Lecture 2: Probability, Statistics, and Basic of machine learning

Maurizio Pierini (CERN), Thea Aarrestad (ETH)

Reminder

Date	Topic	Tutorial
Sep 16	Intro & class description	Linear Algebra in a nutshell + prob and stat
Sep 23	Basic of machine learning + Dense NN	*Basic jupyter + DNN on mnist (give jet dnn as homework)*
Sep 30	Convolutional NN	*Convolutional NNs with MNIST*
Oct 7	Training in practice: regularization, optimization, etc	*Practical methodology*
Oct 14	*Google tutorial **To Be Confirmed***	
Oct 21	Graph NNs	*Tutorial on Graph NNs*
Oct 28	Unsupervised learning and anomaly detection	*Autoencoders with MNIST*
Nov 4	Generative models: GANs, VAEs, etc	Normalizing flows
Nov 11	Transformers	
Nov 18	Network compression (pruning, quantization, Knoweldge Distillation)	
Nov 25	Tutorial on hls4ml **To Be Confirmed**	
Dec 2	To Be Decided	
Dec 9	Quantum Machine Learning	
Dec 16	**Exams To Be Confirmed**	

Some news

- *I re-sent the email to adl-zurich@cern.ch (to which I subscribed all you, in principle). Did anyone receive it?*
- *I created a repository on https://indico.cern.ch/category/20346/*
  - *There, I will log the recording of each lecture from zoom (no guarantees on audio quality)*
  - *I will send access pwd by email once I get confirmation that the e-group works*

Plan for Today

- *Probability*
- *Numerical Computation*
- *Machine Learning Basic*

✔: Parts that we will cover

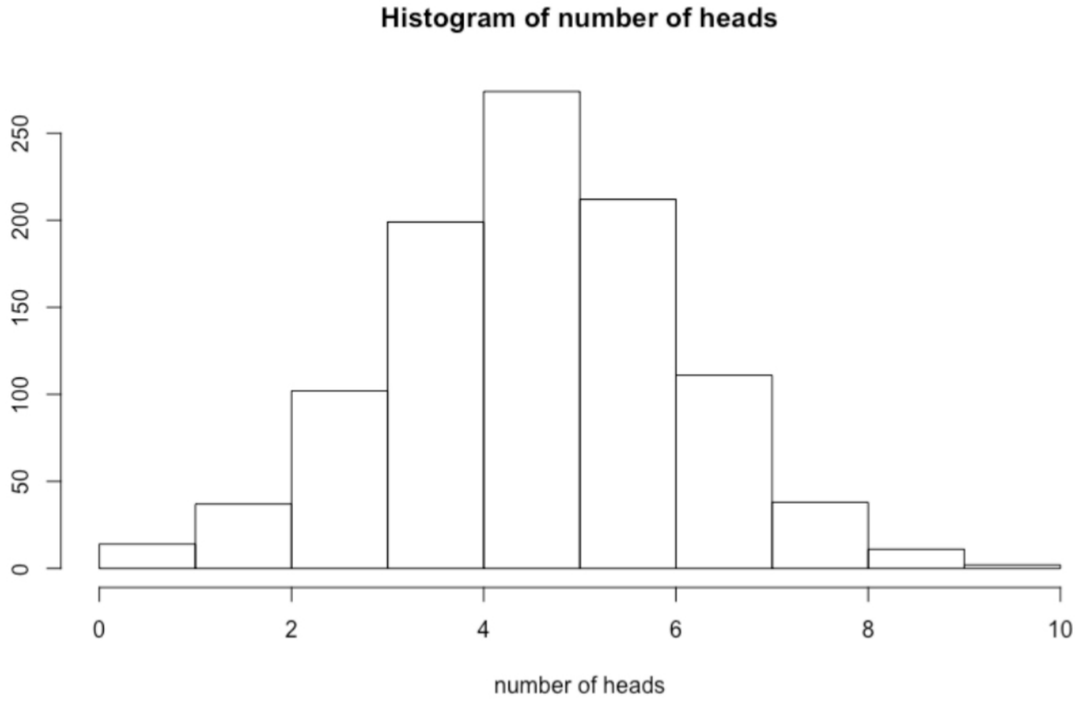
Probability

"I wish we hadn't learned probability 'cause I don't think our odds are good."

Uncertainty and Probability

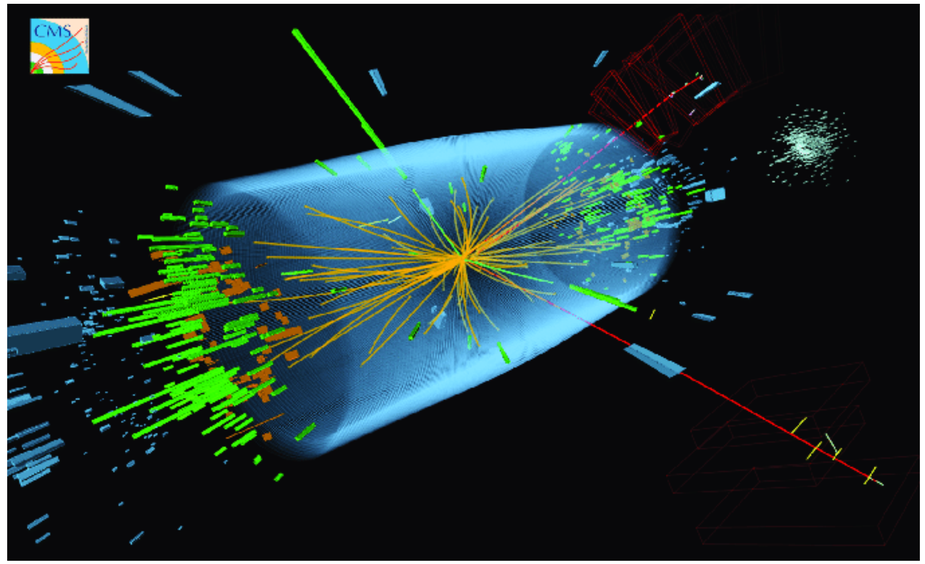
- *Machine Learning, like any branch of Data Science, is not deterministic*
  - *Inherent stochasticity: One tries to learn from a dataset, and datasets come with statistical noise*
  - *Incomplete observability: even in absence of stochastic noise, one might be in an uncertain status for lack of knowledge due to partial observation*
  - *Incomplete modeling: we might miss some of the observed observation, that for various reasons might have been removed from the original dataset*
- *In presence of uncertainty, we need to approach the problem following probability theory and statistics*

Example: 10 coin tossing will give ~ 5 heads, but not always EXACTLY 5 heads



Two schools of statistics

- *Frequentist school: the probability of a certain event is the frequency for that event to occur in the limit of infinite statistics.*
  - *This point of view is suitable for repeatable experiments (e.g., probability to produce a Higgs boson at the LHC)*
  - *It has some circularity (it requires each repetition to have equiprobable outcome, which assumes the concept of equal probability to define probability)*
- *Bayesian school: the probability of a certain event measures the degree of belief that the observed associates to a certain outcome*
  - *It applies well to events which are not repeatable. E.g., what is the probability that it will rain tomorrow?*
  - *It implies a subjectivity (which boils down to specifying a-priori belief on a given event before the observation)*



Probability Distribution

- *A random variable a variable that can take on different values randomly*
  - *Can be a scalar or a vector. Can be discrete or continuous*
- *A probability distribution is a description of how likely a random variable or set of random variables is to take on each of its possible states.*

The probability density function (pdf), aka probability mass function (pmf) for a discrete quantity

- The domain of P must be the set of all possible states of x.
- $\forall x \in \mathrm{x}, 0 \leq P(x) \leq 1$. An impossible event has probability 0, and no state can be less probable than that. Likewise, an event that is guaranteed to happen has probability 1, and no state can have a greater chance of occurring.
- $\sum_{x \in \mathrm{x}} P(x) = 1$. We refer to this property as being **normalized**. Without this property, we could obtain probabilities greater than one by computing the probability of one of many events occurring.

Probability Distribution

- *A random variable a variable that can take on different values randomly*
  - *Can be a scalar or a vector. Can be discrete or continuous*
- *A probability distribution is a description of how likely a random variable or set of random variables is to take on each of its possible states.*

The probability density function (pdf), aka probability mass function (pmf) for a continuous quantity

- The domain of p must be the set of all possible states of x.
- $\forall x \in \mathrm{x}, p(x) \geq 0$. Note that we do not require $p(x) \leq 1$.
- $\int p(x)dx = 1$.

Probability Distribution

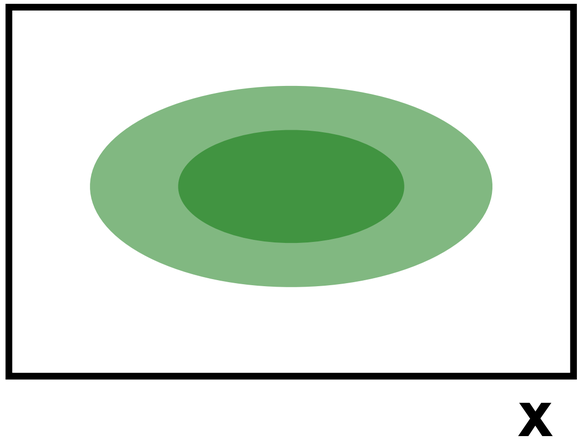
- *EXAMPLE: Uniform distribution of discrete quantity, the pdf for equiprobable outcomes:* $p(x = x_i) = \frac{1}{k} \text{for } i \in [0, k-1]$*. This definition meets all the desired properties. In particular*

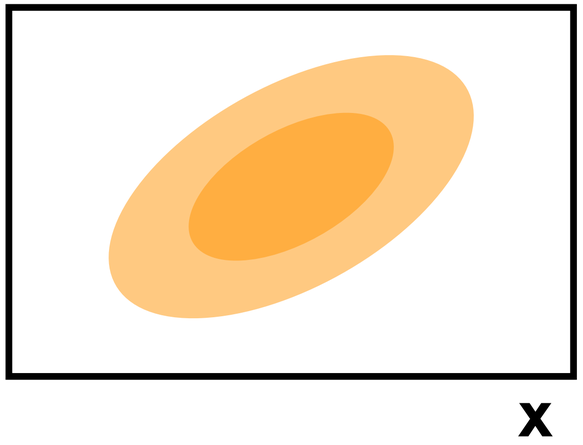
$$0 \leq p(x = x_i) \leq 1 \quad \textit{and} \quad \sum_{i=0}^{k-1} p(x_i) = \sum_{i=0}^{k-1} \frac{1}{k} = 1$$

- *This generalises to the case of a continuous quantity defined between* a *and* b *as* $p(x) = \frac{1}{b-a}$

Joint Probability and Correlation

- The *joint probability* of two quantities x and y is the pdf that describes the probability of all the possible outcomes, expressed as (x, y) pairs
  - If x and y are *independent*, $p(x, y) = q(x)t(y)$, where $q(x)$ and $t(y)$ are the 1D pdfs of x and y
  - If $p(x, y) \neq q(x)t(y)$, it means that the outcome on x depends on y and vice versa. x and y are said to be *correlated*



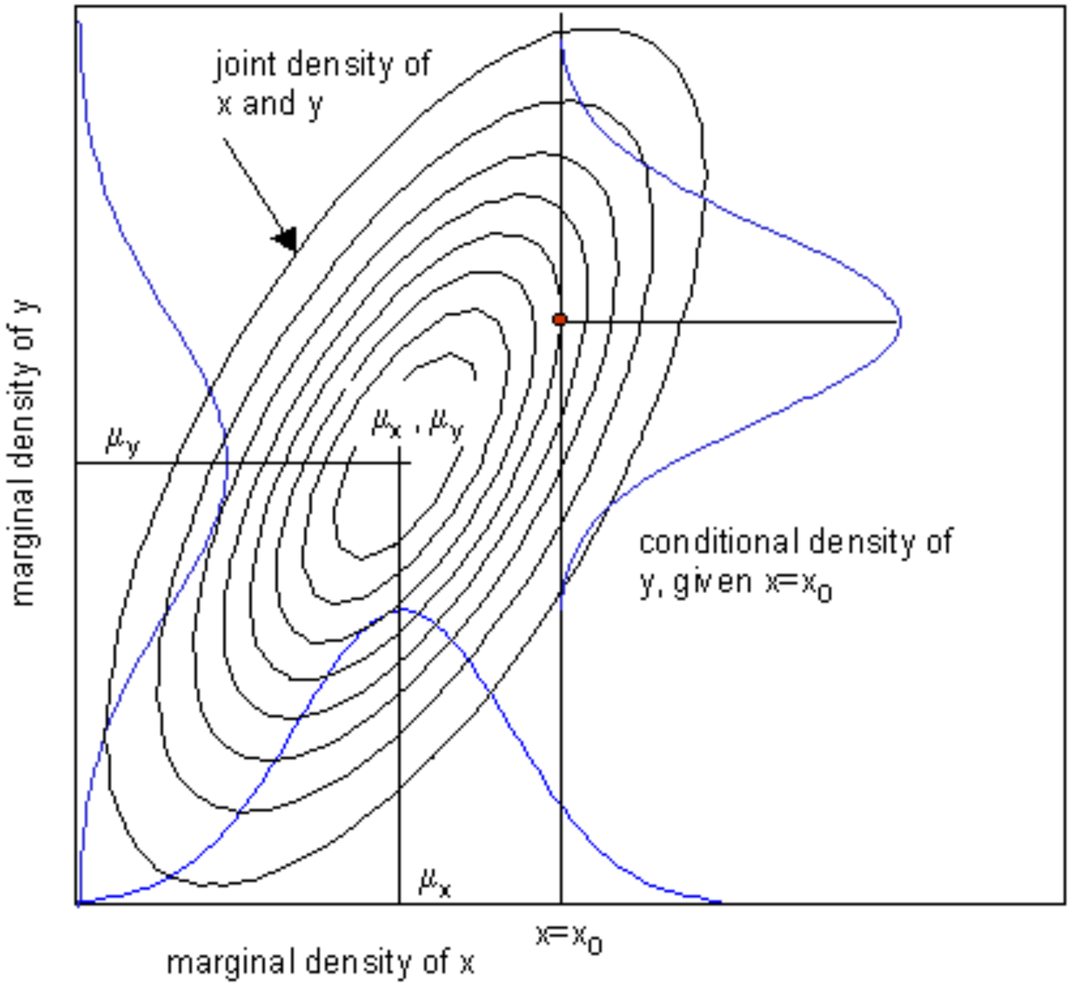


Marginal Probability

- *In some cases, one knows the joint probability distribution of two quantities x and y $p(x,y)$ but being interested to a statement on x regardless of y*

- *in this case, one needs to derive the marginal pdf from the joint pdf*

$$p(x) = \int p(x,y)dy$$



Conditional Probability

- In certain cases, the value of y is known and one wants to restrict the prediction on x to the fact that that value of y occurred. This means deriving the conditional probability from the joint probability

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

- The marginal probability in the denominator guarantees that the conditional probability is $\in [0,1]$

- Inverting the relation above, we derive the probability chain rule

$$p(x, y = \hat{y}) = p(x \mid y = \hat{y})\ p(y = \hat{y})$$

or, with a lighter notation and for any value of y

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

Chain rule

- We saw how to derive the marginal and conditional probabilities from the joint probability
- Often, it is useful to do the opposite: express the joint probability as a function of marginal and conditional probabilities
  - The main advantage is to break down an N-dim function into many 1-dim functions
  - Take the example of three quantities. We have

$$P(x_1, x_2, x_3) = P(x_1, x_2 \,|\, x_3) P(x_3) \quad \text{and} \quad P(x_1, x_2) = P(x_1 \,|\, x_2) P(x_2)$$

which imply

$$P(x_1, x_2, x_3) = P(x_1 \,|\, x_2, x_3) P(x_2 \,|\, x_3) P(x_3)$$

- In general

$$P(x_1, \ldots, n_n) = P(x_1) \prod_{i=2}^{n} P(x_i \,|\, x_1, \ldots, x_{i-1})$$

More on Independence

- We saw that two quantities are said to be *independent* when their joint probability factorizes into the product of two 1-dim pdf

$$p(x, y) = p(x)p(y)$$

- We can generalise this concept introducing *conditional independence* of

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

or $$p(x, y \mid z) = p(x \mid z)p(y \mid z)$$

- x and y are not independent, but their correlation is only driven by the common value of z

Why is this important for ML?

- *Supervised learning is the learning of a conditional pdf. Given some data* $\mathbf{x}$ *and ground truth* y *we want to learn* $p(y|\mathbf{x})$

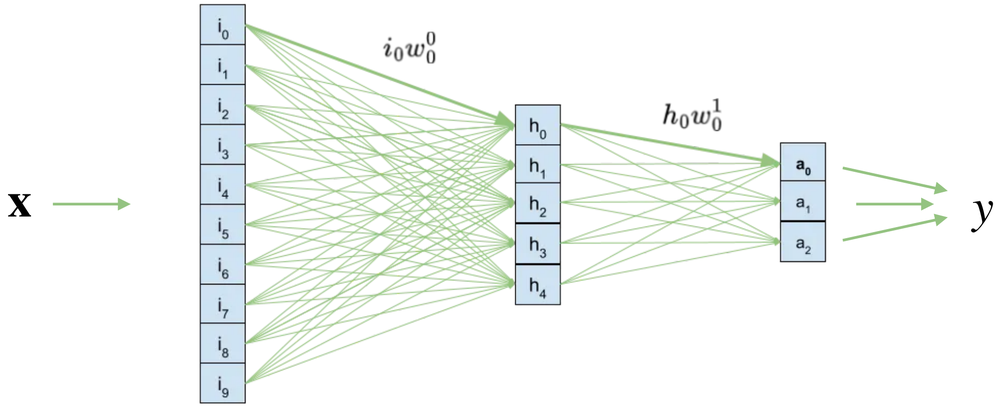
- *When solving certain tasks with ML, you are often confronted to learn the pdf of your data*

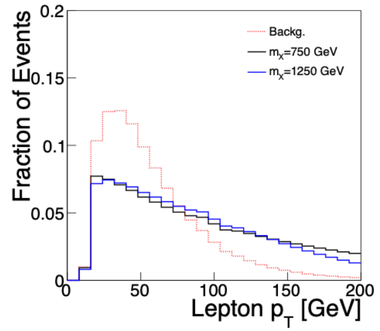
  - *At the LHC, you might want to know if your data were produced by Higgs bosons or by some new kind of particle, for which you don't know the mass.*

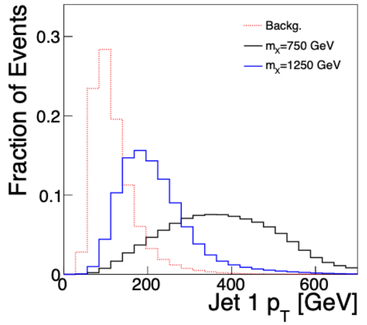
  - *You can assume a mass and train a network for every possible mass value*

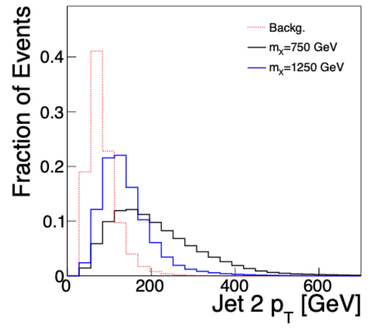
  - *Or you can learn the classifier parametrically to the assumed value* $p_H(D)$ *vs* $p_X(D|m_X)$

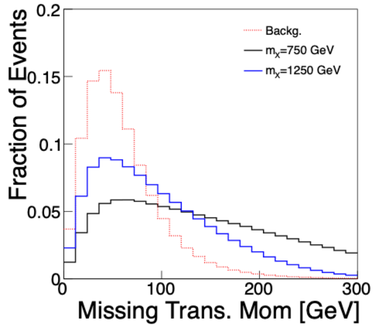
- *This is called parametric learning (see* ***https://arxiv.org/abs/1601.07913****)*











Expectation Value

- *Expectation value: mean value (aka average value) of a quantity f(x) with respect to the pdf of x*
  - *Expectation value is linear:*

$$E_X[\alpha\, f(x) + \beta\, g(x)] =$$

$$\alpha\, E_X[f(x)] + \beta\, E_X[g(x)]$$

DEFINITION:

FOR A GENERIC $P(k|\alpha)$

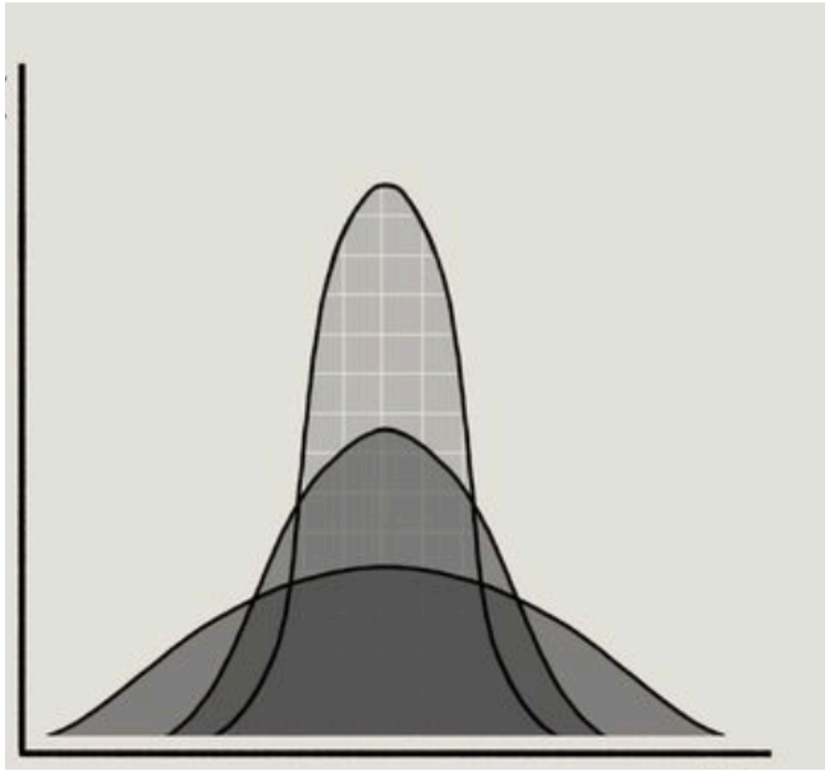
$$E[k|\alpha] = \frac{\sum_k k P(k|\alpha)}{\sum P(k|\alpha)}$$

FOR A GENERIC $P(x|\alpha)$

$$E[x|\alpha] = \frac{\int dx\, x\, P(x|\alpha)}{\int dx\, P(x|\alpha)}$$

Variance

- E[x] is not enough to characterize a distribution
  - distributions with same E[x] can be very different
- It is convenient to have a measure of the dispersion of points around E[x]
  - One typically introduced the variance (aka mean square error)



$$Var[x] = E[(x - E[x])^2] = E[x^2] - E[x]^2$$

- The square root of the variance is called standard deviation

Covariance

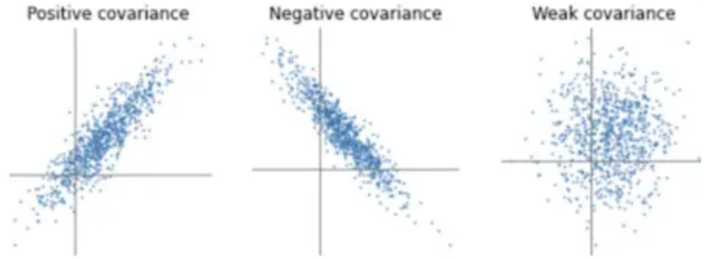
- *Covariance quantifies the correlation between two quantities*

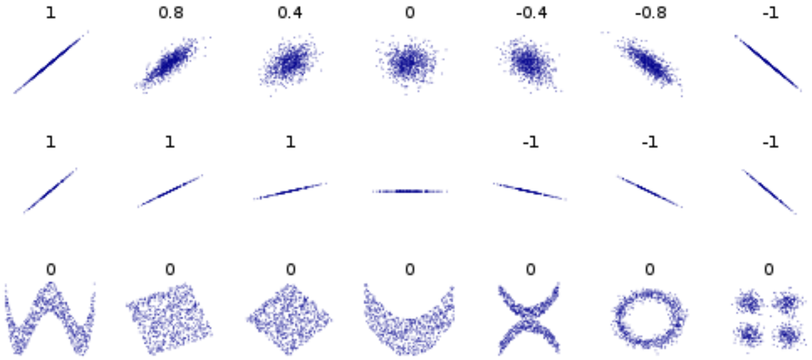
$$Cov(f(x)g(x)) = E_X\left[(f(x) - E_X[f(x)])(g(x) - E_X[g(x)])\right]$$

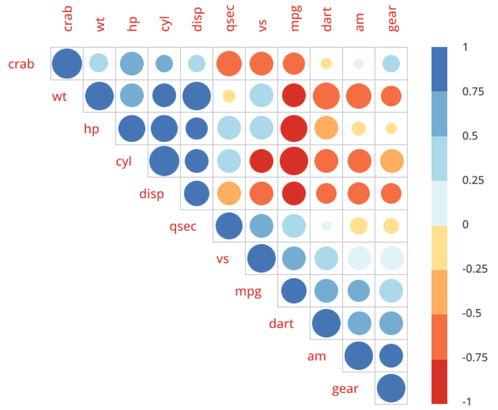
- *For $f(x) = g(x)$, one obtains the variance*
- *NOTICE: the covariance measures a linear dependence. Two quantities can have covariance = 0 without being independent (independence is a stronger requirement)*
- *Often one introduces the linear correlation coefficient*

$$\rho(f(x), g(x)) = \frac{Cov(f(x), g(x))}{\sqrt{Var(f(x))}\sqrt{Var(g(x))}}$$

- *For a vector of quantities, covariance and correlation matrices can be built*







Bernoulli's process

- You have a question with yes/no as possible outcomes, each outcome being independent from the others
  - did I get a head when tossing my coin?
- Let's call
  - p: probability that the answer is Yes
  - q = 1-p : probability that the answer is No

$$P(k=1)=p \qquad P(k=0)=1-p=q$$

$$\mathbb{E}[k]=\frac{p\cdot 1+q\cdot 0}{p+q}=p$$

$$Var(k)=\mathbb{E}[(k-\mathbb{E}[k])^2]$$

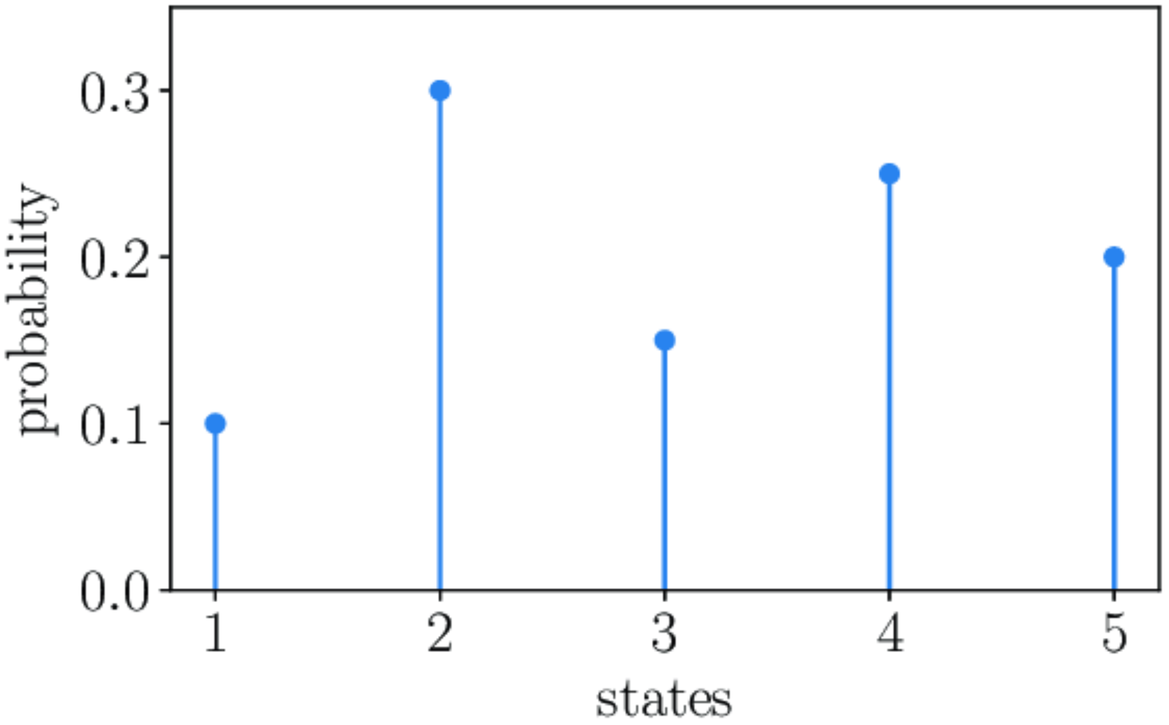
$$=\mathbb{E}[k^2]-2\mathbb{E}[k]^2+\mathbb{E}[k]^2$$

$$=\mathbb{E}[k^2]-\mathbb{E}[k]^2=\frac{1^2\cdot p+0^2\cdot q}{p+q}-p^2$$

$$=p-p^2=p(1-p)=pq$$

Categorical Distribution

- *Bernoulli distribution often generalized to categorical distribution for more than two possible values. In this case, for n possible outcomes, n-1 values of p are given (the n-th being $1 - \sum_{i=1}^{n-1} p_i$)*



Binomial distribution

- *Probability of k out of N items being Y*
- *Probability of one out of two items being Y [order not important!]*
- *Probability of k out of N items being Y [order not important!]*

$$P(k=1\,|\,N=1)=p$$

$$P(k=1\,|\,N=2)=\frac{pq+qp}{p^2+pq+qp+q^2}=2pq$$

$$P(k\,|\,N)=\frac{N!}{k!(N-k)!}p^k q^{N-k}$$

Probability that the selected event is obtained k times out of the total of N trials.

Probabability that something other than the chosen event will occur in all the other trials.

The "combination" expression, which is the permutation relationship (the number of ways to get k occurrences of the selected event) divided by k ! (the number of different orders in which the k events could be chosen, assuming they are distinguishable).

Limit of rare events

- *For $N \to \infty$ with $p \to 0$ so that Np stays finite and positive, the Binomial distribution takes the form of a Poisson distribution*

$$P(K|N,p) = \frac{N!}{(N-K)!\,K!}\,p^K(1-p)^{N-K} =$$

$$= \frac{N!}{(N-K)!\,N^K}\,\frac{(Np)^K}{K!}\,/(1-p)^N\,(1-p)^{-K} =$$

LET'S DEFINE $\lambda = N \cdot p$

$$= \frac{\lambda^K}{K!}\,\frac{N!}{(N-K)!\,N^K}\,\left(1-\frac{\lambda}{N}\right)^N\left(1-\frac{\lambda}{N}\right)^{-K} \to$$

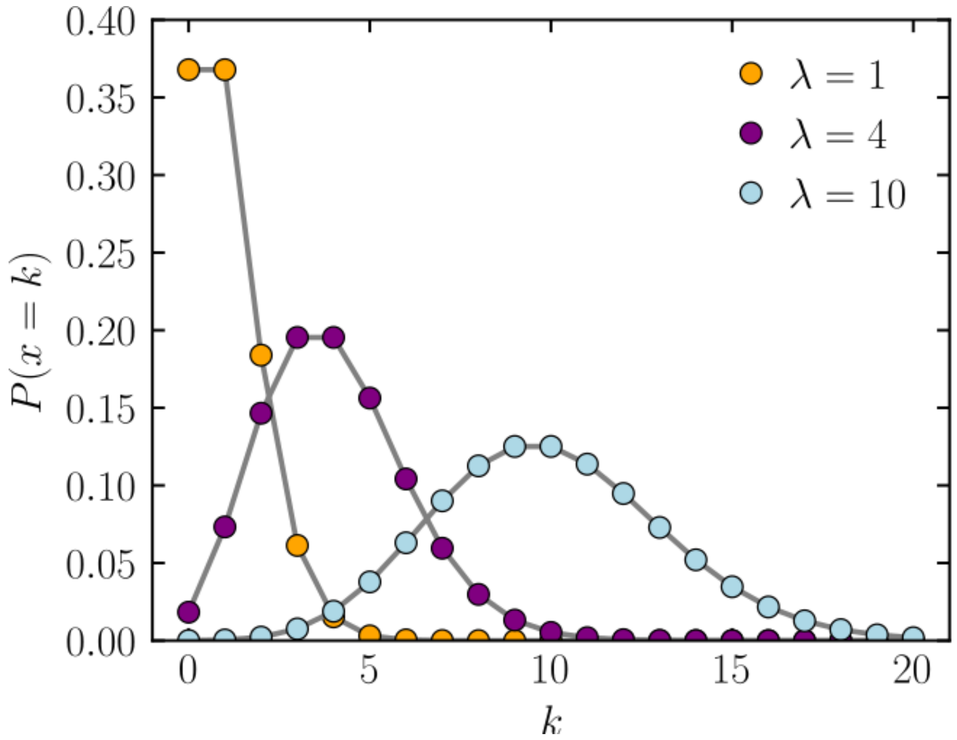
$\frac{N!}{(N-K)!\,N^K} \xrightarrow{N\to\infty} 1$, $\left(1-\frac{\lambda}{N}\right)^N \xrightarrow{N\to\infty} e^{-\lambda}$, $\left(1-\frac{\lambda}{N}\right)^{-K} \xrightarrow{N\to\infty} 1$

$$\xrightarrow[N\to\infty]{} \frac{\lambda^K}{K!}\,e^{-\lambda}$$

Poisson Distribution

$$f(k;\lambda) = \Pr(X=k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

- *k is the unknown (the outcome of our experiment counting). It takes integer values by construction*
- *λ is the parameter determining the distribution shape and it is related to the most probable outcome of our counting experiment. It might be an integer, but in general it is a real number*



Poisson Expectation Value

$$E[k|\lambda] = \frac{0 \cdot P(0|\lambda) + 1 \cdot P(1|\lambda) + 2 \cdot P(2|\lambda) + \dots}{P(0|\lambda) + P(1|\lambda) + P(2|\lambda) + \dots} =$$

$$= \frac{\sum_{k=0}^{\infty} \lambda P(k|\lambda)}{\sum_{k=0}^{\infty} P(k|\lambda)} =$$

$$= \frac{e^{-\lambda} \sum_{k=0}^{\infty} \frac{k \lambda^k}{k!}}{e^{-\lambda} \sum \frac{\lambda^k}{k!}} = \frac{\lambda \sum_{k=1}^{\infty} \frac{k \cdot \lambda^{k-1}}{k \cdot (k-1)!}}{e^{\lambda}} = \frac{\lambda e^{\lambda}}{e^{\lambda}} = \boxed{\lambda}$$

Poisson Variance

$$E[(k - E[k])^2] = E[k^2] - E[k]^2 = \lambda^2 + \lambda - \lambda^2 = \lambda$$

$$E[k^2] = \frac{\sum_{k=0}^{\infty} \frac{e^{-\lambda}\lambda^k k^2}{k!}}{\underbrace{\left(\sum_{k=0}^{\infty} \frac{\lambda^k}{k!}\right) e^{-\lambda}}_{e^{\lambda}}} = \sum_{k=1}^{\infty} \frac{e^{-\lambda}\lambda^k k}{(k-1)!} =$$

$$= \lambda e^{-\lambda}\left[\sum_{k=1}^{\infty} \frac{\lambda^{k-1}(k-1)}{(k-1)!} + \underbrace{\sum_{k=1}^{\infty} \frac{\lambda^{k-1}}{(k-1)!}}_{e^{\lambda}}\right] =$$

$$= \lambda e^{-\lambda}\left[\lambda \underbrace{\sum_{k=2} \frac{\lambda^{k-2}}{(k-2)!}}_{e^{\lambda}} + e^{\lambda}\right] = \lambda(\lambda+1) = \boxed{\lambda^2+\lambda}$$

The limit of large λ

$$\ln(P(k|\lambda)) = \ln\left(\frac{\lambda^k e^{-\lambda}}{k!}\right) \simeq \ln\left(\frac{\lambda^k e^{-\lambda}}{k^k e^{-k}\sqrt{2\pi k}}\right) =$$

STIRLING'S APPROXIMATION $\quad x! \simeq x^x e^{-x}\sqrt{2\pi x}$

$$= k\ln\lambda - \lambda - k\ln k + k - \ln\sqrt{2\pi k} =$$

$$= \dots = -\frac{y^2}{2\lambda} + \frac{y^3}{6\lambda^2} - \ln\sqrt{2\pi(y+\lambda)} \simeq$$

$y = k - \lambda$ WITH $\lambda \to \infty$, $k \sim O(\lambda)$ $\Rightarrow y \ll \lambda$

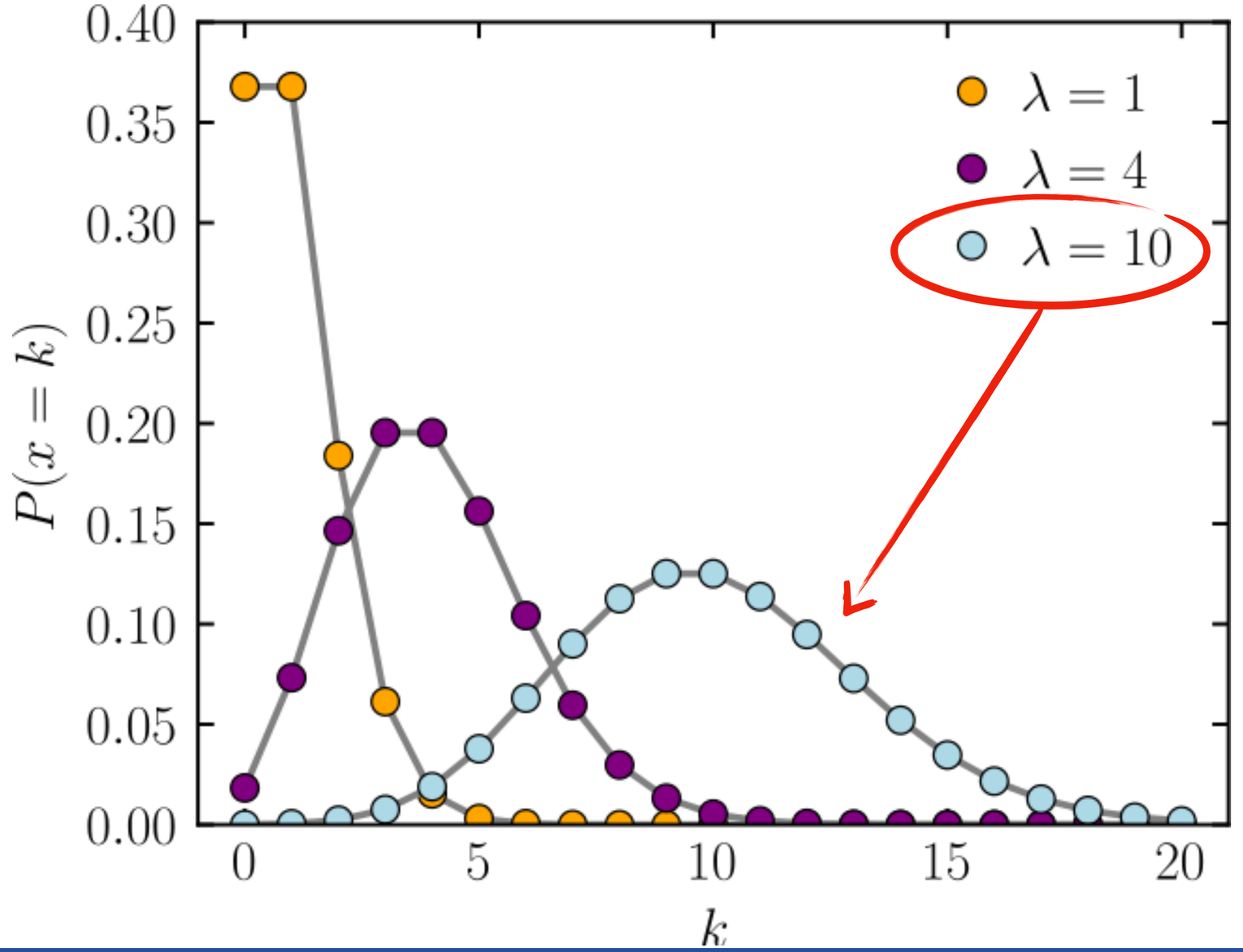
$$\simeq -\frac{y^2}{2\lambda}$$

$$P(y|\lambda) = e^{-\frac{y^2}{2\lambda}}$$

$\ln(1+\varepsilon) \simeq \varepsilon - \varepsilon^2 + \frac{\varepsilon^3}{3} \dots$

GAUSSIAN

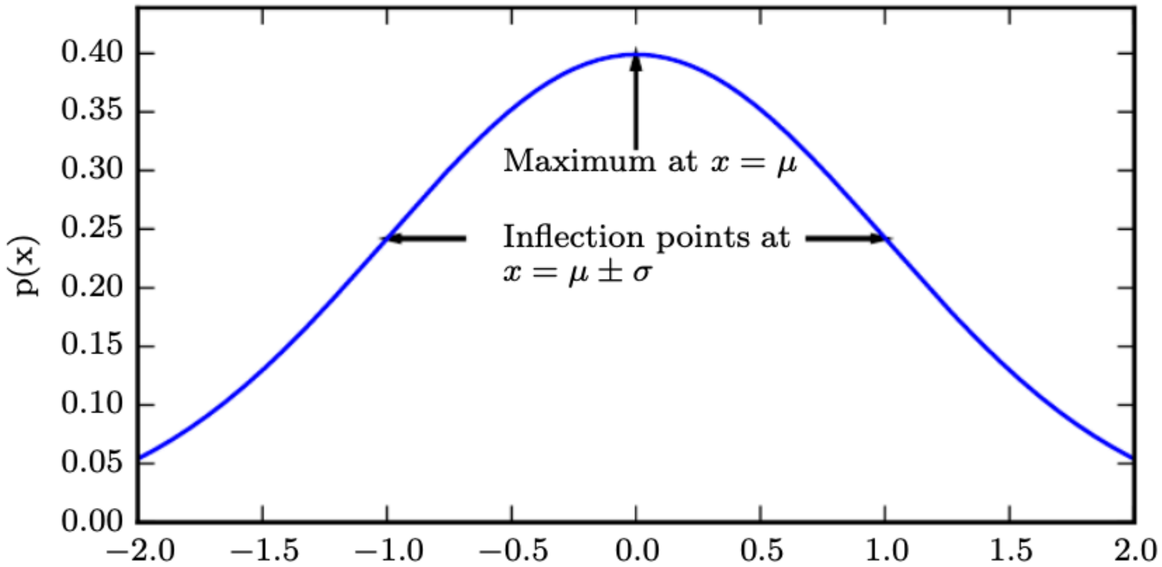
How Large is Large?



Gaussian

- One of the most frequently encountered distributions
  - Rules many stochastic problems, because it is the limit distribution of other pdfs
  - Two parameters
    - μ determines the position of the mean/maximum
    - σ determines the width around the mean

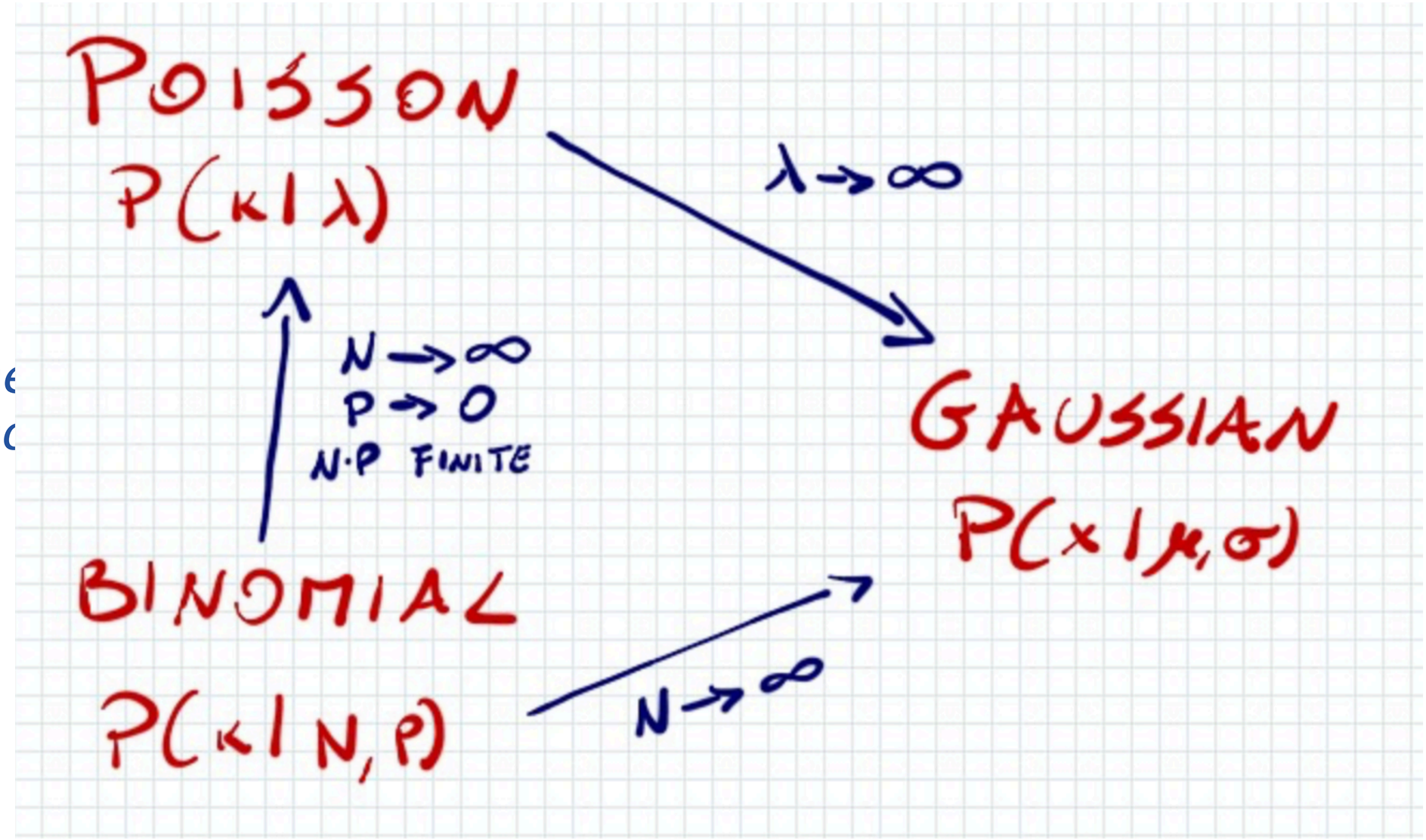
$$G(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



$$\mathbb{E}[x] = \mu \qquad Var(x) = \sigma^2$$

Gaussian as a large-trial limit

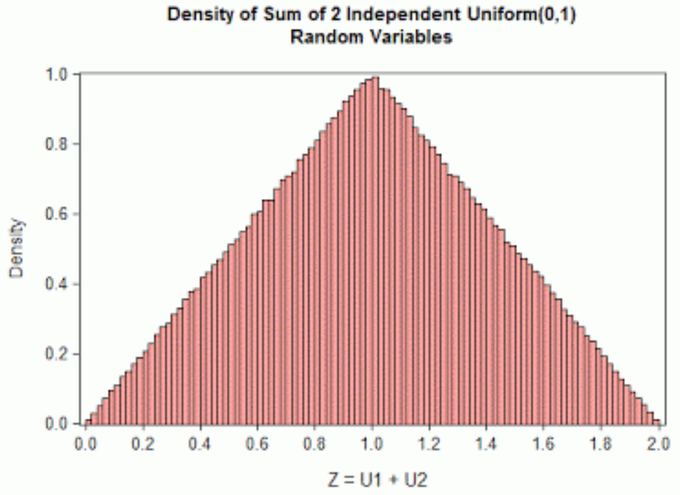
- *The*
  enc

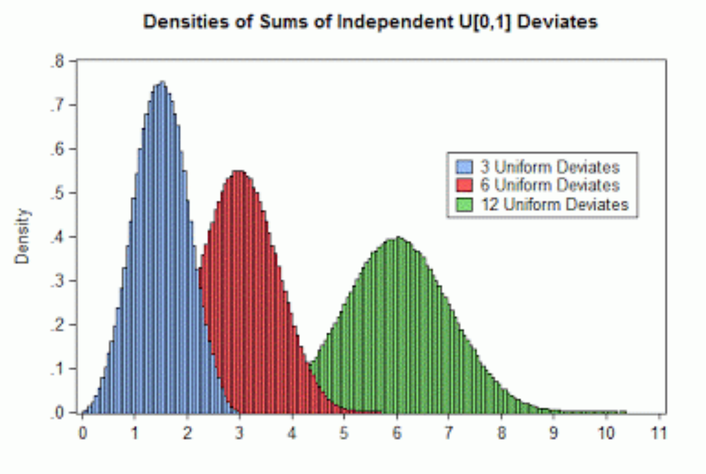


Gaussian as a large-trial limit

- *The central limit theorem establishes the role of the Gaussian distribution as the asymptotic limit of a much broader class of problems*

In probability theory, the central limit theorem establishes that, in many situations, when independent random variables are summe up, their properly normalized sum tends toward a normal distribution even if the original variables themselves are not normally distributed. (from Wikipedia)





- *In practice, in a counting experiment one has to deal with*
  - *The intrinsic variation (statistical uncertainty) associated with the spread of the distribution (Poisson, Binomial, etc.)*
  - *The systematic uncertainty, associated to the uncertainty on the knowledge of the expectation. This is typically the result of many contributions -> it tends to have a Gaussian behavior*

Recap

Function	Distribution	E[x]	Var[x]
Poisson	$P(k \mid \lambda) = \frac{e^{-\lambda}\lambda^k}{k!}$	λ	λ
Binomial	$P(k \mid p, N) = \frac{N!}{k!(N-k)!} p^k (1-p)^{N-k}$	pN	pN(1-p)
Gaussian	$G(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	μ	σ^2

Machine Learning

- *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as descent. We describe how to combine various algorithm components, such as measured by P, improves with experience E."*

- *A task T: classification, regression, etc.*

- *A performance measure P (typically on independent test dataset):*
  - *Example: for classifiers, accuracy/error rate, i.e., fraction of examples correctly/ incorrectly classified*

- *An Experience E, provided as a training dataset on which an algorithm can try to improve its task solving skill*
  - *Supervised learning: learn to reproduce the ground-truth (e.g., flags in classification)*
  - *Unsupervised learning: learn the pdf of the dataset and use it to make probabilistic statements about new data (e.g., anomaly detection)*

learn some quantity x_i

from the other N-1 x_j

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(\mathrm{x}_i \mid \mathrm{x}_1, \ldots, \mathrm{x}_{i-1}).$$

Machine Learning

- *Boundary between supervised and unsupervised are not so strict: the same approach can be used for supervised or unsupervised learning*

Learning the pdf of the data $p(\mathbf{x})$ is an unsupervised task

But it can be broken down in supervised tasks

from the other N-1 x_j

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(\mathbf{x}_i \mid \mathbf{x}_1, \ldots, \mathbf{x}_{i-1}).$$

learn some quantity x_i

Learning some y from some $\mathbf{x}$, i.e., learning $p(y \mid \mathbf{x})$ is a supervised task

But this task can be approached with unsupervised technique, using the probability chain rule

learning the full pdf

$$p(y \mid \mathbf{x}) = \frac{p(\mathbf{x}, y)}{\sum_y p(\mathbf{x}, y')}$$

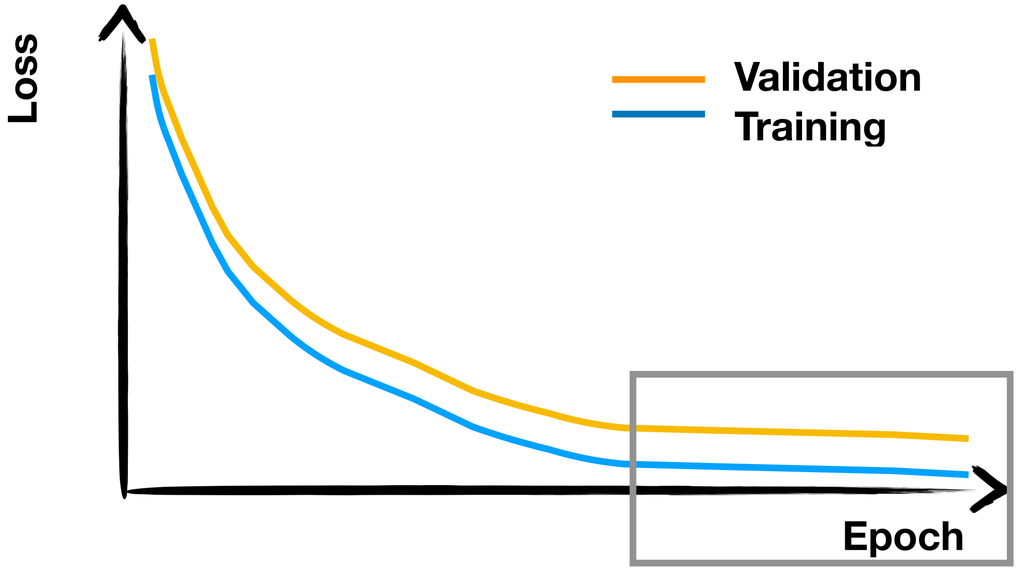
learning the pdf $p(\mathbf{x})$

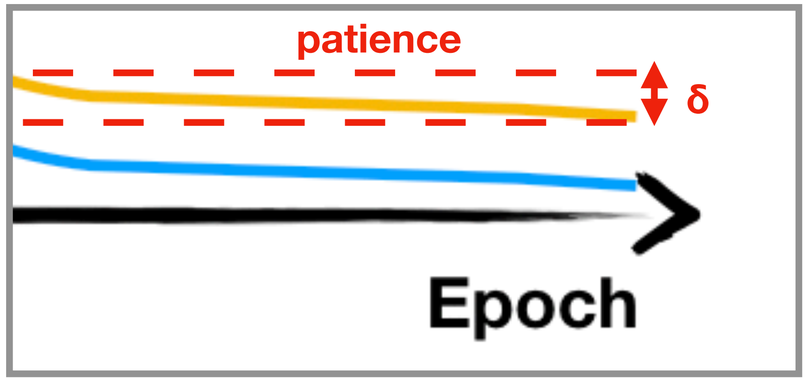
Dataset Split

- *Split your sample in three:*
  - *Training: the biggest chunk, where you learn from*
  - *Validation: an auxiliary dataset to verify generalization and prevent overtraining*
  - *Test: the dataset for the final independent check*

Learning history & Early Stopping

- *Train across multiple epochs*
  - *1 epoch = going once through the full dataset*
- *Use small batches (64, 128, etc)*
- *Check your training history*
  - *on the training data (training loss)*
  - *and the validation ones (validation loss)*
- *Use an objective algorithm to stop (e.g., early stopping)*

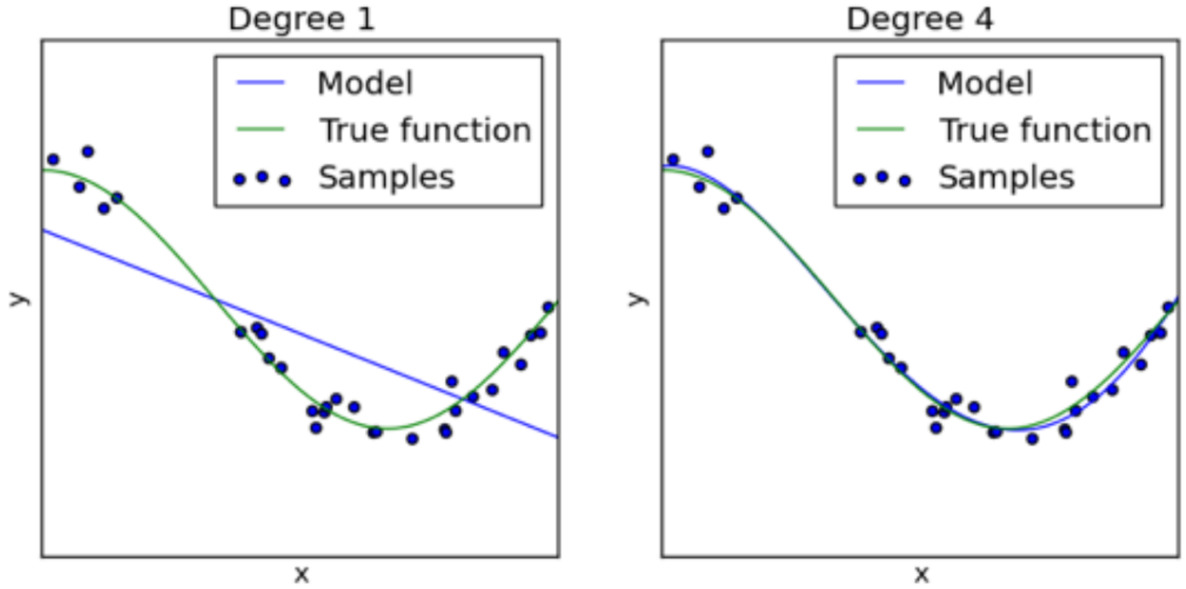


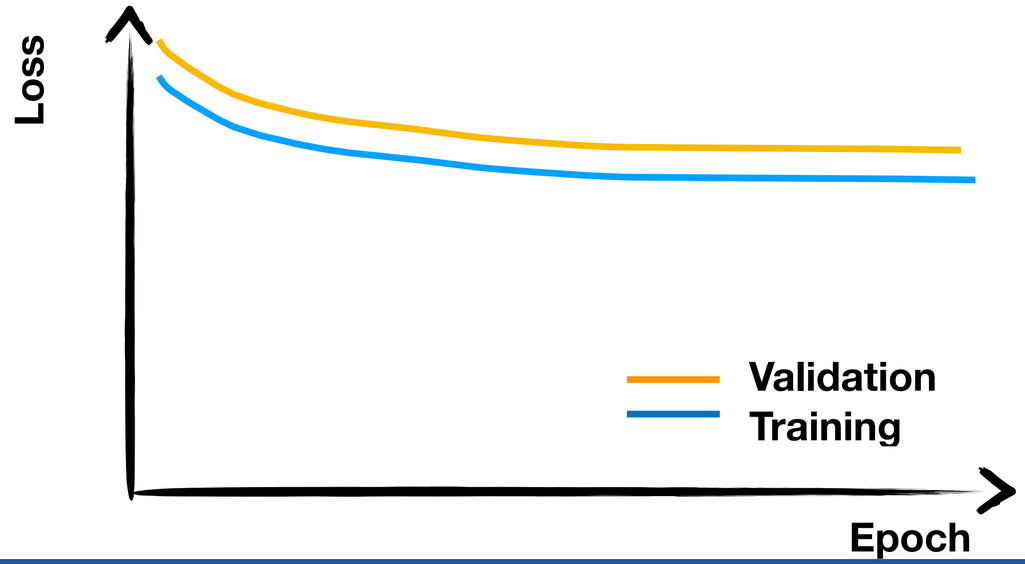


EARLY TOPPING: stop the train if the validation loss didn't change more than δ in the last n epochs (patience)

Undertraining

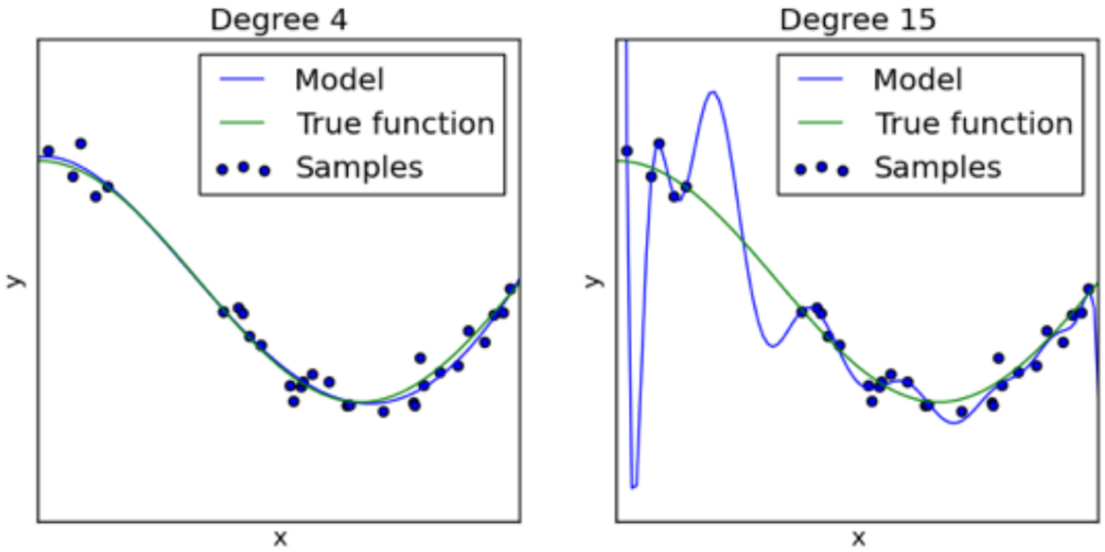
- *If your model has not enough flexibility, it will not be able to describe the data*
- *The training and validation loss will be close, but their value will not decrease*
- *The model is said to be underfitting, or being **biased***

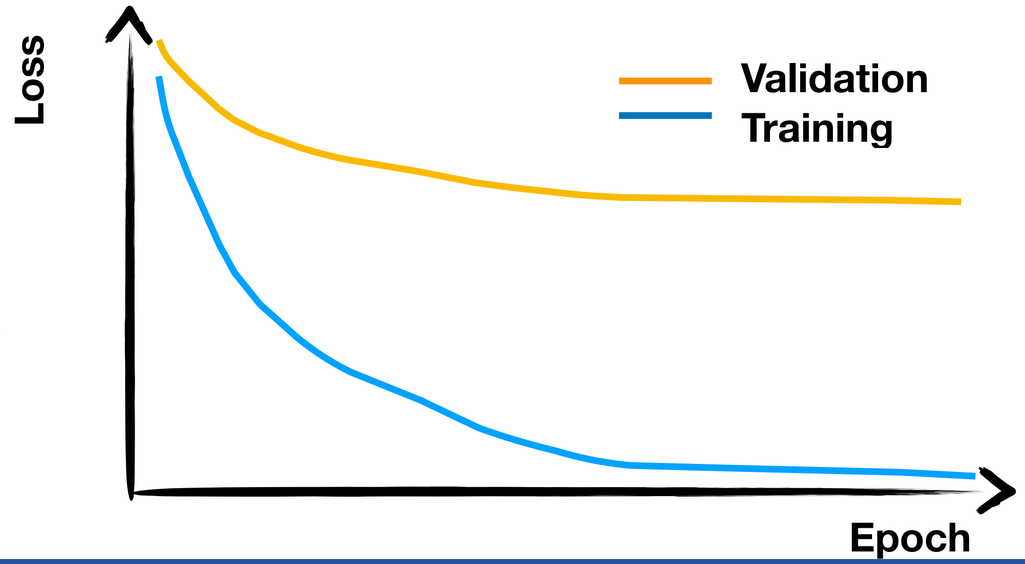




Overtraining

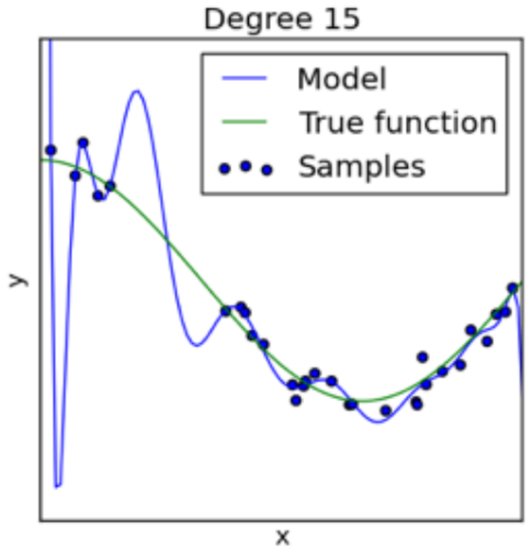
- *Your model can learn too much of your training dataset*
  - *e.g., its statistical fluctuations*
- *Such an overfitted model would not generalise*
- *So, its description of the validation dataset will be bad (i.e., **the mode doesn't generalise**)*
- *This is typically highlighted by a divergence of the training and validation loss*

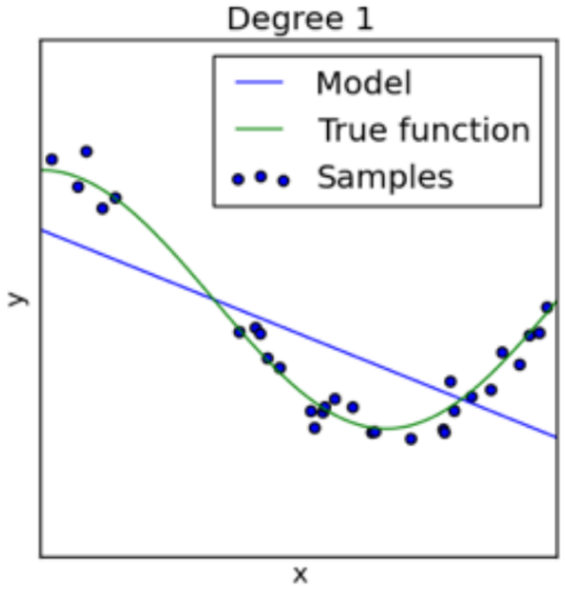




Bias vs Variance

- ***A model would underfit if too simple:*** *it will not be able to model the mean value*
- ***A model would overfit if too complex:*** *it will reproduce the mean value, but it will underestimate the variance of the data*
- *The* ***generalization error*** *is the error made going from the training sample to another sample (e.g., the test sample)*





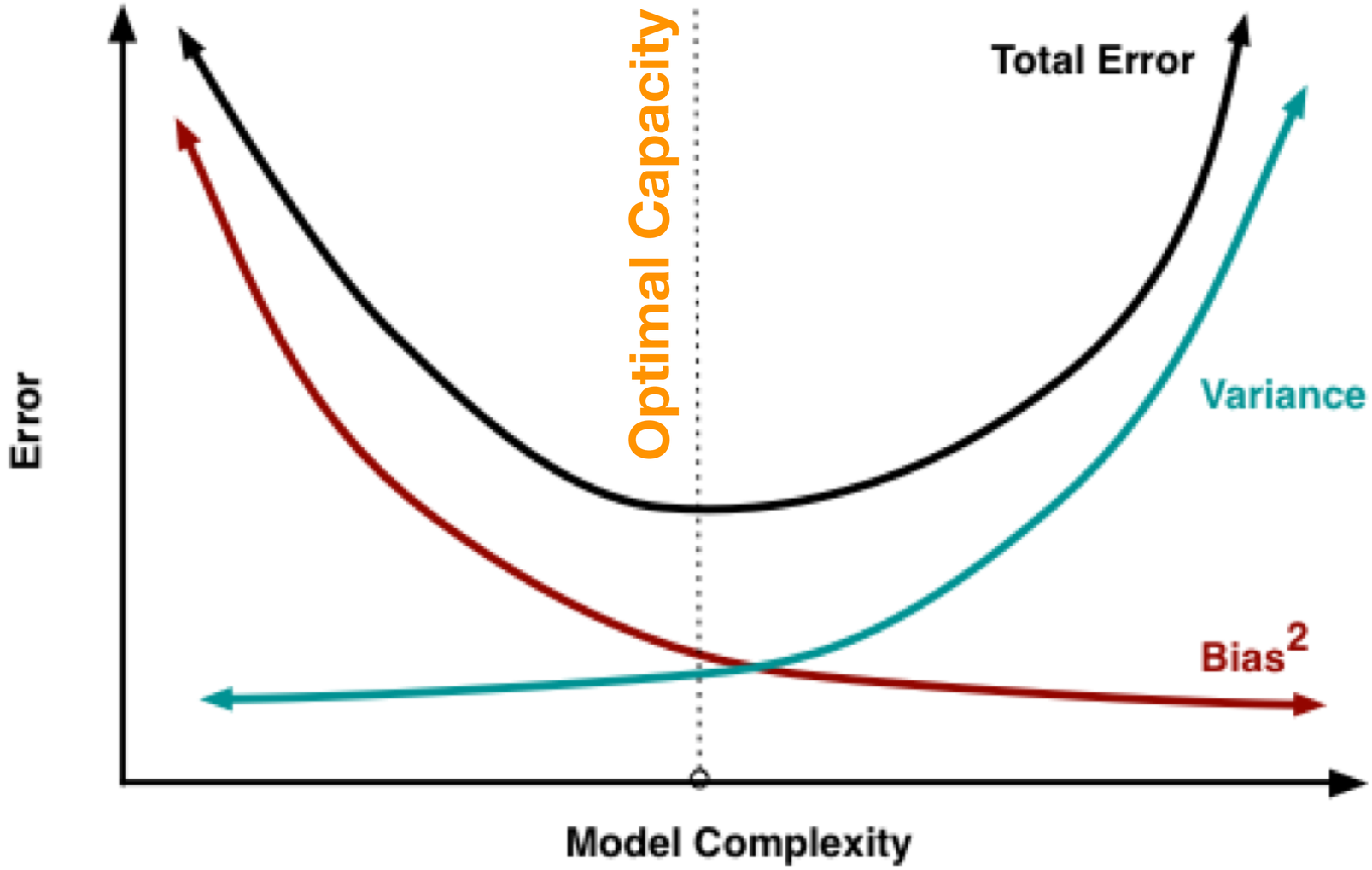
Bias vs Variance

- *Generalization error can be written as the sum of three terms:*
  - *The intrinsic statistical noise in the data*
  - *the bias wrt the mean*
  - *the variance of the prediction around the mean*

$$E[(y - h(x))^2] = E[(y - \bar{y})^2] + (\bar{y} - \bar{h}(x))^2 + E[(h(x) - \bar{h}(x))^2]$$

Noise **Bias Squared** **Variance**

Bias vs Variance



From theory to Practice

- *You can learn all Deep Learning theory, but an effective training requires skills and experience that are linked to your technical insight*
- *Many things can go wrong in training, and many of them have nothing to do with deep learning*
  - *Overflows, Underflows, out-of-memory, slow code due to ineffective I/O when handling the data*
- *Even if the training works, many things can be done better*
  - *Optimize I/O (data parallelism), distribute training on multiple GPUs, etc.*

Few things to keep in mind

- *PROBLEM: You are training on hardware that comes with limited numerical representation (e.g., 64 bits)*
  - *This implies rounding error, that could have large effects when training or using the algorithm in practice*
    - *Underflow: too small numbers are turned into 0, which kills your gradient and stop your training or give error (e.g., 1/0 -> NaN)*
    - *Overflow: number is too big -> NaN. And the following mathematic is compromised*
  - *Example: softmax activation function, used ad the end of multi class classifier (takes a vector of scores and turns them into probabilities)*

$$\mathrm{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}$$

You can avoid NaN evaluating the function at z = x -max(x)

Few things to keep in mind

- *PROBLEM: You are looking for an optimal weight configuration computing gradients. You want to avoid 0 gradient*
  - *This is called conditioning, i.e., how rapidly a function changes wrt its input*
  - *Example: matrix inversion. You need to find* $f(x) = A^{-1}x$*, given* A*. Given the* λ *eigenvalues of* A*, the condition number for A is* $\max_{i,j} \left| \frac{\lambda_i}{\lambda_j} \right|$*. Matrix inversion becomes numerically complicated when this number is large, i.e., when there is a big gap between the largest and smallest eigenvalue*
  -

Regularisation in the loss

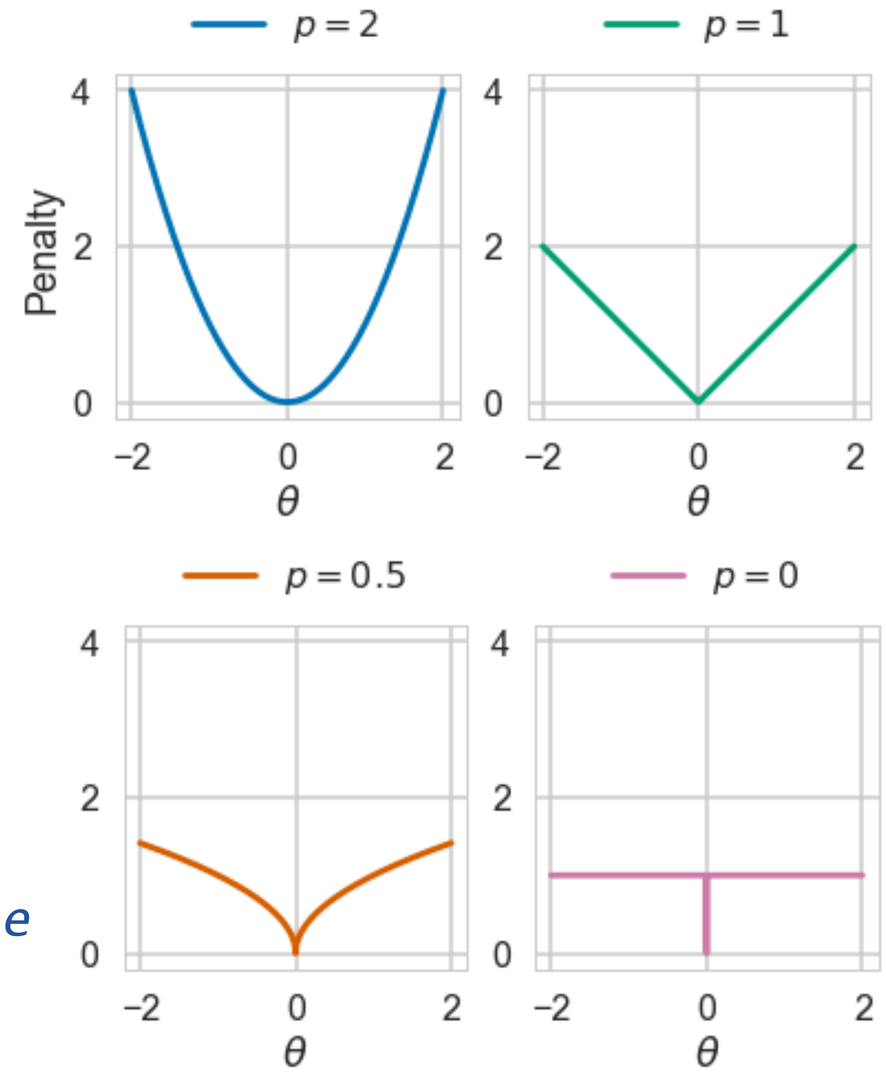
- *Model complexity can be "optimized" when minimizing the loss*
  - *A modified loss is introduced, with a penalty term attache to each model parameter*

$$L_{reg} = L + \Omega(w)$$

- *For instance, Lp regularisation*

$$L_p = \|\mathbf{w}\|^p = \sum_i |w_i|^p$$

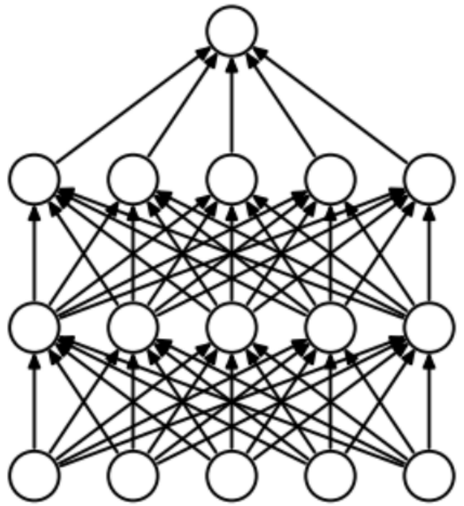
- *The minimisation is a tradeoff between:*
  - *pushing down the 1st term by taking advantage of the parameters*
  - *pushing down the 2nd term by switching off the parameters*



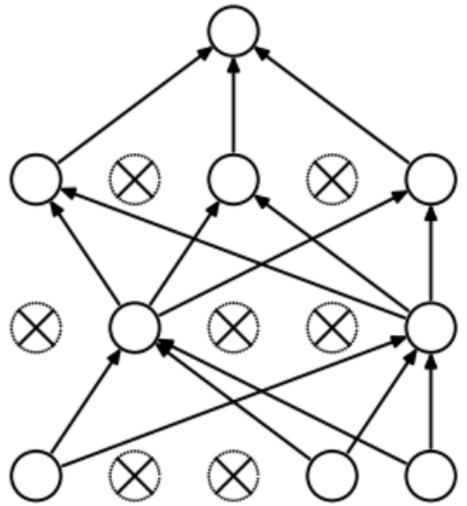
https://openreview.net/pdf?id=H1Y8hhg0b

Regularisation with Drop out

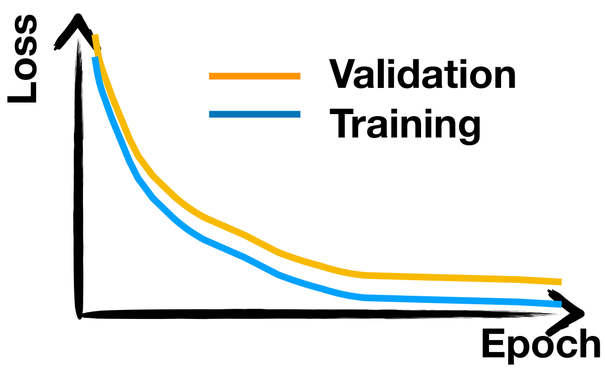
- *A special kind of layer, introduced for regularisation purpose*
- *Randomly drop links between neurons, with probability p*
- *The connections are re-established during the validation and inference steps*
- *Typical sign of it: invert hierarchy between training and validation loss*

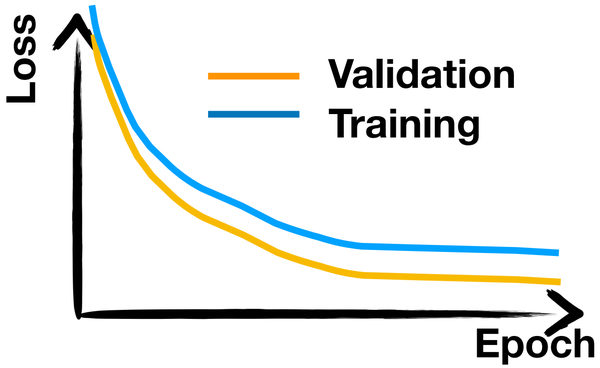


(a) Standard Neural Net



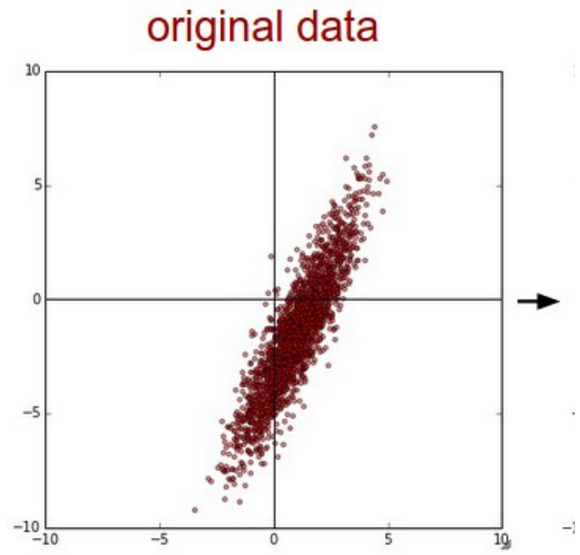
(b) After applying dropout.

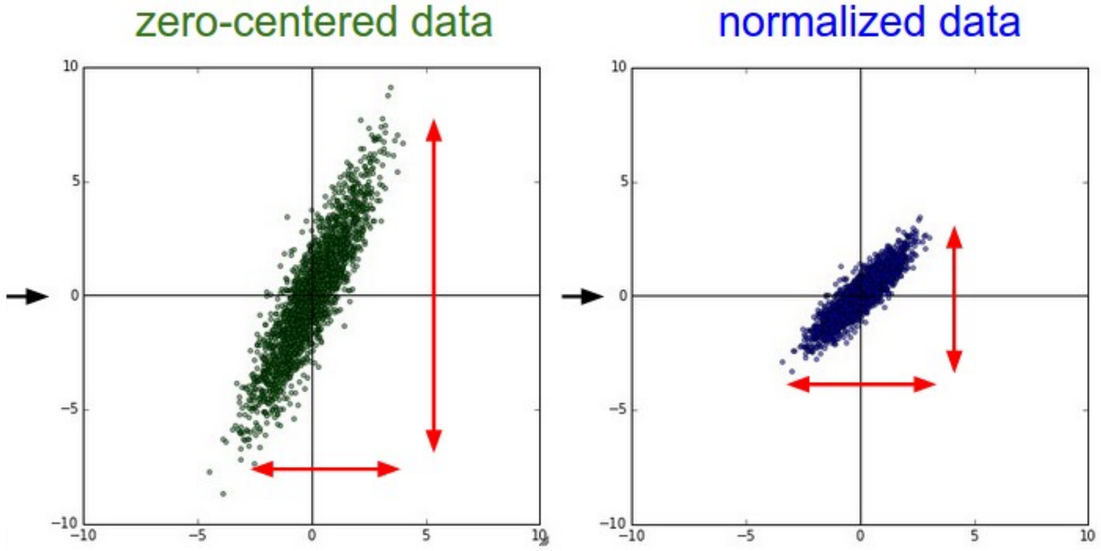




Data normalization

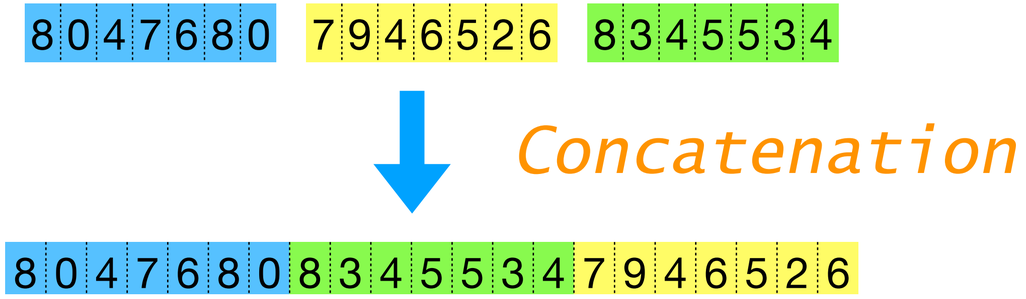
- *It is good practice to give normalized inputs to a layer*
  - *With all inputs having the same order of magnitude, all weights are equal important in the gradient*
  - *Prevents explosion of the loss function*
- *This can be done automatically with BatchNormalization*
  - *non-learnable shift and scale parameters, adjusted batch by batch*

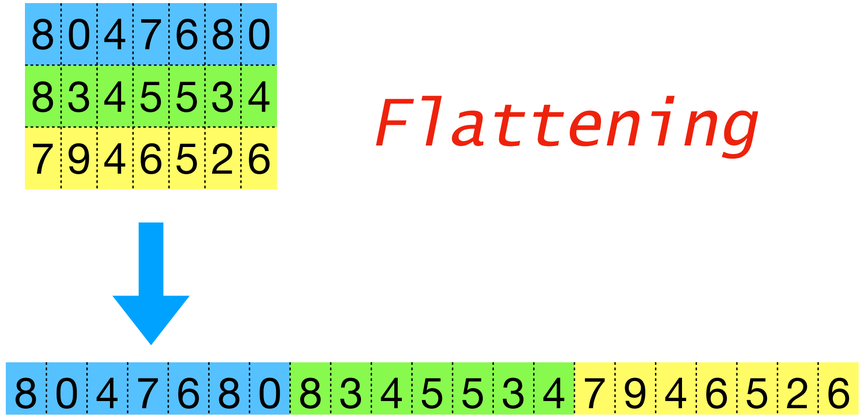




Array Manipulation

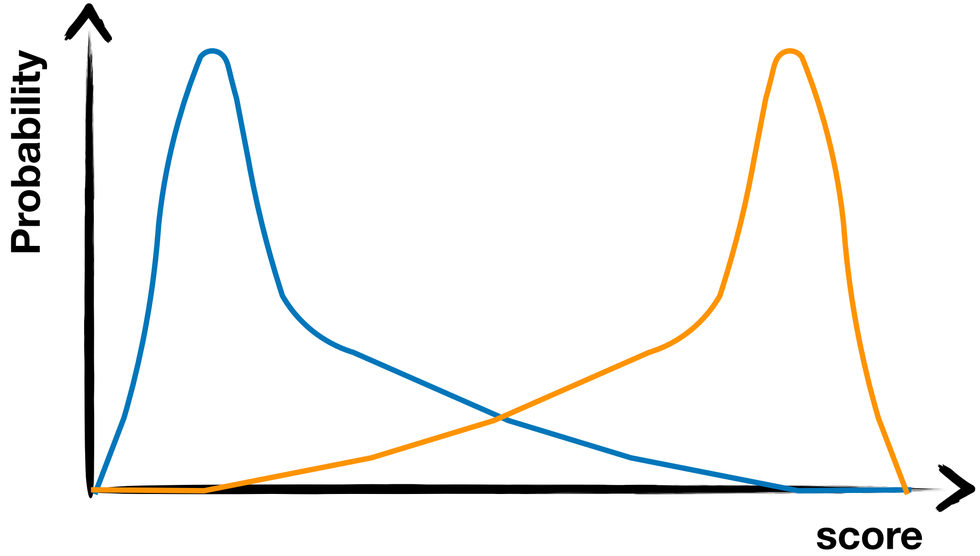
- Dense NN architectures can be made more complex
  - Multiple inputs
  - Multiple outputs
  - Different networks branches
- This is possible thanks to layer-manipulation layers
  - Add, Subtract, etc.
  - Concatenation
  - Flattening
- All these operations are usually provided with NN training libraries

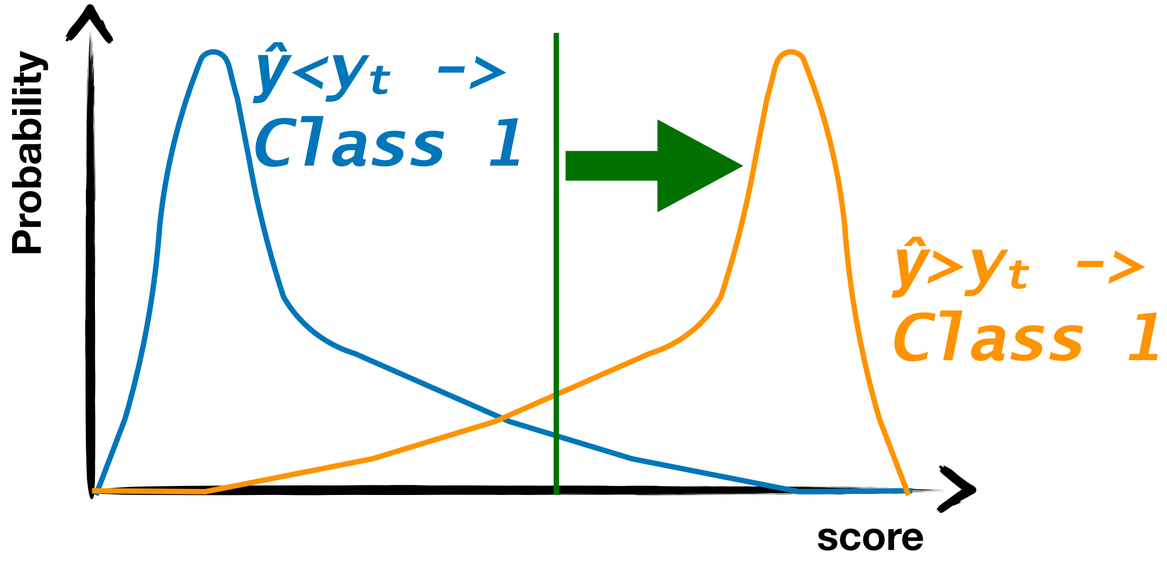




Classifier metrics

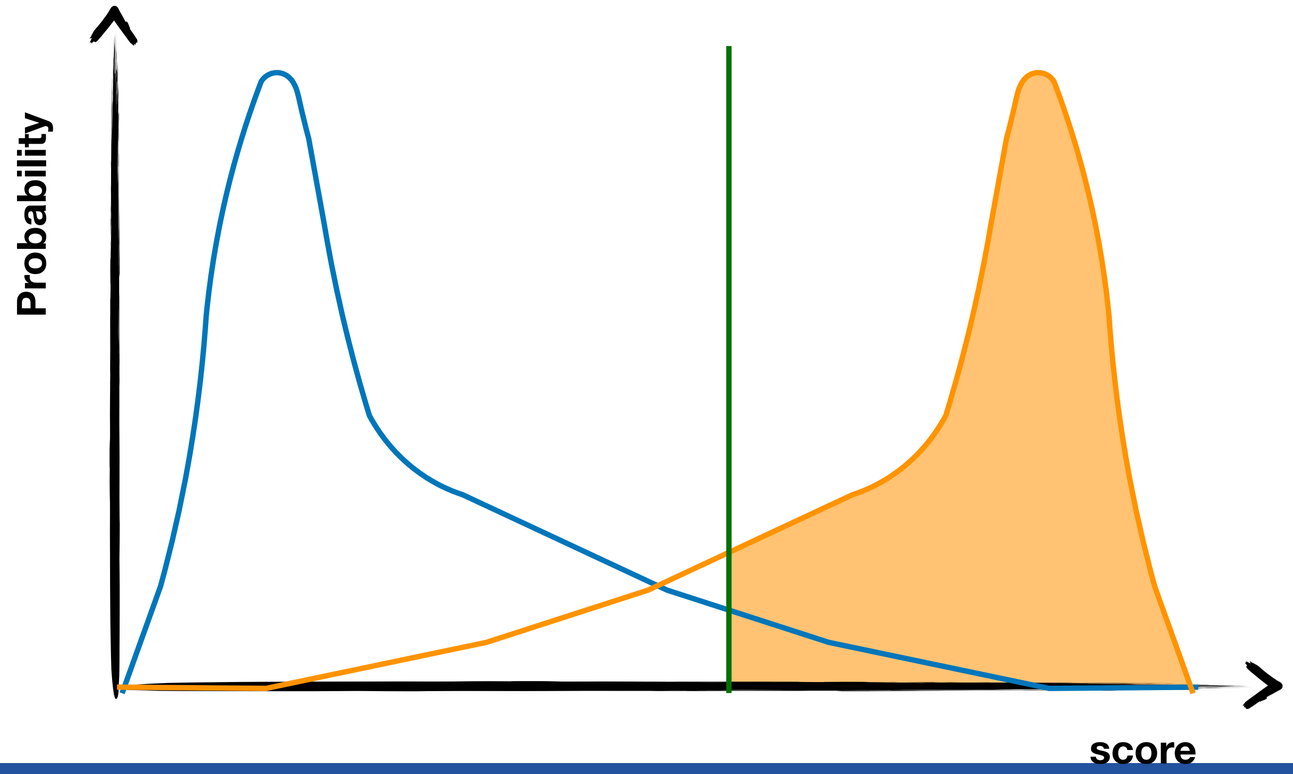
- Consider a binary classifier
- Its output $\hat{y}$ is a number in [0,1]
- If well trained, value should be close to 0 (1) for class-0 (class-1) examples
- One usually defines a threshold y_t such that:
  - $\hat{y}>y_t$ -> Class 1
  - $\hat{y}<y_t$ -> Class 0





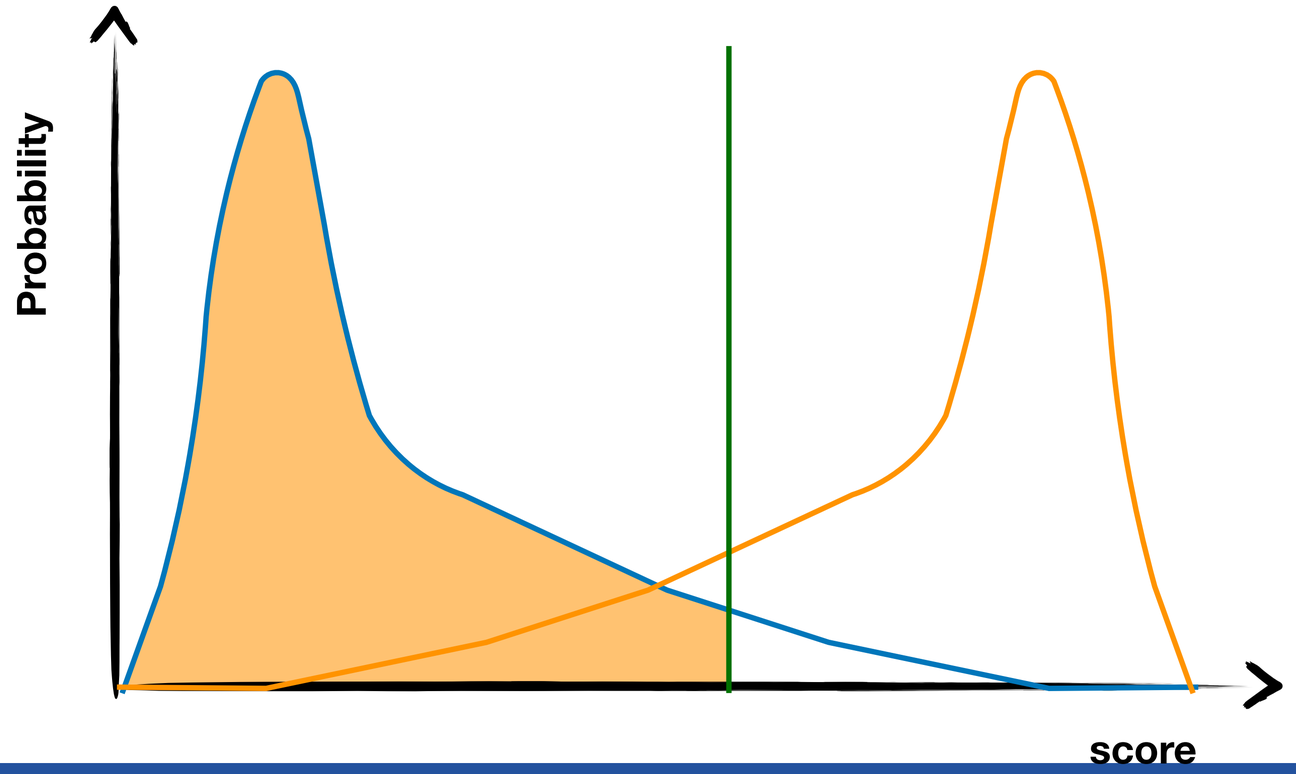
Classifier metrics

- A given threefold defines the following qualities
  - True-positives: Class-1 events above the threshold
  - True-negatives: Class-0 events below the threshold
  - False-positives: Class-0 events above the threshold
  - False-negatives: Class-1 events below the threshold



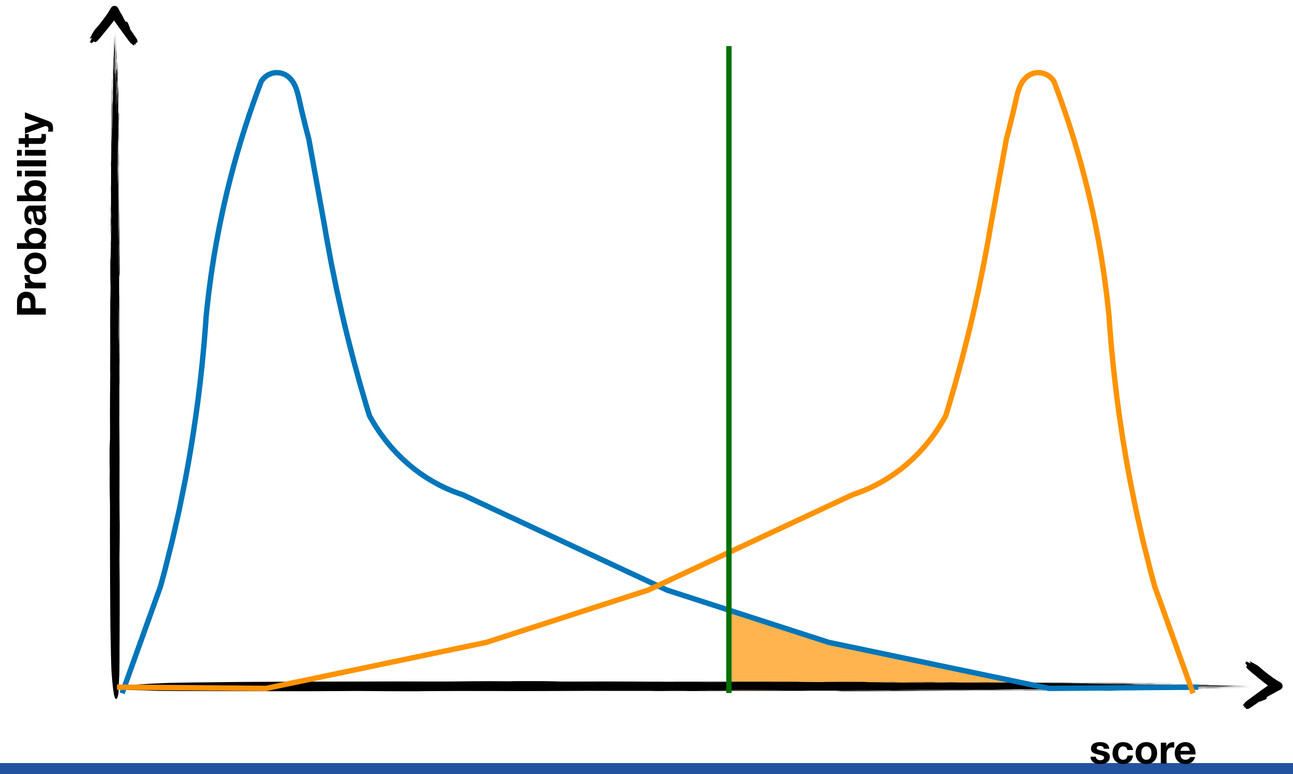
Classifier metrics

- A given threefold defines the following qualities
  - True-positives: Class-1 events above the threshold
  - True-negatives: Class-0 events below the threshold
  - False-positives: Class-0 events above the threshold
  - False-negatives: Class-1 events below the threshold



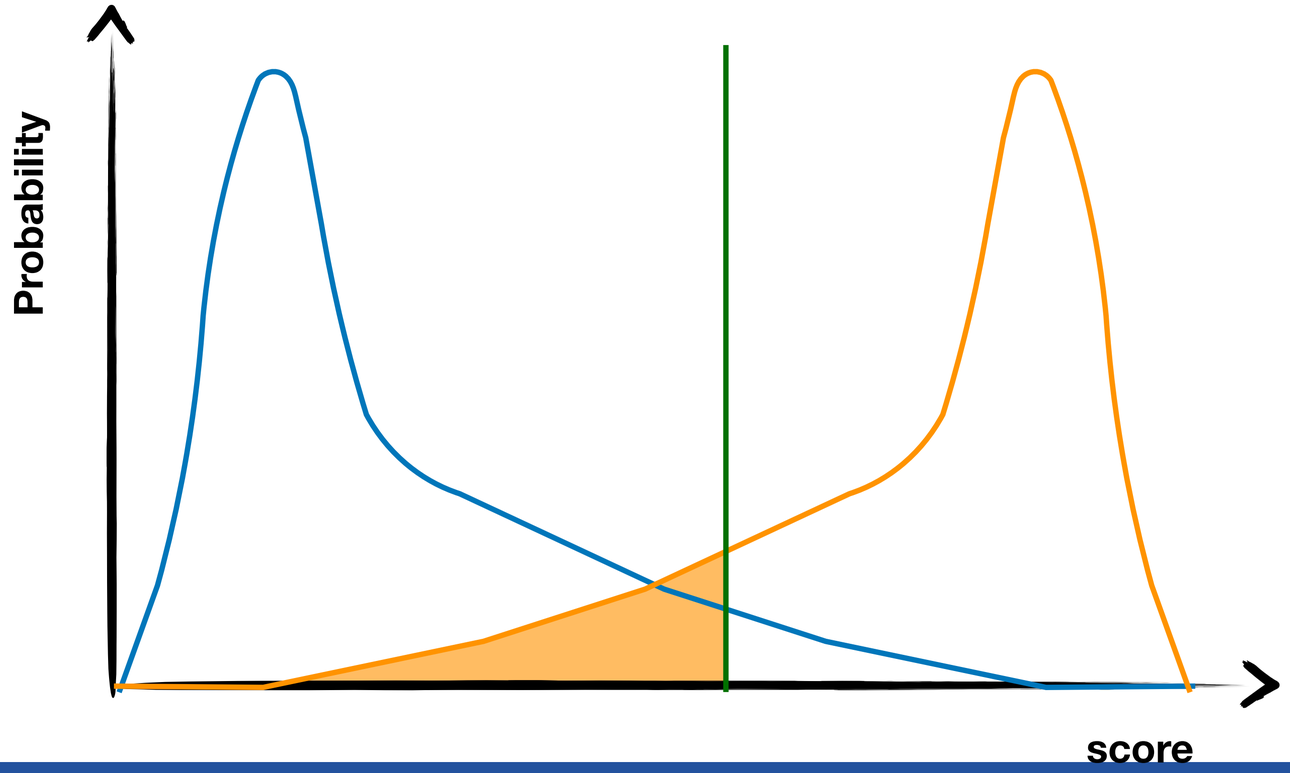
Classifier metrics

- *A given threefold defines the following qualities*
  - *True-positives: Class-1 events above the threshold*
  - *True-negatives: Class-0 events below the threshold*
  - *False-positives: Class-0 events above the threshold*
  - *False-negatives: Class-1 events below the threshold*



Classifier metrics

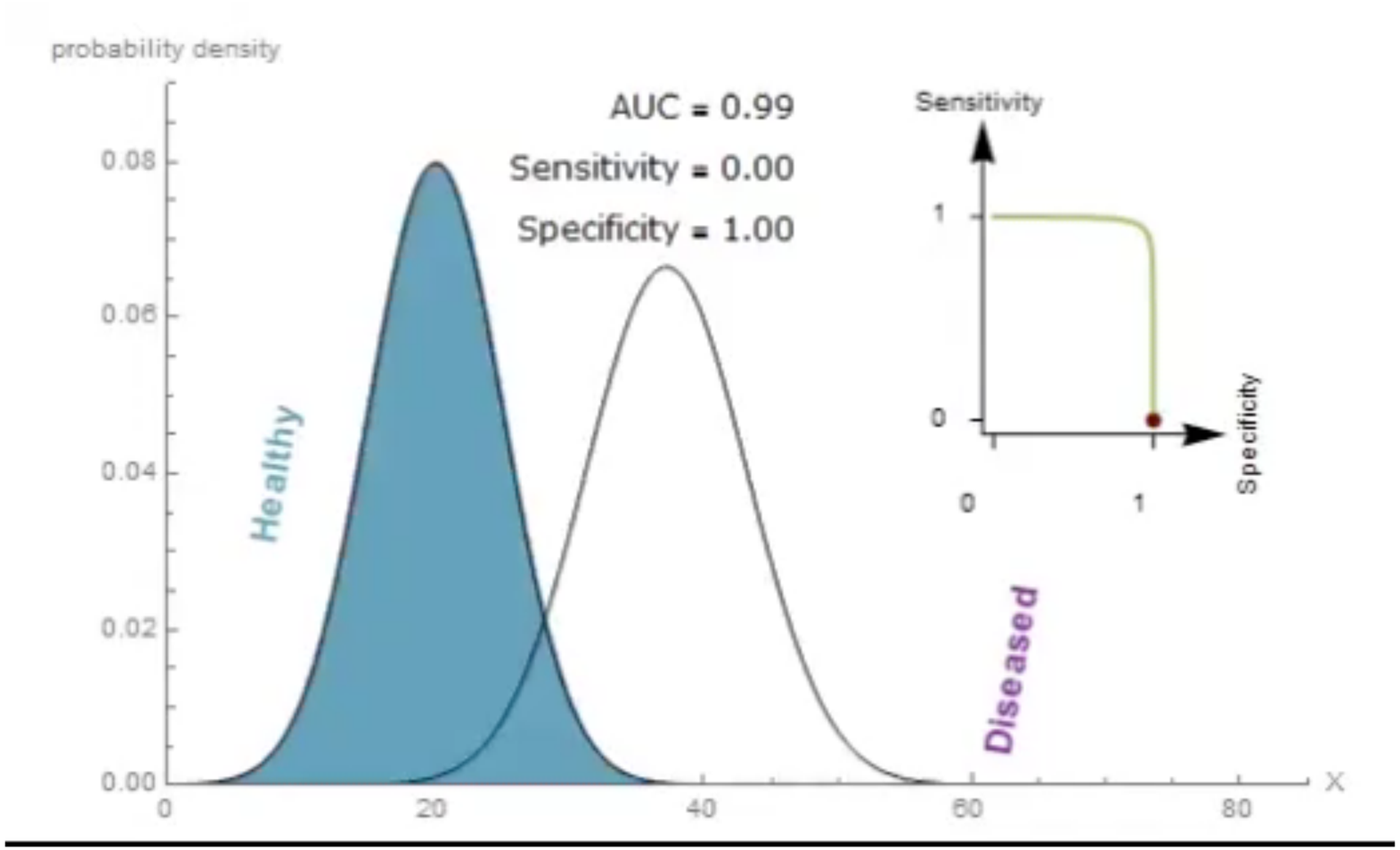
- *A given threefold defines the following qualities*
  - *True-positives: Class-1 events above the threshold*
  - *True-negatives: Class-0 events below the threshold*
  - *False-positives: Class-0 events above the threshold*
  - *False-negatives: Class-1 events below the threshold*



Classifier metrics

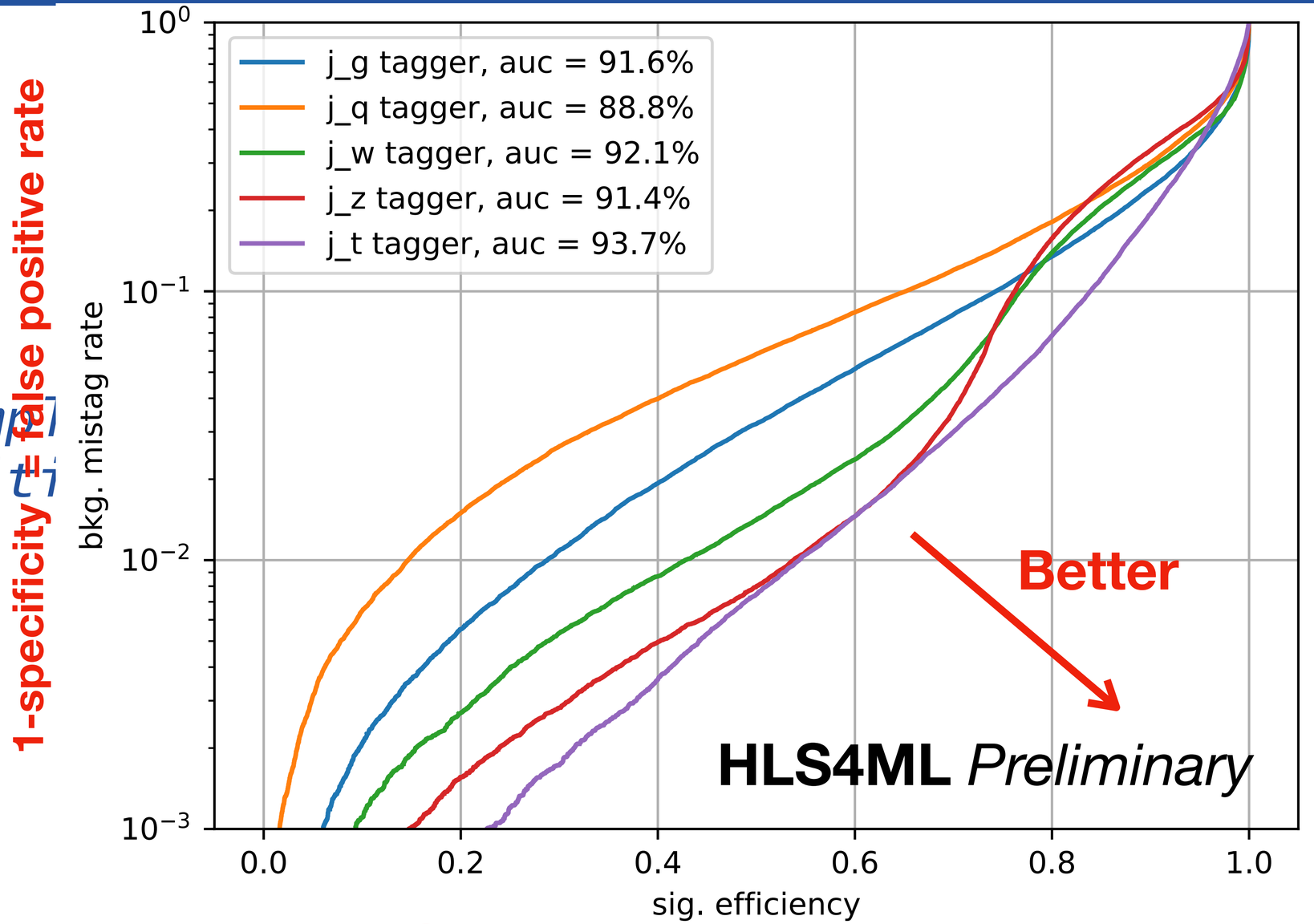
- *Starting ingredients are true positive (TP) and true negative (TN) rates*
- *Accuracy: (TP+TN)/Total*
  - *The fraction of events correctly classified*
- *Sensitivity: TP/(Total positive)*
  - *AKA signal efficiency in HEP*
- *Specificity: TN/(Total negative)*
  - *AKA mistag rate in HEP*

Receiver operating characteristic



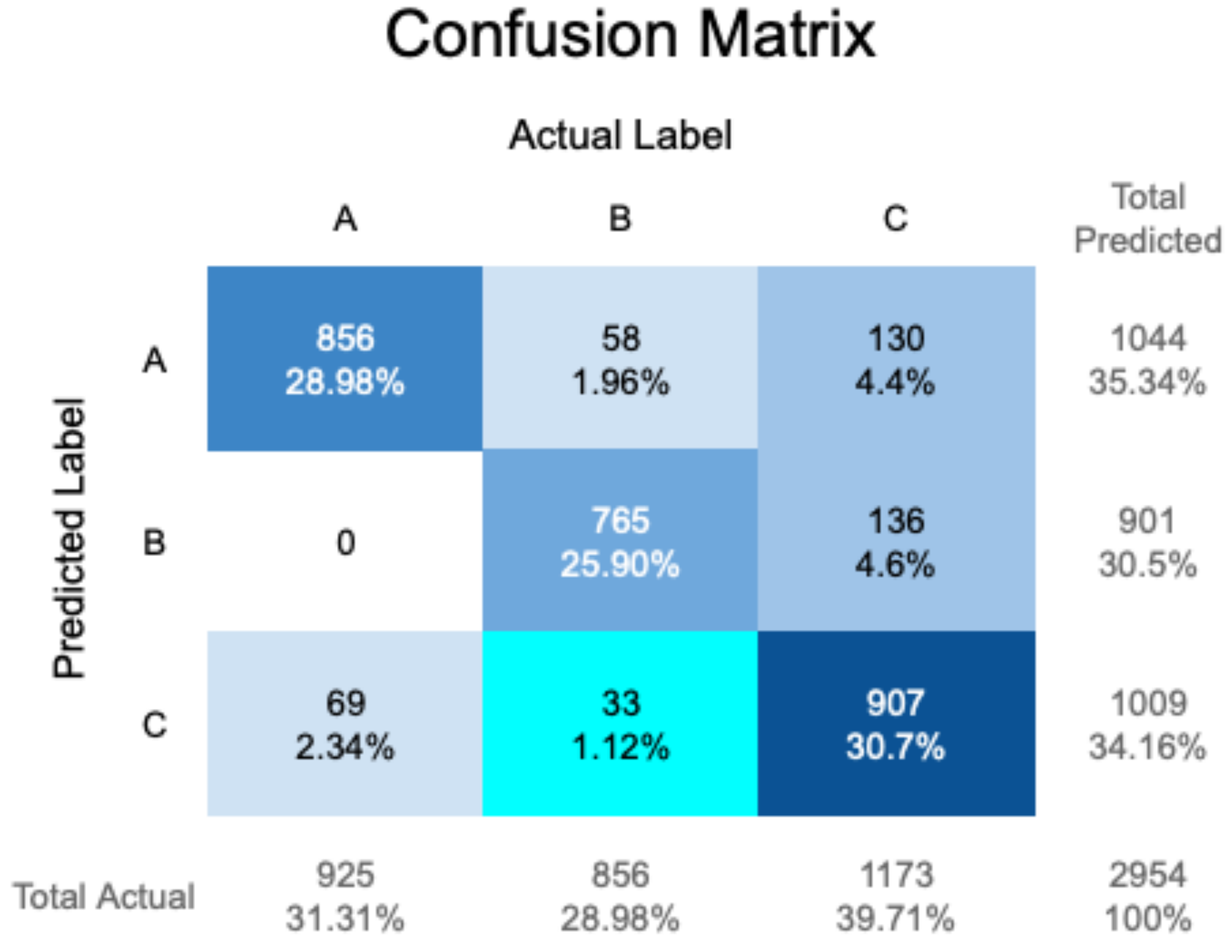
ROC curve

- *Simpl… …ses, multi…*



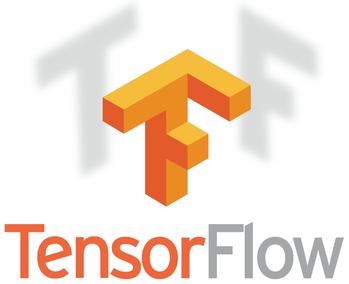
Confusion matrix

- *A very effective visual tool to understand where your classifier works and where it doesn't*
  - *Fill a matrix A_{ij} the fraction of examples of class C_i classified as class C_j*
  - *Ideally, one would like to see the identity matrix*
  - *In practise, one can look for large off-diagonal elements to see which classification is problematic*



Training Libraries

- Many solutions exist. Most popular softwares live in a python ecosystem
  - Google's TensorFlow
  - Facebook's Pytorch
  - Apache MXnet
- All of them integrated in a data science ecosystem
  - with numpy, scikit, etc.
- Convenient libraries built on top, with pre-coded ingredients
  - Keras for TF (this is what we will be using)

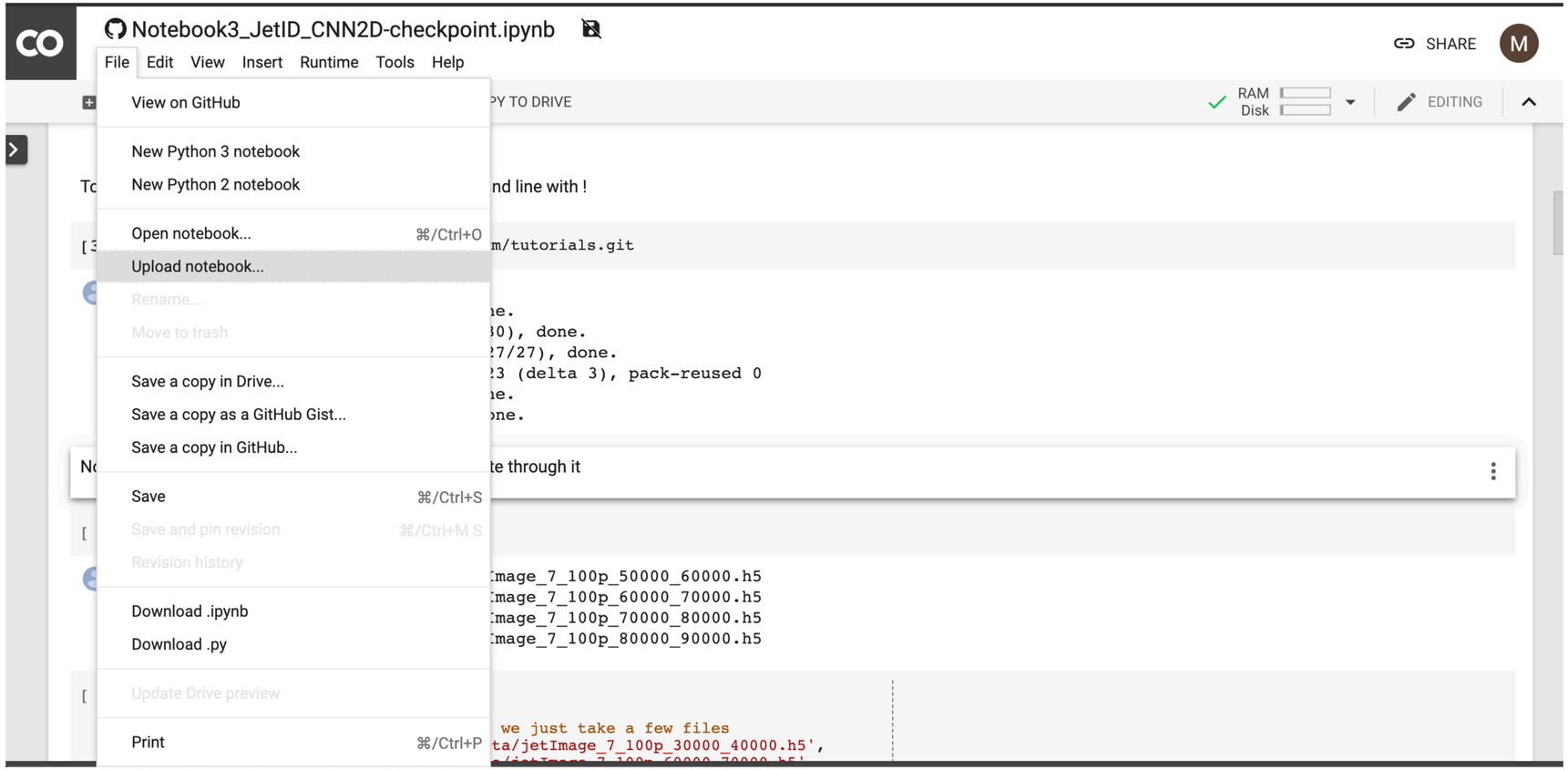


Summary

- *Training a network requires work*
  - *To prepare the data (dataset split, normalization, etc)*
  - *To optimize the training efficiency (early stopping, regularization, etc.)*
  - *To perform post-training diagnostic: training convergence, bias vs variance, accuracy, confusion*
- *Things might not work out of the box, but insight on pre-training, training and post-training techniques make your work easier*

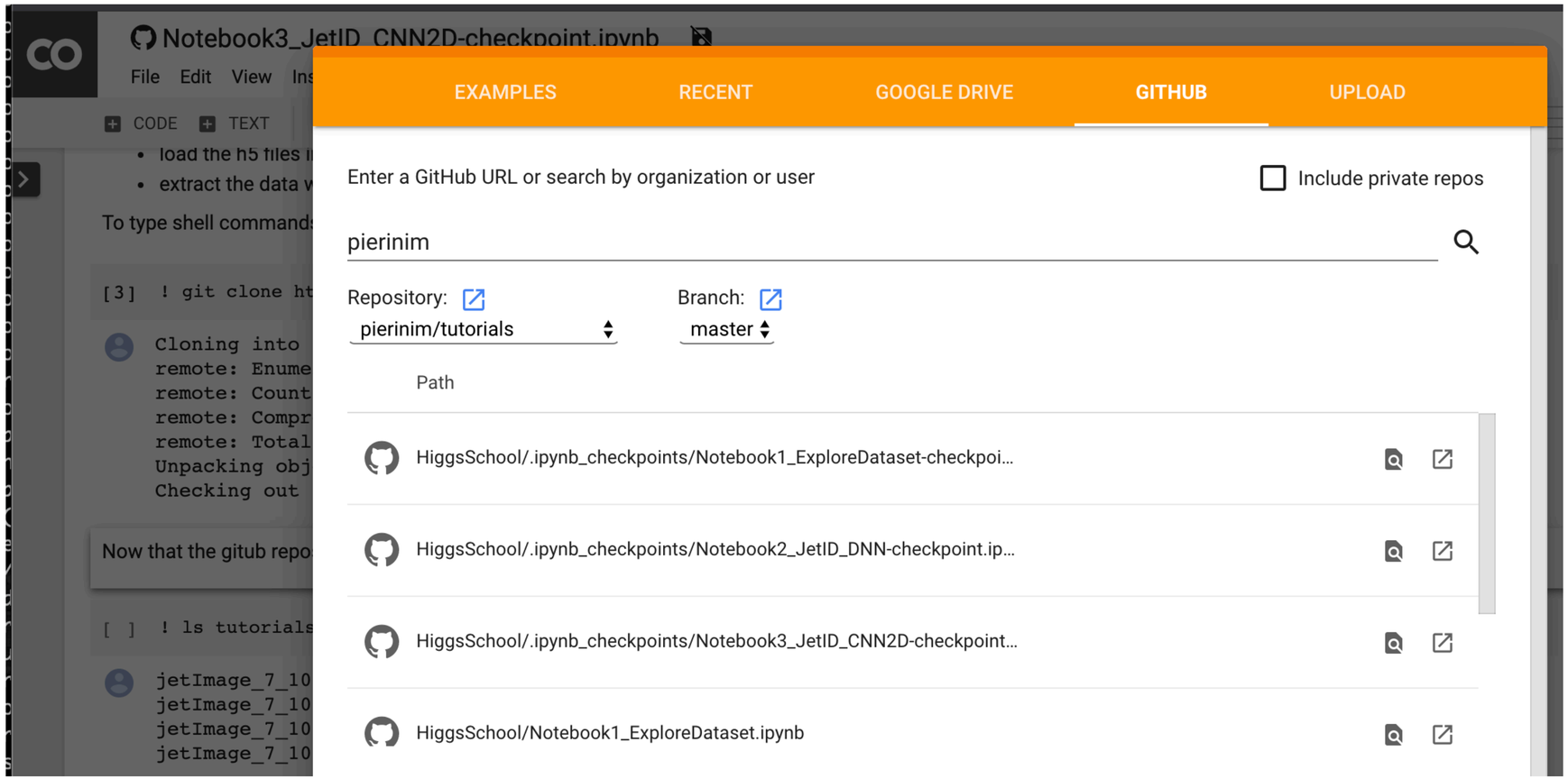
Step1: Open Notebook on Colab

- Go to https://colab.research.google.com

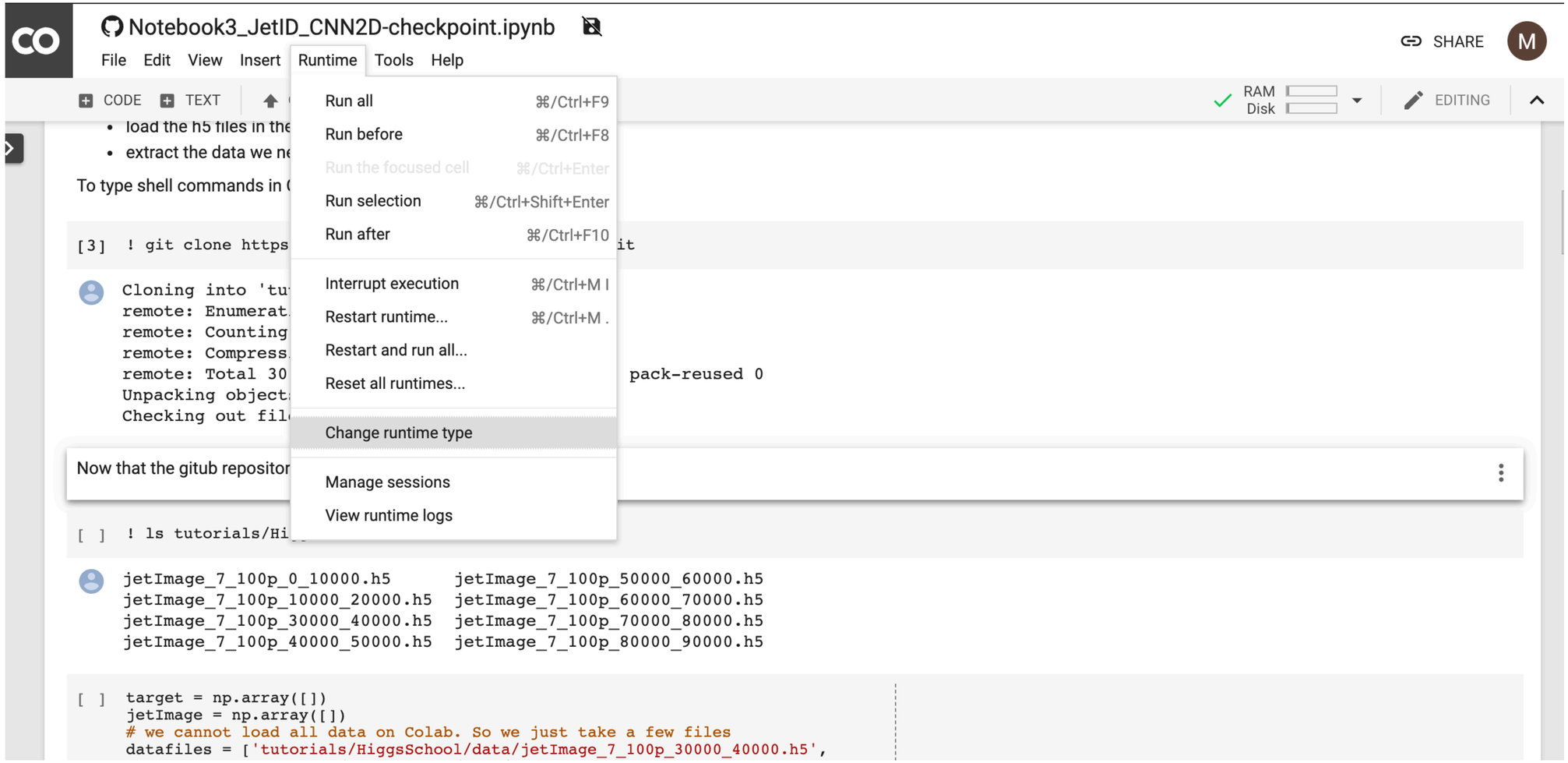


Step2: import the Tutorial from gitlab

- *Click on the GITHUB tab*
- *Specify the repository pierinim/tutorials/SMARTHEP*
- *Click on the notebook*



Set your resources to use GPUs



Set your resources to use GPUs