Machine Learning
A Non-Technical Introduction

University of Zurich[UZH]

Dr. Markus Meierer - markus.meierer@business.uzh.ch | Dr. Margot Löwenberg - margot.loewenberg@business.uzh.ch

The warrior is always trying to improve.

Model Tuning (1/2)

Dr. Markus Meierer - markus.meierer@business.uzh.ch | Dr. Margot Löwenberg - margot.loewenberg@business.uzh.ch

This session has 3 learning goals

After this lecture you should be able to:

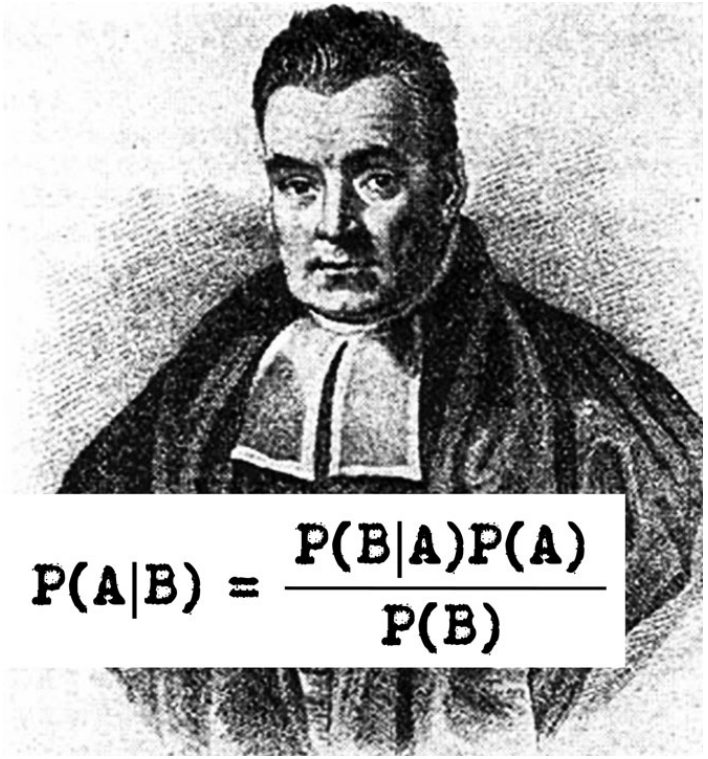
1. Understand Naïve Bayes.
2. Understand the advantages of hyperparameter optimization.
3. Understand the advantages and structure of ensemble models.

Your second ML algorithm: (Gaussian) Naïve Bayes

Dr. Markus Meierer - markus.meierer@business.uzh.ch | Dr. Margot Löwenberg - margot.loewenberg@business.uzh.ch

Why “naïve”? Why “Bayes”?

- The algorithm is called “naïve”, because all predictors are assumed to be uncorrelated.
- It is based on Bayes’ Theorem (based on work from Thomas Bayes, 1701-1761, but only published posthumously).



$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

For metric features a variation is used, i.e. **Gaussian Naïve Bayes**. This variation relies on the probability density function of the normal distribution, thus the reference to “Gaussian” in its name.

Steps of estimating (Gaussian) Naïve Bayes

1. Define P(A), i.e. class prior probability.
2. Define P(B|A), i.e. the likelihood.
3. Define P(B), i.e. unconditional probability.
4. Predict the class of a new point.

Step 1: Define P(A), i.e. class prior probability
Calculate the probability

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
0	7.7	3.5
5/10 = 0.5	**P (Y=1)**	
5/10 = 0.5	**P (Y=0)**	

We **calculate the probability** for a case being classified as class 0 or class 1. Here, the probability is 5/10 = 0.5, thus 50%.

Step 2: Define P(B|A), i.e. the likelihood
Separate the data

"**Conditional probability of B given A**": measure of the probability of an event given that another event has occurred.

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
0	7.7	3.5
5/10 = 0.5	**P (Y=1)**	
5/10 = 0.5	**P (Y=0)**	

Y = 0 →

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
0	7.7	3.5

Y = 1 →

Y	X1	X2
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2

Separate the data into Y=0 and Y=1 cases.

Step 2: Define P(B|A), i.e. the likelihood
Calculate the mean

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
0	7.7	3.5
5/10 = 0.5	**P (Y=1)**	
5/10 = 0.5	**P (Y=0)**	

Y = 0 →

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
0	7.7	3.5
Mean	3.22	1.64

Y = 1 →

Y	X1	X2
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
Mean	6.28	4.16

Calculate the mean for all Y=0 and Y=1.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Step 2: Define P(B|A), i.e. the likelihood
Calculate the standard deviation

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
0	7.7	3.5
5/10 = 0.5	**P (Y=1)**	
5/10 = 0.5	**P (Y=0)**	

Y = 0 →

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
0	7.7	3.5
Mean	3.22	1.64
SD	2.55	1.06

Y = 1 →

Y	X1	X2
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
Mean	6.28	4.16
SD	2.41	1.32

Calculate the standard deviation for all Y=0 and Y=1.

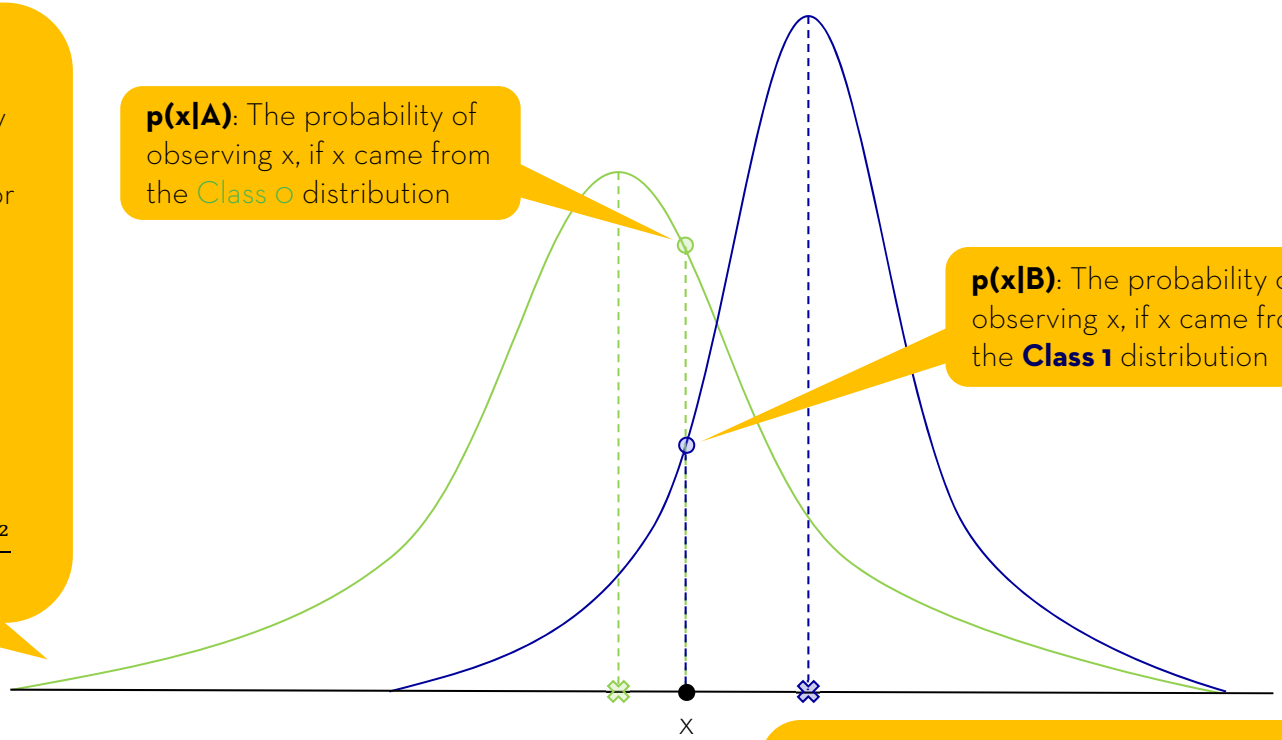
$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1}}$$

Step 2: Define P(B|A), i.e. the likelihood
How does the Gaussian Naïve Bayes classifier work?

Calculate probability under assumption of normal distribution by using the value of a variable x, the mean (for class 0 and 1), and the standard deviation (for class 0 and 1), i.e. insert those values to the probability density function of the normal distribution.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

p(x|A): The probability of observing x, if x came from the Class 0 distribution



p(x|B): The probability of observing x, if x came from the **Class 1** distribution

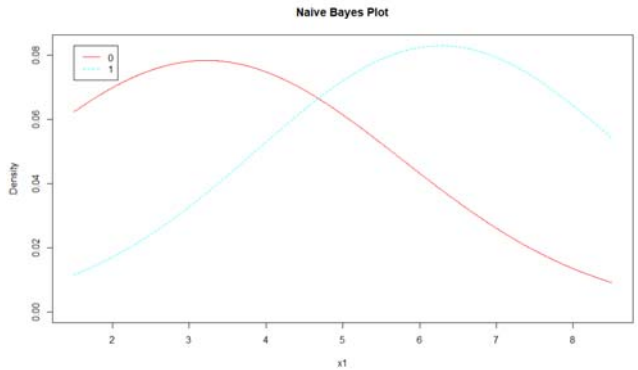
In essence, the approach takes each data point, and assigns it to whichever class it is nearest to. However, rather than calculating that nearness by using Euclidean distance from the class-means, the **GNB takes into account not only the distance from the mean but also how this compares to the class variance**

https://www.researchgate.net/publication/255695722_Smoothness_without_Smoothing_Why_Gaussian_Naive_Bayes_Is_Not_Naive_for_Multi-Subject_Searchlight_Studies/figures?lo=1

Step 2: Define P(B|A), i.e. the likelihood
How does the Gaussian Naïve Bayes classifier work?

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
0	7.7	3.5
Mean	3.22	1.64
SD	2.55	1.06

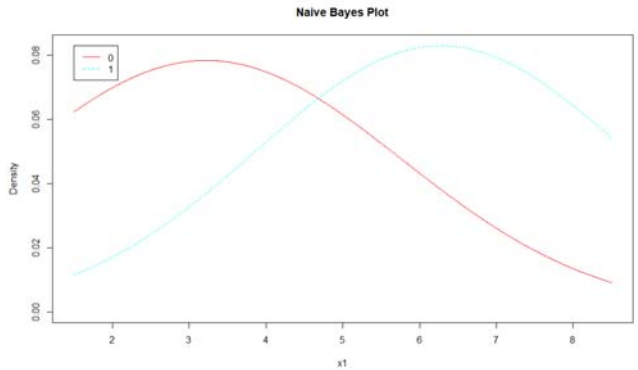
Y	X1	X2
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
Mean	6.28	4.16
SD	2.41	1.32



Step 2: Define P(B|A), i.e. the likelihood
How does the Gaussian Naïve Bayes classifier work?

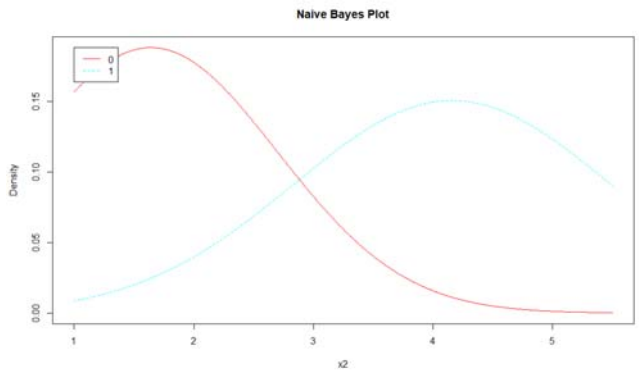
Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
0	7.7	3.5
Mean	3.22	1.64
SD	2.55	1.06

Y	X1	X2
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
Mean	6.28	4.16
SD	2.41	1.32

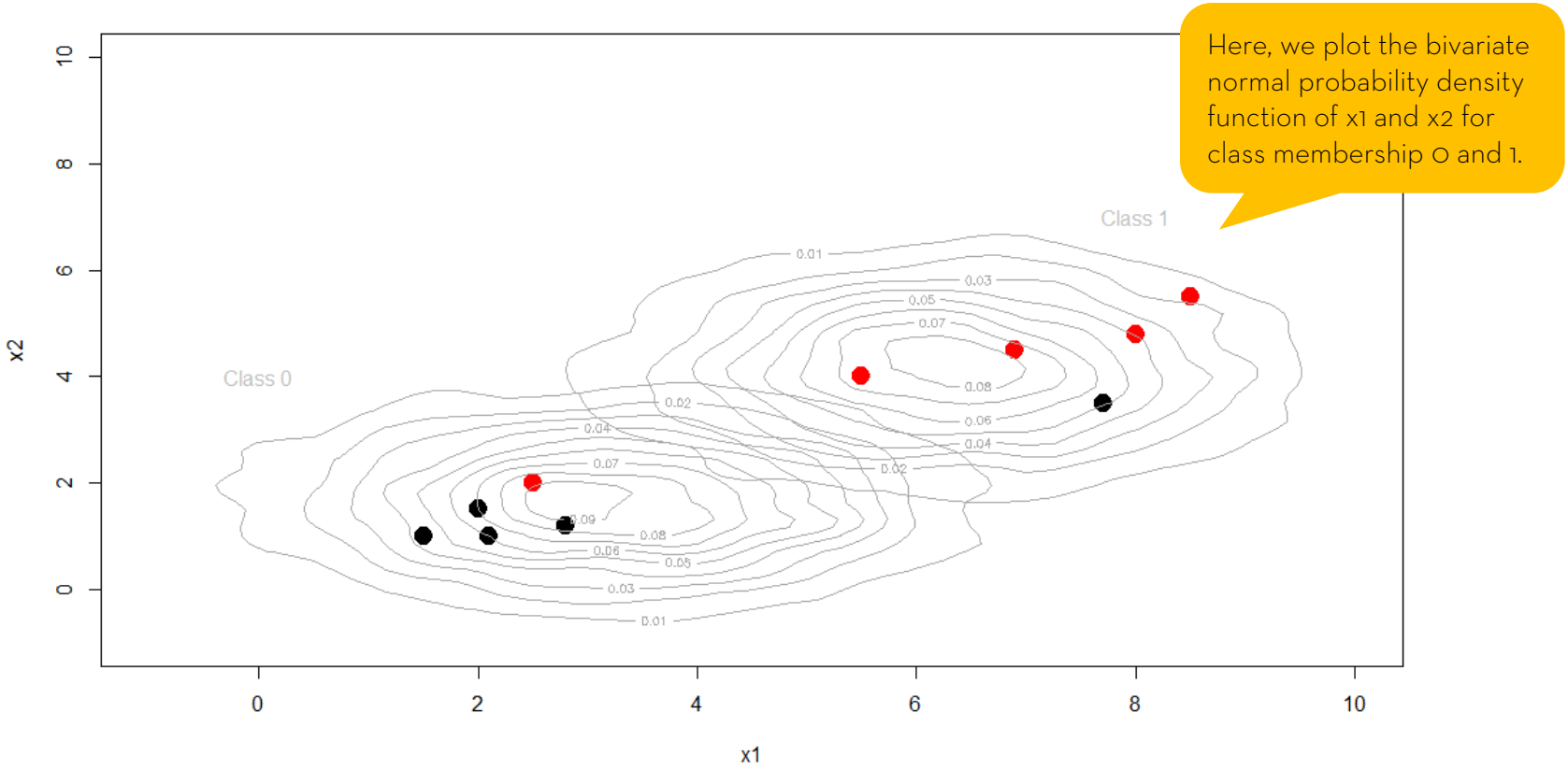


Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
0	7.7	3.5
Mean	3.22	1.64
SD	2.55	1.06

Y	X1	X2
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
Mean	6.28	4.16
SD	2.41	1.32



Step 2: Define P(B|A), i.e. the likelihood
How does the Gaussian Naïve Bayes classifier work?



Step 3: Define P(B), i.e. unconditional probability

- Not necessary for solving the classification problem, because it is effectively only a **normalizing constant**.
- It is necessary for solving the ranking problem where you need to ensure comparability of probabilities between observations, but again it does not affect which class is most likely for each observation.
- Thus, we only compute the **relative likelihood**, i.e.:

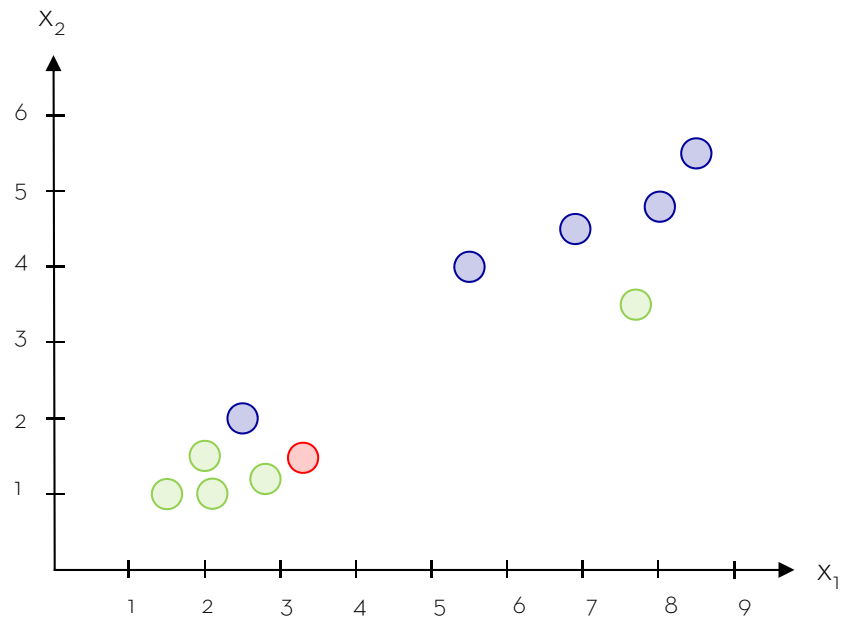
rel(A|B) = P(B|A) * P(A)

https://stats.stackexchange.com/questions/129666/why-normalizing-factor-is-required-in-bayes-theorem

Step 4: Predict the class of a new point

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
0	7.7	3.5
	3.19	1.50

Naïve Bayes is now used to **predict in which class the new point falls**.



Step 4: Predict the class of a new point
Calculate P(B|A) (1/4)

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
0	7.7	3.5
	3.19	1.50

Y = 0 →

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
0	7.7	3.5
Mean	3.22	1.64
SD	2.55	1.06

Calculate probability under assumption of normal distribution by using the value of x1, the mean, and the standard deviation, i.e. insert those values to the probability density function of the normal distribution.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Step 4: Predict the class of a new point
Calculate P(B|A) (2/4)

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
0	7.7	3.5
	3.19	1.50

Y = 0 →

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
0	7.7	3.5
Mean	3.22	1.64
SD	2.55	1.06

$$P(x_1 = 3.19 \mid y = 0) = \frac{1}{\sqrt{2\pi} * 2.55} e^{-\frac{(3.19-3.22)^2}{2*2.55^2}} = 0.157$$

$$P(x_2 = 1.50 \mid y = 0) = \frac{1}{\sqrt{2\pi} * 1.06} e^{-\frac{(1.50-1.64)^2}{2*1.06^2}} = 0.373$$

Calculate probability under assumption of normal distribution by using the value of x1, the mean, and the standard deviation, i.e. insert those values to the probability density function of the normal distribution.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Step 4: Predict the class of a new point
Calculate P(B|A) (3/4)

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
0	7.7	3.5
	3.19	1.50

Y = 0 →

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
0	7.7	3.5
Mean	3.22	1.64
SD	2.55	1.06

Calculate probability under assumption of normal distribution by using the value of x1, the mean, and the standard deviation, i.e. insert those values to the probability density function of the normal distribution.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(x_1 = 3.19 \mid y = 0) = \frac{1}{\sqrt{2\pi} * 2.55} e^{-\frac{(3.19-3.22)^2}{2*2.55^2}} = 0.157$$

$$P(x_2 = 1.50 \mid y = 0) = \frac{1}{\sqrt{2\pi} * 1.06} e^{-\frac{(1.50-1.64)^2}{2*1.06^2}} = 0.373$$

$$P(x_1 = 3.19, x_2 = 1.50 \mid y = 0)$$

$$= P(x_1 = 3.19 \mid y = 0) * P(x_2 = 1.50 \mid y = 0)$$
$$= 0.157 * 0.373$$
$$= 0.059$$

P(B|A)

Step 4: Predict the class of a new point
Calculate P(B|A) (4/4)

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
0	7.7	3.5
	3.19	1.50

Y = 1

Y	X1	X2
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
Mean	6.28	4.16
SD	2.41	1.32

$$P(x_1 = 3.19 \mid y = 1) = \frac{1}{\sqrt{2\pi} * 2.41} e^{-\frac{(3.19-6.28)^2}{2*2.41^2}} = 0.073$$

We do the same for Y=1.

$$P(x_2 = 1.50 \mid y = 1) = \frac{1}{\sqrt{2\pi} * 1.32} e^{-\frac{(1.50-4.16)^2}{2*1.32^2}} = 0.040$$

$$P(x_1 = 3.19, x_2 = 1.50 \mid y = 1)$$

$$= P(x_1 = 3.19 \mid y = 1) * P(x_2 = 1.50 \mid y = 1)$$
$$= 0.073 * 0.040$$
$$= 0.003$$

P(B|A)

Step 4: Predict the class of a new point
Compute posterior probabilities by class (1/2)

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
0	7.7	3.5
	3.19	1.50

Y = 0

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
0	7.7	3.5
Mean	3.22	1.64
SD	2.55	1.06

Y = 1

Y	X1	X2
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
Mean	6.28	4.16
SD	2.41	1.32

rel(A|B)

$$rel(y=0 \mid x_1, x_2)$$
$$= P(x_1 = 3.19, x_2 = 1.50 \mid y=0) * P(y=0)$$
$$= 0.059 * 0.5$$
$$= 0.0295$$

P(B|A) * P(A)

(see slide 5 for calculation of P(A))

Step 4: Predict the class of a new point
Compute posterior probabilities by class (2/2)

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
0	7.7	3.5
	3.19	1.50

Y = 0 →

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
0	7.7	3.5
Mean	3.22	1.64
SD	2.55	1.06

rel(A|B)

$$rel(y=0 \mid x_1, x_2)$$
$$= P(x_1 = 3.19, x_2 = 1.50 \mid y=0) * P(y=0)$$
$$= 0.059 * 0.5$$
$$= 0.0295$$

P(B|A) * P(A)
(see slide 5 for calculation of P(A))

Y = 1 →

Y	X1	X2
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
Mean	6.28	4.16
SD	2.41	1.32

rel(A|B)

$$rel(y=1 \mid x_1, x_2)$$
$$P(x_1 = 3.19, x_2 = 1.50 \mid y=1) * P(y=1)$$
$$= 0.003 * 0.5$$
$$= 0.001$$

P(B|A) * P(A)

Step 4: Predict the class of a new point
Find the largest posterior probability by class

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
0	7.7	3.5
	3.19	1.50

Y = 0 →

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
0	7.7	3.5
Mean	3.22	1.64
SD	2.55	1.06

rel(A|B)

$$rel(y=0 \mid x_1, x_2)$$
$$= P(x_1 = 3.19, x_2 = 1.50 \mid y=0) * P(y=0)$$
$$= 0.059 * 0.5$$
$$= 0.0295$$

P(B|A) * P(A)
(see slide 5 for calculation of P(A))

Y = 1 →

Y	X1	X2
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
Mean	6.28	4.16
SD	2.41	1.32

rel(A|B)

$$rel(y=1 \mid x_1, x_2)$$
$$P(x_1 = 3.19, x_2 = 1.50 \mid y=1) * P(y=1)$$
$$= 0.003 * 0.5$$
$$= 0.001$$

P(B|A) * P(A)

The class with the highest probability is considered as the most likely class (also known as Maximum A Posteriori (MAP)).

→ **The new point is predicted to be class 0.**

Step 4: Predict the class of a new point
Compute class probabilities

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
0	7.7	3.5
	3.19	1.50

Y = 0 →

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
0	7.7	3.5
Mean	3.22	1.64
SD	2.55	1.06

$$rel(y=0 \mid x_1, x_2)$$
$$= P(x_1 = 3.19, x_2 = 1.50 \mid y=0) * P(y=0)$$
$$= 0.059 * 0.5$$
$$= 0.0295$$

$$rel(y=0 \mid x_1, x_2)_{\text{standardized}}$$
$$= 0.0295 / (0.0295 + 0.001)$$
$$= 0.967$$

Normalize the values by the sum of the posterior probabilities of both classes.

Y = 1 →

Y	X1	X2
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
Mean	6.28	4.16
SD	2.41	1.32

$$rel(y=1 \mid x_1, x_2)$$
$$P(x_1 = 3.19, x_2 = 1.50 \mid y=1) * P(y=1)$$
$$= 0.003 * 0.5$$
$$= 0.001$$

Step 4: Predict the class of a new point
Compute class probabilities

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
0	7.7	3.5
	3.19	1.50

Y = 0 →

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
0	2.1	1
0	7.7	3.5
Mean	3.22	1.64
SD	2.55	1.06

Y = 1 →

Y	X1	X2
1	5.5	4
1	8	4.8
1	6.9	4.5
1	8.5	5.5
1	2.5	2
Mean	6.28	4.16
SD	2.41	1.32

$$rel(y=0 \mid x_1, x_2)$$
$$= P(x_1 = 3.19, x_2 = 1.50 \mid y=0) * P(y=0)$$
$$= 0.059 * 0.5$$
$$= 0.0295$$

$$rel(y=0 \mid x_1, x_2)_{\text{standardized}}$$
$$= 0.0295 / (0.0295 + 0.001)$$
$$= 0.967$$

$$rel(y=1 \mid x_1, x_2)$$
$$P(x_1 = 3.19, x_2 = 1.50 \mid y = \quad (y=1)$$
$$= 0.003 * 0.5$$
$$= 0.001$$

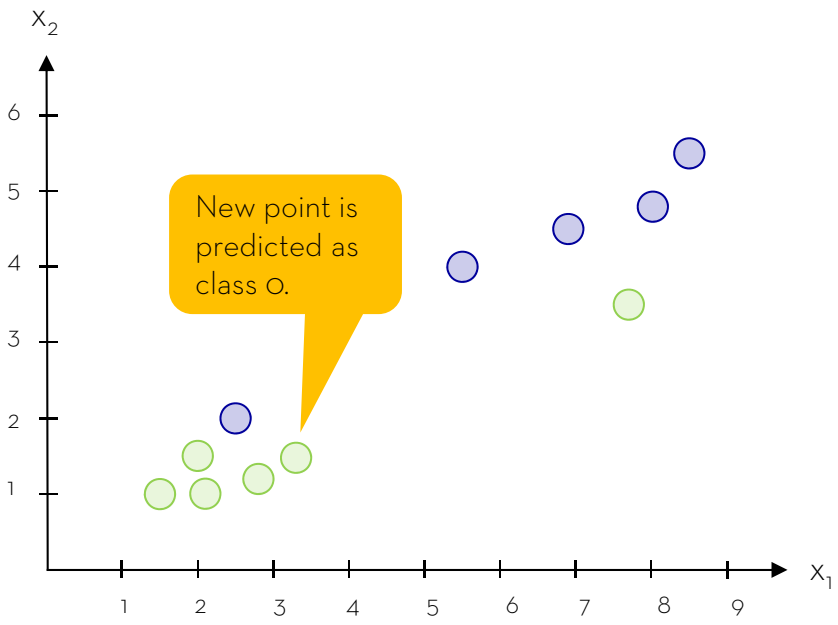
$$rel(y=1 \mid x_1, x_2)_{\text{standardized}}$$
$$= 0.001 / (0.0295 + 0.001)$$
$$= 0.033$$

Normalize the values by the sum of the posterior probabilities of both classes.

The **class with the highest probability** is considered as the most likely class
→ The new point is **predicted to be class 0.**

Step 4: Predict the class of a new point

Y	X1	X2	Y predicted
0	2	1.5	
0	2.8	1.2	
0	1.5	1	
0	2.1	1	
1	5.5	4	
1	8	4.8	
1	6.9	4.5	
1	8.5	5.5	
1	2.5	2	
0	7.7	3.5	
	3.19	1.50	0



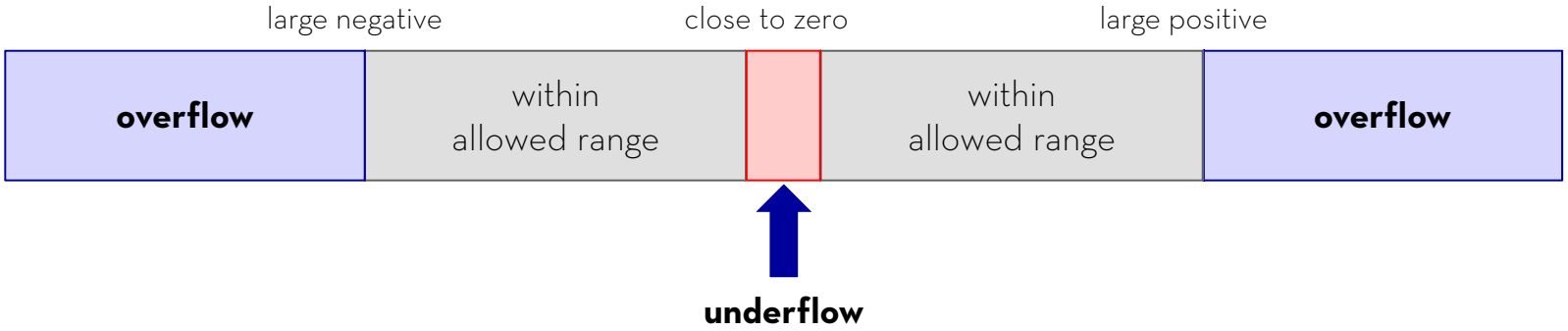
Underflow is a problem occurring in (Gaussian) Naïve Bayes models with huge number of features (1/2)

- **A posterior density** is (proportional to) a likelihood function times a prior distribution.
- The **likelihood function is a product (P(B|A)*P(A)).**
- The **number of data points** is the number of terms in the product.
- If these numbers are less than 1, and you multiply enough of them together, the result will be too small to represent in a floating point number and your **calculation will underflow to zero** (imagine: 0.00008 * 0.0006 * 0.000004 * ..... = 0.0000000...).
- Then, subsequent operations with this number, such as dividing it by another number that has also underflowed to zero, may **produce an infinite or NaN result**.

https://www.johndcook.com/blog/2012/07/26/avoiding-underflow-in-bayesian-computations/

Underflow is a problem occurring in (Gaussian) Naïve Bayes models with huge number of features (2/2)

Problem



Solution

Use **logs** to avoid underflow (or overflow), i.e.: $c_{NB} = \arg\max[\log P(c_j) + \sum \log P(x_i \mid c_j)]$

https://www.johndcook.com/blog/2012/07/26/avoiding-underflow-in-bayesian-computations/

Model variants of Naïve Bayes can make use of binary/discrete features

1. Classic Naïve Bayesian model
2. Bernoulli Naïve Bayes model
3. Multinomial Naïve Bayes model

→ The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

http://blog.datumbox.com/machine-learning-tutorial-the-naive-bayes-text-classifier/

Hybrid Naïve Bayes can use both, continuous as well as binary features for classification

- **Various approaches** exist to consider bot, continuous and binary features.
- A simple approach is to compute the likelihoods of binary variables via a Bernoulli Naïve Bayes model and compute the likelihoods of the continuous variables via a Gaussian Naïve Bayes model.
- Since we have the **conditional independence assumption** in Naive Bayes, we see that mixing variables is not a problem.

Advantages and disadvantages of the Naïve Bayes classifier

Advantages:

- Easily trained, even with a small dataset.
- Fast and highly scalable algorithm.
- Can be used for both binary and multiclass classification.
- Outperforms highly sophisticated classification methods, given the independence assumption holds.

Disadvantages:

- Independence assumption: Considers all features to be unrelated, thus cannot learn the relationship between features.
- Given a categorical variable, if a category is missing in the training set, it is impossible to predict that category.

Exercise
Naïve Bayes estimation

Hint: set the right method in the function `train()`.

1. Apply a Naïve Bayes model on the dataset.
2. Test the out-of-sample performance.

Hint: think about which data to use here.

Model tuning strategies & hyperparameter optimization

Dr. Markus Meierer - markus.meierer@business.uzh.ch | Dr. Margot Löwenberg - margot.loewenberg@business.uzh.ch

4 ways of tuning a machine learning model (1/7)

Classifier selection

Estimate **multiple classifiers** with the same data and pick the best one.

4 ways of tuning a machine learning model (2/7)

Hyperparameter tuning

Classifier selection

Use a **single classifier** which you have a particular preference for, try different settings for this classifier, and pick the best one.

4 ways of tuning a machine learning model (3/7)

Training strategy

one classifier	multiple classifiers
Hyperparameter tuning	Classifier selection

4 ways of tuning a machine learning model (5/7)

Training strategy

Final predictions made with	one classifier	multiple classifiers
one model	Hyperparameter tuning	Classifier selection
multiple models		

4 ways of tuning a machine learning model (5/7)

Training strategy

Final predictions made with	one classifier	multiple classifiers
one model	Hyperparameter tuning	Classifier selection
multiple models		Voting Averaging Stacking

Estimate **multiple classifiers** with the same data and combine the predictions.

4 ways of tuning a machine learning model (6/7)

Training strategy

Final predictions made with	one classifier	multiple classifiers
one model	Hyperparameter tuning	Classifier selection
multiple models	Bagging Boosting	Voting Averaging Stacking

Use a **single classifier** which you have a particular preference for, apply this classifier to different samples of the data and combine the predictions.

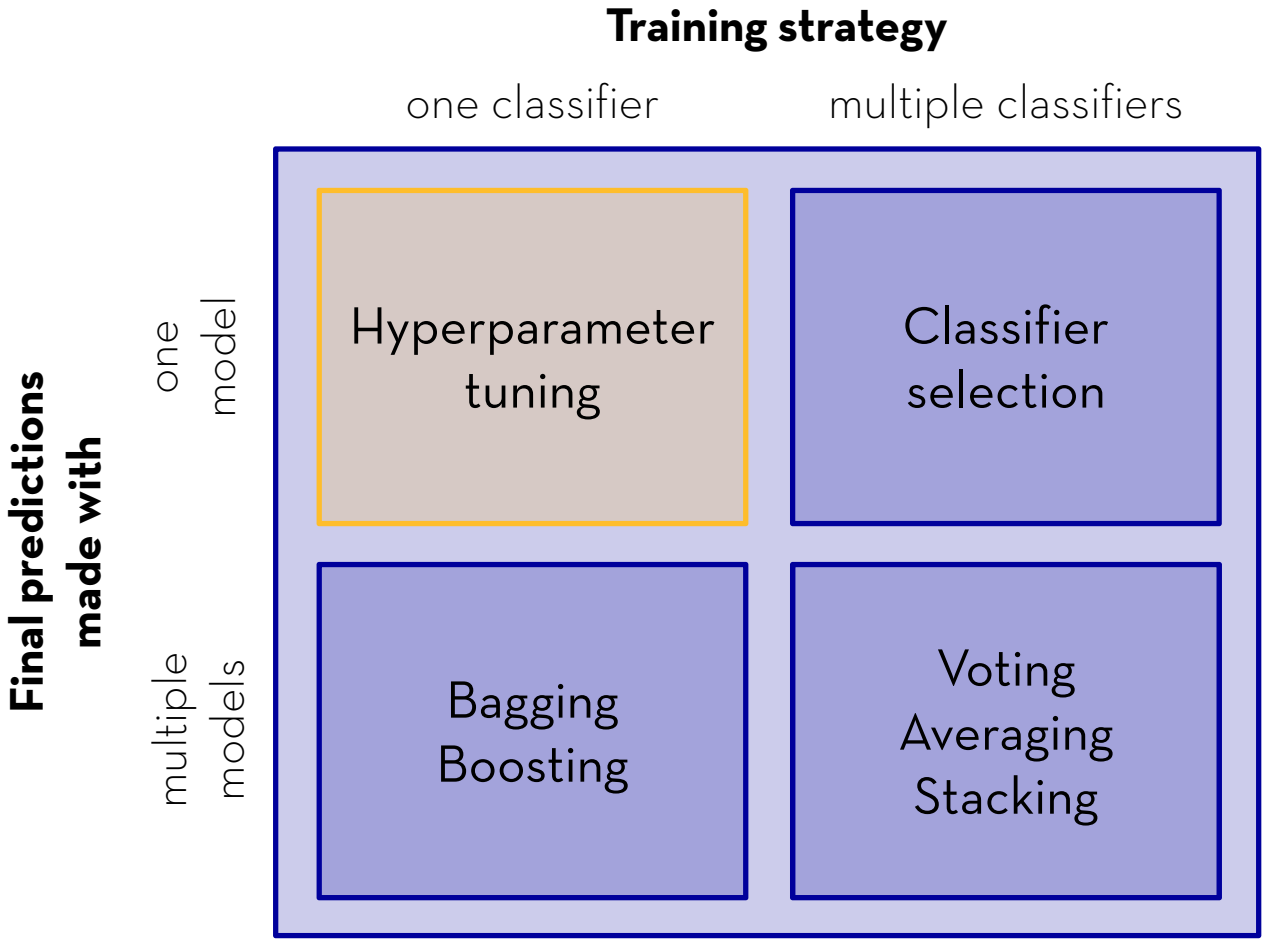
4 ways of tuning a machine learning model (7/7)

Training strategy

Final predictions made with	one classifier	multiple classifiers
one model	Hyperparameter tuning	Classifier selection
multiple models	Bagging Boosting	Voting Averaging Stacking

These model tuning techniques are not exclusive, i.e. you can **"mix-and-match"** multiple techniques.

Hyperparameter optimization is a model tuning technique based on finding the best configuration of a single classifier



Use hyperparameter tuning to improve the performance of a specific classification algorithm

Hyperparameter

- Default settings that can be changed and need to be set previous to training.
- For example:
  - Number of neighbors in a kNN.
  - Depth of a decision tree.
  - Number of trees in a random forest.

Hyperparameters impact the parameters and can significantly impact the performance of a machine learning model.

Parameter

- Settings that are directly inferred from the data set.
- For example:
  - Estimated beta coefficients of a logistic regression.
  - Estimated weights of a neural network.

The **starting values** of those parameters are hyperparameters.

https://www.quora.com/What-exactly-is-a-hyperparameter-in-machine-learning-terminology

Hyperparameter tuning is a challenging task

- Conceptually, **hyperparameter tuning is an optimization task**, just like training a machine learning model.
- However, when tuning hyperparameters, the **quality of those hyperparameters cannot be derived mathematically**, because it depends on the outcome of a black box (the model training process).

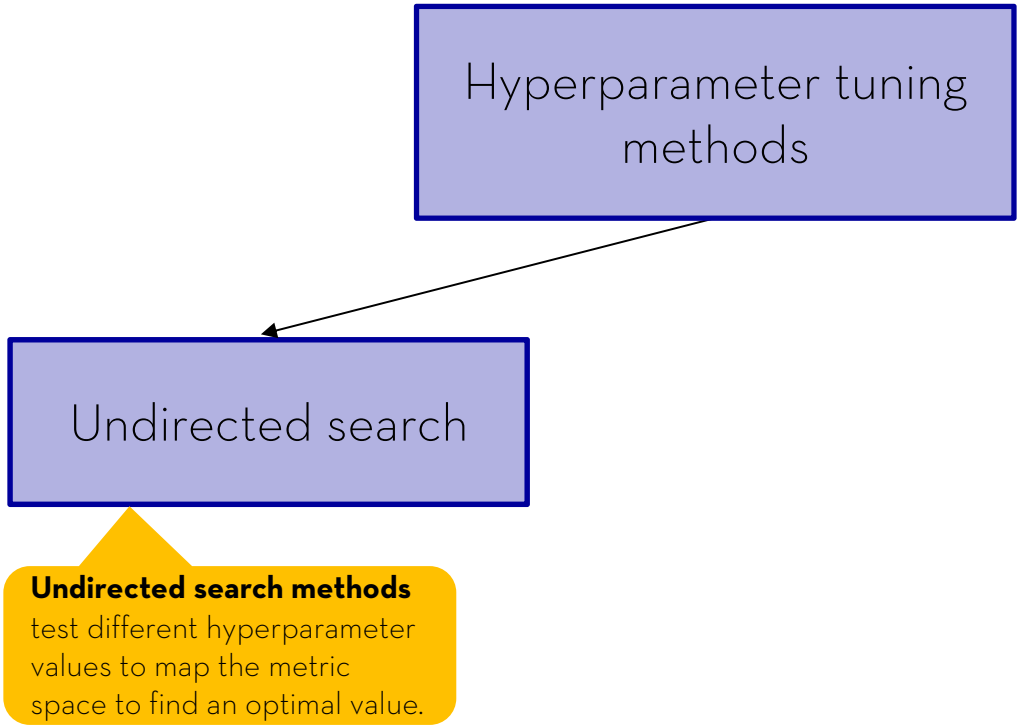
There are various ways to conduct hyperparameter optimization

Hyperparameter tuning methods

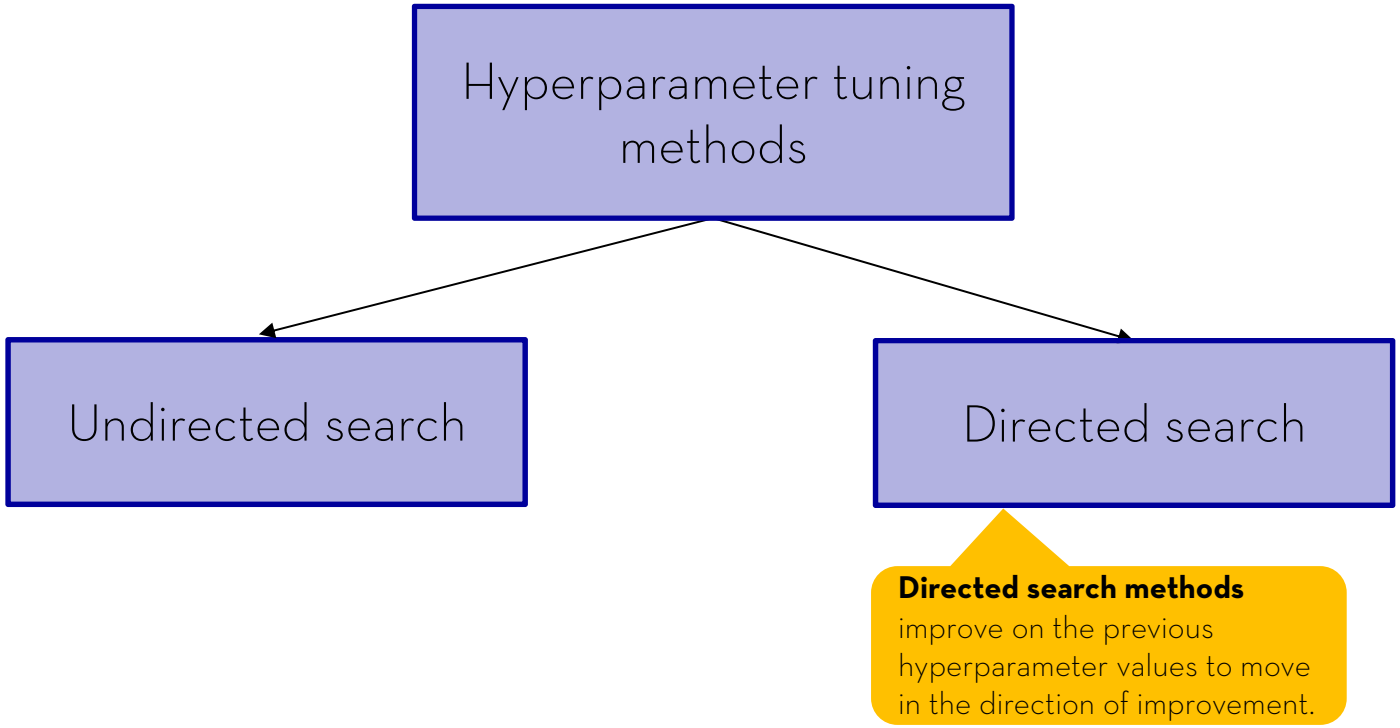
https://en.wikipedia.org/wiki/Hyperparameter_optimization

http://stats.stackexchange.com/questions/95495/guideline-to-select-the-hyperparameters-in-deep-learning

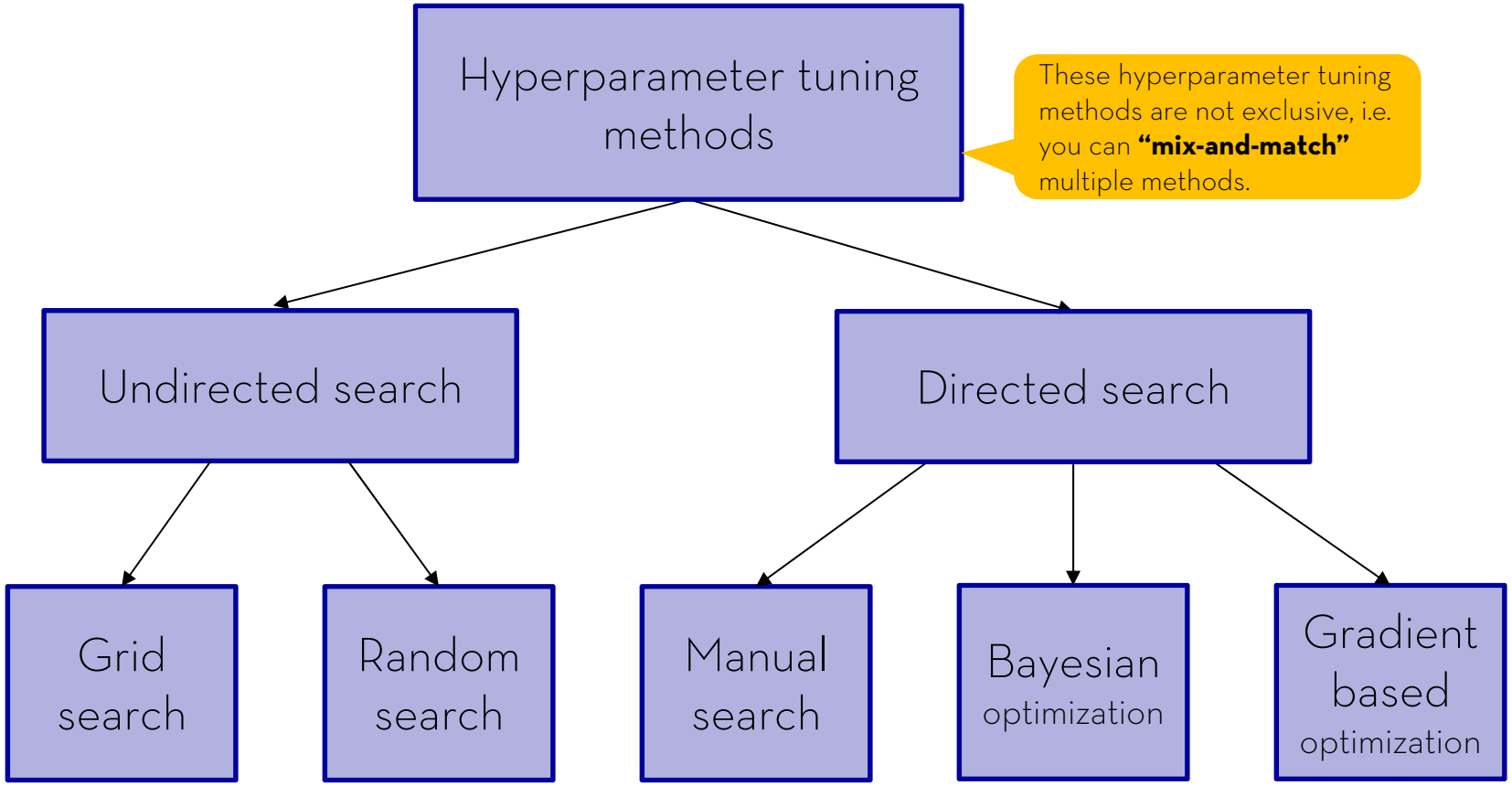
There are various ways to conduct hyperparameter optimization



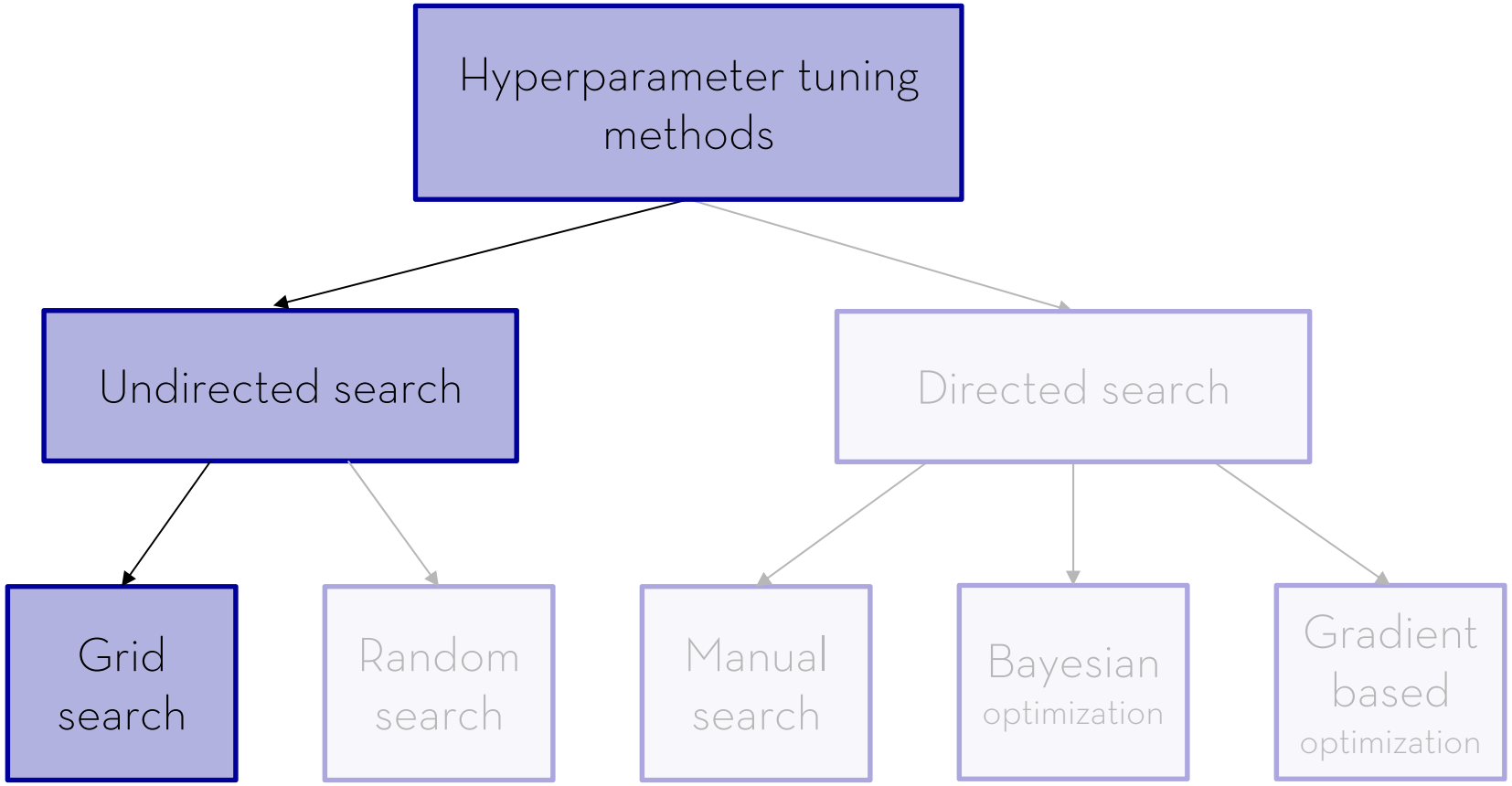
There are various ways to conduct hyperparameter optimization



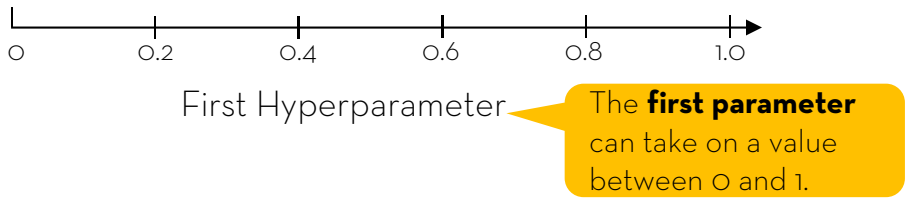
There are various ways to conduct hyperparameter optimization



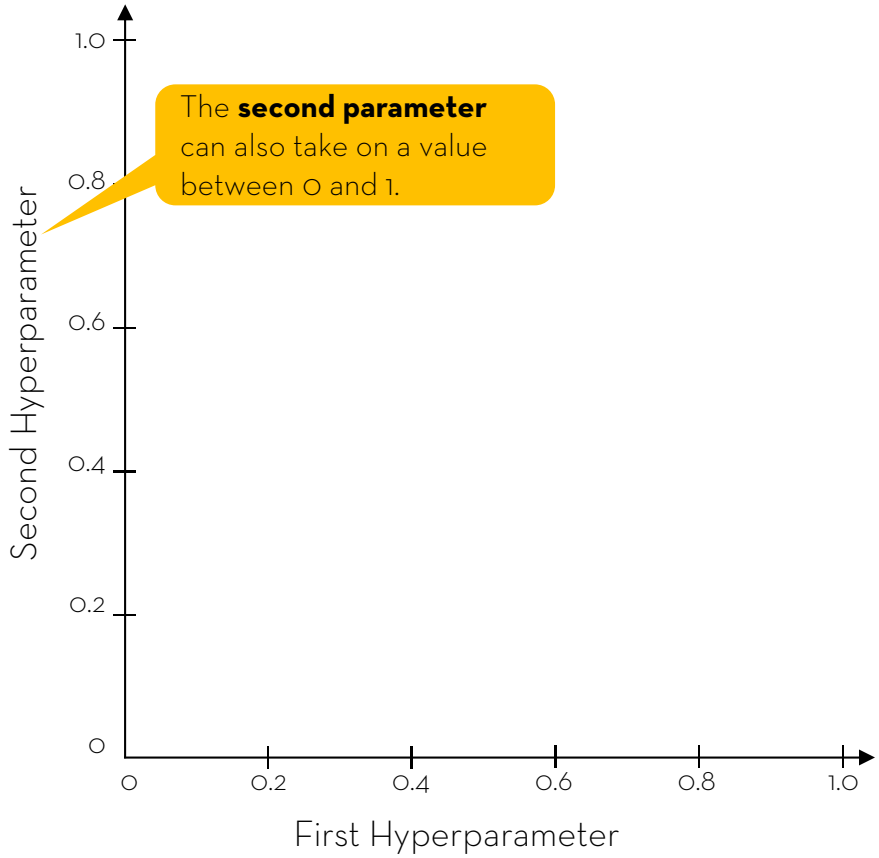
There are various ways to conduct hyperparameter optimization



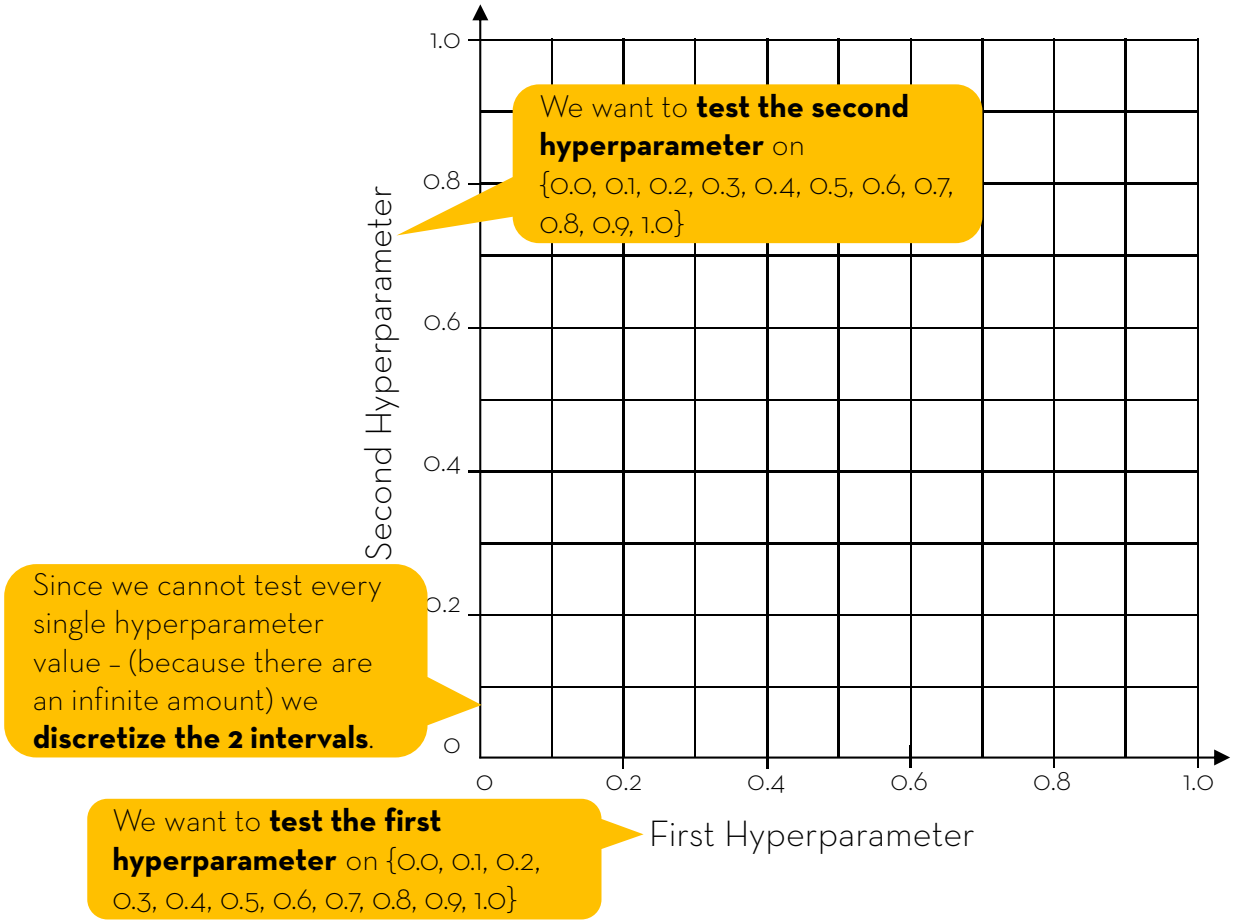
Grid search: We want to optimize 2 hyperparameters



Grid search: We want to optimize 2 hyperparameters



Grid search: Create a grid by discretizing the intervals

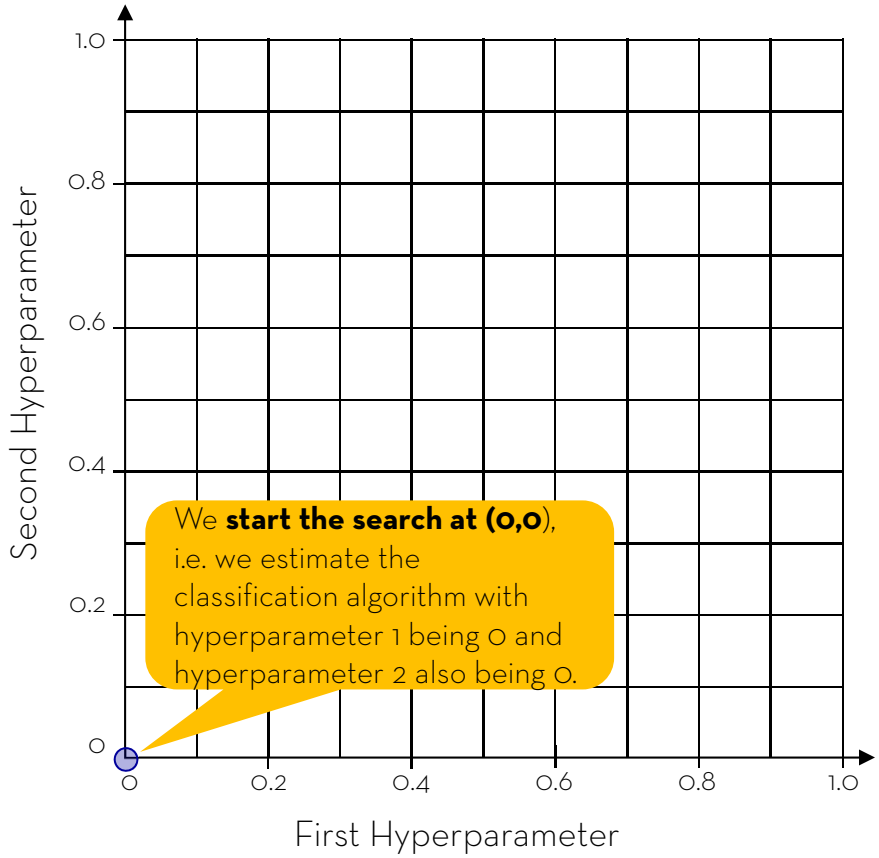


Grid search: To conduct a grid search we iterate through the parameter values

HP 1	HP 2	Accuracy
0.0	0.0	0.67

The **classification algorithm trained with (0,0)** has an overall accuracy of 67%.

We **store the predictions**, thus we are able to calculate performance metrics such as overall accuracy.

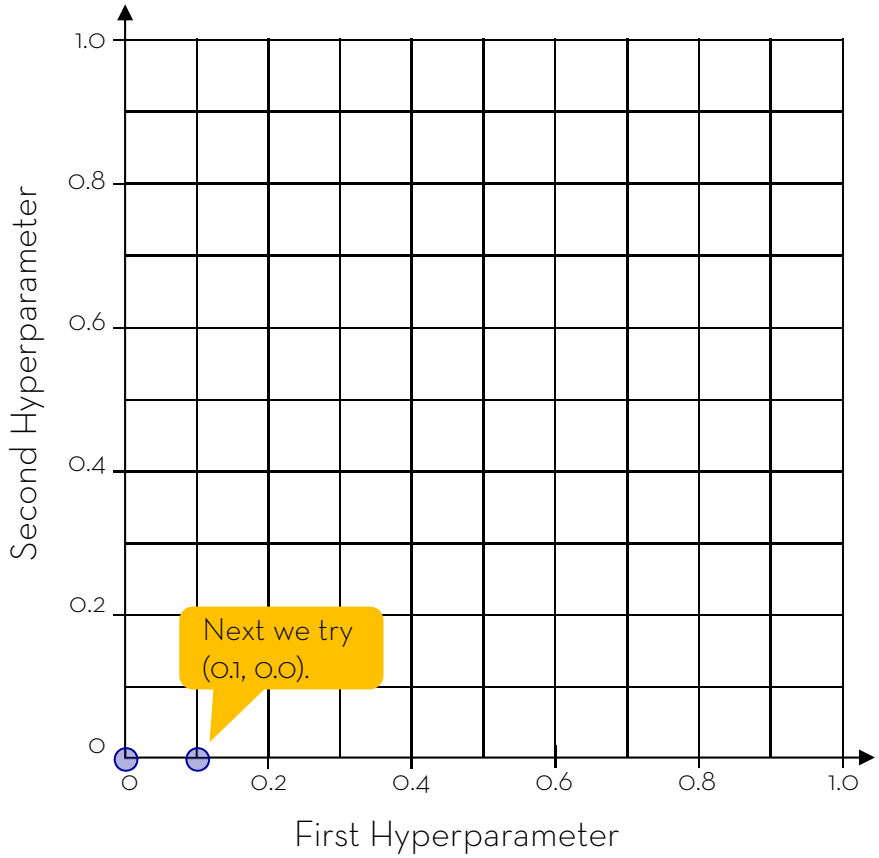


We **start the search at (0,0)**, i.e. we estimate the classification algorithm with hyperparameter 1 being 0 and hyperparameter 2 also being 0.

Grid search: To conduct a grid search we iterate through the parameter values

HP 1	HP 2	Accuracy
0.0	0.0	0.67
0.1	0.0	0.68

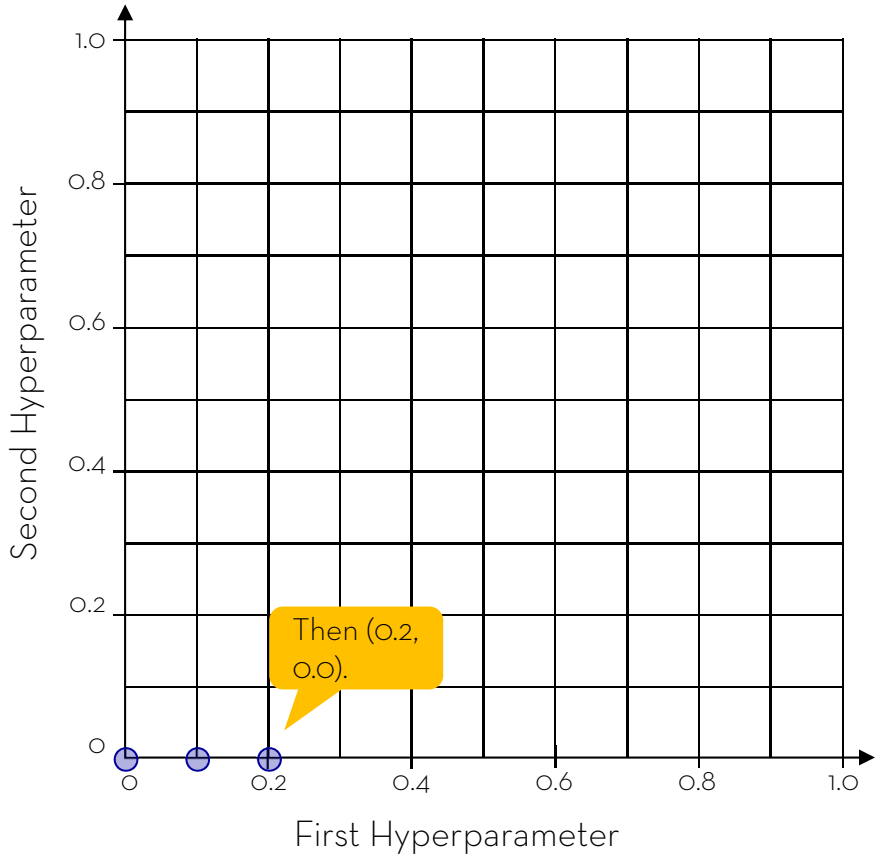
The classification algorithm trained with (0.1,0) has an overall accuracy of 68%.



Grid search: To conduct a grid search we iterate through the parameter values

HP 1	HP 2	Accuracy
0.0	0.0	0.67
0.1	0.0	0.68
0.2	0.0	0.70

The classification algorithm trained with (0.2,0) has an overall accuracy of 70%.

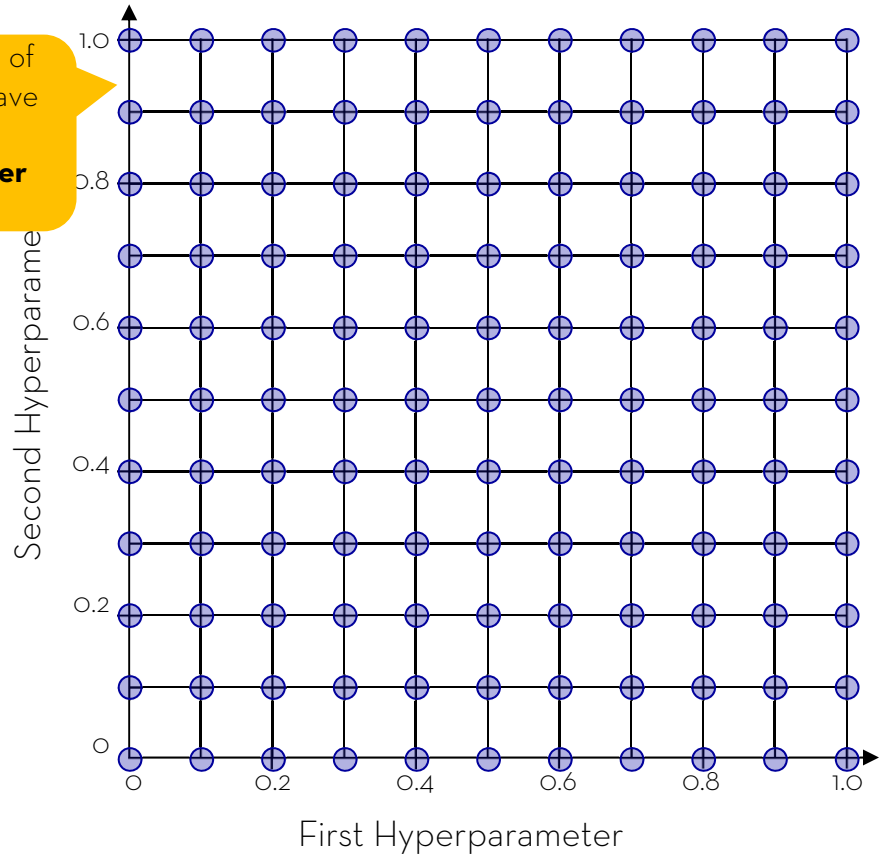


Grid search: To conduct a grid search we iterate through the parameter values

HP 1	HP 2	Accuracy
0.0	0.0	0.67
0.1	0.0	0.68
0.2	0.0	0.70
...	...	...
0.8	0.7	0.89
...	...	...
1.0	1.0	0.80

The classification algorithm trained with (1.0,1.0) has an overall accuracy of 80%.

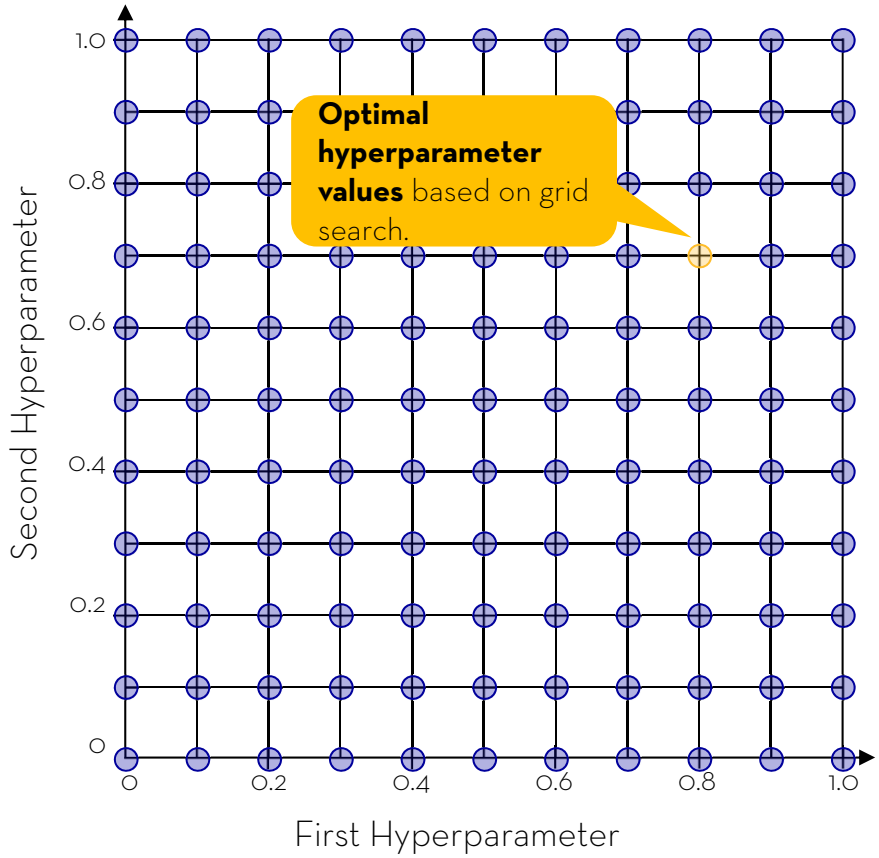
After a number of iterations we have **tested all the hyperparameter combinations**.



Grid search: Selecting the optimal hyperparameters

HP 1	HP 2	Accuracy
0.0	0.0	0.67
0.1	0.0	0.68
0.2	0.0	0.70
...	...	...
0.8	0.7	0.89
...	...	...
1.0	1.0	0.80

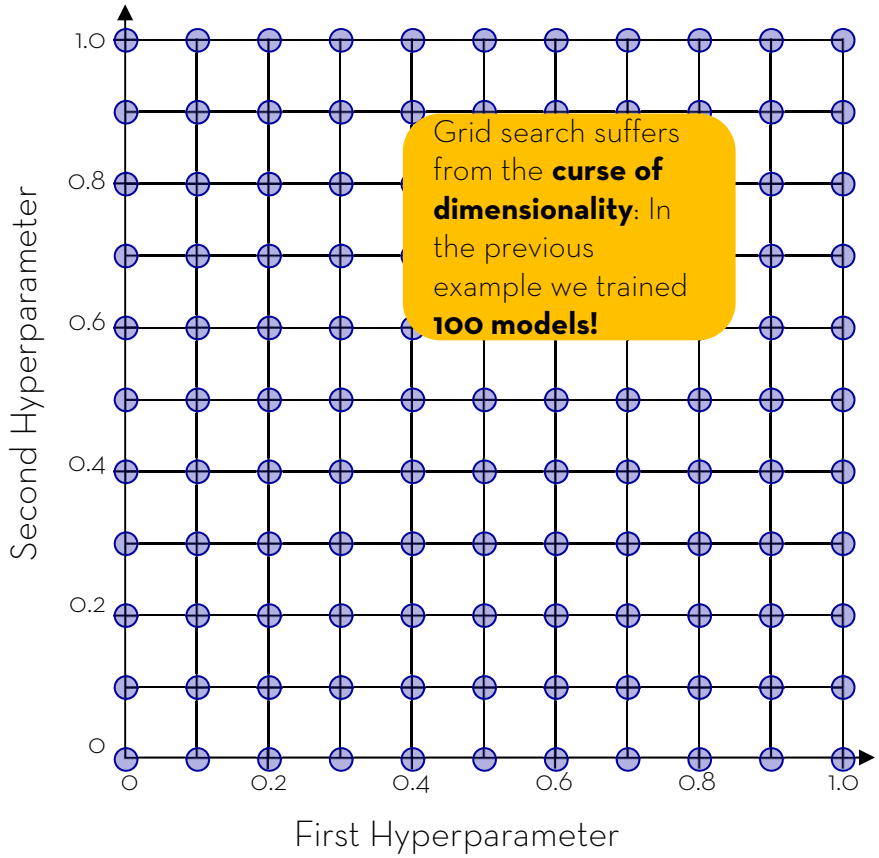
The **optimal hyperparameters based on grid search** are (0.8, 0.7) with an accuracy of 89%.



Grid search: This hyperparameter optimization is the most popular approach

- At the leading ML conference in 2014, 82 out of 86 papers used grid search to tune hyperparameters.
- Advantages:
  - Easy to implement.
  - Parallelizable.
- Disadvantages:
  - Computationally expensive, e.g. if number of hyperparameter increases.

  Possible solution: Use sparser grid.

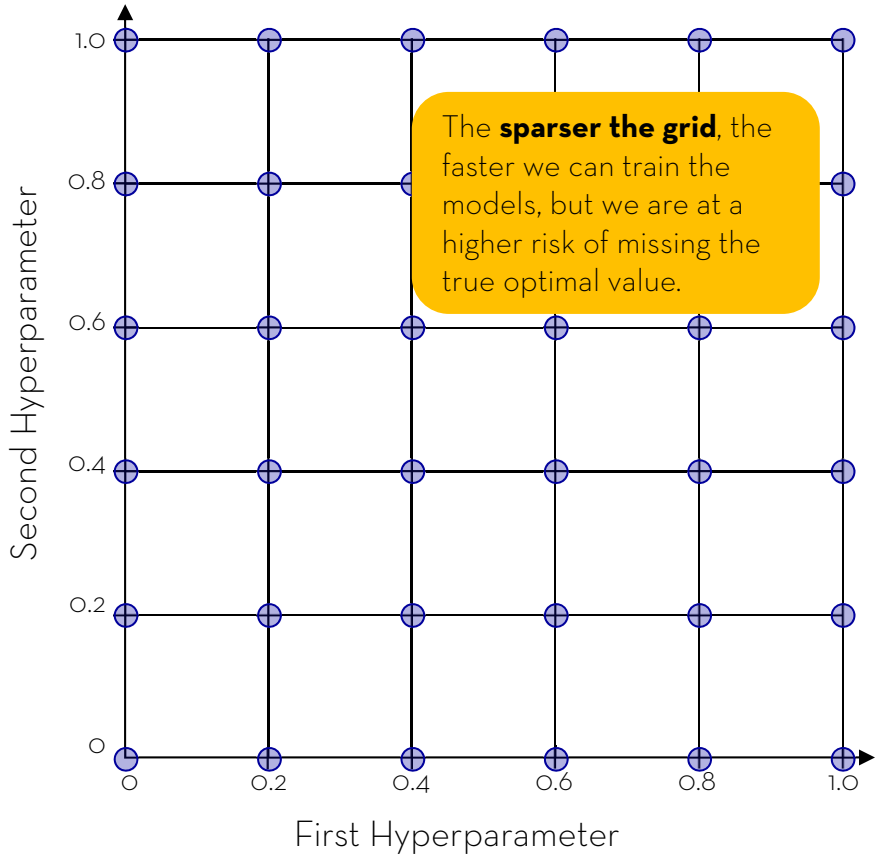


https://github.com/jaak-s/nips2014-survey

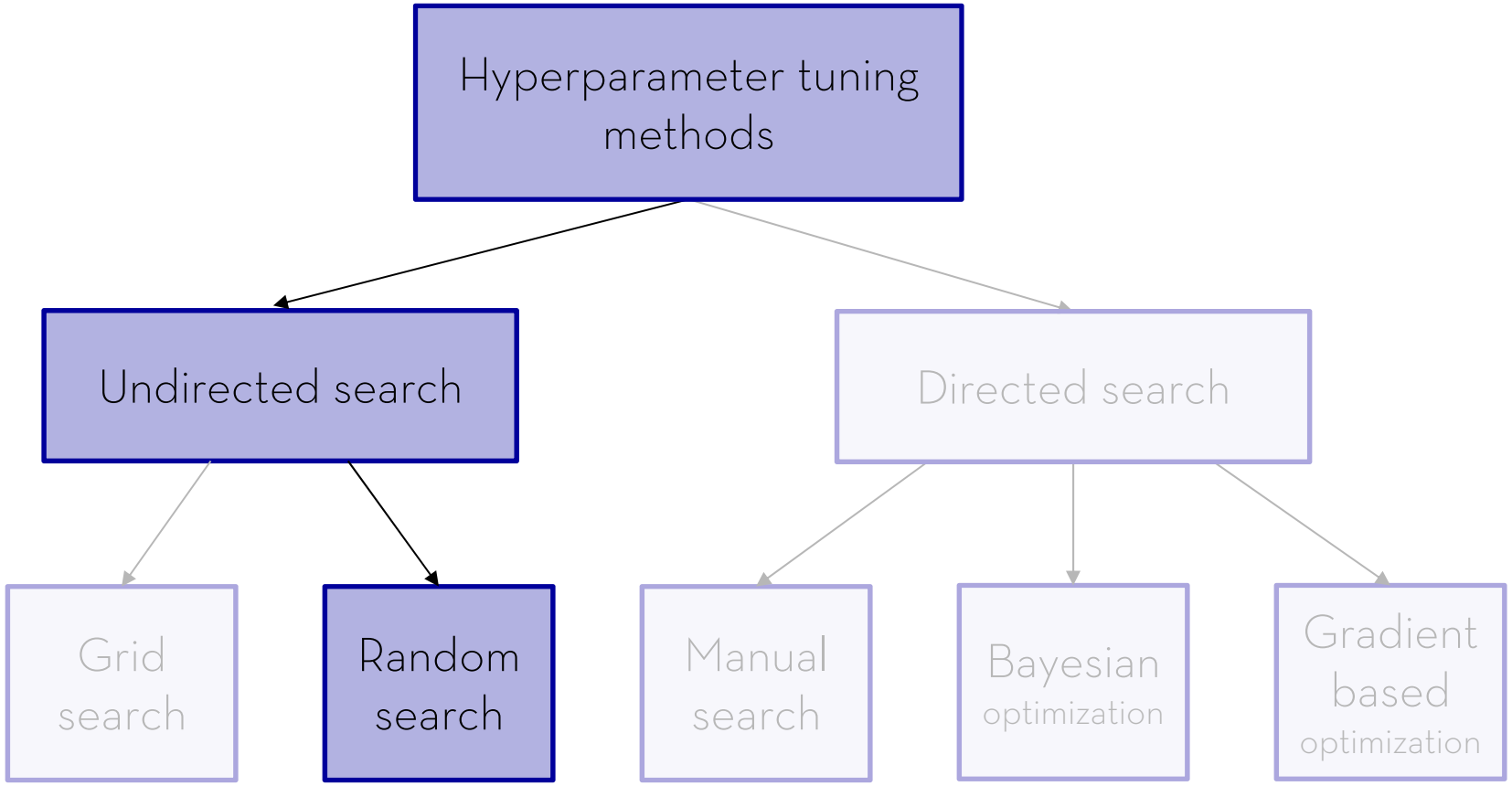
Grid search: This hyperparameter optimization is the most popular approach

- At the leading ML conference in 2014, 82 out of 86 papers used grid search to tune hyperparameters.
- Advantages:
  - Easy to implement.
  - Parallelizable.
- Disadvantages:
  - Computationally expensive, e.g. if number of hyperparameter increases.

  Possible solution: Use sparser grid.



There are various ways to conduct hyperparameter optimization



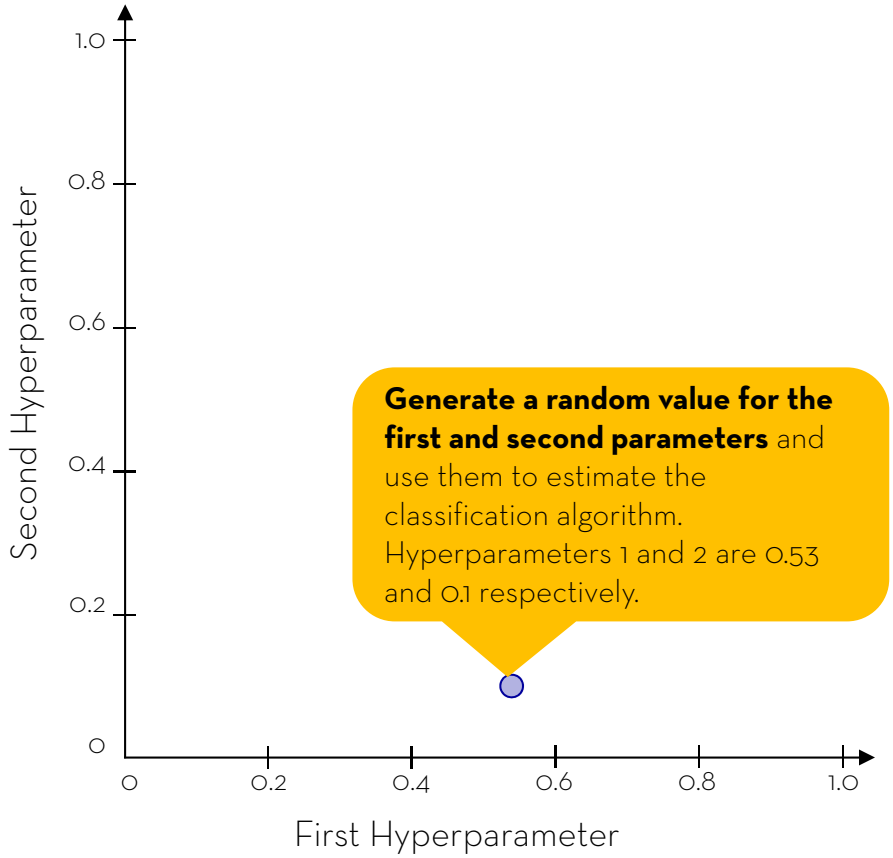
https://en.wikipedia.org/wiki/Hyperparameter_optimization

http://stats.stackexchange.com/questions/95495/guideline-to-select-the-hyperparameters-in-deep-learning

Random search: To conduct a random search we randomly generate hyperparameter values to test

HP 1	HP 2	Accuracy
0.53	0.10	0.72

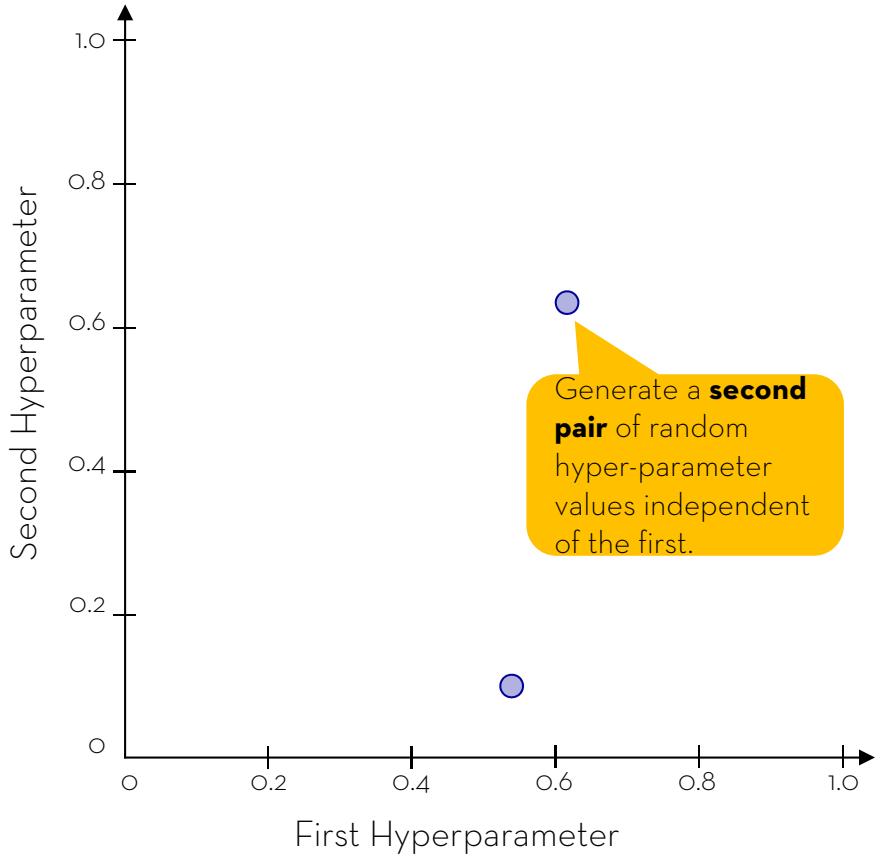
The **classification algorithm trained** with (0.53, 0.10) has an overall accuracy of 72%.



Random search: To conduct a random search we randomly generate hyperparameter values to test

HP 1	HP 2	Accuracy
0.53	0.10	0.72
0.62	0.67	0.76

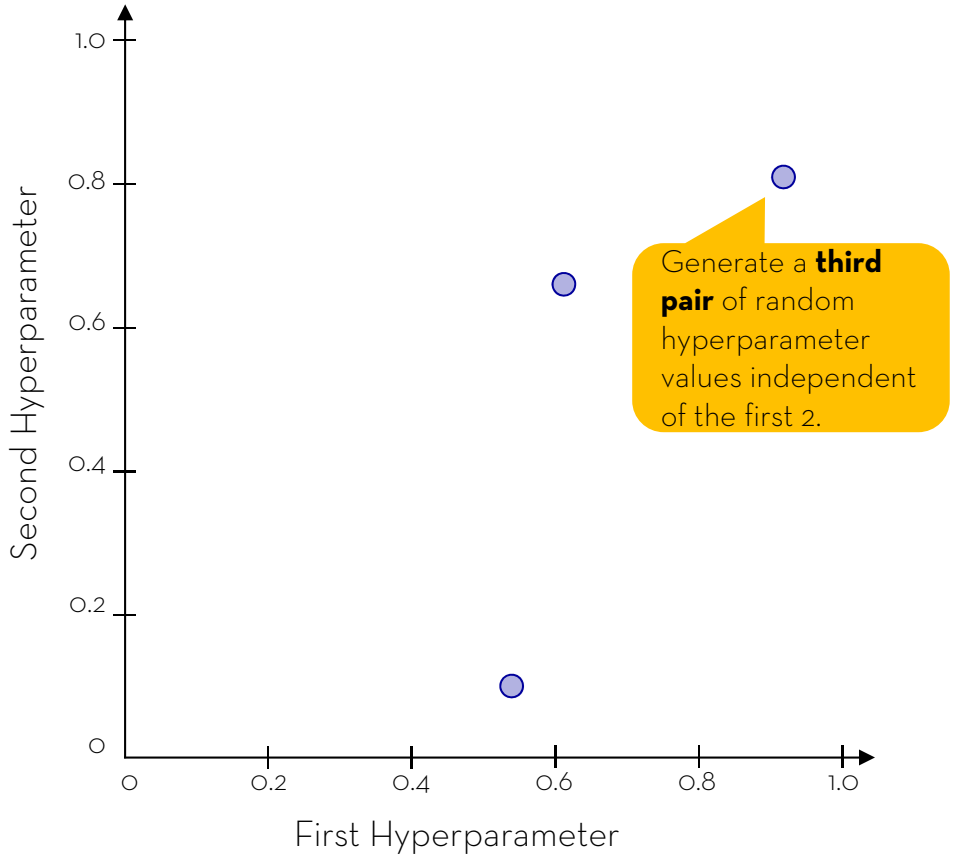
The **classification algorithm trained** with (0.62, 0.67) has an overall accuracy of 76%.



Generate a **second pair** of random hyper-parameter values independent of the first.

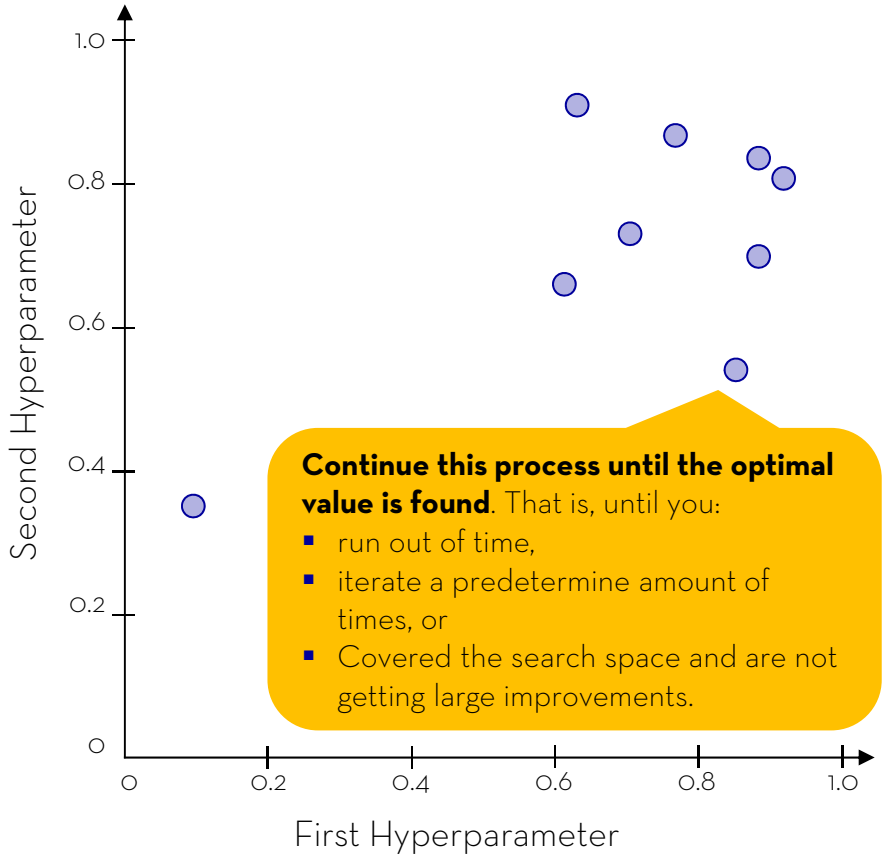
Random search: To conduct a random search we randomly generate hyperparameter values to test

HP 1	HP 2	Accuracy
0.53	0.10	0.72
0.62	0.67	0.76
0.92	0.81	0.80



Random search: To conduct a random search we randomly generate hyperparameter values to test

HP 1	HP 2	Accuracy
0.53	0.10	0.72
0.62	0.67	0.76
0.92	0.81	0.80
...	...	...
0.87	0.70	0.85
...	...	...
0.09	0.36	0.79



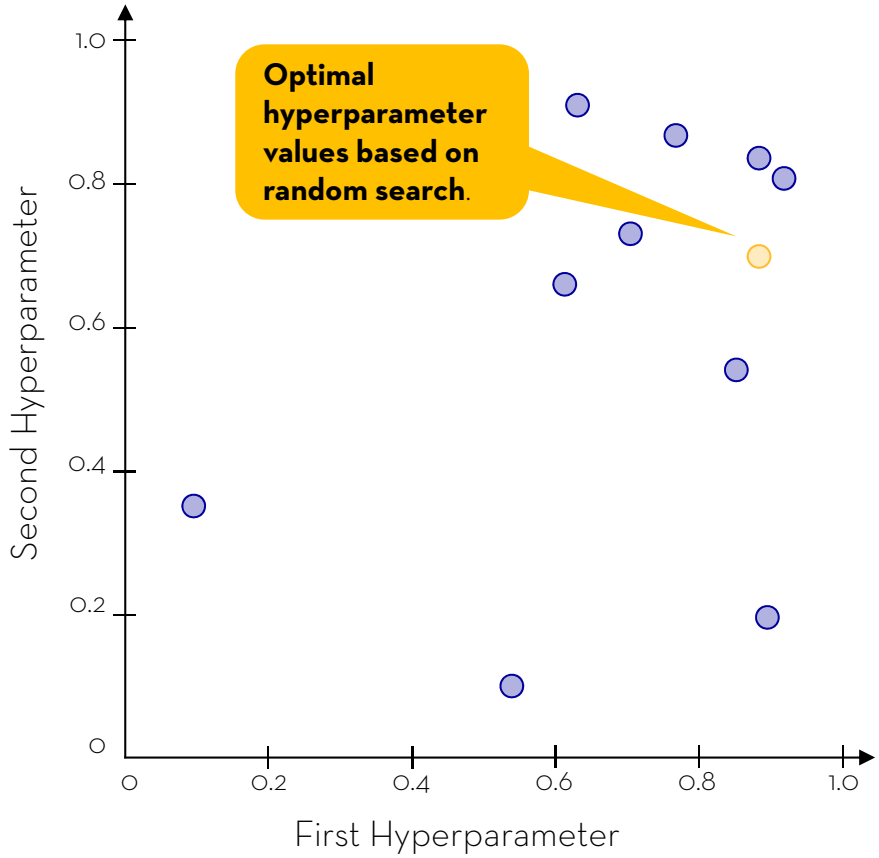
Continue this process until the optimal value is found. That is, until you:

- run out of time,
- iterate a predetermine amount of times, or
- Covered the search space and are not getting large improvements.

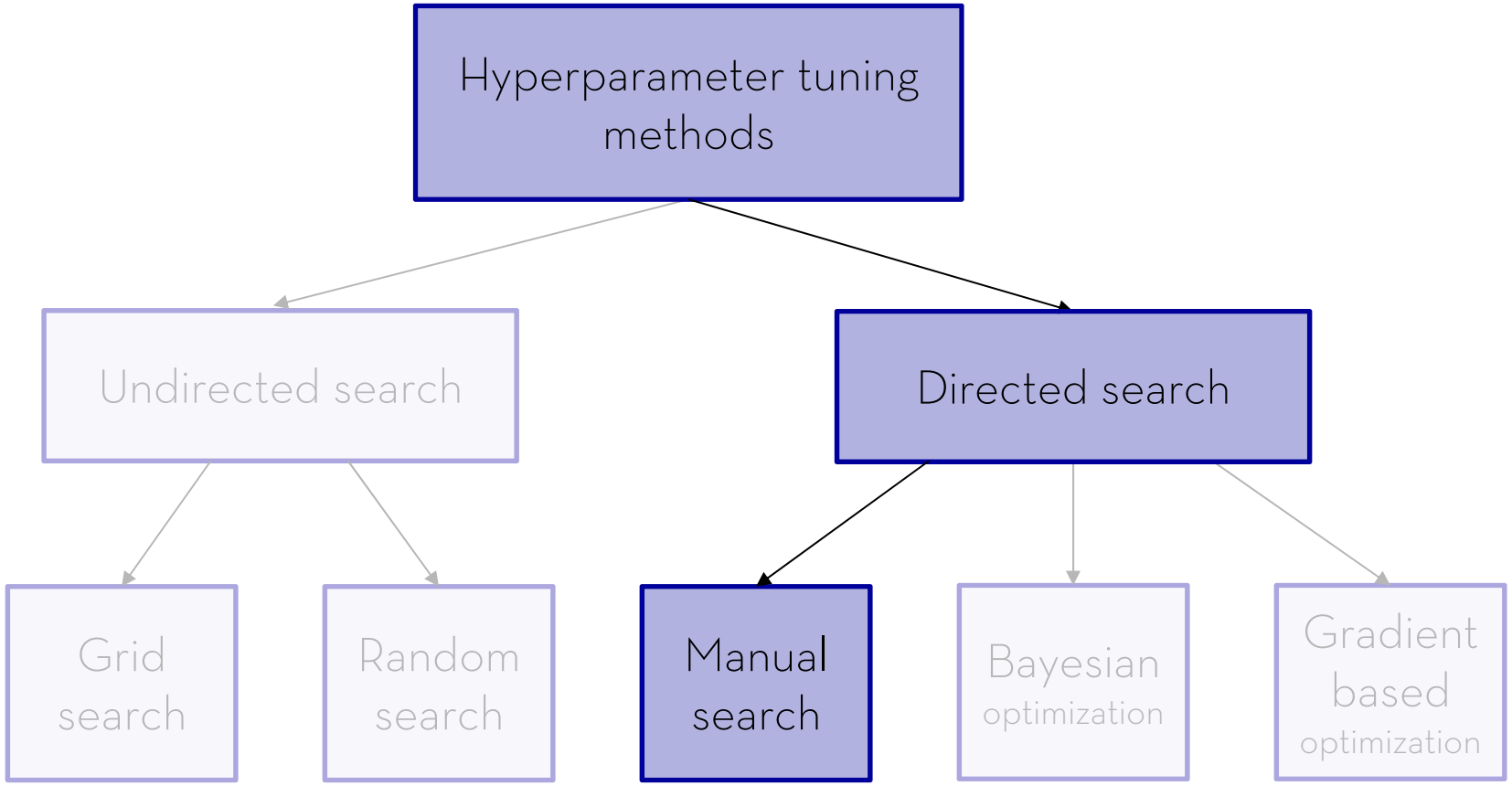
Random search: Selecting the optimal hyperparameters

HP 1	HP 2	Accuracy
0.53	0.10	0.72
0.62	0.67	0.76
0.92	0.81	0.80
...	...	...
0.87	0.70	0.85
...	...	...
0.09	0.36	[illegible]

The **optimal hyperparameters** based on random search are (0.87, 0.70) with an accuracy of 85%.



There are various ways to conduct hyperparameter optimization



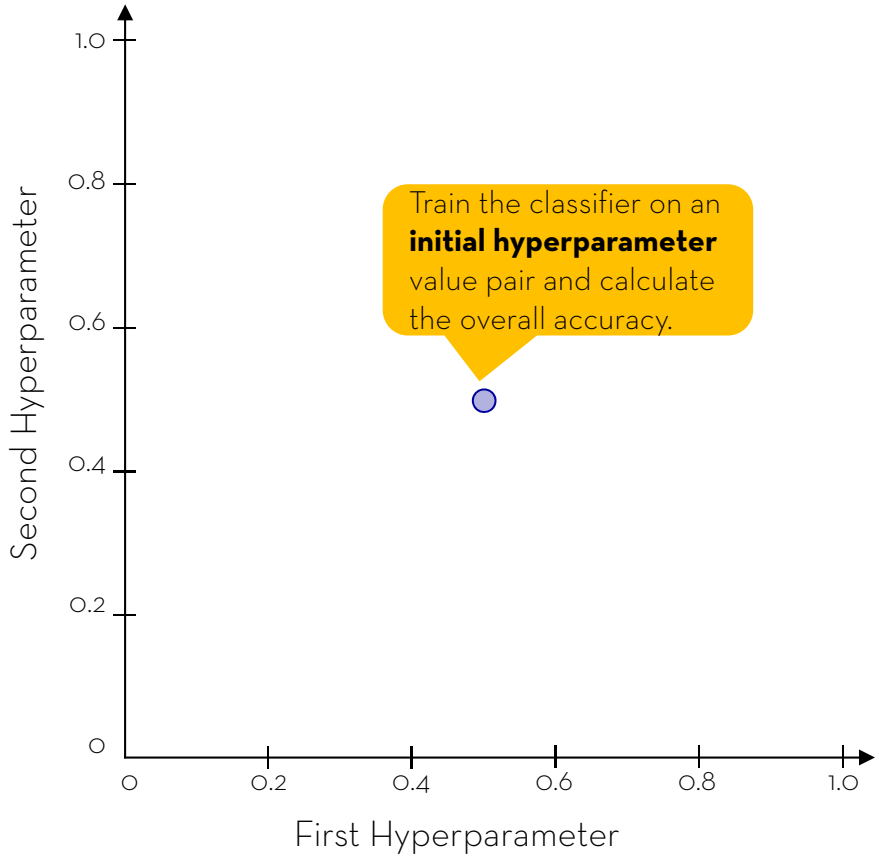
https://en.wikipedia.org/wiki/Hyperparameter_optimization

http://stats.stackexchange.com/questions/95495/guideline-to-select-the-hyperparameters-in-deep-learning

Manual search: Uses knowledge (e.g. from previous analyses) to guess and adapt parameters

HP 1	HP 2	Accuracy
0.5	0.5	0.75

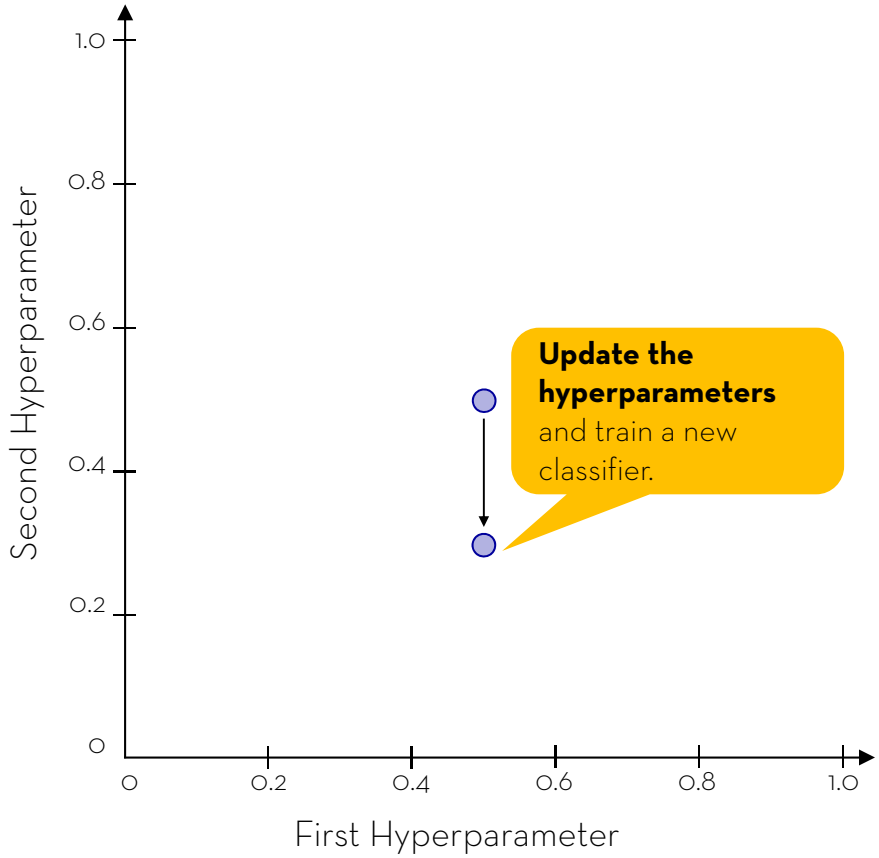
A model trained with (0.5, 0.5) has an overall accuracy of 75%.



Manual search: Uses knowledge (e.g. from previous analyses) to guess and adapt parameters

HP 1	HP 2	Accuracy
0.5	0.5	0.75
0.5	0.3	0.72

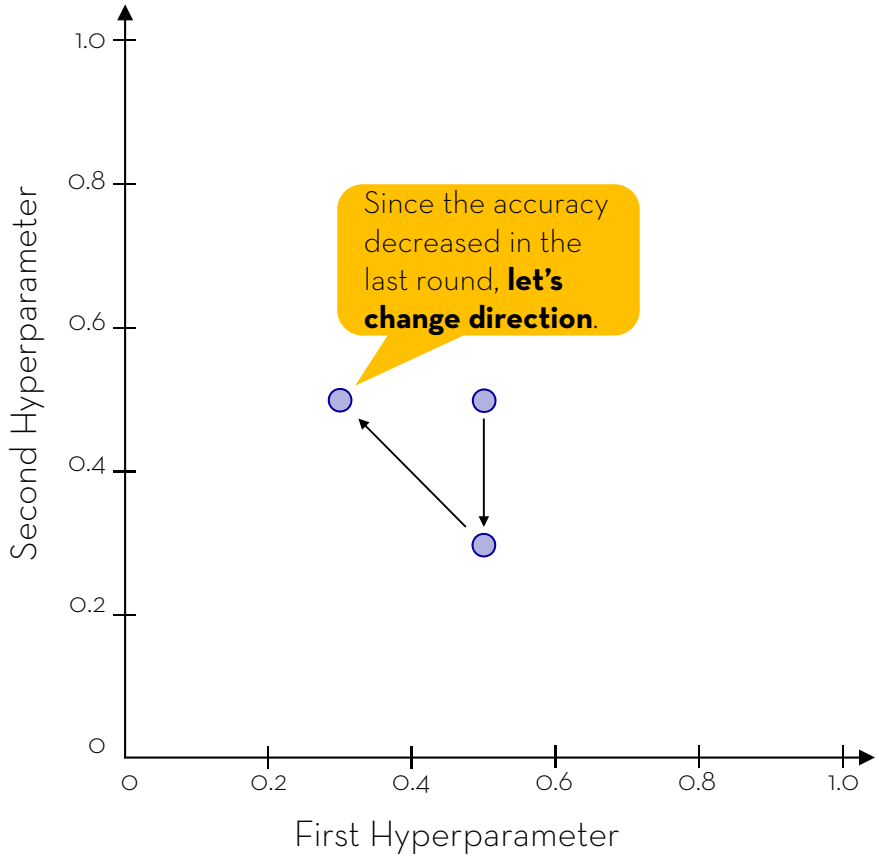
The accuracy decreased to 72% with the new hyperparameters.



Manual search: Uses knowledge (e.g. from previous analyses) to guess and adapt parameters

HP 1	HP 2	Accuracy
0.5	0.5	0.75
0.5	0.3	0.72
0.3	0.5	0.71

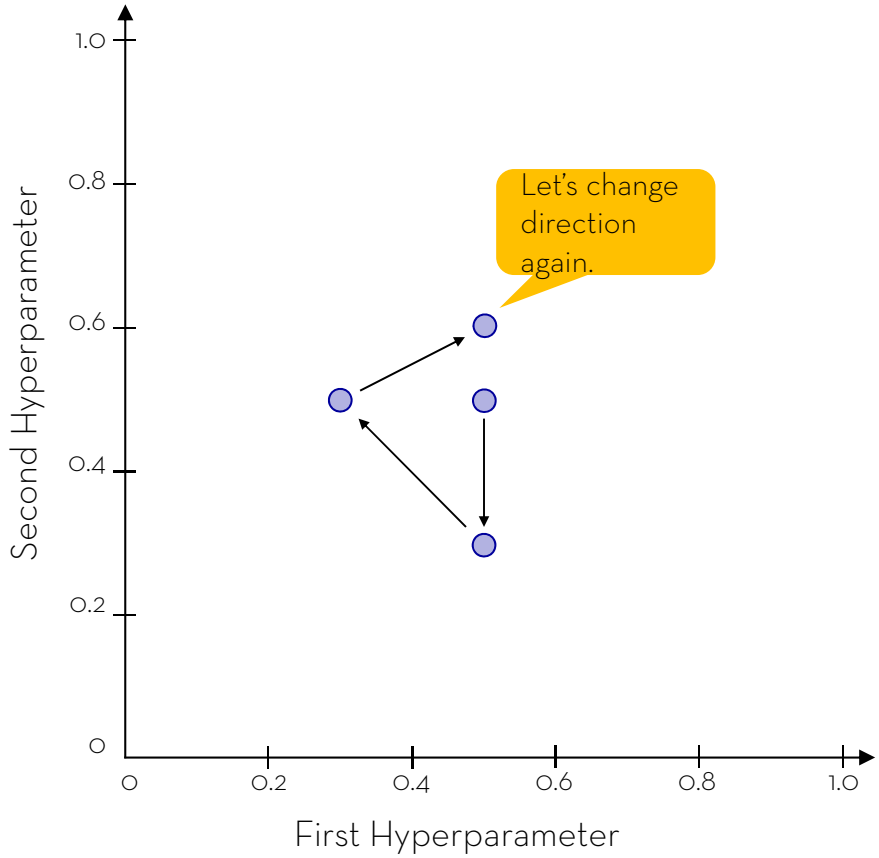
The accuracy has decreased again to 71%.



Manual search: Uses knowledge (e.g. from previous analyses) to guess and adapt parameters

HP 1	HP 2	Accuracy
0.5	0.5	0.75
0.5	0.3	0.72
0.3	0.5	0.71
0.5	0.6	0.76

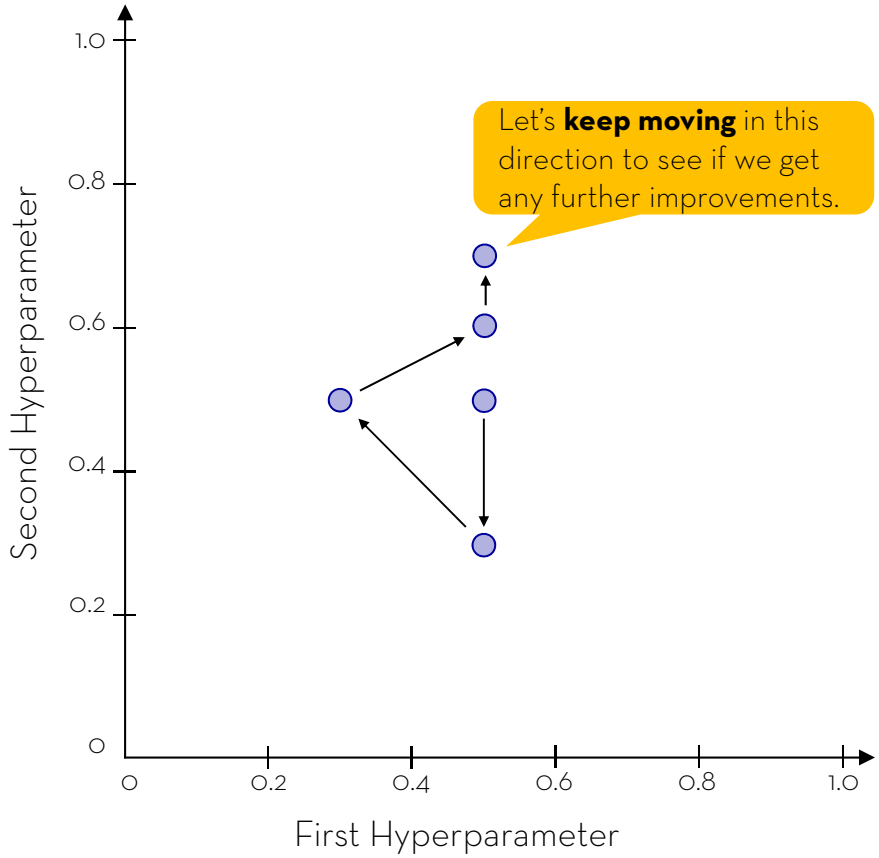
The new hyperparameters have increased the overall accuracy.



Manual search: Uses knowledge (e.g. from previous analyses) to guess and adapt parameters

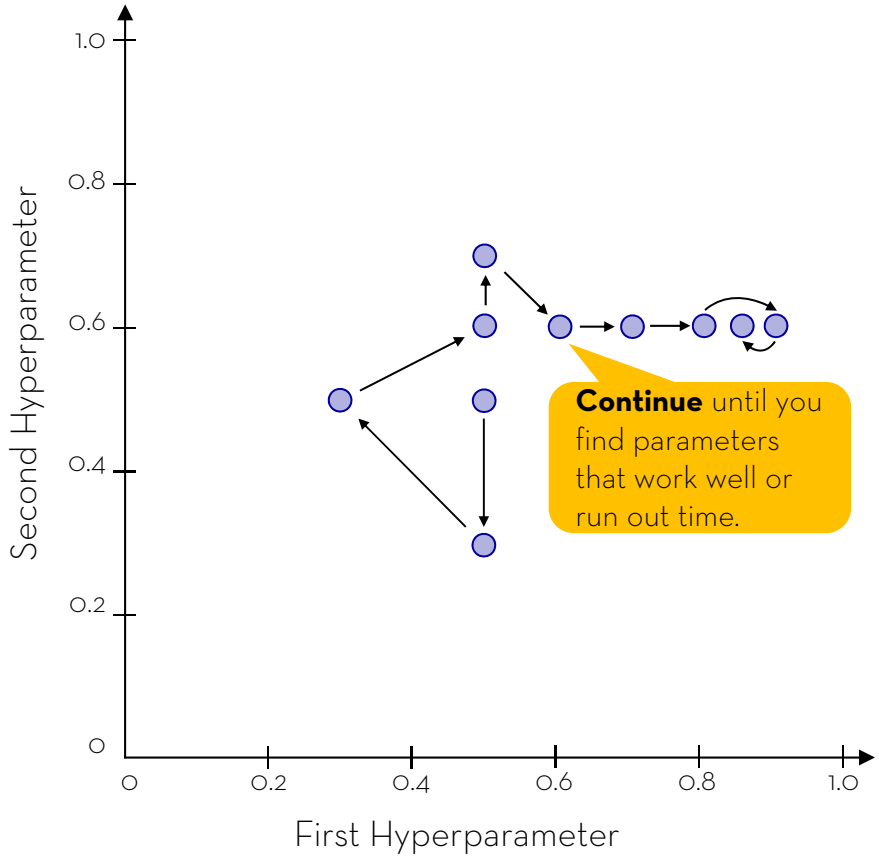
HP 1	HP 2	Accuracy
0.5	0.5	0.75
0.5	0.3	0.72
0.3	0.5	0.71
0.5	0.6	0.76
0.5	0.7	0.79

The accuracy increased again.



Manual search: Uses knowledge (e.g. from previous analyses) to guess and adapt parameters

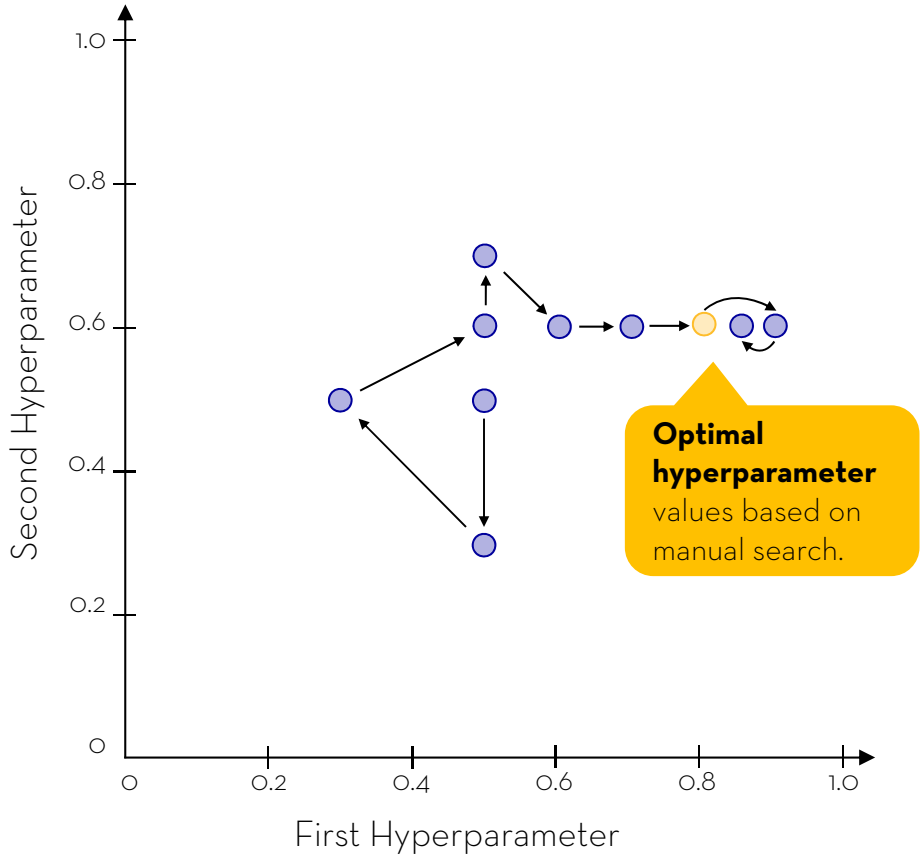
HP 1	HP 2	Accuracy
0.5	0.5	0.75
0.5	0.3	0.72
0.3	0.5	0.71
0.5	0.6	0.76
0.5	0.7	0.79
0.5	0.8	0.78
0.6	0.7	0.81
0.7	0.7	0.85
0.8	0.7	0.89
0.9	0.7	0.88
0.85	0.7	0.88



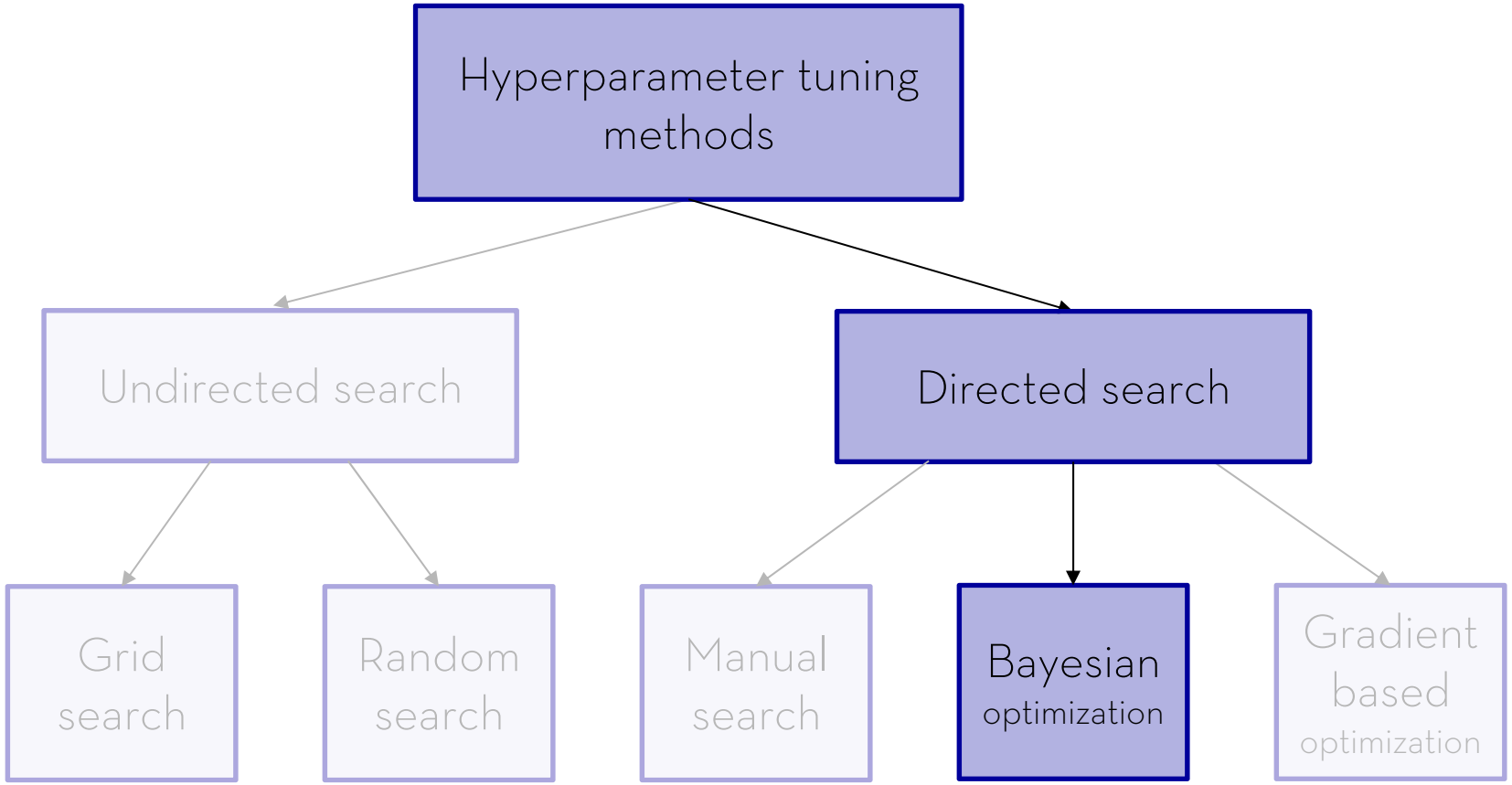
Manual search: Selecting the optimal hyperparameters

HP 1	HP 2	Accuracy
0.5	0.5	0.75
0.5	0.3	0.72
0.3	0.5	0.71
0.5	0.6	0.76
0.5	0.7	0.79
0.5	0.8	0.78
0.6	0.7	0.81
0.7	0.7	0.85
0.8	0.7	0.89
0.9	0.7	0.88
0.85	0.7	[illegible]

The **optimal hyperparameters** based on random search are (0.8, 0.7) with an accuracy of 89%.



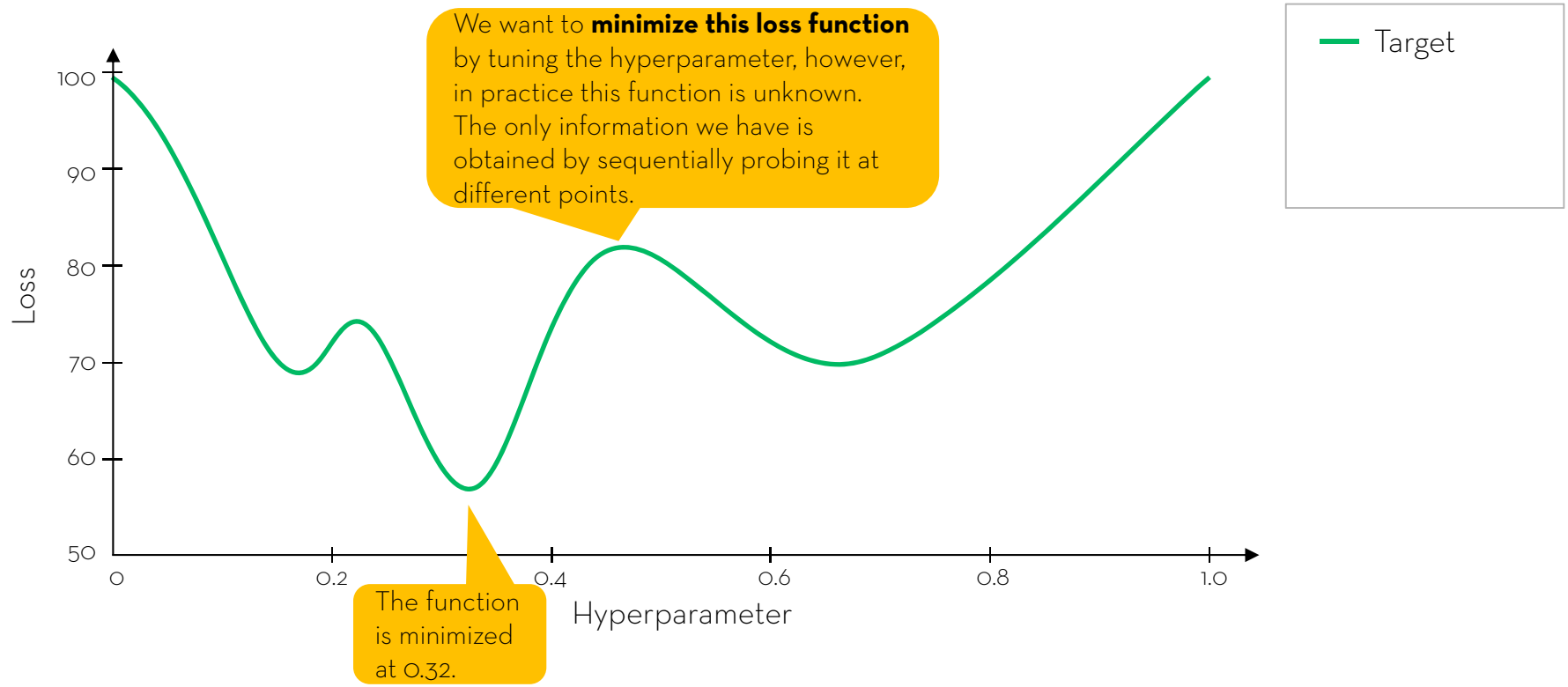
There are various ways to conduct hyperparameter optimization



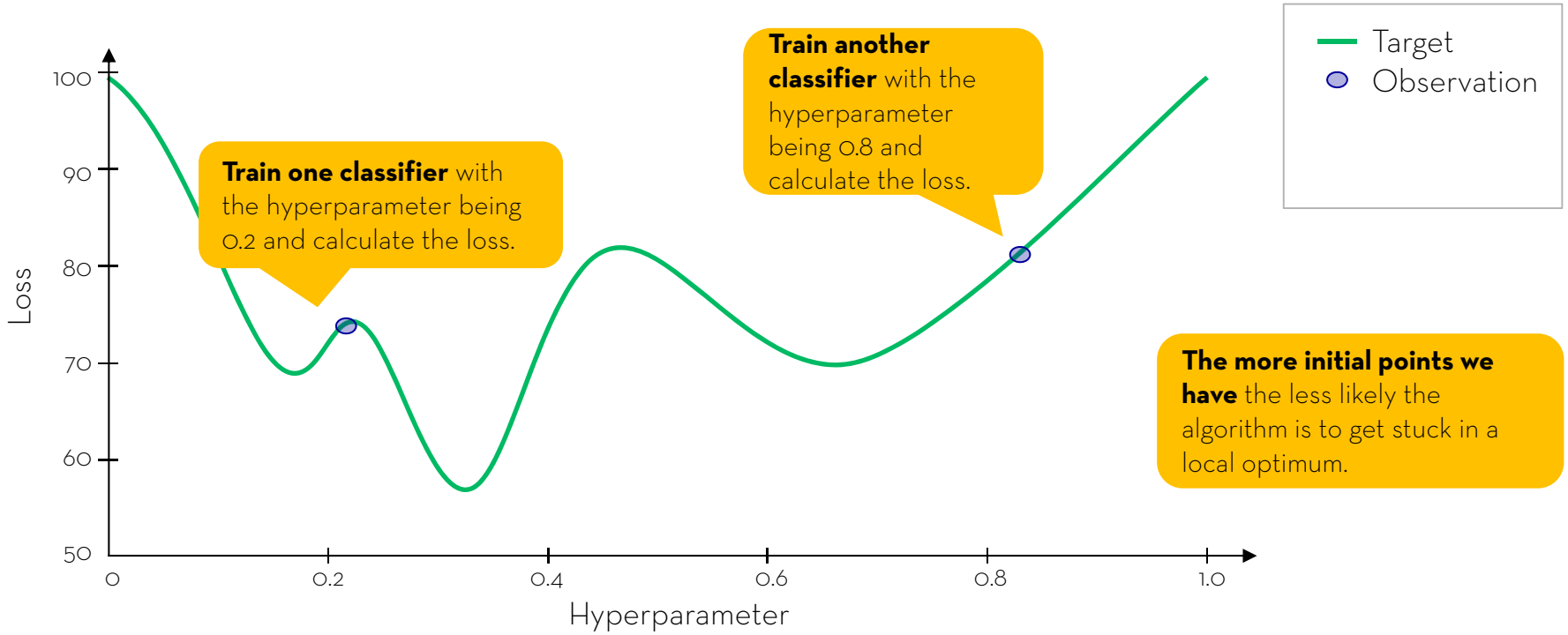
https://en.wikipedia.org/wiki/Hyperparameter_optimization

http://stats.stackexchange.com/questions/95495/guideline-to-select-the-hyperparameters-in-deep-learning

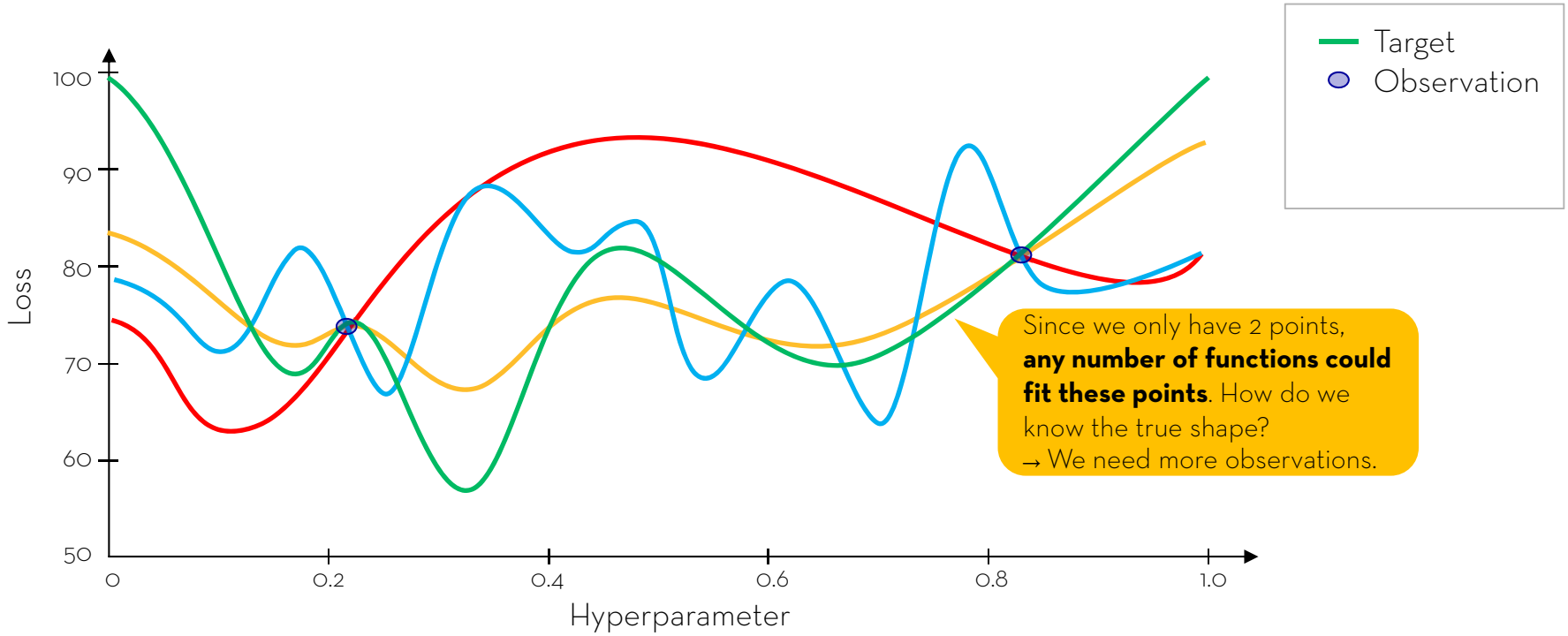
Bayesian optimization: We want to optimize 1 hyperparameter



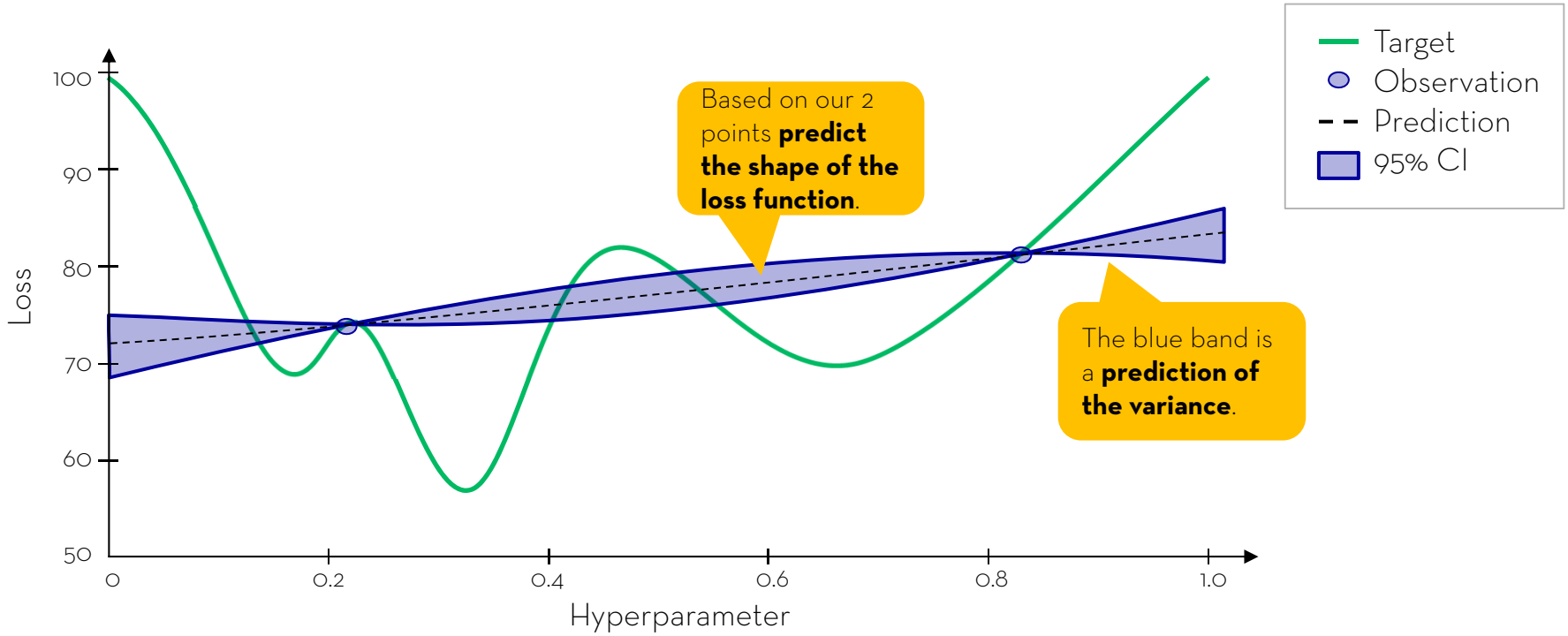
Bayesian optimization: We need (at least) 2 initial guesses to start the algorithm



Bayesian optimization: These two observations could fit a number of curves

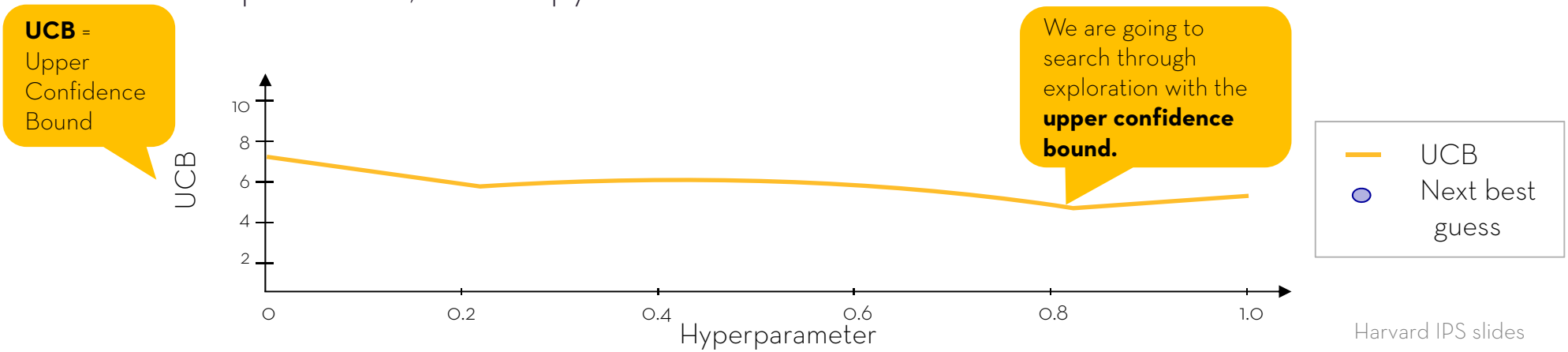


Bayesian optimization: Use a Gaussian process to predict the shape of the loss function



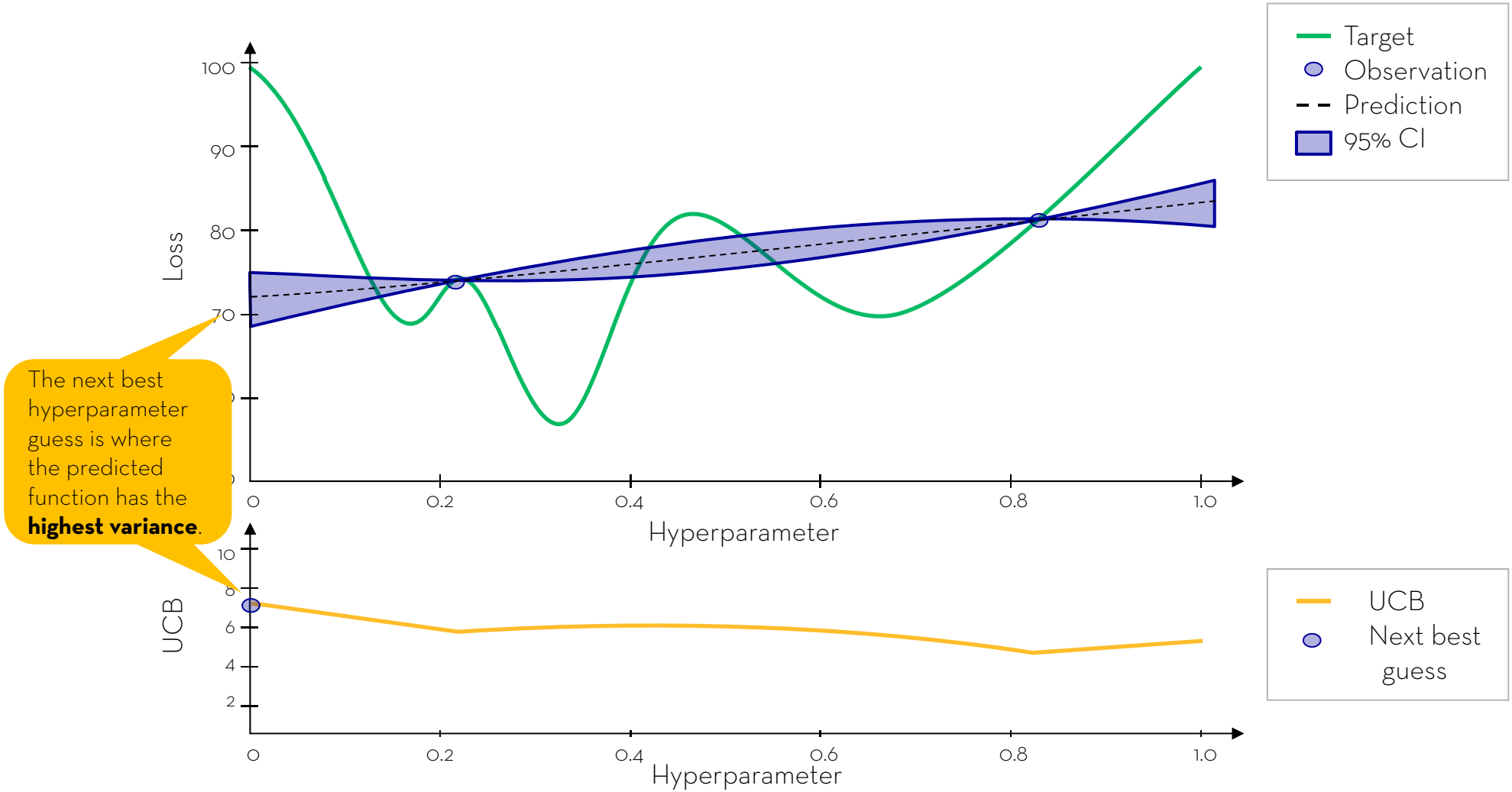
Bayesian optimization: Search the hyperparameter space through exploration or exploitation

- We want the **next observation to give us as much information about the shape of the loss function as possible**:
  - **Exploration**: Next guess is where the variance in our prediction is highest.
  - **Exploitation**: Next guess is where the mean of our prediction is lowest.
- **Possible metrics**: upper (lower) confidence bound, expected improvement, probability of improvement, or entropy search.

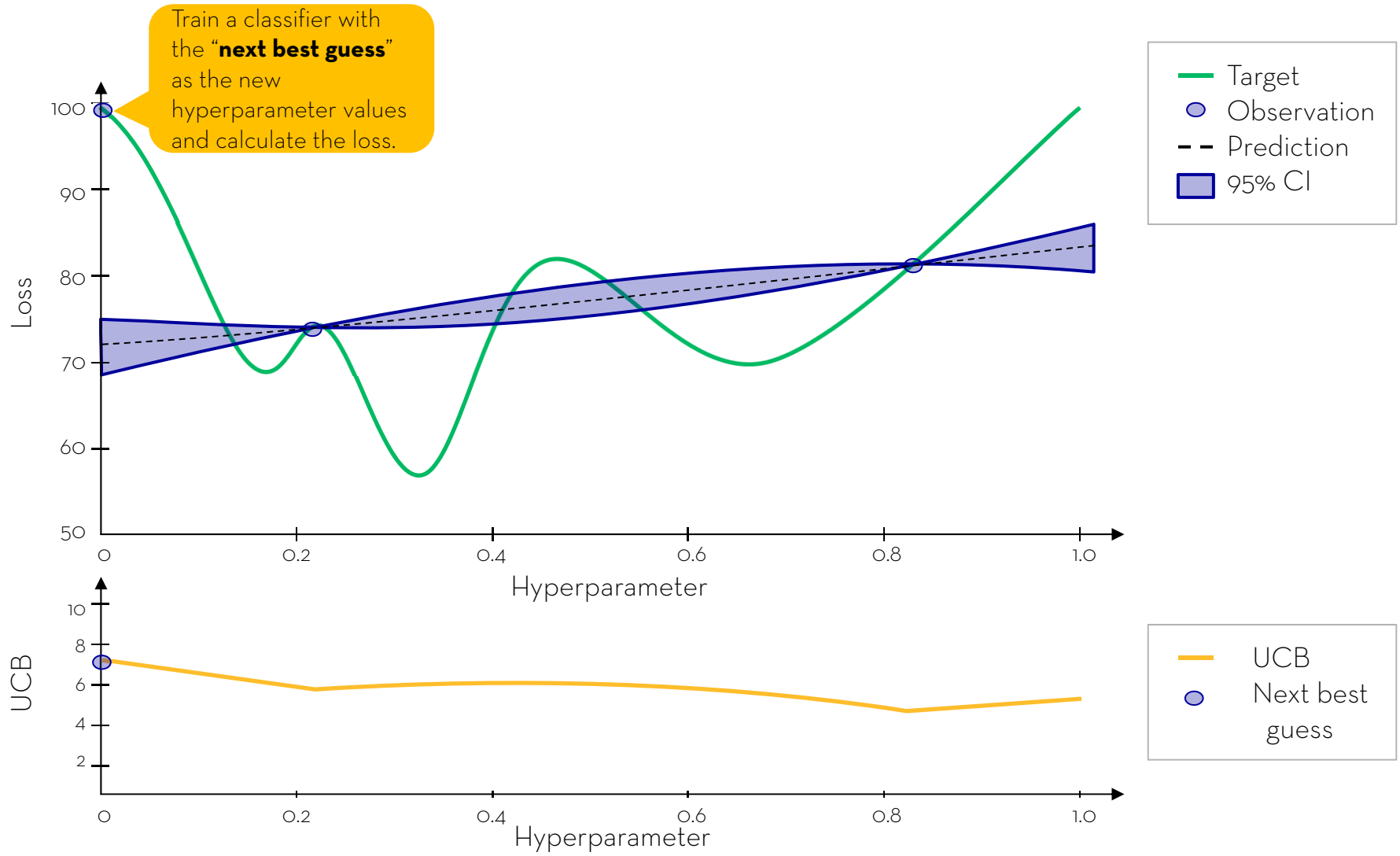


Harvard IPS slides

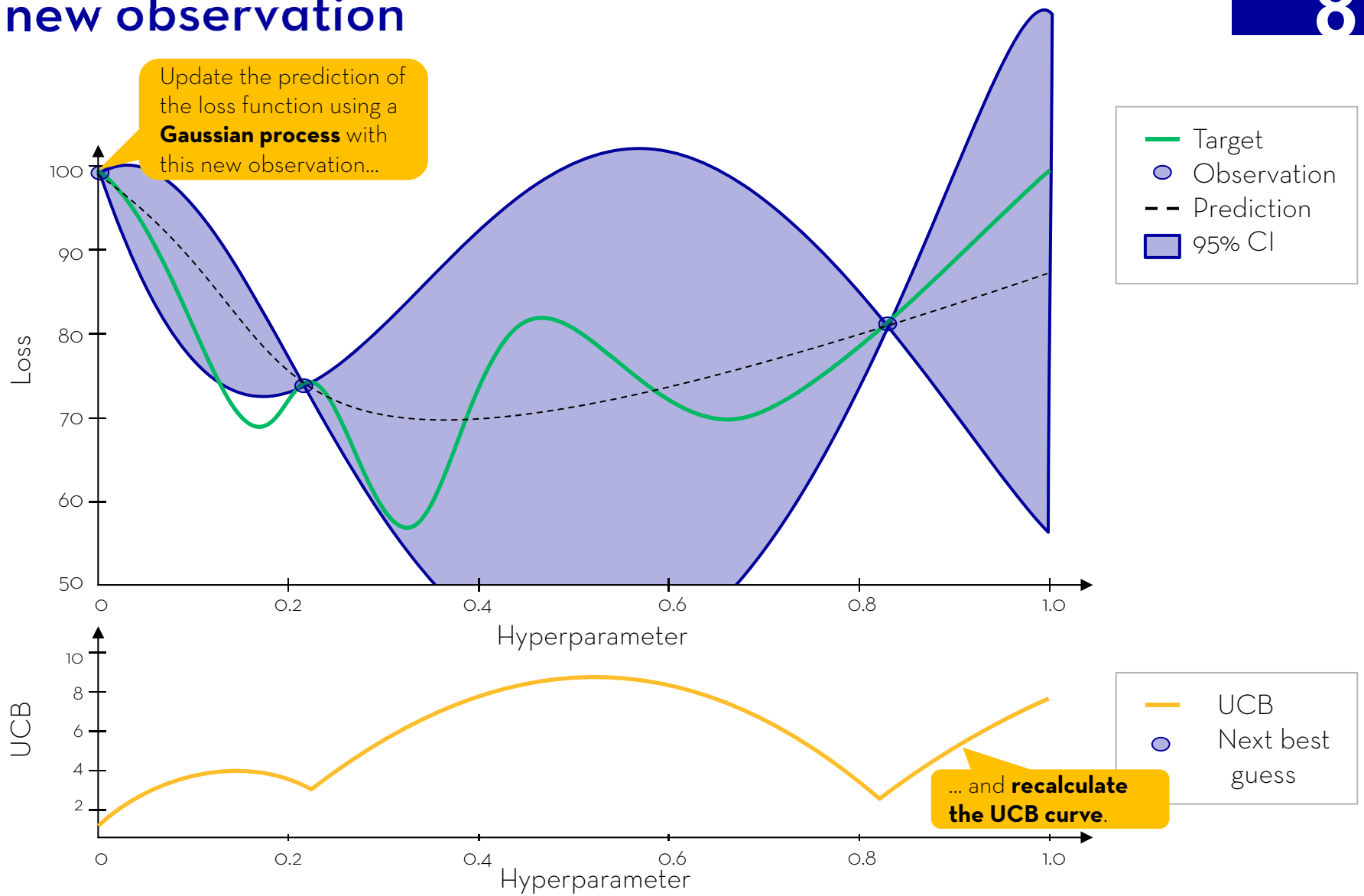
Bayesian optimization: Based on the upper confidence bound identify the next best guess



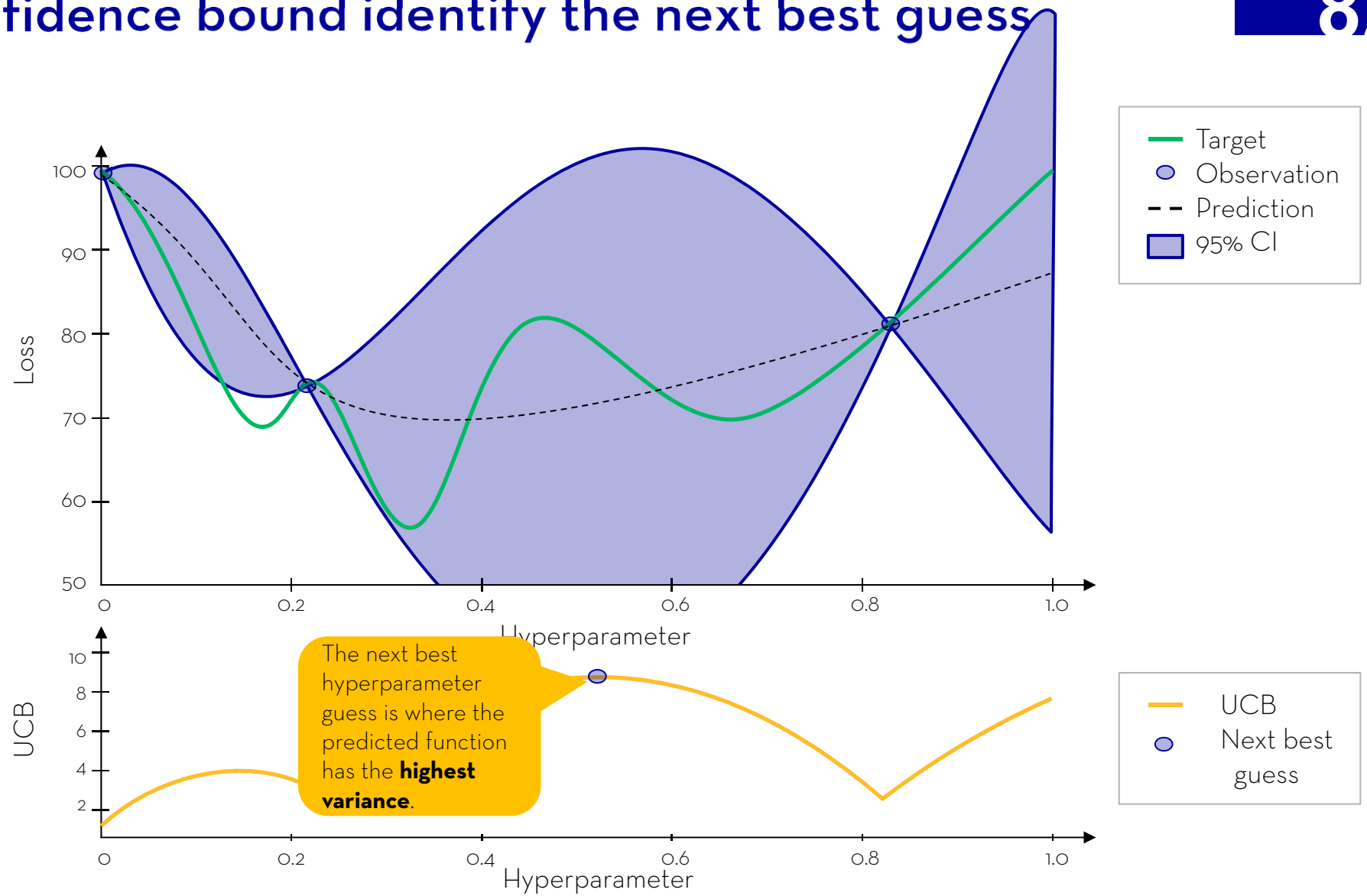
Bayesian optimization: Train a new classifier with the next best guess as the new hyperparameter



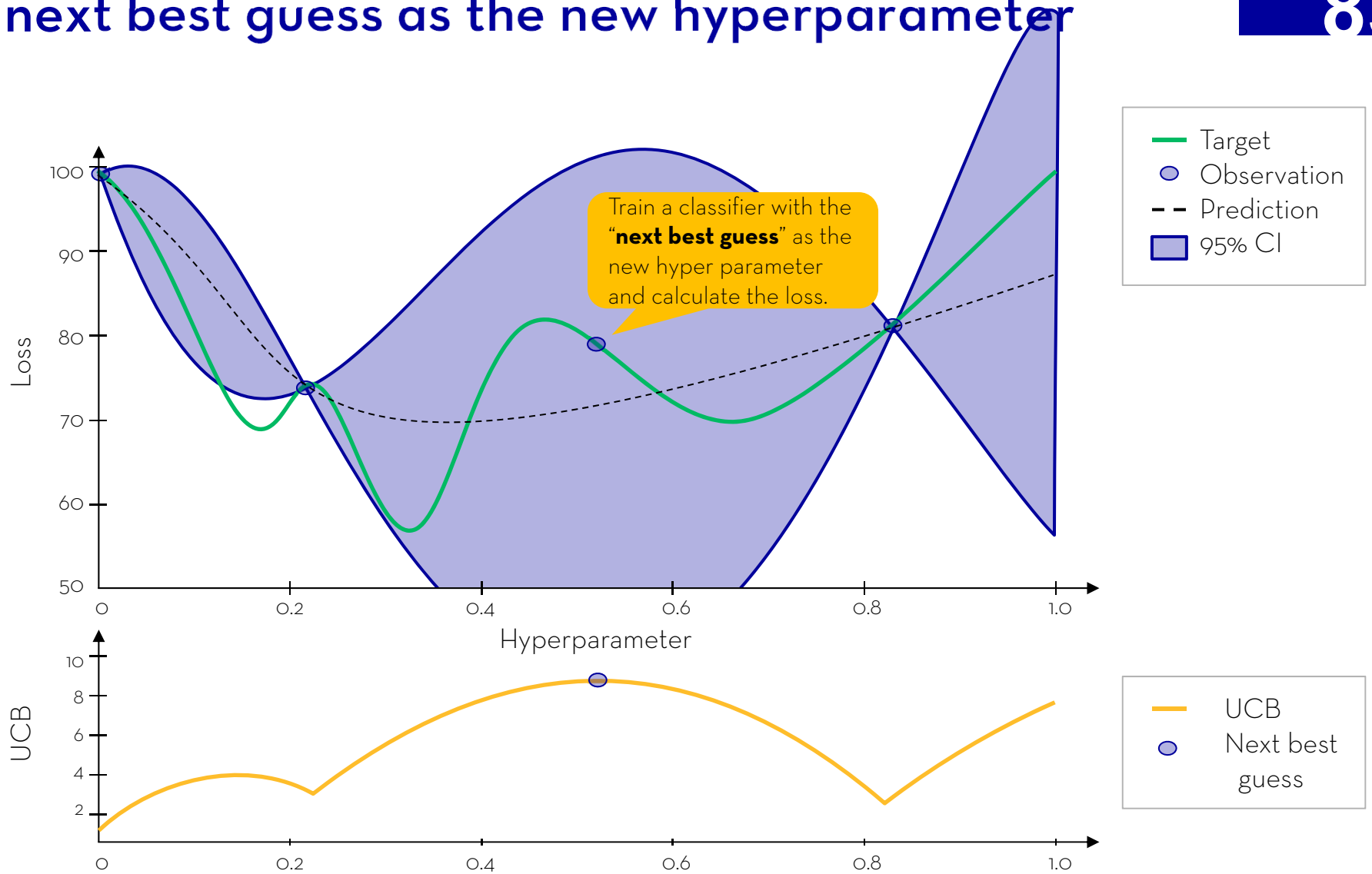
Bayesian optimization: Update the prediction using the new observation



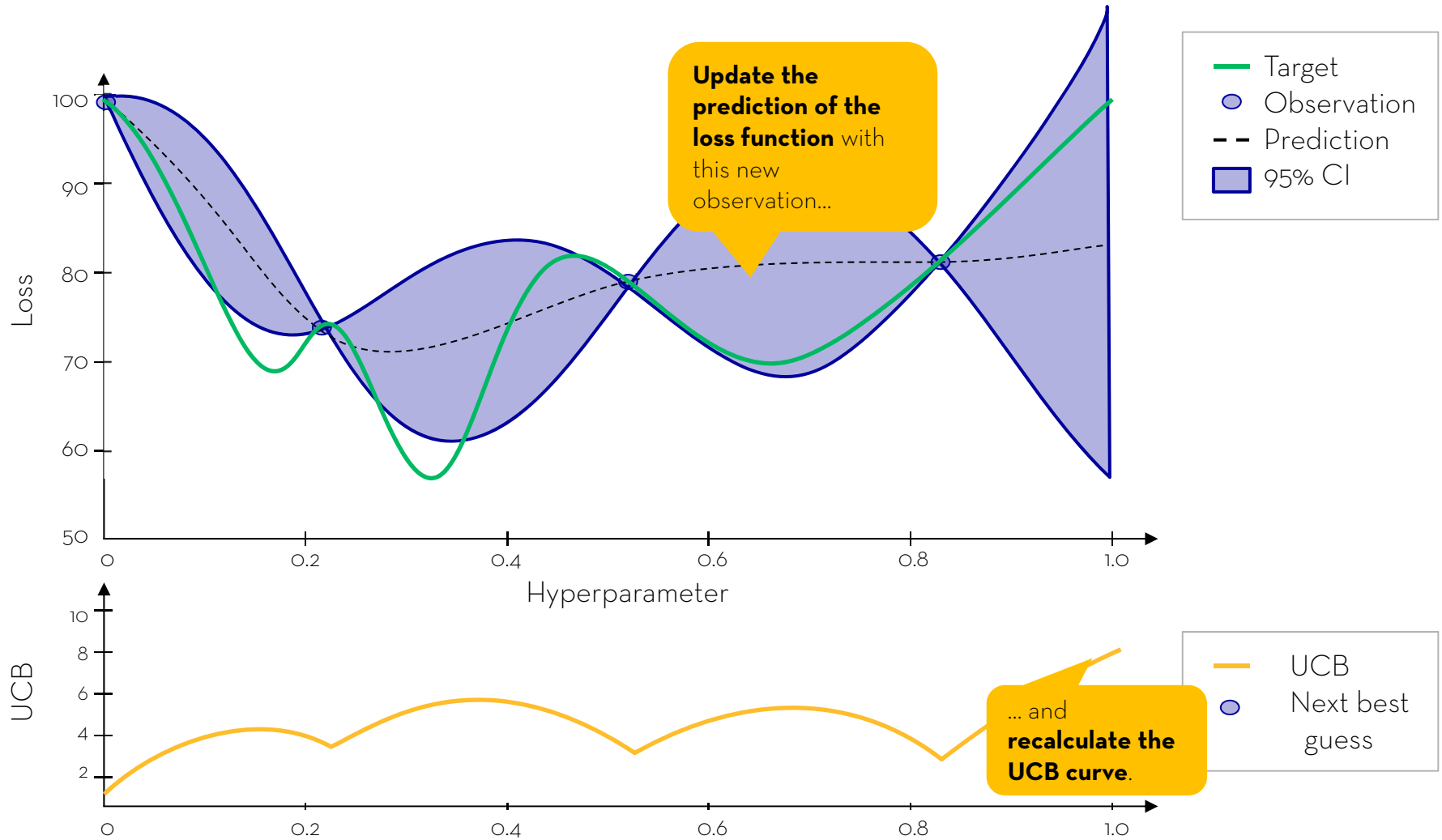
Bayesian optimization: Based on the upper confidence bound identify the next best guess



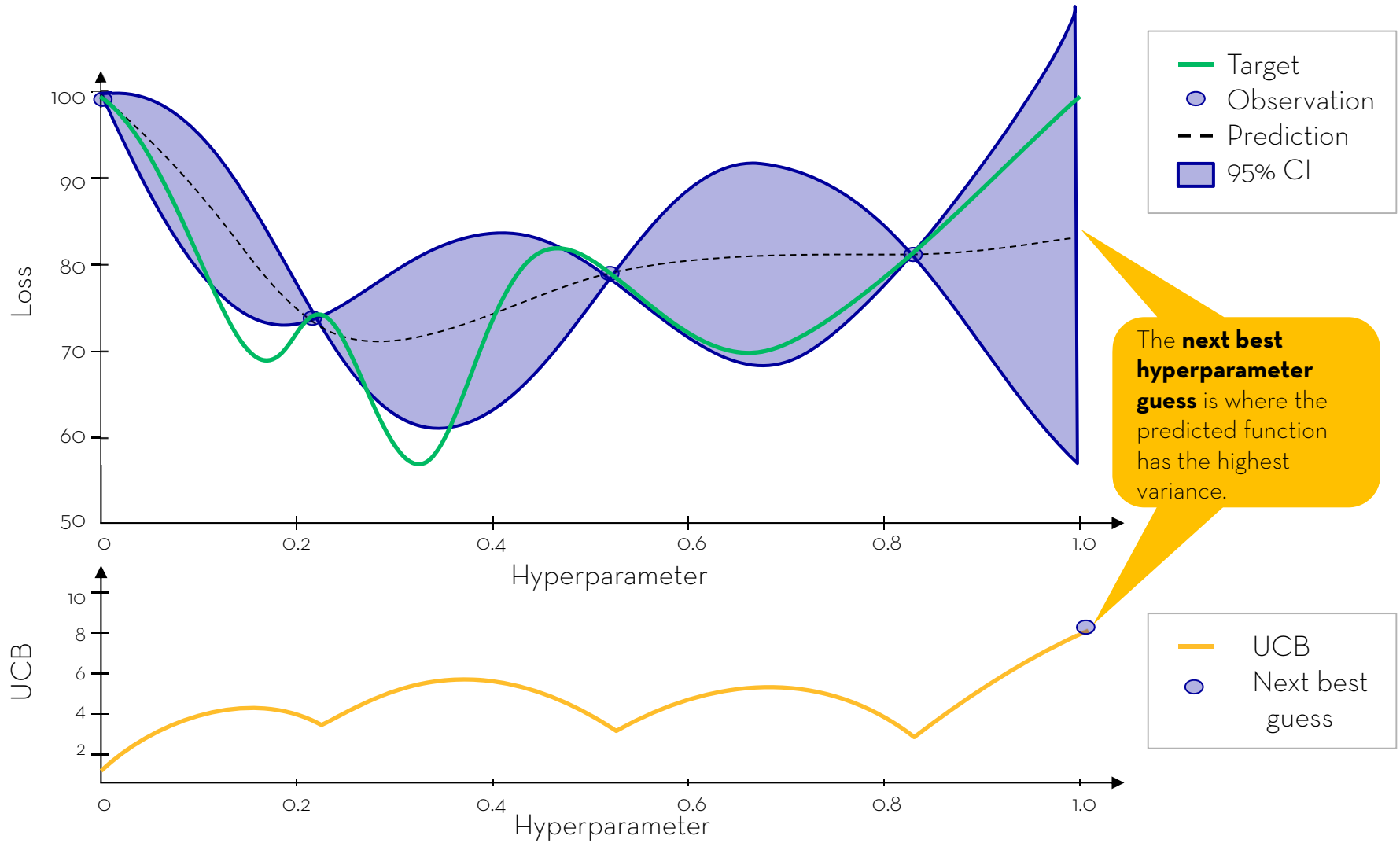
Bayesian optimization: Train a new classifier with the next best guess as the new hyperparameter



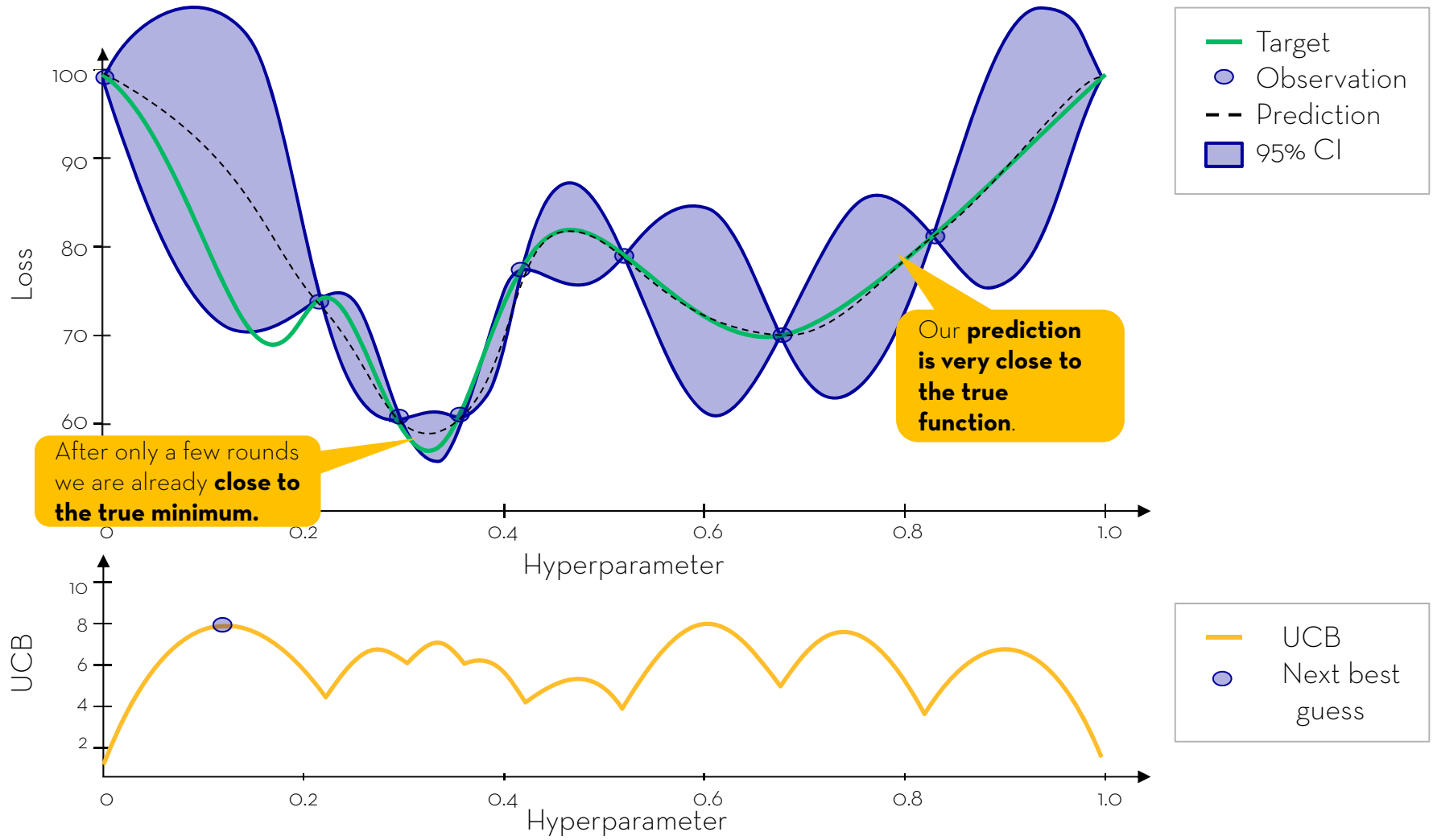
Bayesian optimization: Update the prediction using the new observation



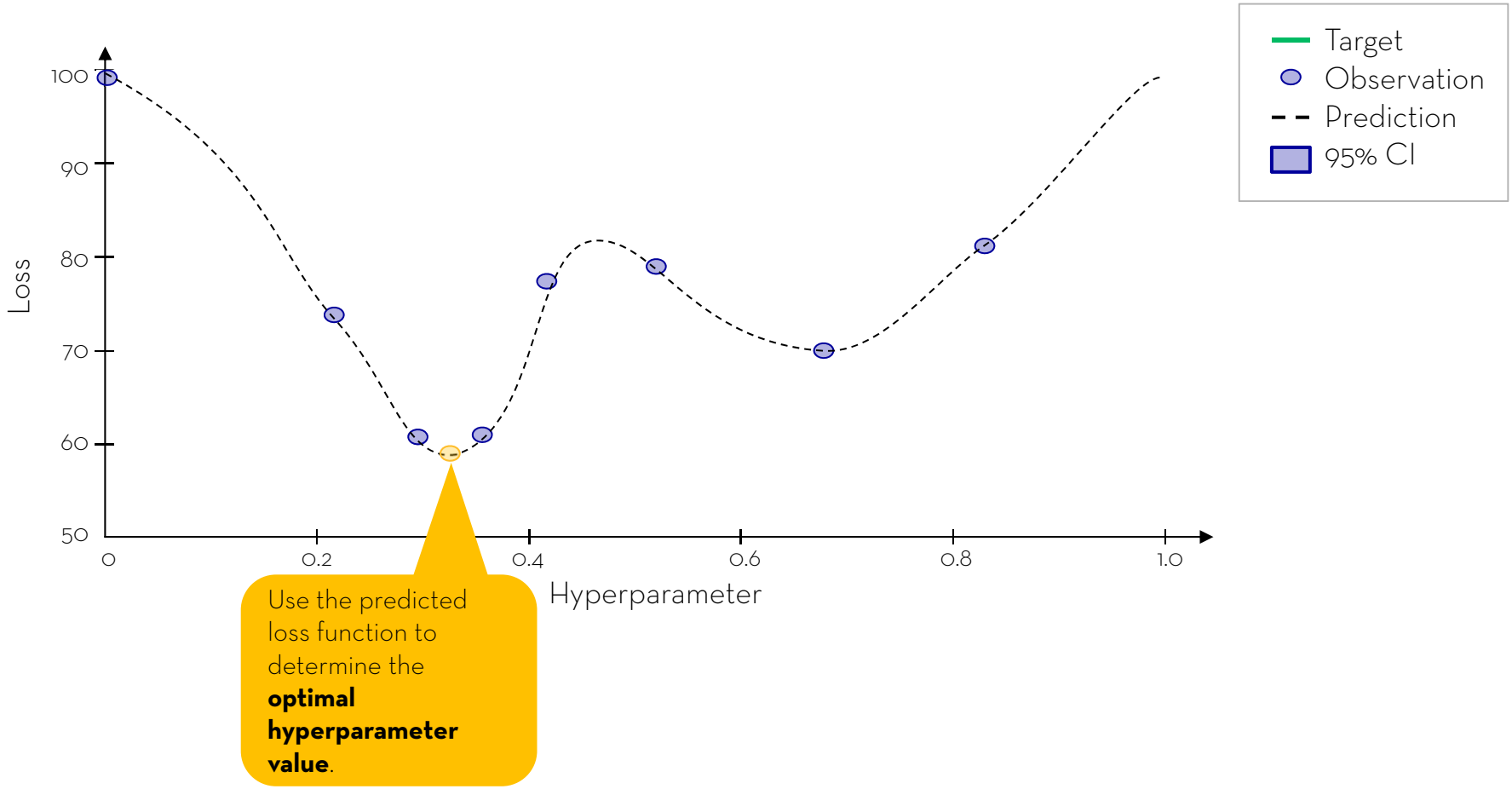
Bayesian optimization: Based on the upper confidence bound identify the next best guess



Bayesian optimization: Stop the algorithm after a few iterations

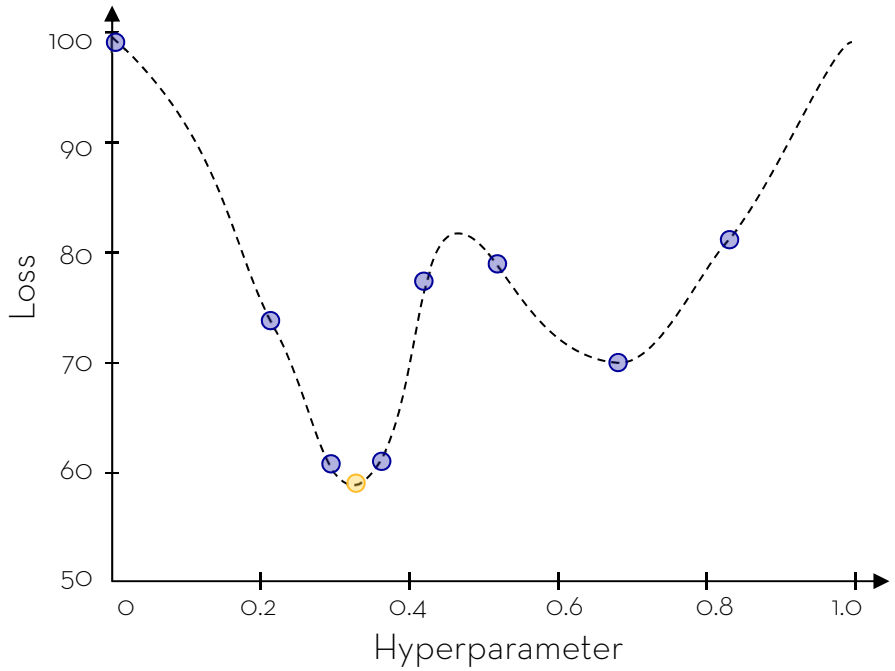


Bayesian optimization: Selecting the optimal hyperparameter

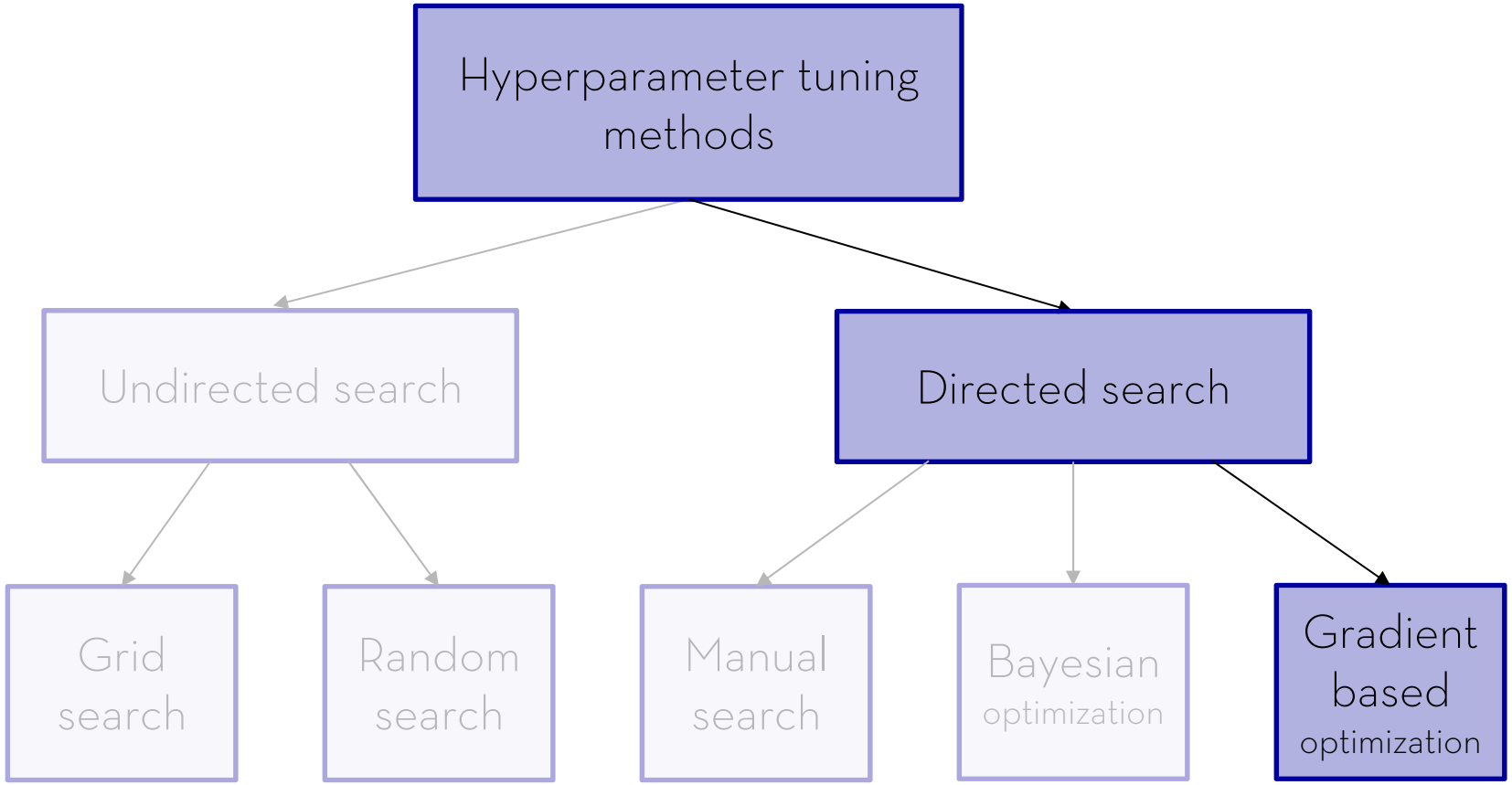


Bayesian optimization: why doesn't everyone use Bayesian optimization?

- Difficult to parallelize since experiments are run sequentially.
  - We have to train a model. Wait until it finishes. Repeat.
- Since it has its own parameters the results are fragile.
  - The parameters can be adjusted to explore or extrapolate, but there is no guideline when to use which.



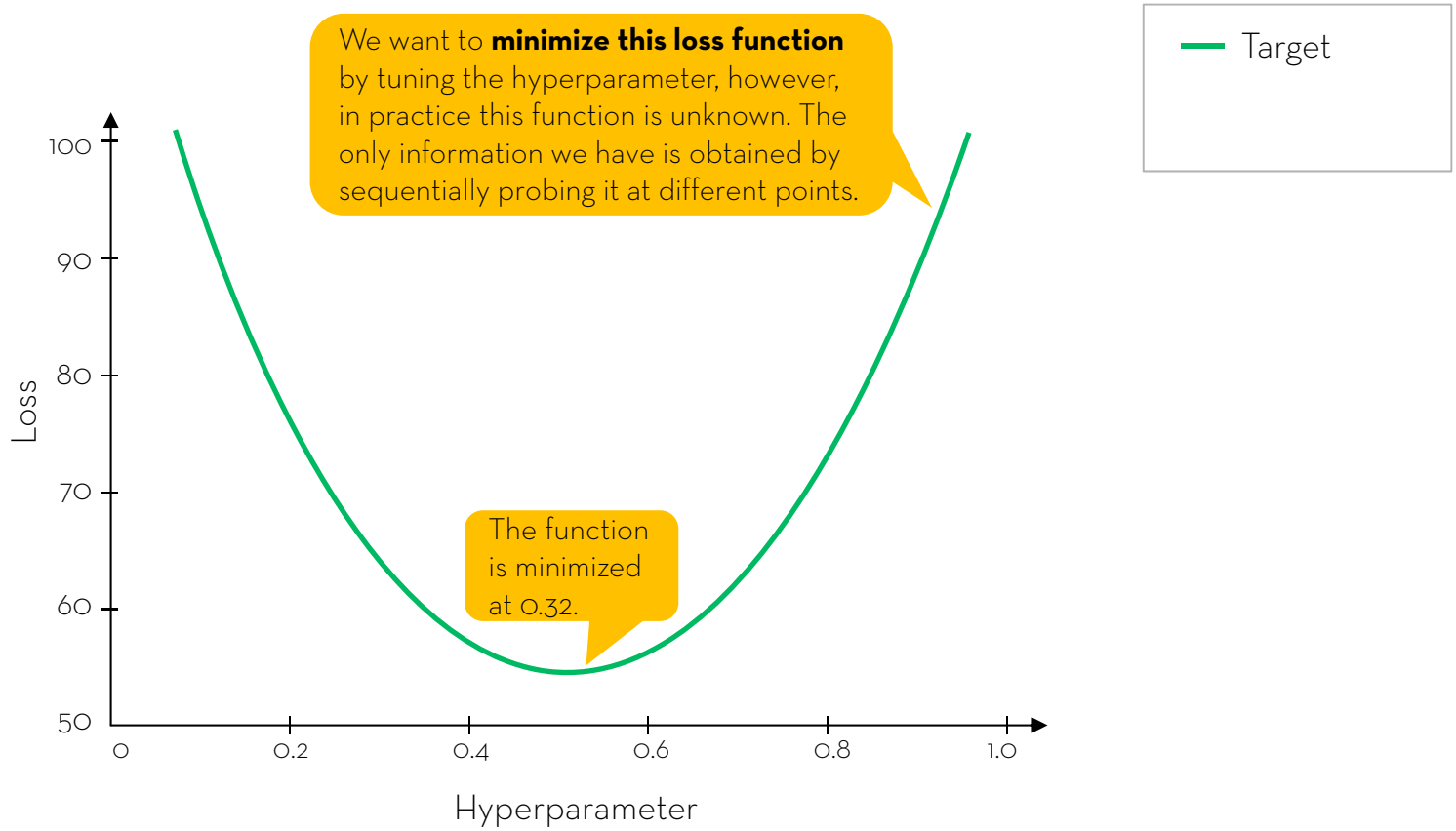
There are various ways to conduct hyperparameter optimization



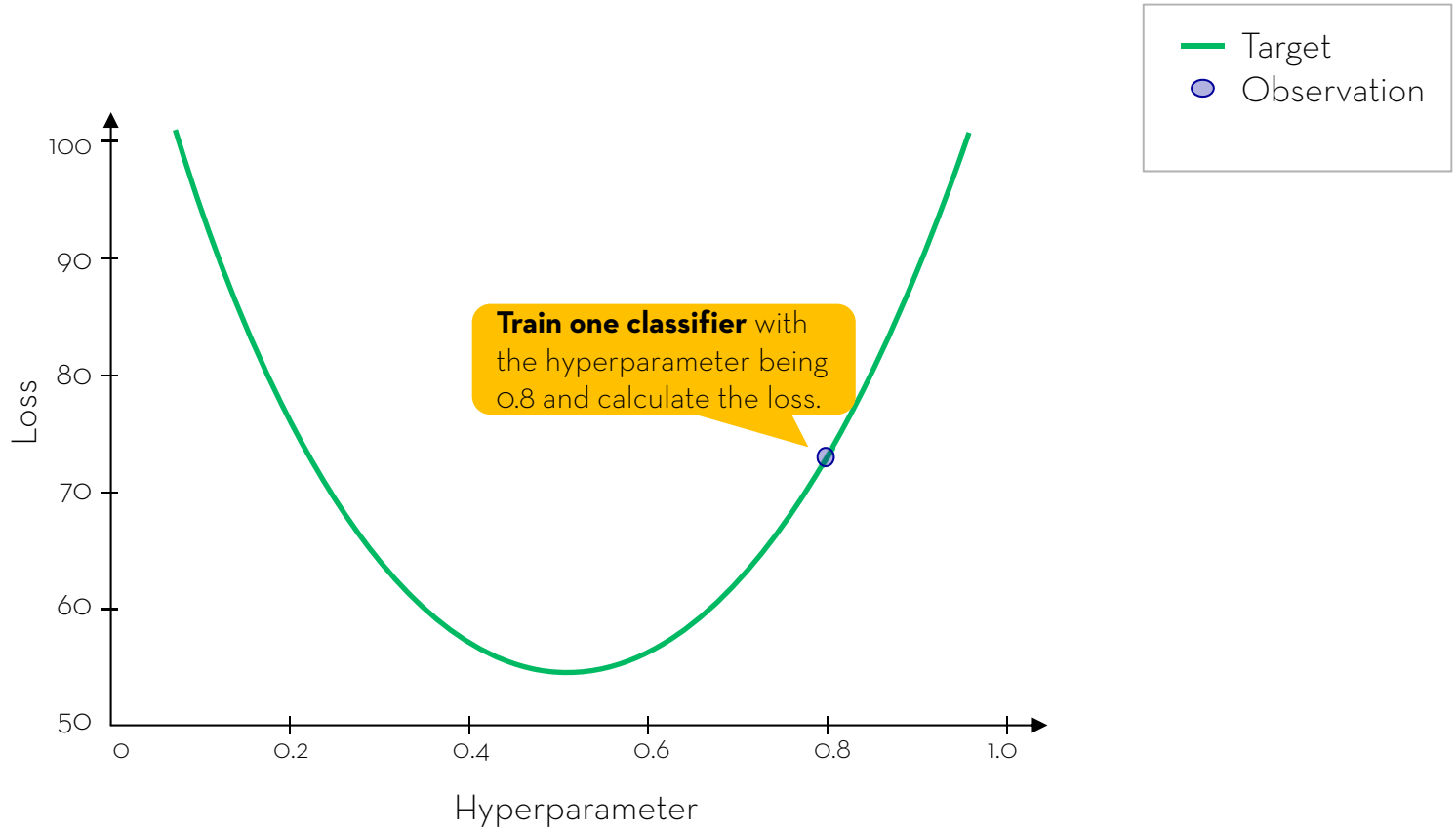
https://en.wikipedia.org/wiki/Hyperparameter_optimization

http://stats.stackexchange.com/questions/95495/guideline-to-select-the-hyperparameters-in-deep-learning

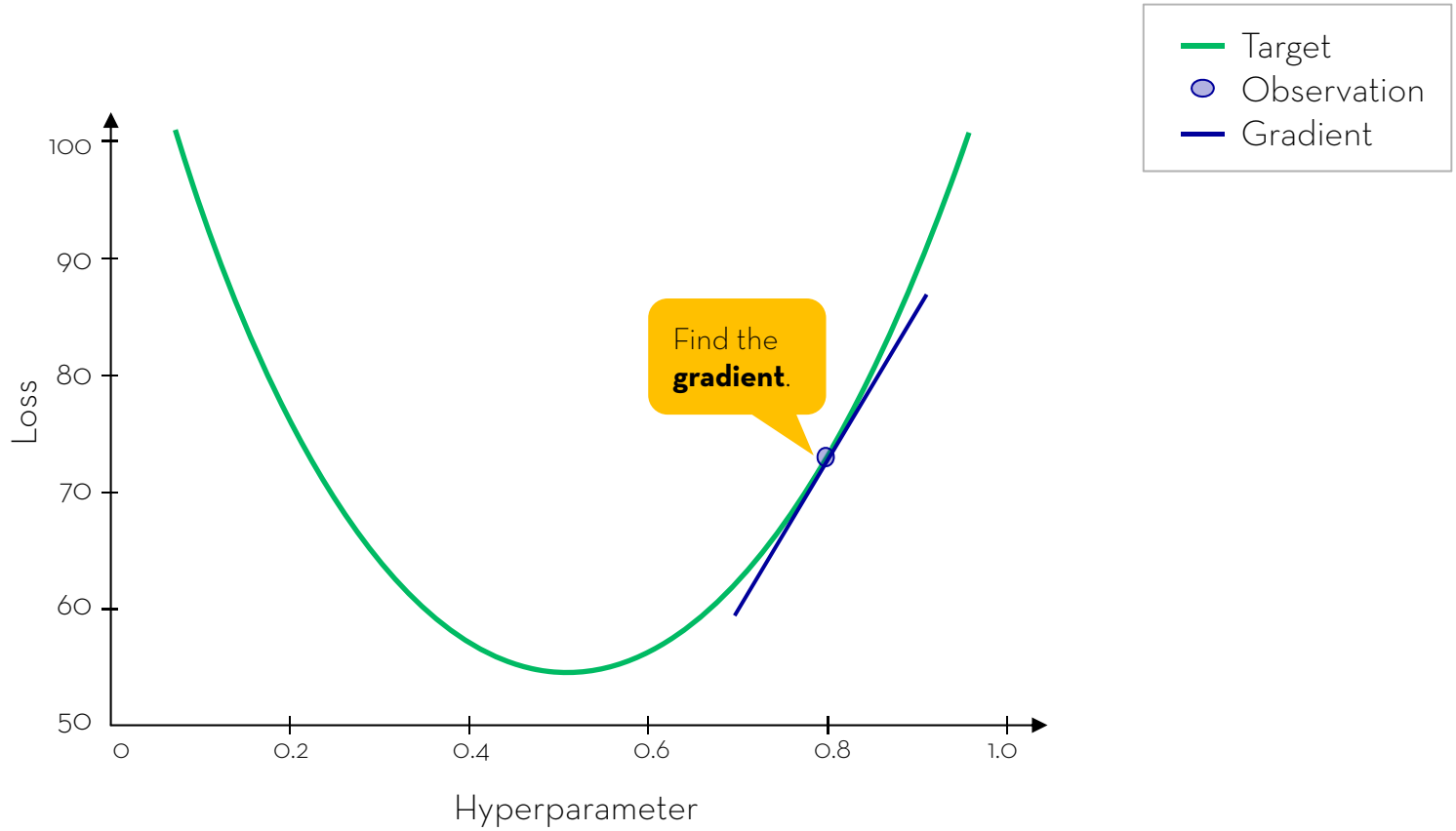
Gradient-based optimization: We want to optimize one hyperparameter



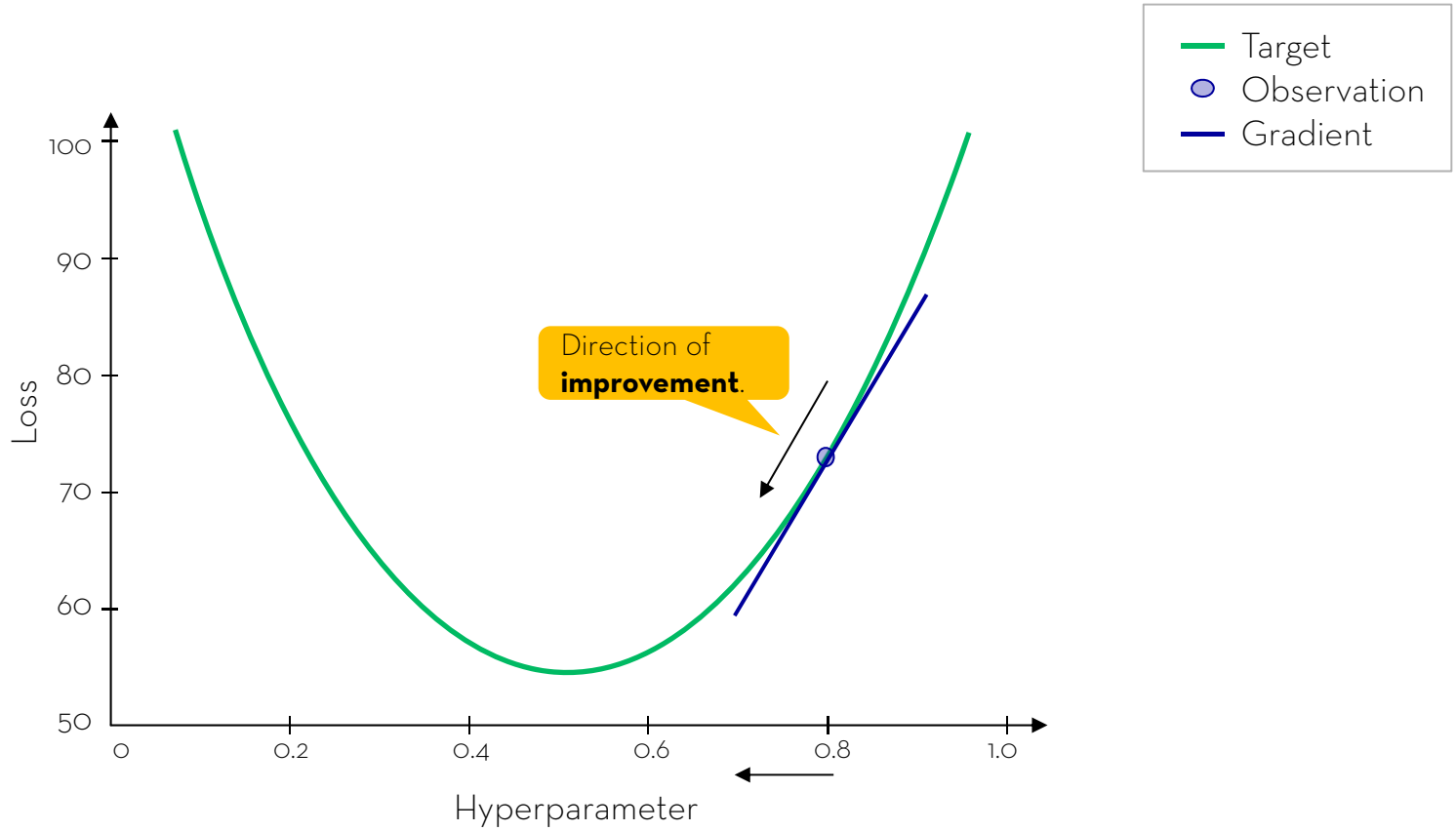
Gradient-based optimization: Train a classifier for an initial hyperparameter value



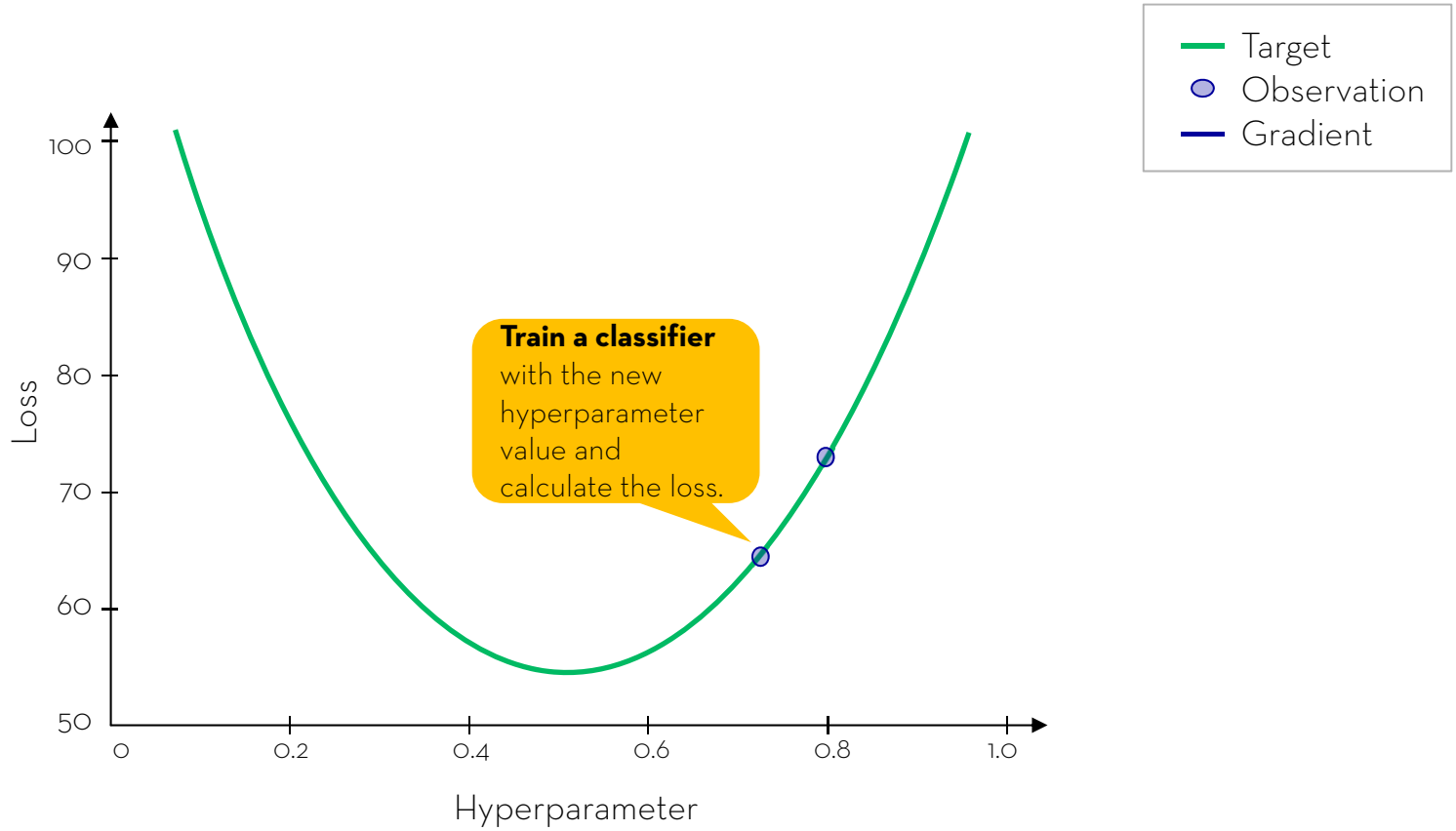
Gradient-based optimization: Find the gradient with respect to the hyperparameter



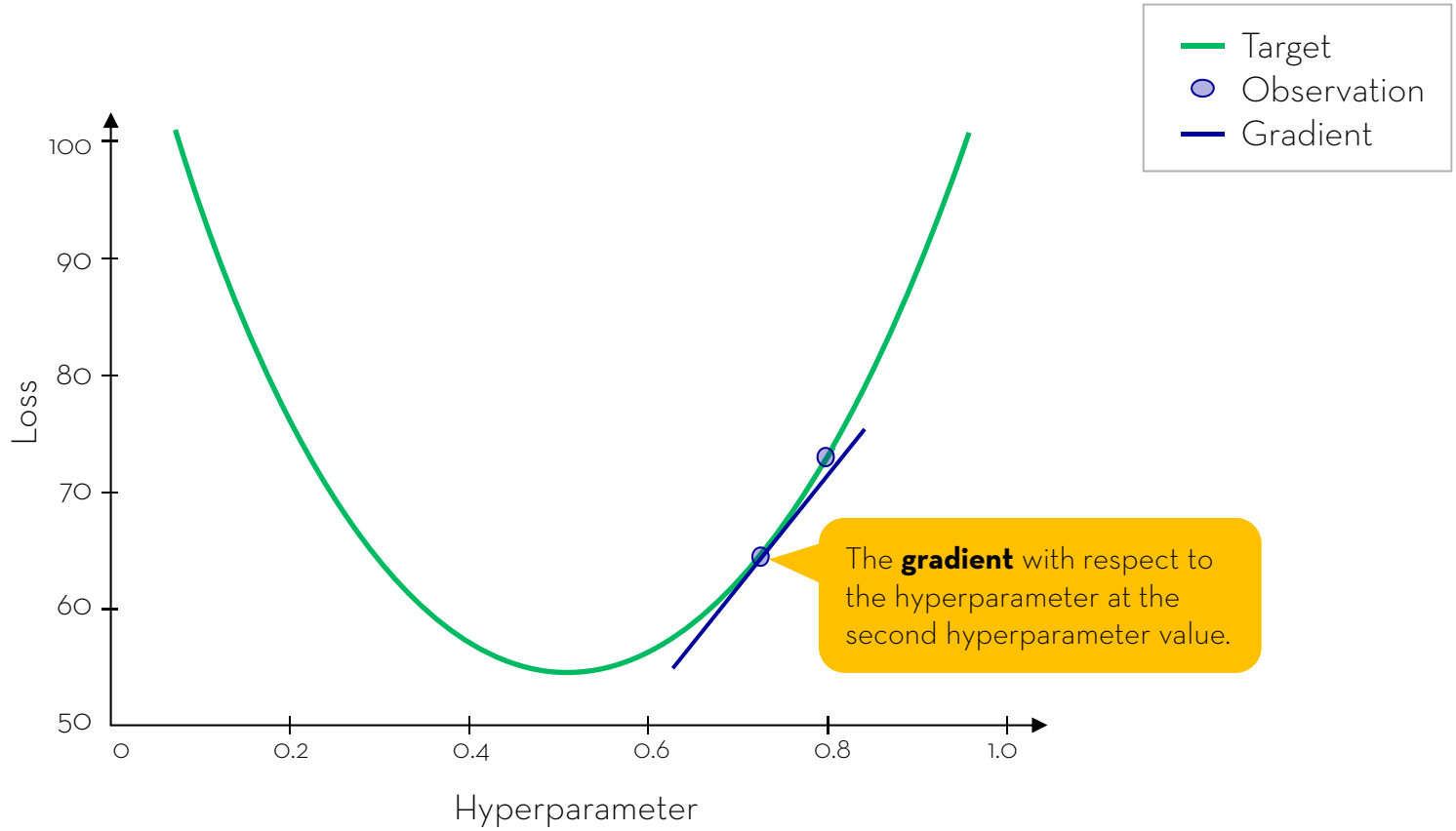
Gradient-based optimization: Descend down the gradient to the next hyperparameter value



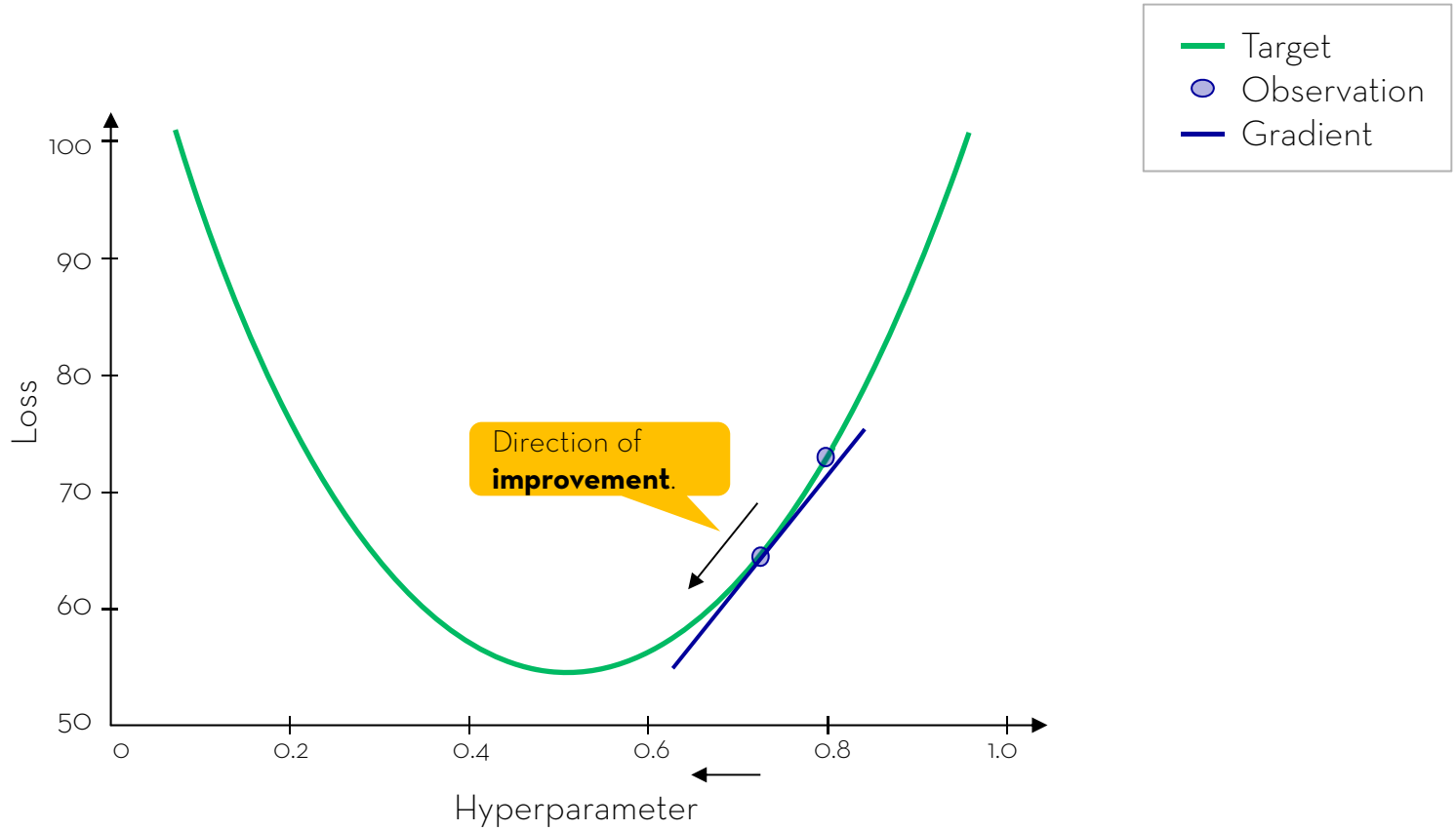
Gradient-based optimization: Train a new classifier with the next hyperparameter value



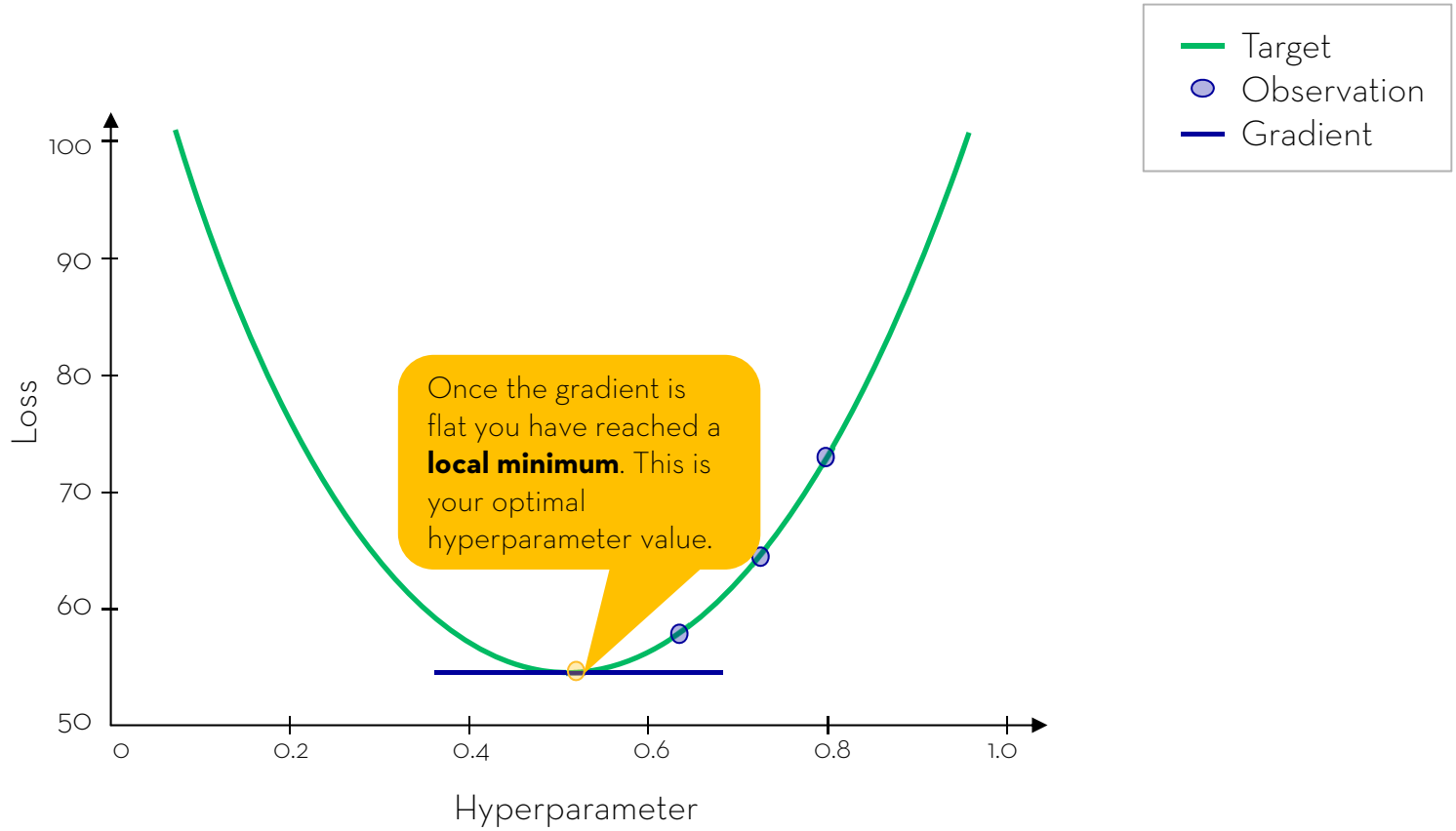
Gradient-based optimization: Find the gradient with respect to the hyperparameter



Gradient-based optimization: Descend down the gradient to the next hyperparameter value



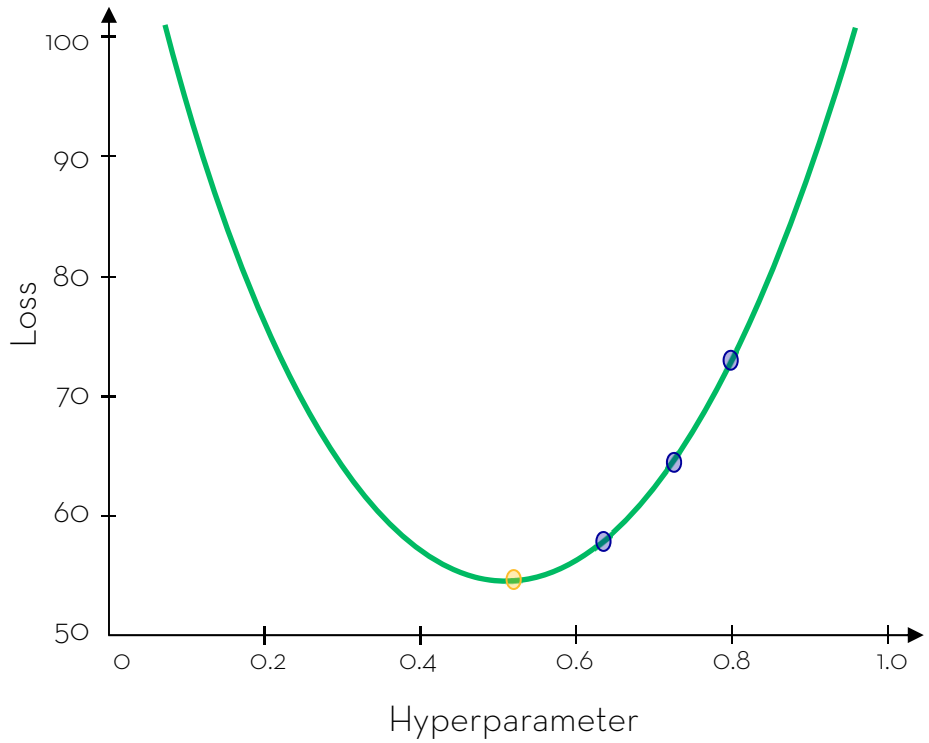
Gradient-based optimization: Continue until the gradient is zero



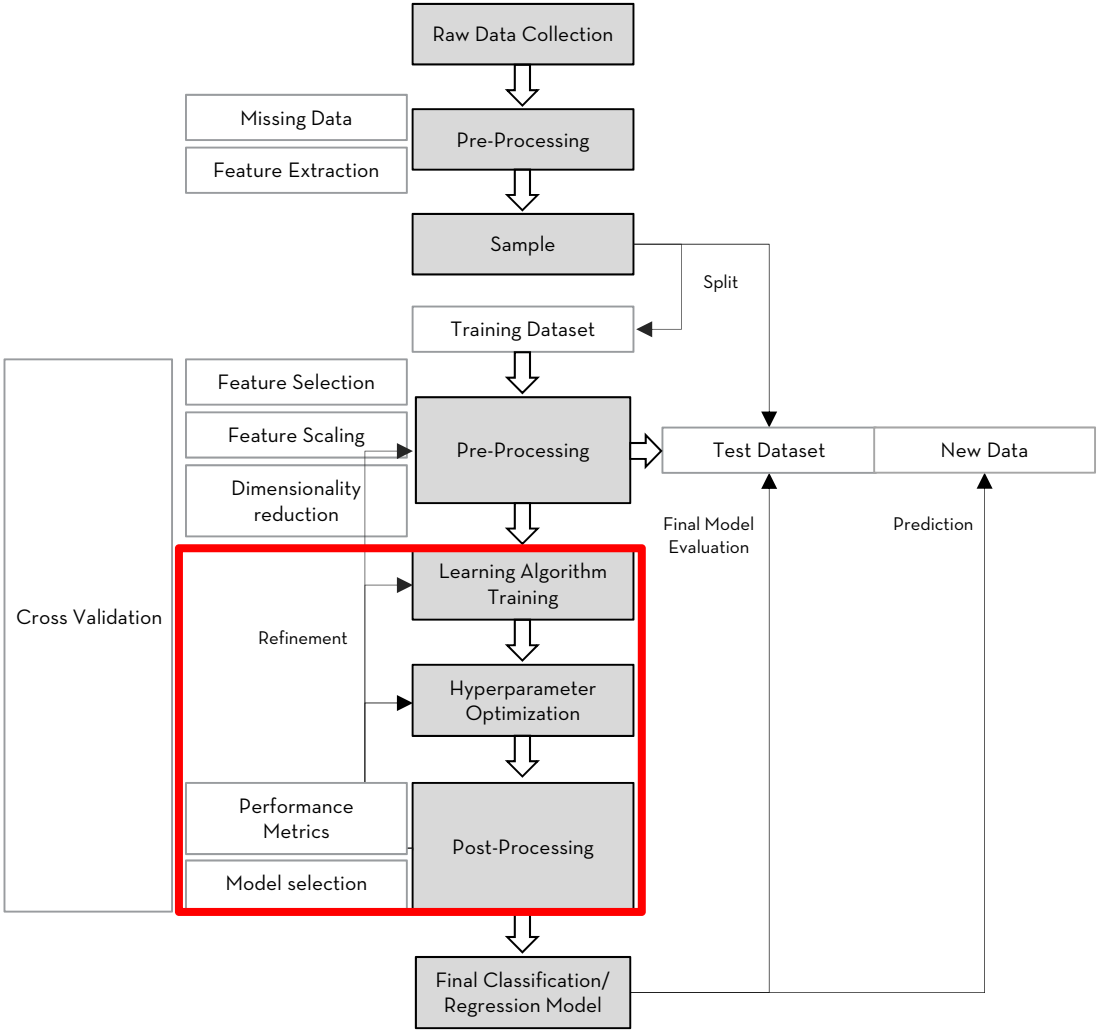
Gradient-based optimization is

- This method only works for some machine learning algorithms (neural network and SVM).
- Disadvantages:
  - It's possible to get stuck in a local optimum or jump over the true depending on step size.

    → Try different starting points.



When to actually do hyperparameter optimization?



http://sebastianraschka.com/Articles/2014_intro_supervised_learning.html

Exercise
Apply hyperparameter optimization to KNN

Hint: check out the argument `tuneGrid` from the `caret` package.

1. What are the hyperparameters of the kNN method?
2. Improve the predictive accuracy of a kNN model by optimizing the hyperparameters through a grid search: Evaluate the models performance for number of neighbors between 1 and 30. Use 50 resampling iterations. Estimate the model on 80% of the data while using the remaining 20% for evaluating the model's performance.

Hint: set the training percentage and number of iterations in the `trControl` argument.

Ensemble learning and its advantages

Dr. Markus Meierer - markus.meierer@business.uzh.ch | Dr. Margot Löwenberg - margot.loewenberg@business.uzh.ch

If the final prediction is made with multiple models, we talk about ensemble learning

Final predictions made with	Training strategy: one classifier	Training strategy: multiple classifiers
one model	Hyperparameter tuning	Classifier selection
multiple models	Bagging Boosting	Voting Averaging Stacking

Ensembles

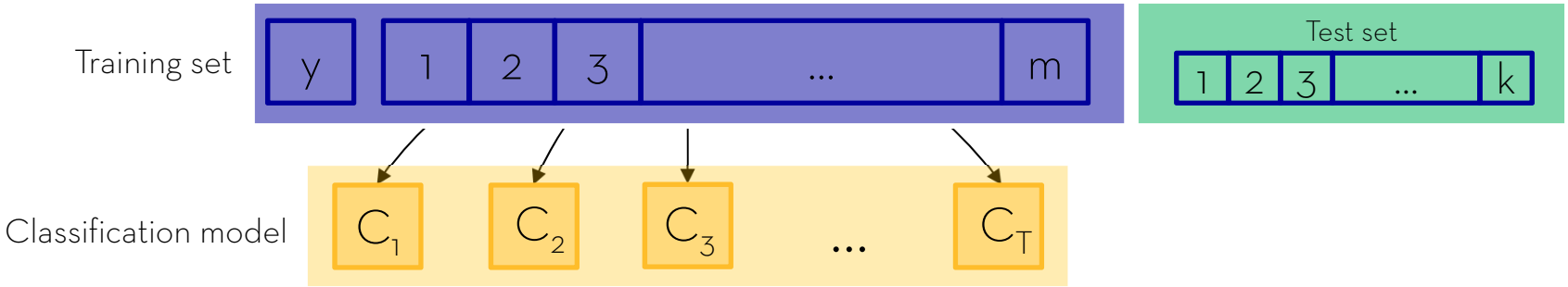
Theoretically, ensembles are based on the concept of the "wisdom of the crowds"

The aggregation of information in groups and thereby, derived decisions, are often better than the decision made by any single member of the group.

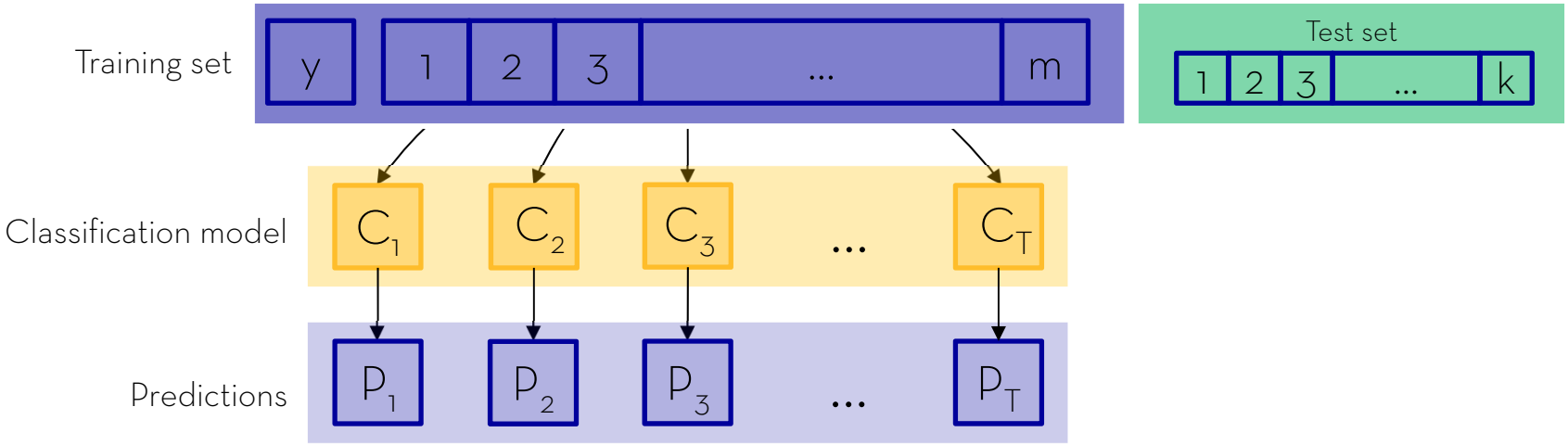
Ensemble learning lecture 10, A. Maki, J. Sullivan

General principle: Split the data into a training and test set

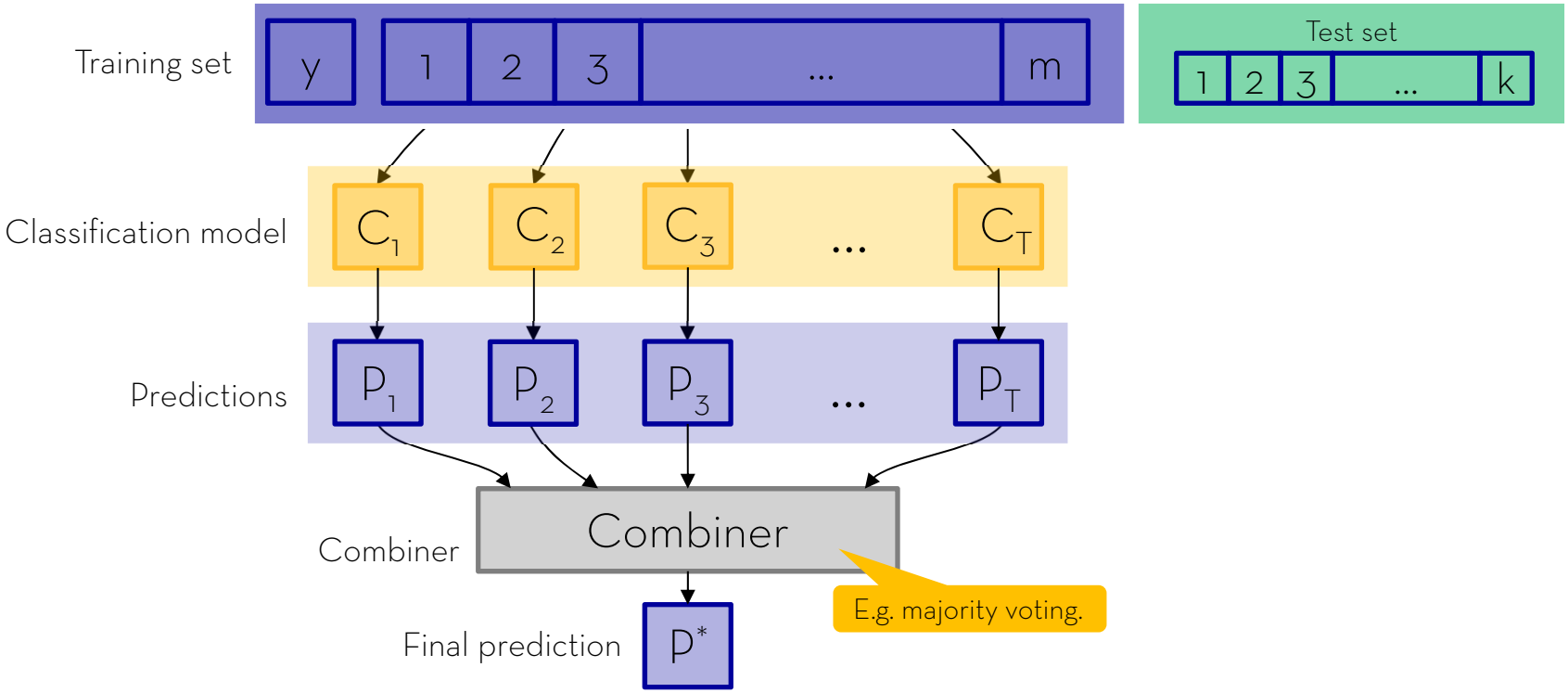
General principle: Build classifiers on the whole training set



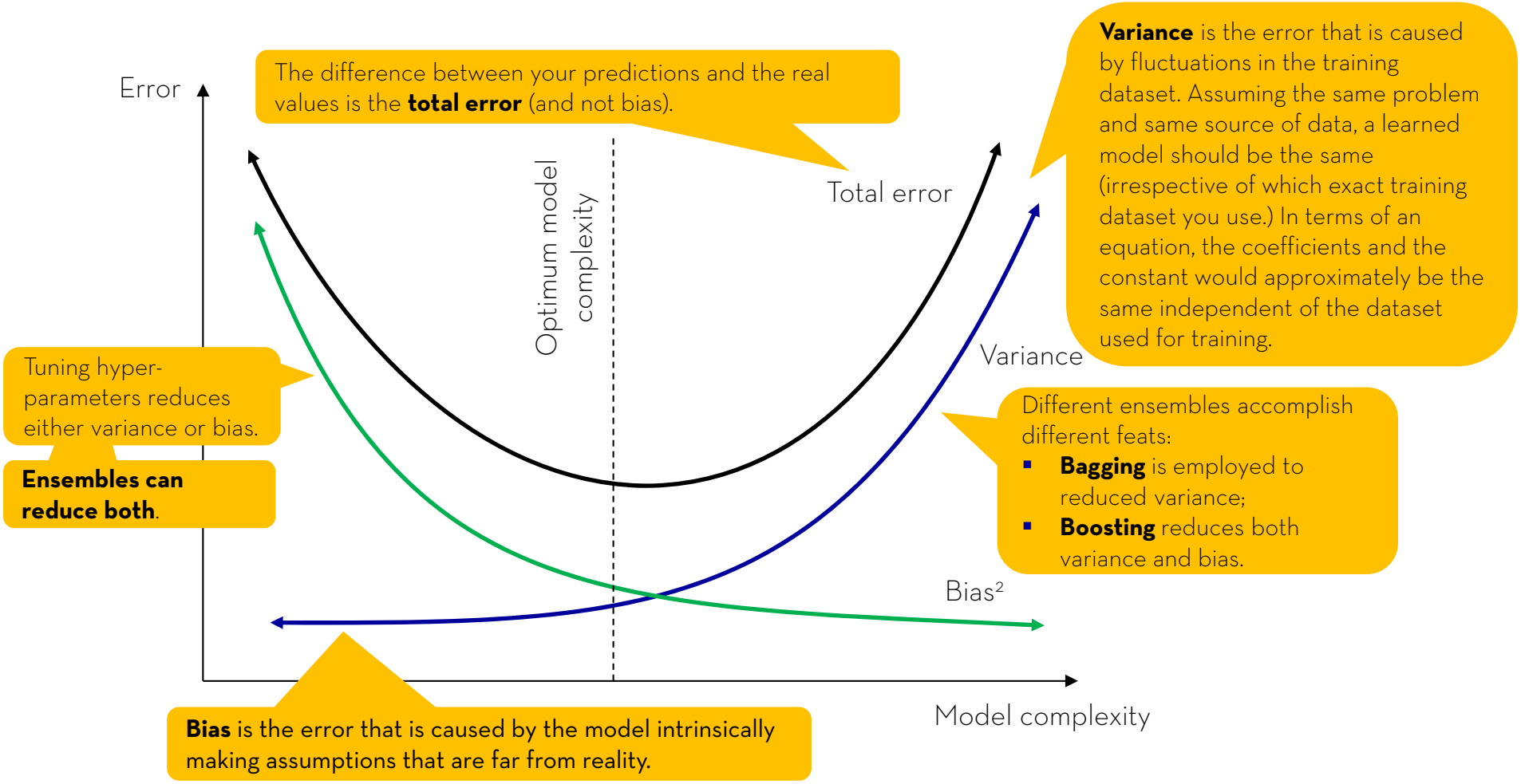
General principle: Generate predictions from each classifier



General principle: Combine the predictions to obtain a final prediction



Statistically, ensembles have two advantages: (1) bias reduction and (2) variance reduction



https://codesachin.wordpress.com/2015/08/05/on-the-biasvariance-tradeoff-in-machine-learning/comment-page-1/

Ensembles have 4 main building blocks

Diversity generator	Ensemble size	Members dependency	Combiner

Rokach (2009)

Ensembles have 4 main building blocks

Diversity generator	Ensemble size	Members dependency	Combiner

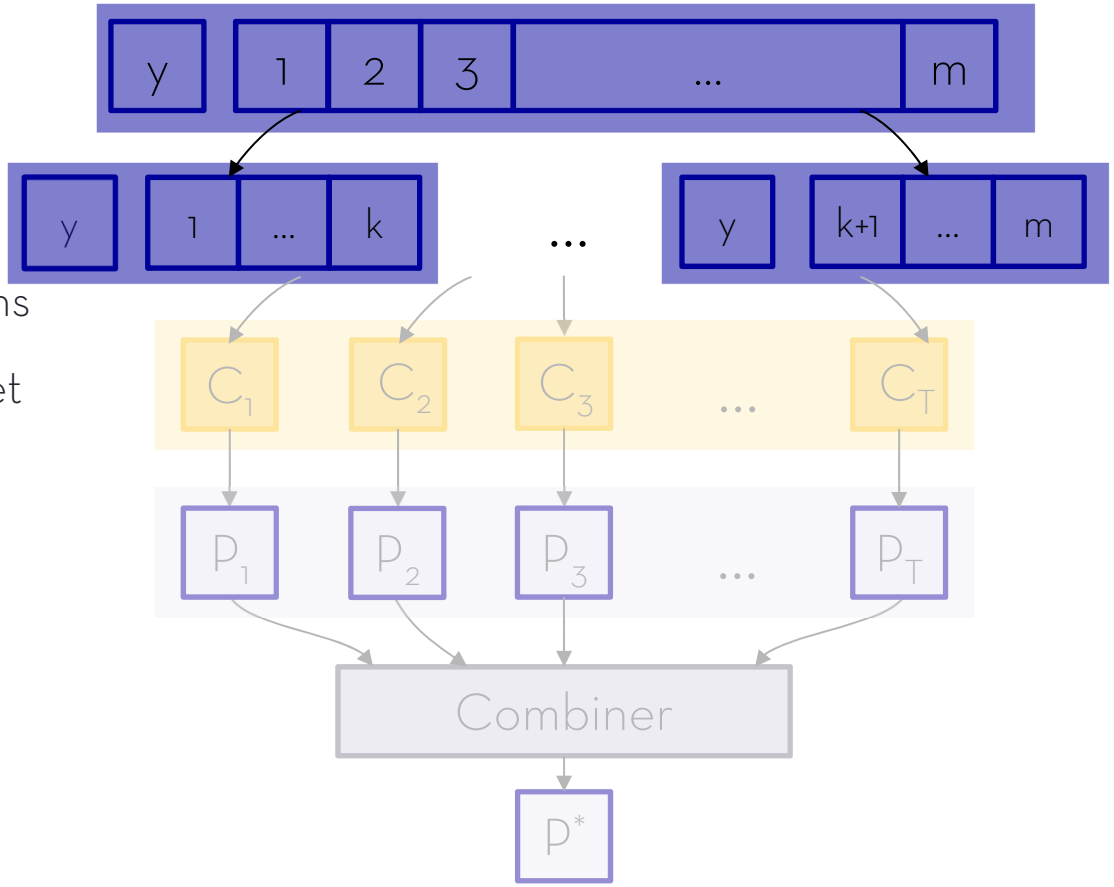
Rokach (2009)

(1) Generating diversity by manipulating the training data

(a) Split the training data into smaller training samples, e.g.:

- **Horizontal**:
  Each partitioned data set contains the same feature set and a subset of the observations.

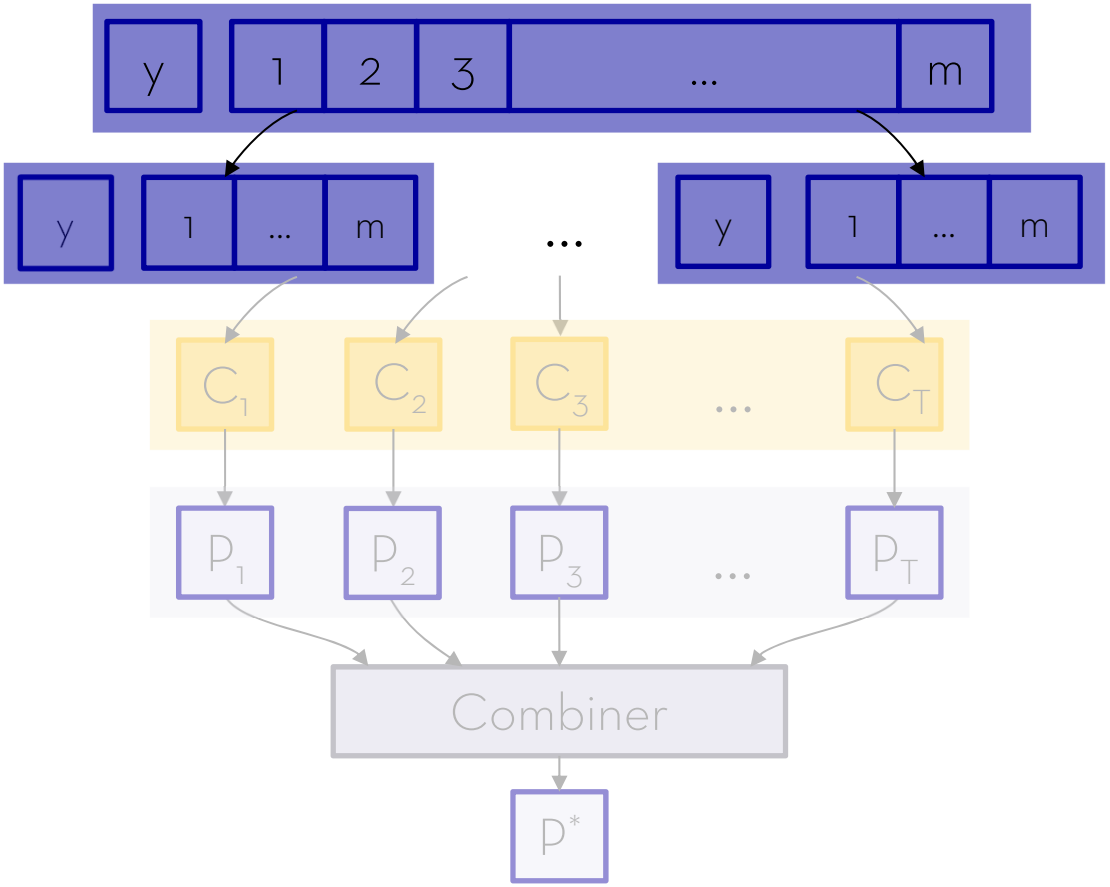
- **Vertical**:
  Each partitioned data set is a subset of the features (variables) and all the observations.



(1) Generating diversity by manipulating the training data

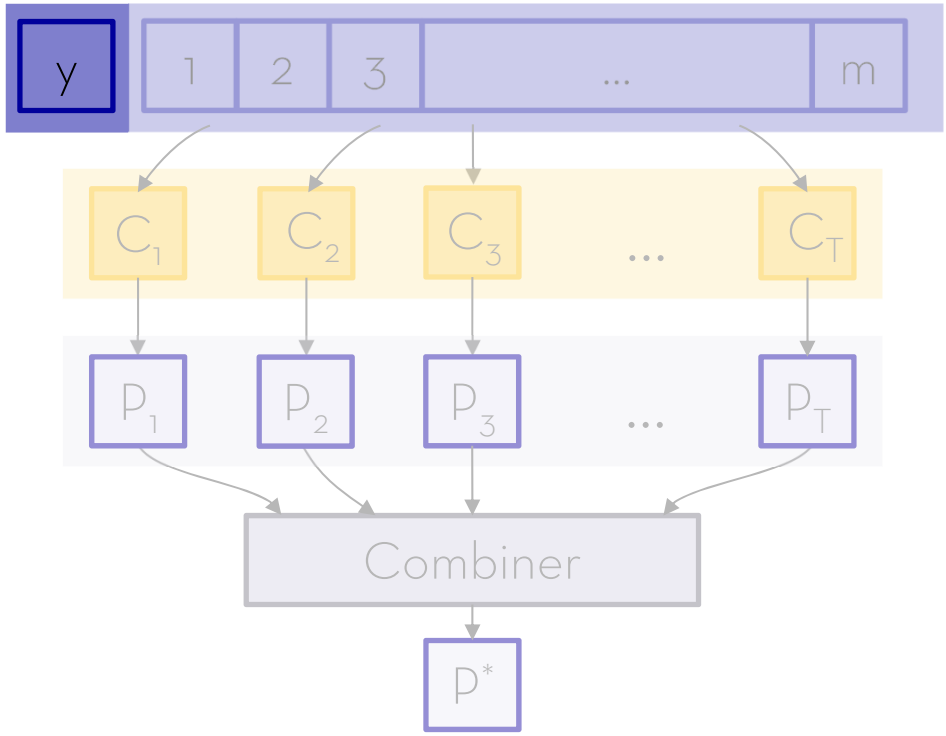
(b) Train each classifier on a **different variation of the training data**, e.g.:

- Sampling without replacement
- Sampling with replacement (bootstrapping)
- Combine original training data with synthetic training data (e.g. DECORATE algorithm)



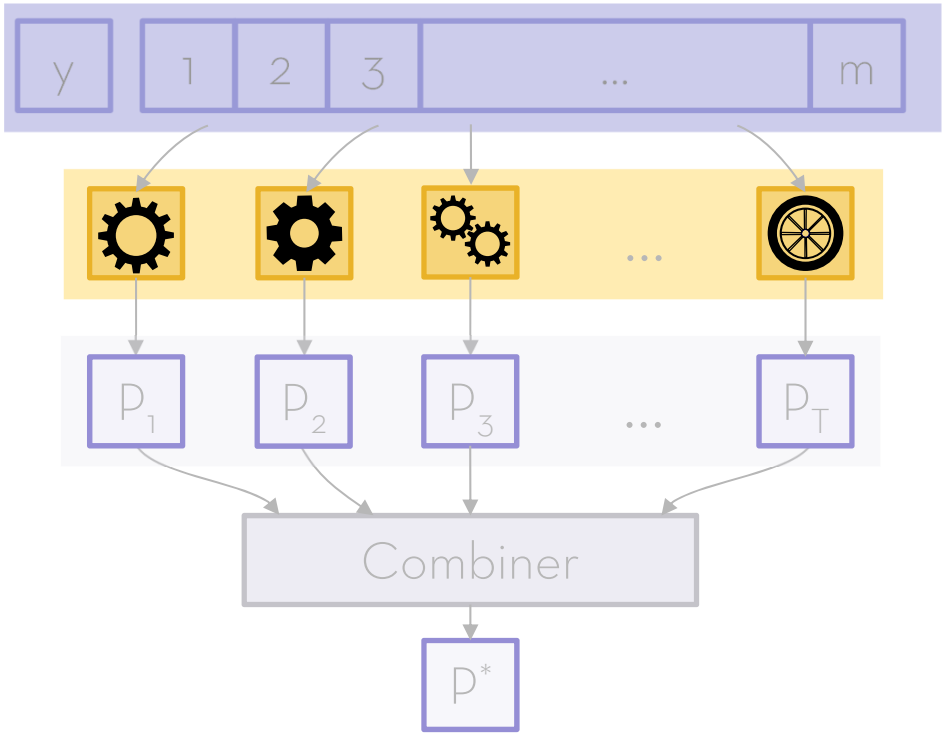
(1) Generating diversity by manipulating the training data

(c) **Change the target representation**, e.g. by using binary representations of the class membership variable in a multiclass classification setting.



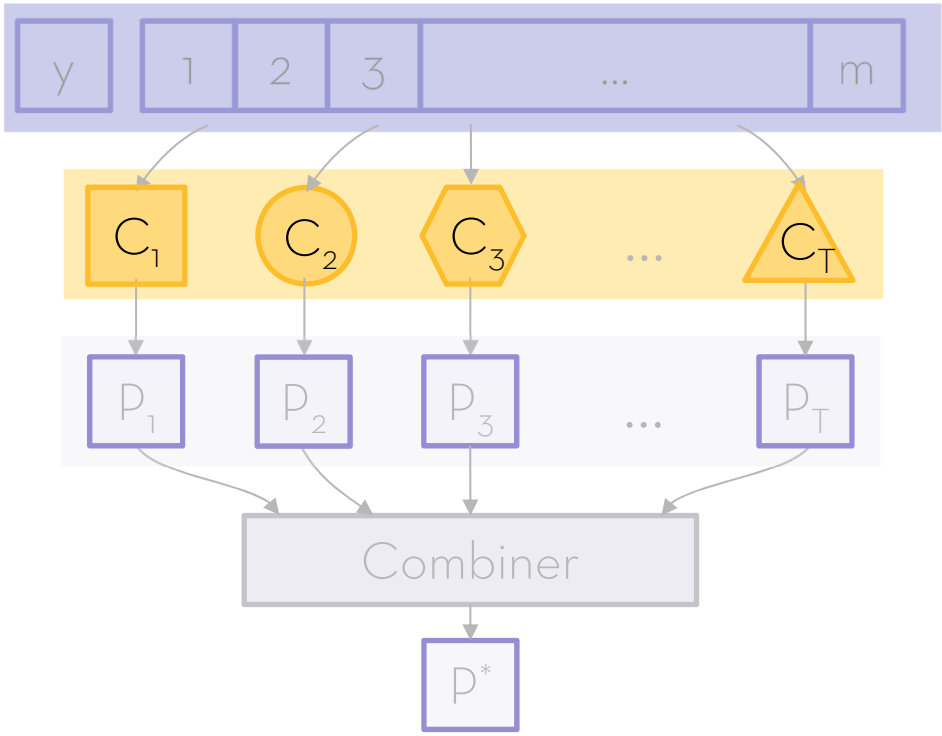
(2) Generating diversity using different configurations of the same classifier

Manipulate how the same classifier learns by changing its **hyperparameters**.



(3) Generating diversity by using multiple classifiers

Use **multiple classifiers**.



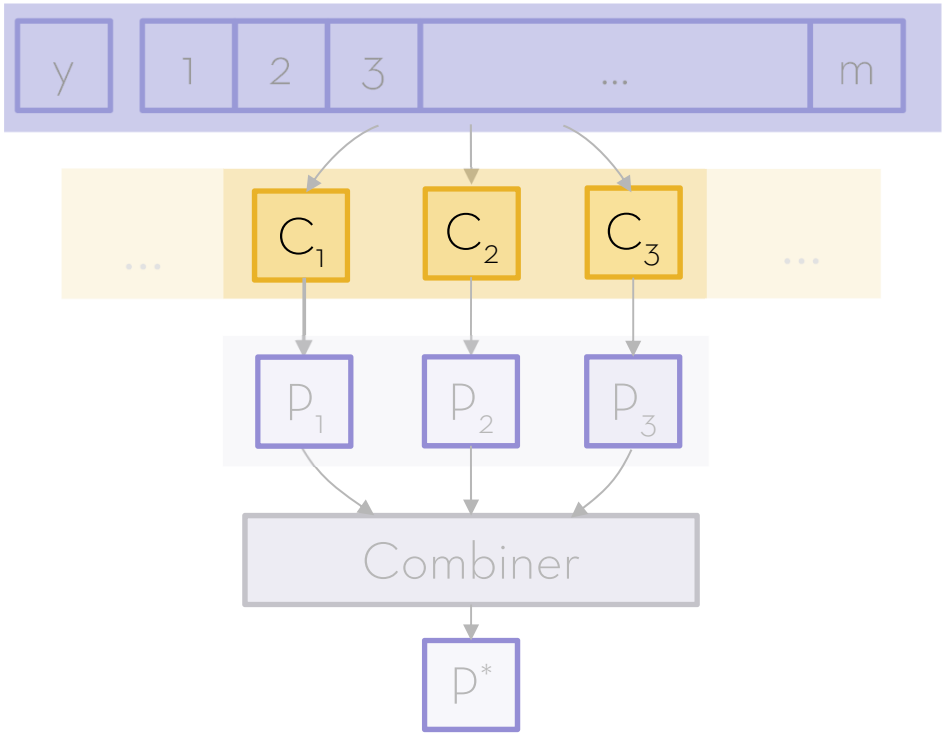
Ensembles have 4 main building blocks

Diversity generator	Ensemble size	Members dependency	Combiner

The ensemble size can be determined either before, during, or after training

Select the number of classifiers used

- prior to model estimation.
- during model estimation (e.g. random forests which rely on out of bag procedure to determine if enough trees have been generated).
- after model estimation (e.g. based on classification performance).



The ensemble size is important due to the trade off between computational cost and accuracy

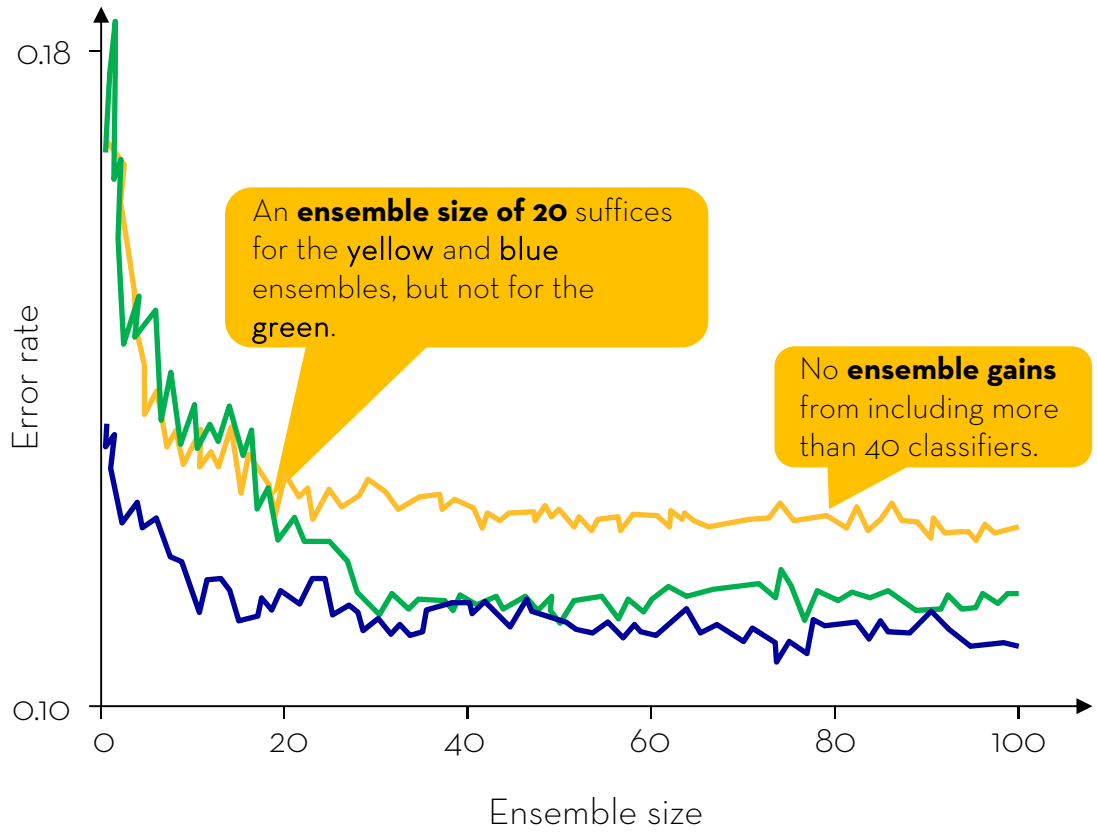


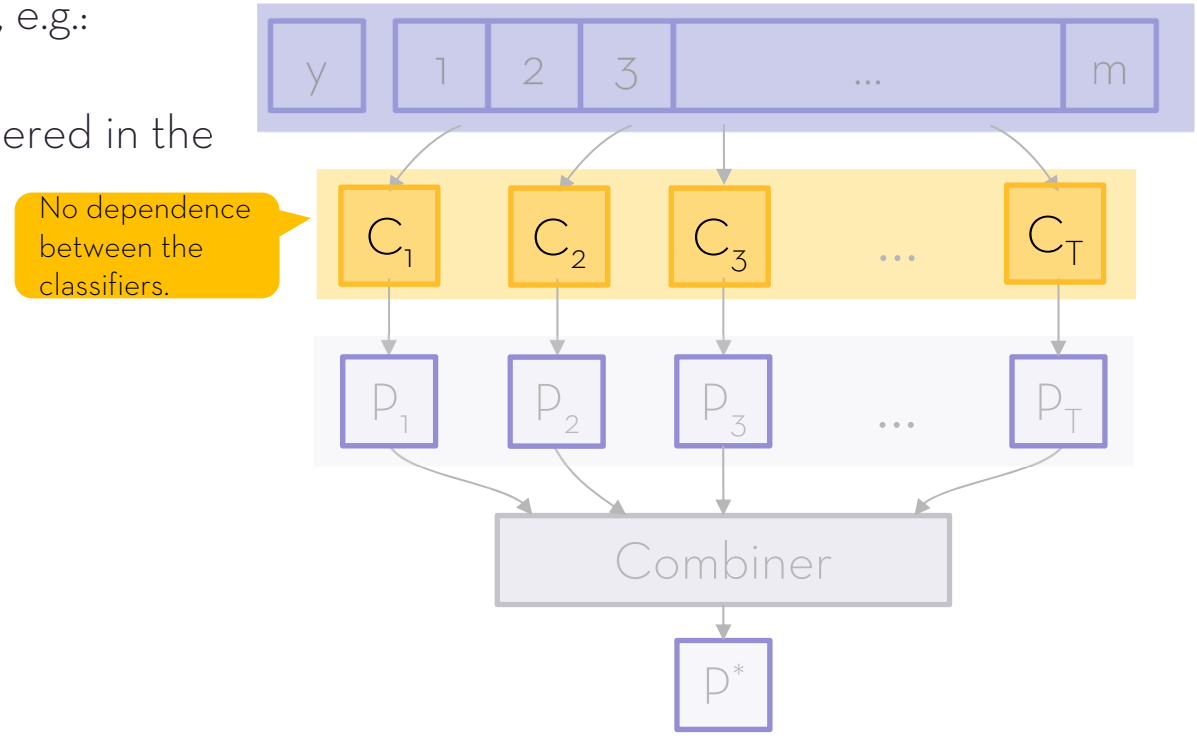
Figure adapted from https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume11/opitz99a-html/node10.html

Ensembles have 4 main building blocks

Diversity generator	Ensemble size	Members dependency	Combiner

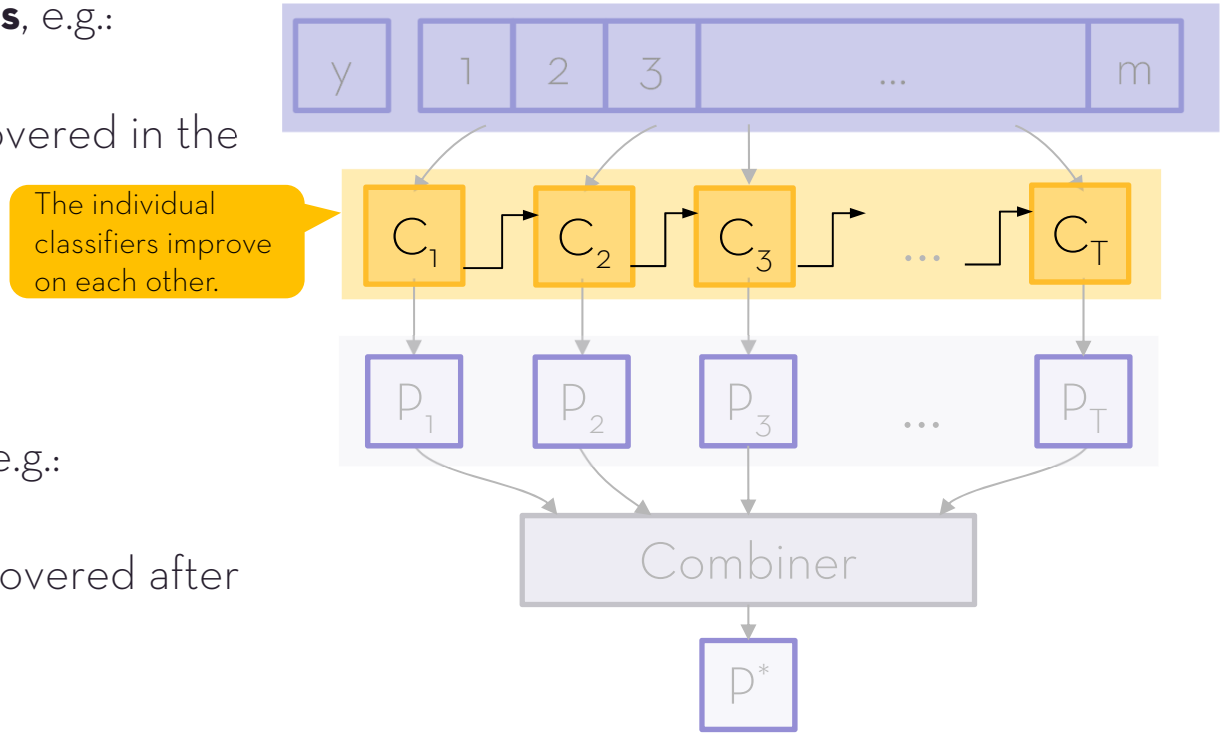
Classifiers can be trained independently from one another...

- **Independent methods**, e.g.:
  - Bagging (will be covered in the next section).



… or they can consecutively improve on the performance of the previous classifier

- **Independent methods**, e.g.:
  - Bagging (will be covered in the next section).



- **Dependent method**, e.g.:
  - Boosting (will be covered after bagging).

Ensembles have 4 main building blocks

Diversity generator	Ensemble size	Members dependency	Combiner

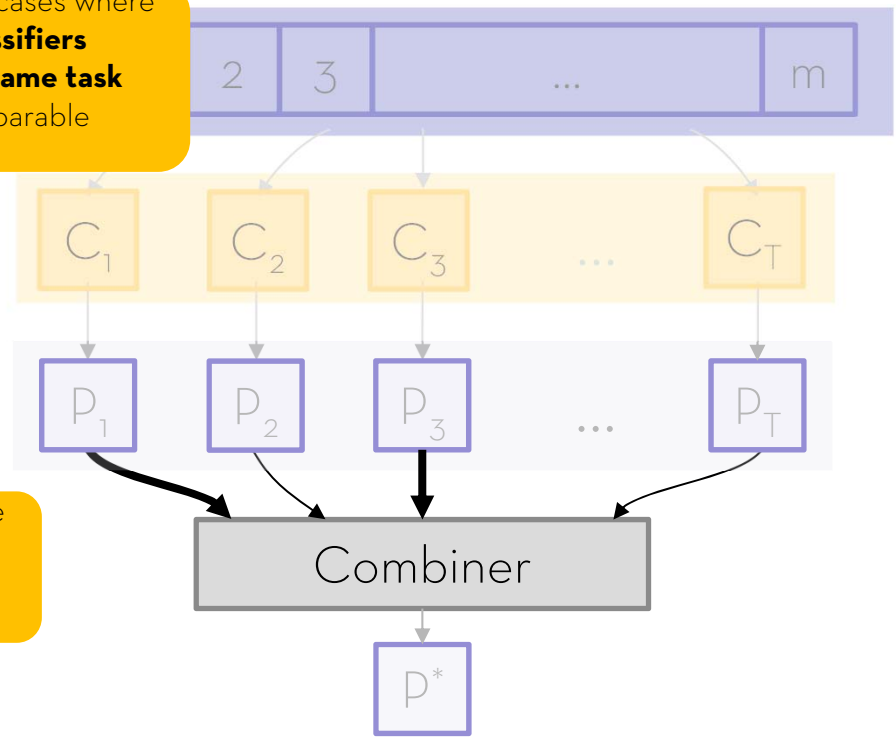
The combiner combines the individual classifiers into a single prediction

- **Weighting methods**, e.g.:
  - Majority voting.
  - Performance weighting.
  - Distribution summation.

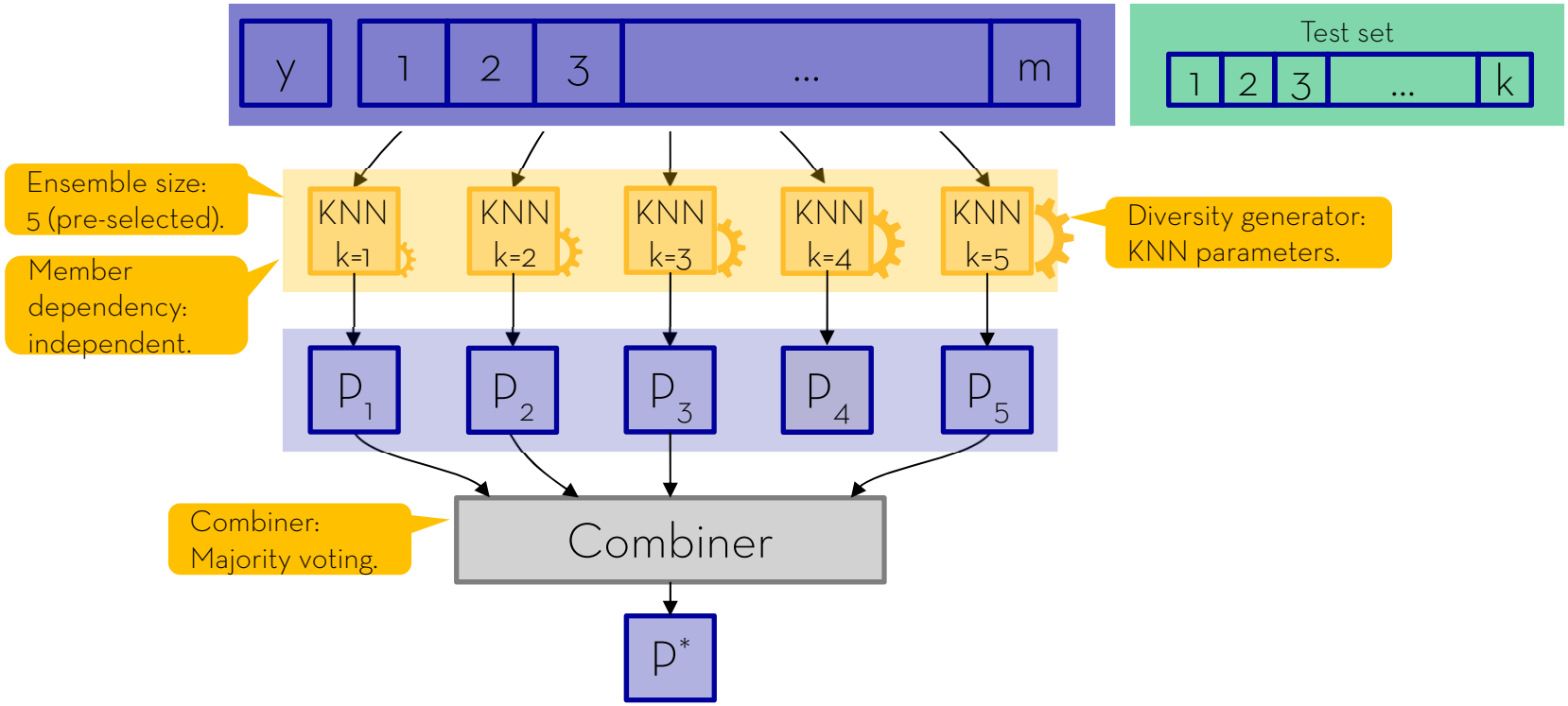
Best suited in cases where **individual classifiers perform the same task** and have comparable success.

- **Meta learners**, e.g.:
  - Stacking.
  - Arbiter trees.
  - Grading.

Best suited in cases where **classifiers consistently correctly or incorrectly classify certain patterns**.



The simplest application of an ensemble is majority voting



Ensembles have 4 main building blocks

Diversity generator	Ensemble size	Members dependency	Combiner

Combining these building blocks arbitrarily does not guarantee ensemble success. Some ensemble compositions are highly effective in narrow contexts.

Exercise
Ensemble learning and its advantages

1. Use the training data to create a majority-vote based ensemble of 3 machine learning models:

   - kNN (k=5)
   - kNN (k=19)
   - Naïve Bayes model

   Predict the class membership for the observations in the holdout set based on ensemble classifier and compare it to the performance of the individual models.

Hint: run the three machine learning models first.

Hint: Think about how to set up the majority-vote based ensemble classifier,: the class is predicted 1 in case at least 2 out of the 3 models predict class 1.

The warrior is always trying to improve.

Model Tuning (1/2)

Dr. Markus Meierer - markus.meierer@business.uzh.ch | Dr. Margot Löwenberg - margot.loewenberg@business.uzh.ch