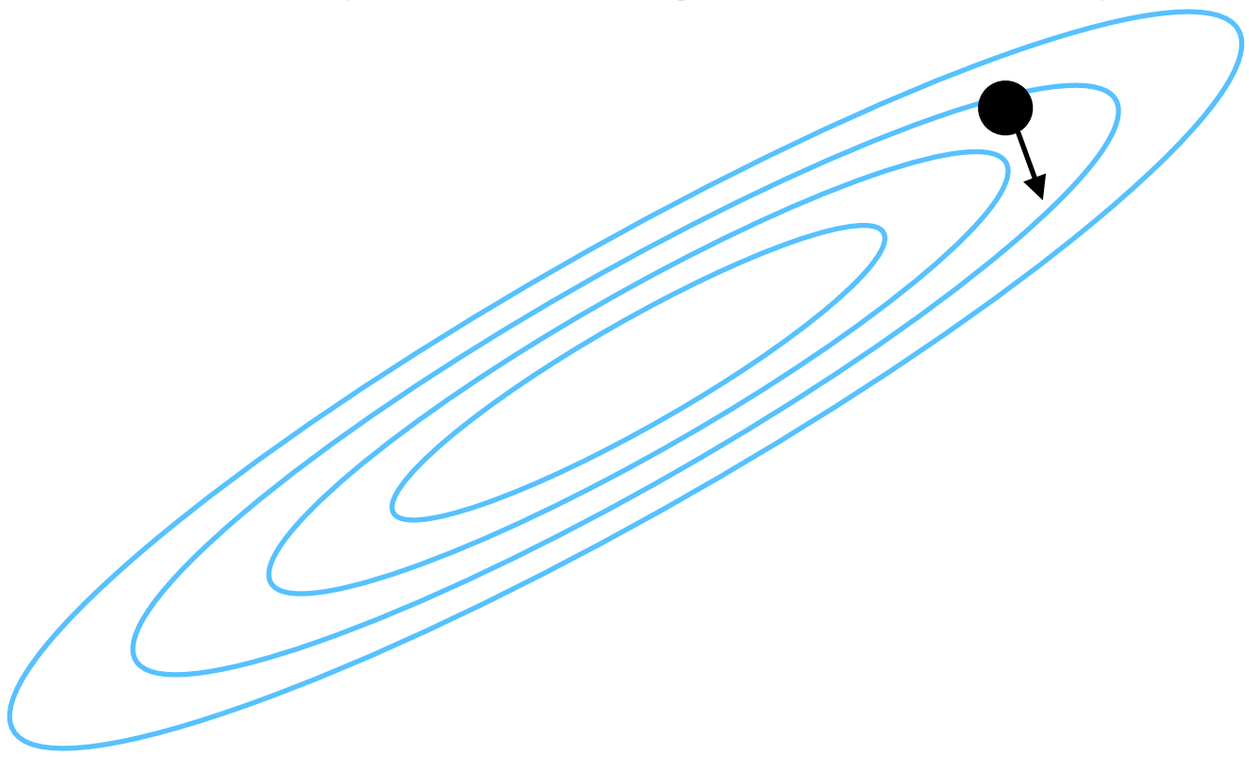
- If we are unlucky, convergence is very slow.

Optimization: what can go wrong?

- With multidimensional weight vectors, badly chosen learning rates can cause more subtle problems.

- But what if the level sets are ellipsoids instead of spheres?

- If we are unlucky, convergence is very slow.

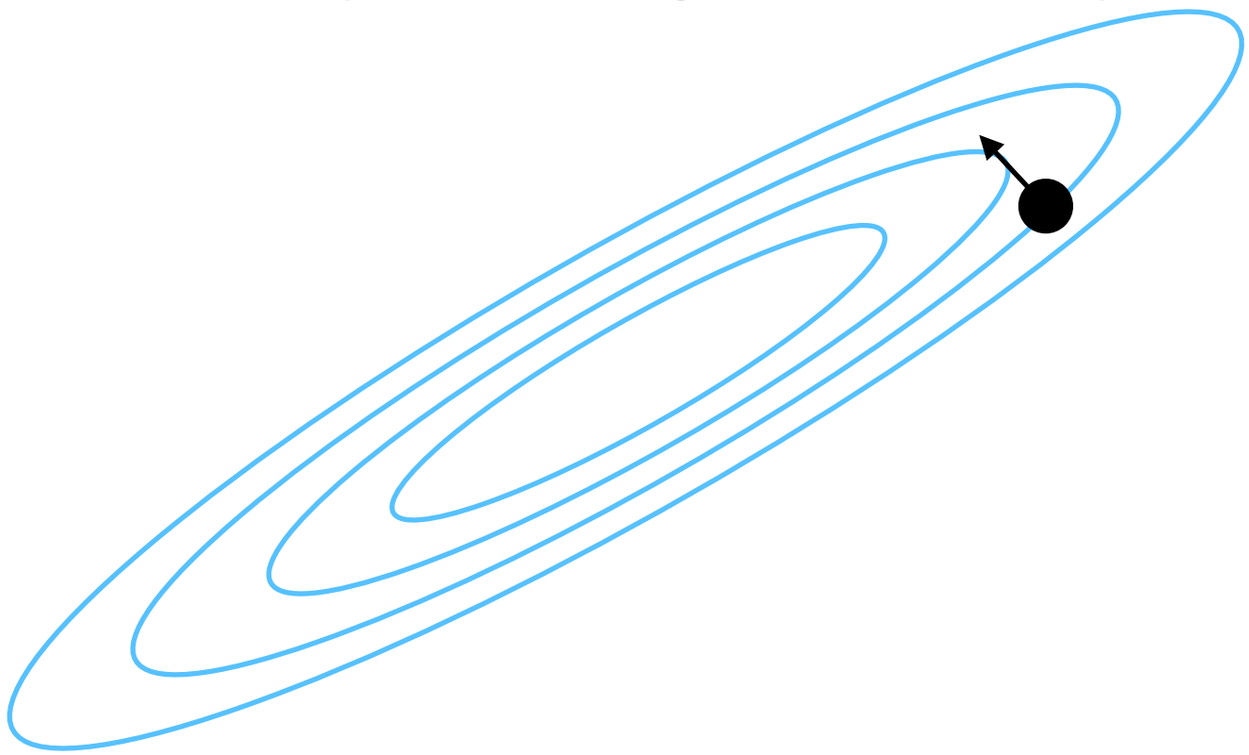


Optimization: what can go wrong?

- With multidimensional weight vectors, badly chosen learning rates can cause more subtle problems.

- But what if the level sets are ellipsoids instead of spheres?

- If we are unlucky, convergence is very slow.



Optimization: what can go wrong?

- With multidimensional weight vectors, badly chosen learning rates can cause more subtle problems.

- But what if the level sets are ellipsoids instead of spheres?

- If we are unlucky, convergence is very slow.