Queen Mary School Hainan
Queen Mary University of London

QHM5703 Principles of Machine Learning Methodology I

Dr Nikesh Bajaj

Week 10 - 10/11 Nov 2025

Ventris' decisive check

Linear B

"... and a decisive check, preferably with the aid of **virgin material**, to ensure that the apparent results are not due to **fantasy**, **coincidence** or **circular reasoning**"

Don't fool yourself!

What's different about machine learning?

Machine learning aims at building solutions that work well on **samples** coming from a **target population**.

Many other areas have similar goals, so what's different?

- In machine learning, we **lack a description** of the target population.
- All we can do is extract samples from the population (known as **sampling the population**).

A collection of samples extracted from a population forms a dataset. A dataset provides an **empirical** description of our target population. In other words, they are population **surrogates**.

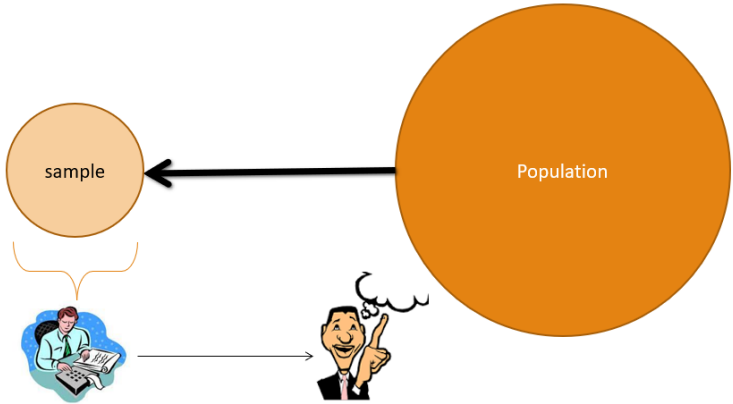
About the unknown: Deployment quality

Examples from Engineering:

- Building a rover for moon
- Going to deep water
- ..

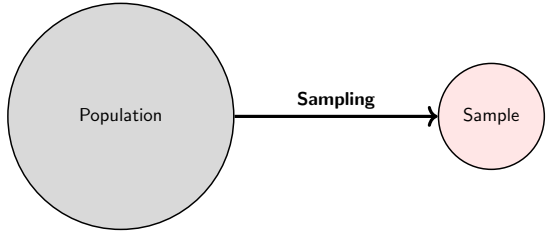
About the unknown: Inferential Statistics

- Inferential Statistics - inferencing about population from sample
- Sampling: a little peek into unknown (indirect/partial)



Let's take a look at Statistics (Inferential)

- Estimation, Quality, Inferencing.

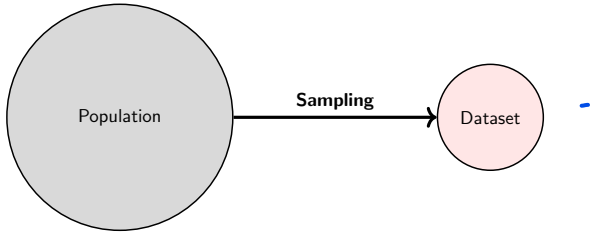


Sampling a population

We need to ensure that datasets are **representative** and provide a complete picture of the target population by:

- Extracting samples **randomly** and **independently**.
- Ensuring they come from the target population (**identically distributed**).
- Having a **sufficiently large** number of samples.

Independent and identically distributed datasets are known as **IID**.



Agenda

Testing a model to evaluate its deployment quality

Optimisation and the error surface

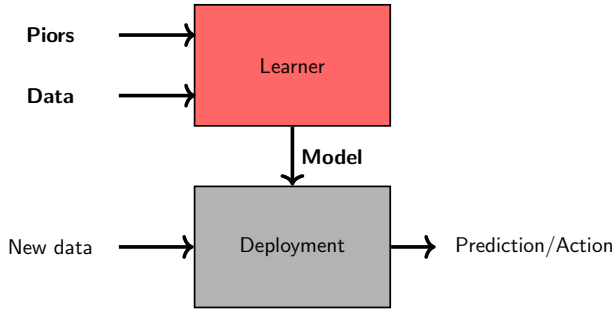
Training a model

Validating our models

Summary

Deployment quality

Machine learning models can be built, sold and deployed.



Take 2 definition: The best model is the one with the highest **deployment quality**, i.e. the best on the **target population**.

Deployment quality

Every machine learning project needs to include a strategy to **evaluate the quality** of a model during deployment.

In machine learning, a quality evaluation strategy includes:

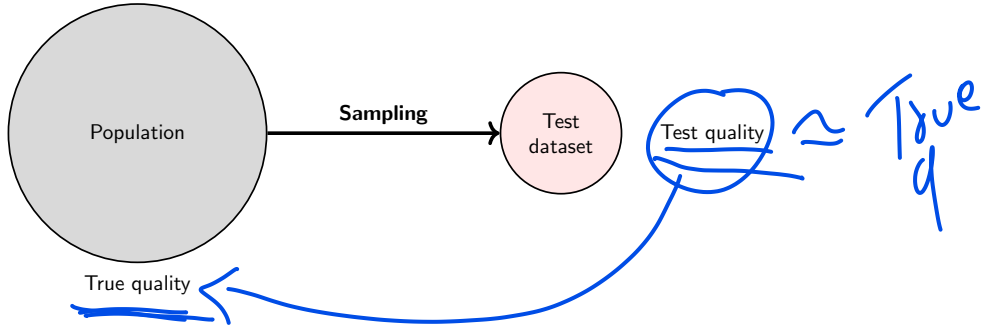
1. A **quality metric** used to quantify the quality.
2. How **data** will be used to assess the quality of a model.

The quality evaluation strategy has to be designed **before** creating a model to avoid falling into our own data-traps, such as confirmation bias. Changing it retrospectively is known as *moving the goalposts*.

Assessing the deployment quality

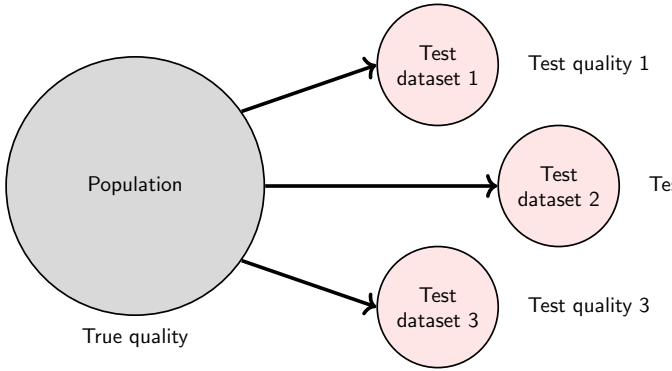
If we could use **all the data** that models can be shown (population), we would be able to quantify their **true** deployment quality.

Instead we use a subset of data, the **test dataset**, to compute the **test quality** as an **estimation** of the true deployment quality.



Random nature of the test quality

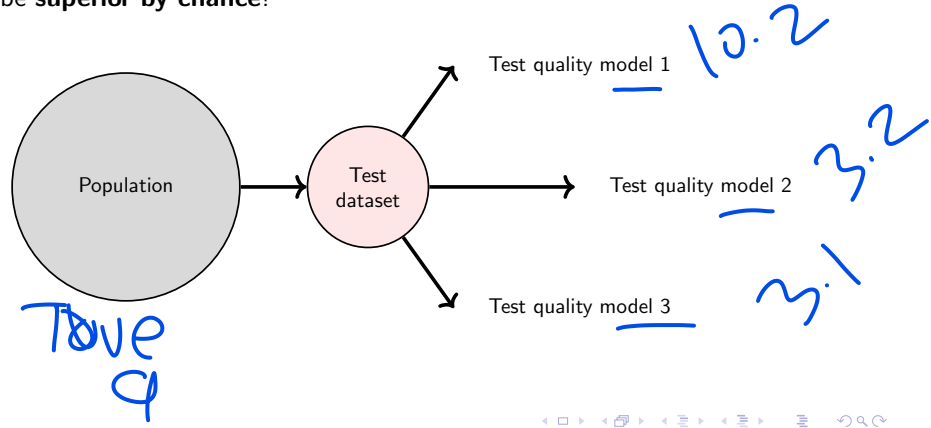
Test datasets are extracted randomly. Hence, the **test quality** is itself **random**, as different datasets will in general produce different values.



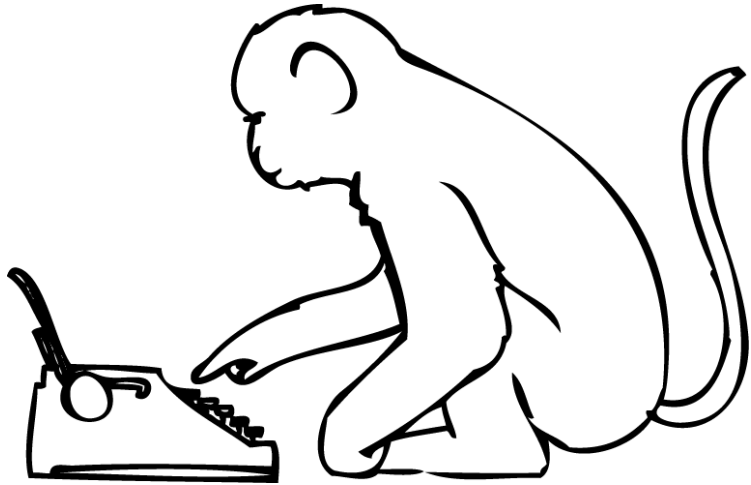
Comparing models

Models built by different teams can be compared based on their test qualities.

Caution should be used, as the test quality is a random quantity, hence some models might appear to be **superior by chance**!



The Infinite Monkey Theorem



Agenda

Testing a model to evaluate its deployment quality

Optimisation and the error surface

Training a model

Validating our models

Summary

Optimisation theory

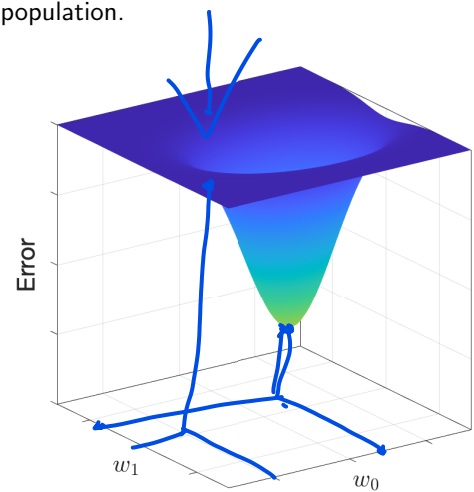
Assume that we have:

- A collection of **candidate models**, e.g. all the models resulting from tuning the linear model $f(x) = w_0 + w_1 x$.
- A **quality metric**, e.g. a notion of error.
- A **perfect description** of the target population.

Optimisation allows us to identify among all the candidate models the one that achieves the highest quality on the target population, i.e. the **optimal** model.

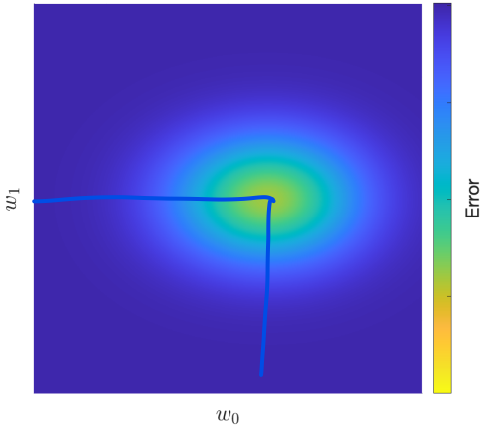
The error surface

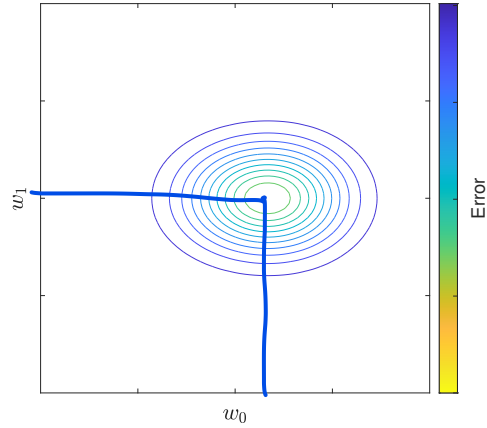
The error surface (a.k.a. error, objective, loss or cost function) denoted by $E(\boldsymbol{w})$ maps each **candidate model** $\boldsymbol{w}$ to its **error**. We will assume that we can obtain it using the ideal description of our target population.



The error surface

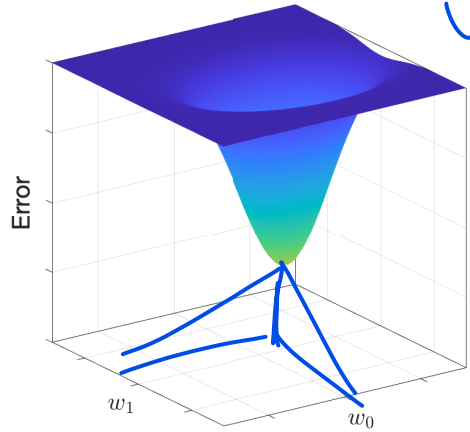
Error surfaces can also be represented as colour-coded and contour maps, where a colour scheme encodes the error values.





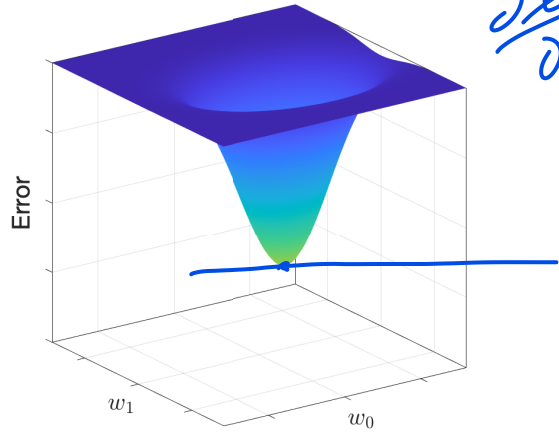
The error surface and the optimal model

The **optimal** model can be identified as the one with the lowest error.



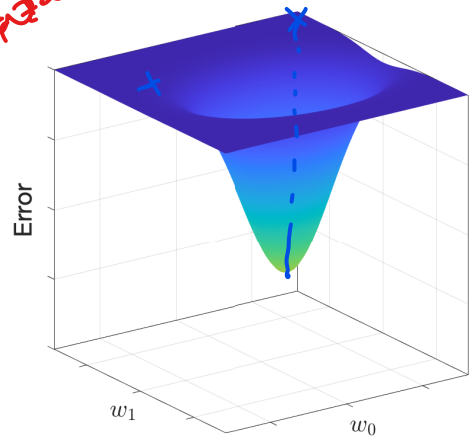
The error surface and the optimal model

The **gradient** (slope) of the error surface, $\nabla E(\boldsymbol{w})$, is zero at the optimal model. Hence we can look for it by identifying where $\nabla E(\boldsymbol{w}) = \vec{0}$.



The error surface and the optimal model

What if do not have enough computation power to obtain the error or the gradient of **every candidate model**, how can we find the optimal model?



Gradient descent

Gradient descent is a numerical optimisation method where we **update iteratively** our model using the gradient of the error surface.

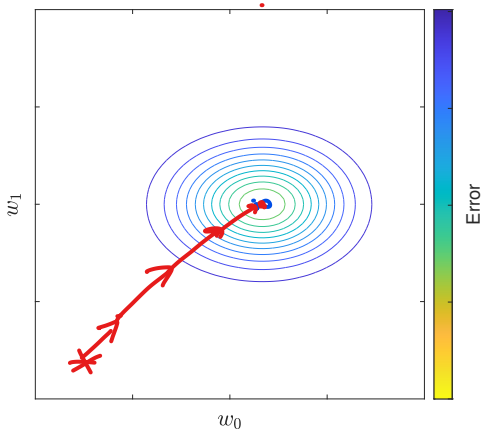
The gradient provides the direction along which the error increases the most. Using the gradient, we can create the following update rule:

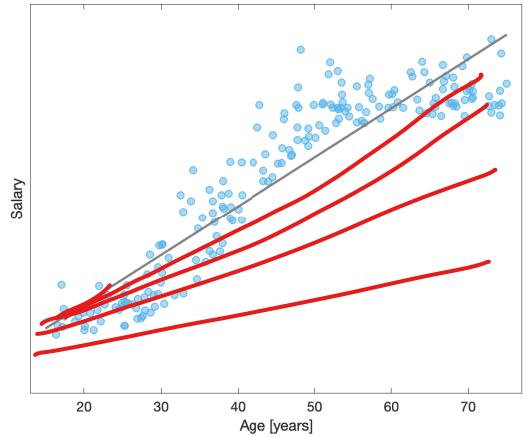
$$\boldsymbol{w}_{\text{new}} = \boldsymbol{w}_{\text{old}} - \epsilon \nabla E(\boldsymbol{w}_{old})$$

where ϵ is known as the **learning rate** or **step size**.

With every iteration we adjust the parameters $\boldsymbol{w}$ of our model. This is why this process is also known as **parameter tuning**.

Gradient descent



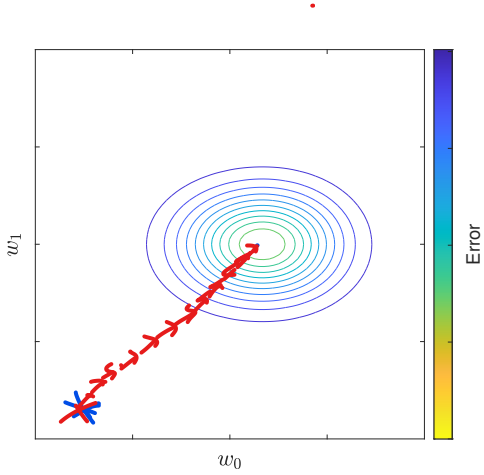


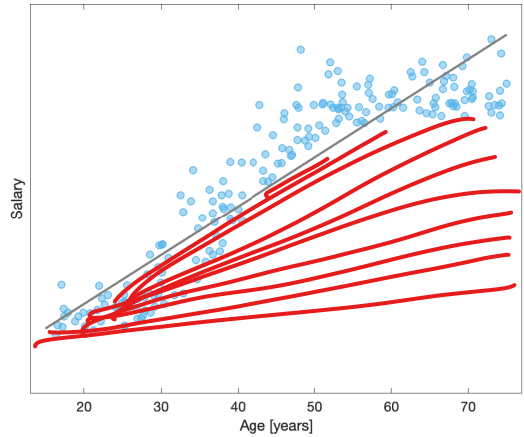
The step size

The step size ϵ controls how much we change the parameters $\boldsymbol{w}$ of our model in each iteration of gradient descent:

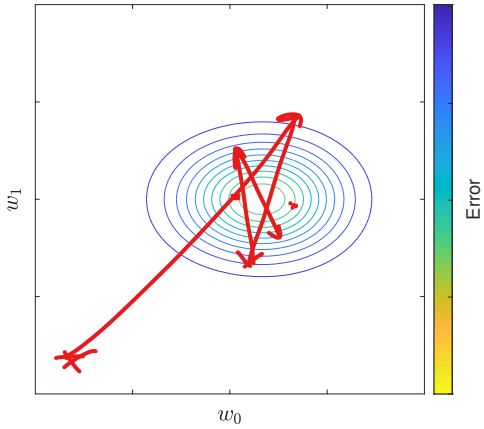
- Small values of ϵ result in slow convergence to the optimal model.
- Large values of ϵ risk overshooting the optimal model.

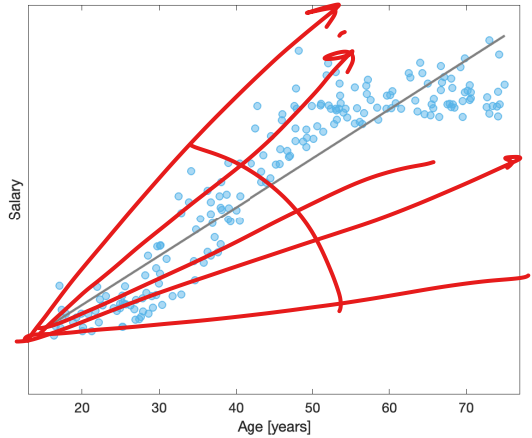
Small steps



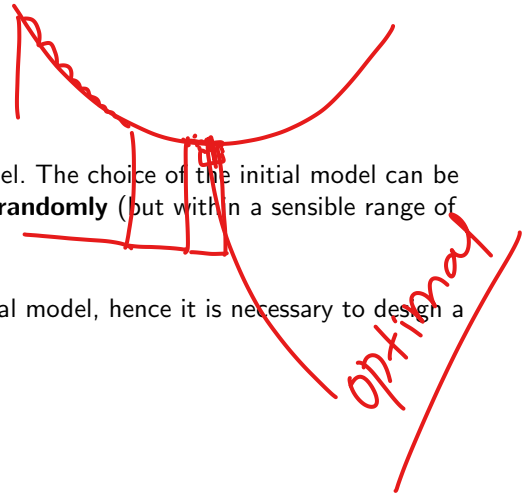


Large steps





Starting and stopping



For gradient descent to start, we need an initial model. The choice of the initial model can be crucial. The initial parameters $\boldsymbol{w}$ are usually chosen **randomly** (but within a sensible range of values).

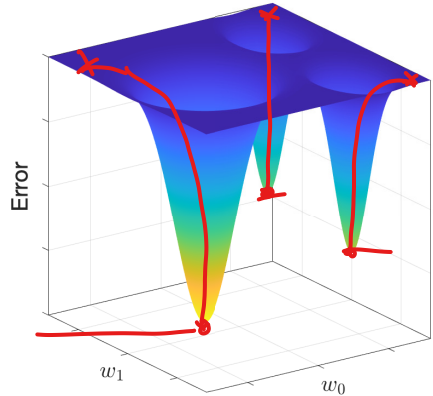
In general, gradient descent will not reach the optimal model, hence it is necessary to design a stopping strategy. Common choices include:

- Number of iterations.
- Processing time.
- Error value.
- Relative change of the error value.

Local and global solutions

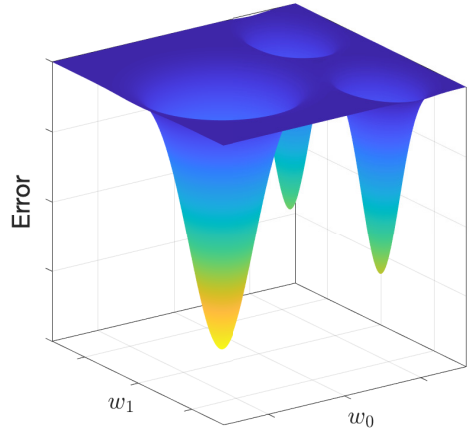
So far we have considered **convex** error surfaces. Error surfaces can however be complex and have

- **Local** optima (model with the lowest error within a region).
- **Global** optima (model with the lowest error among all the models).



Local and global optimal solutions

Gradient descent can get **stuck** in local optima. To avoid them, we can repeat the procedure from several **initial models** and select the best.



Agenda

Where is my error surface?

In machine learning we have:

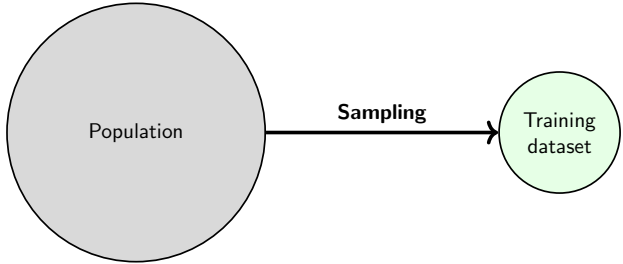
- A family of **candidate models** (e.g. linear models).
- A **quality metric** (e.g. the error).
- **Dataset** extracted from the population (i.e. **not** a perfect description).

If we had an ideal description of the target population we could calculate the error surface and its gradient to find the optimal model.

In machine learning, our starting point is that we do not have a perfect description of the population and we only have datasets extracted from it.

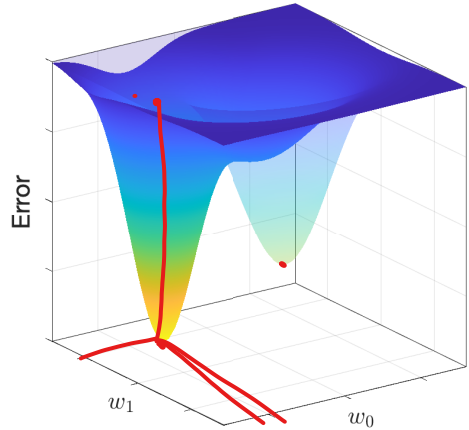
The training dataset

We use a subset of samples, known as the **training dataset**, to **reconstruct** the true error surface needed during optimisation. We will call this new surface the **empirical error surface**.



The empirical error surface

The empirical and true error surfaces are in general different. Hence, their optimal models might differ, i.e. the best model for the training dataset might not be the best for the population.

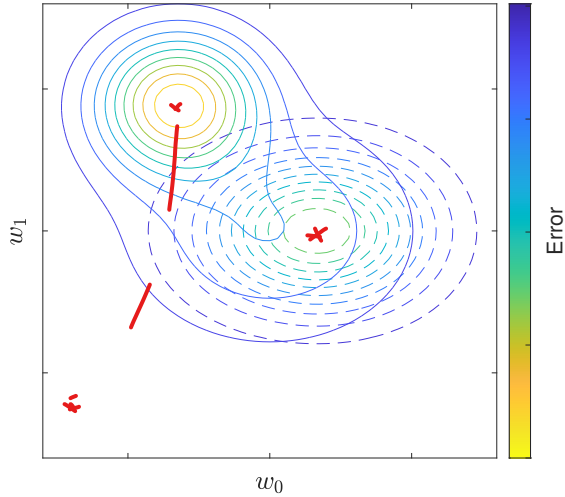


The empirical error surface: Question

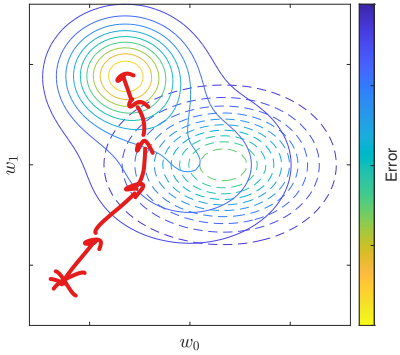
Question: When do we have different error surfaces?

Examples of scenarios:

The empirical error surface



The empirical error surface



Least squares: minimising the error on a training dataset

Least squares defines an **empirical error surface whose gradient can be obtained exactly**. A linear model applied to a training dataset can be expressed as

$$\hat{\boldsymbol{y}} = \mathbf{X}\boldsymbol{w}$$

where $\mathbf{X}$ is the design matrix, $\hat{\boldsymbol{y}}$ is the predicted label vector and $\boldsymbol{w}$ is the coefficients vector. The MSE on the training dataset can be written as

$$\begin{aligned} E_{MSE}(\boldsymbol{w}) &= \frac{1}{N}(\boldsymbol{y} - \hat{\boldsymbol{y}})^T (\boldsymbol{y} - \hat{\boldsymbol{y}}) \\ &= \frac{1}{N}(\boldsymbol{y} - \mathbf{X}\boldsymbol{w})^T (\boldsymbol{y} - \mathbf{X}\boldsymbol{w}) \end{aligned}$$

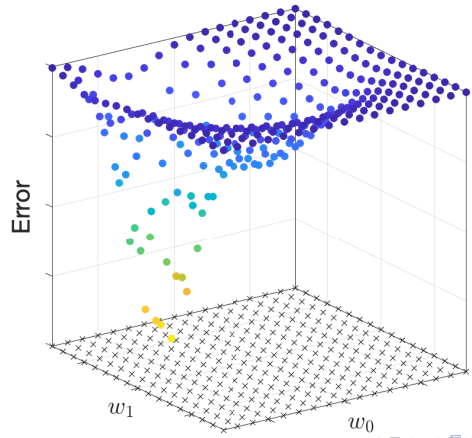
where $\boldsymbol{y}$ is the true label. The resulting gradient of the MSE is

$$\nabla E_{MSE}(\boldsymbol{w}) = \frac{-2}{N}\mathbf{X}^T (\boldsymbol{y} - \mathbf{X}\boldsymbol{w})$$

and it is zero for $\boldsymbol{w}_{opt} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\boldsymbol{y}$.

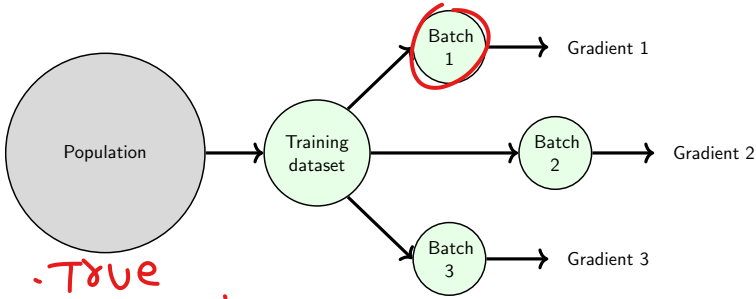
Exhaustive approaches

In general, we will not have analytical solutions. We can reconstruct the empirical error surface by evaluating each model on training data. This is called **brute-force** or **exhaustive search**. Simple, but often **impractical**.



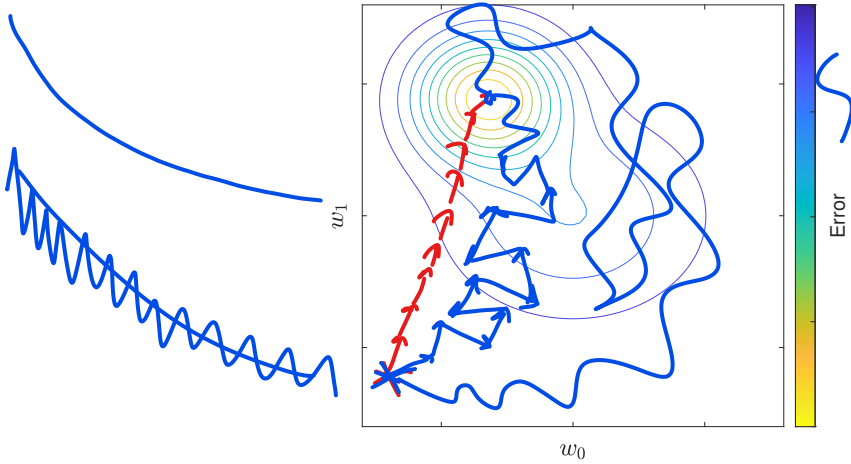
Data-driven gradient descent

Gradient descent can be implemented by **estimating the gradient** of the empirical error surface. During each iteration, a subset (**batch**) of the training dataset is used to compute this gradient.



The batch size determines how close the estimated gradient is to the true gradient of the empirical error surface.

Batch size in data-driven gradient descent



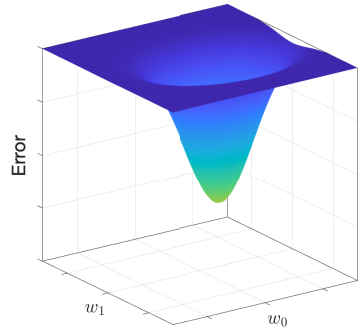
Stochastic gradient de

Why to use gradient from a batch, if we can use entire training data?

The empirical error surface: Question

Question: Examine the error surface carefully associate with a linear model f_1. If we change the family of model; from f_1 to f_2 or to f_3, will this error surface change?

- $f_1(x) = w_0 + w_1 x$
- $f_2(x) = w_0 + w_1 x + w_2 x^2$
- $f_3(x) = w_0 + w_1 e^{-x}$



Think about it! (HW)

Overfitting and fooling ourselves

The empirical and true error surfaces are in general different. When **small datasets** and **complex models** are used, the differences between the two can be very large, resulting in trained models that work very well on the training dataset but poorly when deployed.

This is, of course, another way of looking at **overfitting**. By increasing the size of the training dataset, the empirical error surface gets closer to the true error surface and the risk of overfitting decreases.

Important

Never use the same data for testing and training a model. The test dataset needs to remain **inaccessible** to avoid using it (inadvertently or not) during training.

Regularisation [we will see in more details]

Regularisations modifies the empirical error surface by adding a term that constrains the values that the model parameters can take on. A common option is the regularised error surface $E_R(\boldsymbol{w})$ defined as:

$$E_R(\boldsymbol{w}) = E(\boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{w}$$

For instance, the MSE in regression can be regularised as follows:

$$E_{MSE+R} = \frac{1}{N} \sum_{i=1}^{N} e_i^2 + \lambda \sum_{i=1}^{K} w_k^2$$

and its **least square solution** is

$$\boldsymbol{w} = \left(\mathbf{X}^T \mathbf{X} + N\lambda \mathbf{I} \right)^{-1} \mathbf{X}^T \boldsymbol{y}$$

As λ increases, the complexity of the resulting solution decreases and so does the risk of overfitting.

Cost vs quality

Regularisation provides an example where we use a notion of quality during training (E_{MSE+R}) that is different from the notion of quality during deployment (E_{MSE}).

Our goal is always to produce a model that achieves the highest **quality during deployment**. How we achieve it, is a different question.

In addition to using constraints that limit the risk of overfitting, the notion of training quality might be different from the deployment quality, because we cannot use the latter for training.

We usually call our notion of quality during training **cost** or **objective function**, to distinguish it from the **target quality metric**.

We will see examples where the notion of training cost is different from the notion of deployment quality. In many cases, we cannot use the notion of deployment quality during training.

Agenda

Testing a model to evaluate its deployment quality

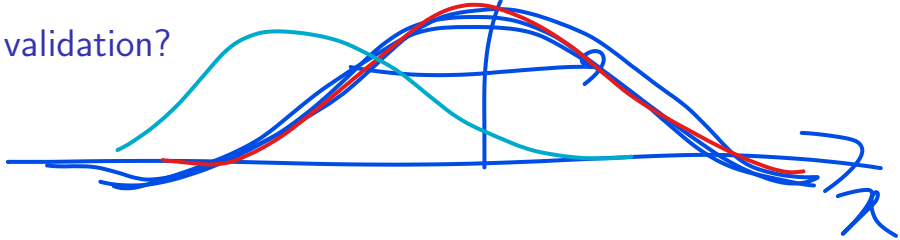
Optimisation and the error surface

Training a model

Validating our models

Summary

Why do we need validation?

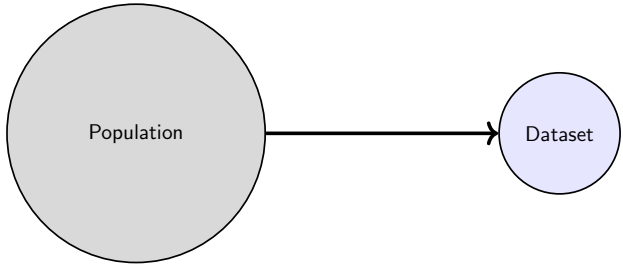


Machine learning uses datasets for different purposes, for instance to assess the deployment quality of a final model (**test dataset**) or to tune a model (**training dataset**).

Often, we need to explore different options before training a final model. For example, consider polynomial regression. The polynomial degree D is a **hyperparameter**, as for each value of D a different family of models is obtained. How can we select the right value of a D?

Why do we need validation?

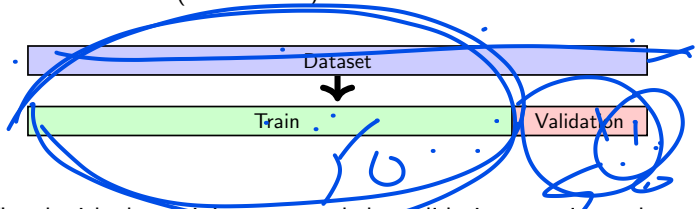
Validation methods allow us to use data for **assessing** and **selecting** different families of models (**model selection**). The same data used for validation can then be used to train a **final model**.



Validation involves one or more **training** and **quality estimation** rounds per model family followed by **quality averaging**.

Validation set approach

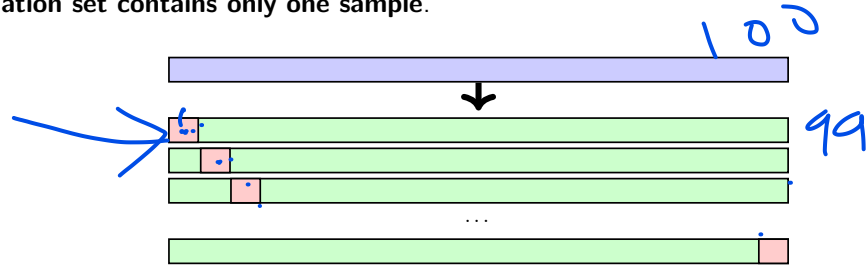
The validation set approach is the simplest method. It randomly splits the available dataset into a **training** and a **validation** (or hold-out) dataset.



Models are then fitted with the training part and the validation part is used to estimate its quality.

Leave-one-out cross-validation (LOOCV)

This method also splits the available dataset into training and validation sets. However, the **validation set contains only one sample**.

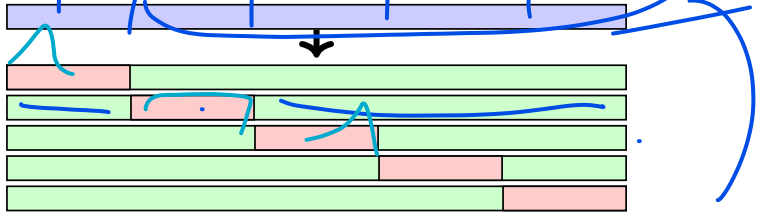


Multiple splits are considered and the final quality is calculated as the average of the individual qualities (for N samples, we produce N splits and obtain N different qualities).

k-fold cross-validation

In this approach the **available dataset is divided into k groups** (also known as folds) of approximately equal size:

- We carry k **rounds of training followed by validation**, each one using a different fold for validation and the remaining for training.
- The final estimation of the quality is the average of the qualities from each round.



LOOCV is a special case of k-fold cross-validation, where $k = N$.

Validation approaches: Comparison

- The **validation set** approach involves one training round. Models are however trained with fewer samples and the final quality is highly variable due to random splitting.
- **LOOCV** requires as many training rounds as samples there are in the dataset, however in every round almost all the samples are used for training. It always provides the same quality estimation.
- k-**fold** is the most popular approach. It involves fewer training rounds than LOOVC. Compared to the validation set approach, the quality estimation is less variable and more samples are used for training.

Agenda

Machine learning methodology: Basic tasks

In machine learning we can identify the following tasks:

- **Test**: This is the **most** important task. It allows us to estimate the deployment quality of a model.
- **Training**: Used to find the best values for the parameters of a model, i.e. to tune a model.
- **Validation**: Necessary to compare different modelling options and select the best one, the one that will be trained.

The role of data

In machine learning we do not have an ideal description of the target population, all we can do is **extract data**.

- Test, training and validation tasks all involve data.
- Hence we talk about test dataset, as the data used for testing a model, training dataset, as the data used for training a model, etc.
- Think about the tasks first, then create the datasets that you need.

So you've read about **splitting a dataset**? Splitting datasets is **not** an ML task. What we need is to **create the right dataset** for each task. This might involve splitting an existing dataset, but not necessarily.

Using data correctly

- Datasets need to be **representative** of the target population and its samples need to have been extracted **independently**.
- Any test strategy has to be **designed before training**. Avoid looking at the test dataset during training and test a final model only once.
- Any quality estimation that is obtained from a dataset is a **random quantity** (within a range): use with caution.
- The quality of a final model depends on the **type** of model, the **optimisation** strategy and the representativity of the training **data**.

Disappointed you didn't decipher Linear B?

Don't worry, we still have a few undeciphered scripts:

- Proto-Elamite
- Indus
- Meroitic
- Linear A
- Rongorongo
- Zapotec
- Voiynichese

The Voynich Manuscript, 15th century

Queen Mary
University of London