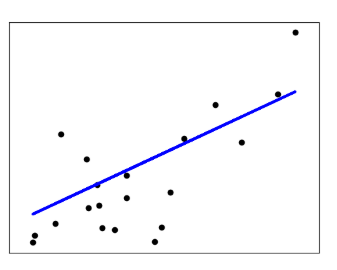
**OLS (Ordinary Least Squares)**

LinearRegression fits a linear model with coefficients w=(w1,...,wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

|  |
| --- |
|  |



Linear regression:

|  |
| --- |
|  |
|  |
|  |

The idea of Simple Linear Regression is finding those parameters α and β for which the error term is minimized. To be more precise, the model will minimize the squared errors: indeed, we do not want our positive errors to be compensated by the negative ones, since they are equally penalizing for our model.

This procedure is called Ordinary Least Squared error — OLS.

you can consider the OLS as a strategy to obtain, from your model, a ‘straight line’ which is as close as possible to your data points. Even though OLS is not the only optimization strategy, it is the most popular for this kind of tasks, since the outputs of the regression (that are, coefficients) are unbiased estimators of the real values of alpha and beta.

Cons: (features are not independent)

Columns of the design matrix **X** have an approximate linear dependence, the design matrix becomes close to singular (not max rank or Det = 0) and as a result, the least-squares estimate becomes highly sensitive to random errors in the observed target, producing a large variance.

**Ridge regression and classification** (A OLS with some Bias)

Ridge regression addresses some of the problems of [Ordinary Least Squares](https://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares) by imposing a penalty on the size of the coefficients. The ridge coefficients minimize a penalized residual sum of squares (DISTANZA punto vero, punto predetto su un cartesiano xy):

|  |
| --- |
|  |

So, ridge regressor, is a variation of linear regression

*“Is a model with the lowest RSS truly the best model?” “Not really.”*

We need to consider BIAS too: how equally a model cares about its predictors.

Unbiased model (overfitting): to find the relationship between the two features and the prices. just as the OLS method does. This model will fit the observations as perfectly as possible to minimize the RSS. The model will not perform as well with new data because it is built for the given data so specifically that it may not fit new data.

|  |
| --- |
|  |

Biased model: Accepts its variables unequally to treat each predictor differently.

Low bias, Low variance 🡪 in Machine learning the ideal algorithm has a low bias and can accurately model the true relationship. Also, it has low variability by producing consistent predictions across different datasets. This is done by finding the sweet spot between a simple model and a complex model.

Bias is related with a model failing to fit the training set and variance is related with a model failing to fit the testing set

There are three use methods for finding the sweet spot:

* Regularization
* Boosting
* Bagging (più modelli dello stesso tipo vengono addestrati su dataset diversi, ciascuno ottenuto dal dataset iniziale tramite campionamento casuale con rimpiazzo, bootstrap).

|  |
| --- |
|  |

Bias and variance are in a trade-off relationship over model complexity, which means that a **simple** model **(predicting bettere in a test set)** would have high-bias and low-variance, and vice versa.

**Where Ridge Regression Comes Into Play**

Looking at *Bias vs. Variance* figure, the Y-axis is ‘Error’ which is the ‘Sum of Bias and Variance’. Since both of them **are basically related with failing, we would like to minimize those.**

|  |
| --- |
|  |

Let’s recall that OLS treats all the variables equally (unbiased). Therefore, an OLS model becomes more complex as new variables are added. It can be said that an OLS model is always on the rightest of the figure, having the lowest bias and the highest variance. It is fixed there, never moves, but we **want to move it to the sweet spot. This is when ridge regression would shine**, also referred to as Regularization. In ridge regression, you can tune the lambda parameter so that model coefficients change.

**A geometric point of view**

|  |  |
| --- | --- |
|  |  |
| Fig 1 | Fig 2 |

Each contour is a connection of spots where the RSS is the same, centered with the OLS estimate where the RSS is the lowest. Also, the OLS estimate is the point where it best fits the training set (low-bias). Unlike the OLS estimate, the ridge estimate changes as the size of the blue circle changes. It is simply where the circle meets the most outer contour. How ridge regression works is how we tune the size of the circle. The key point is that *β***’s** change at a different level.

Let’s say β1 is ‘shine’ and β2 is ‘sweetness’. As you can see, ridge β1 relatively drops more quickly to zero than ridge β2 does as the circle size changes **(compare the two figures).** The reason why this happens is because **the β’s change differently by the RSS**. More intuitively, **the contours are not circles but ellipses positioned tilted.**

Ridge *β*’s can never be zero but only*converge*to it, and this will be explained

**Mathematical Formula**

|  |
| --- |
|  |
|  |

C is equivalent to the radius of the circle, thus, the *β*’s should fall in the circle area, probably somewhere on the edge.

We still want to understand the very first equation. To do so, we need to brush up on vector norm, which is nothing but the following definition.

|  |
| --- |
|  |

L2 norm 🡪 The L2 norm calculates the distance of the vector coordinate from the origin of the vector space. As such, it is also known as the Euclidean norm as it is calculated as the Euclidean distance from the origin. The result is a positive distance value.

The L2 norm is calculated as the square root of the sum of the squared vector values.

|  |  |
| --- | --- |
| 1 | ||v||2 = sqrt(a1^2 + a2^2 + a3^2) |

**Ridge regression:**

|  |
| --- |
|  |

Notice the first term in the following equation is basically OLS, and then the second term with lambda is what makes ridge regression

The term with lambda is often called ‘Penalty’ since it increases RSS. We iterate certain values onto the lambda and evaluate the model with a measurement such as ‘Mean Square Error (MSE)’. So, the lambda value that minimizes MSE should be selected as the final model. This ridge regression model is generally better than the OLS model in prediction. Ridge *β*’s change with lambda and becomes the same as OLS *β*’s if lambda is equal to zero (no penalty).

**Example of RIDGE test Set MSE:**

|  |
| --- |
|  |

The green dotted line is from OLS on the graph above with the X-axis being drawn by increasing lambda values. The MSE values decreases in the beginning as the lambda value increases, which means the model prediction is improved (less error) to a certain point. In short, an OLS model with some bias is better at prediction than the pure OLS model, we call this modified OLS model as the ridge regression model.

**In a nutshell**

* OLS simply finds the best fit for given data
* Features have different contributions to RSS
* Ridge regression gives a bias to important features
* MSE or R-square can be used to find the best lambda

Fonte: <https://towardsdatascience.com/ridge-regression-for-better-usage-2f19b3a202db> ( con anche esempio su python)

**Bayesian Regressor:**

In the Bayesian viewpoint, we formulate linear regression **using probability distributions rather than point estimate**. The response, y, is not estimated as a single value, but is assumed to be drawn from a probability distribution. The model for Bayesian Linear Regression with the response sampled from a normal distribution is:

|  |
| --- |
|  |

The output, y is generated from a normal **(Gaussian) Distribution** characterized by a mean and variance. The mean for linear regression is the transpose of the weight matrix multiplied by the predictor matrix. The variance is the square of the standard deviation σ (multiplied by the Identity matrix because this is a multi-dimensional formulation of the model).

The aim of Bayesian Linear Regression is not to find the single “best” value of the model parameters, but rather **to determine the posterior distribution for the model parameters (Bayes, probabilità condizionata).**

The **posterior probability** **of the model parameters** is conditional upon the **training inputs and outputs**:

|  |
| --- |
|  |
|  |

* P (β| y, X): posterior, estimate pron of B param of the modello, given input X (training x) and output Y (training y). The result of performing Bayesian Linear Regression is a distribution of possible model parameters based on the data and the prior. This allows us to quantify our uncertainty about the model: if we have fewer data points, the posterior **distribution** will be more spread out **(sparsa).**
* P (y|β, X): Likelihood
* P (β | X): Prior

If we have domain knowledge, or a guess for what the model parameters should be, we can include them in our model, unlike in the frequentist approach which assumes everything there is to know about the parameters comes from the data.  (action units?)

If we don’t have any estimates ahead of time, we can use [non-informative priors](https://stats.stackexchange.com/questions/27813/what-is-the-point-of-non-informative-priors" \t "_blank) for the parameters such as a normal distribution.

* P (y|X): Normalization constant

As the amount of data points increases, the likelihood washes out the prior (con più dai il modello tende ad essere più verosimile a quello reale), and in the case of infinite data, the outputs for the parameters converge to the values obtained from OLS.

The formulation of model parameters as distributions encapsulates the Bayesian worldview: we start out with an initial estimate, our prior, and as we gather more evidence, our model becomes less **wrong.**Bayesian reasoning is a natural extension of our intuition. Often, we have an initial hypothesis, and as we collect data that either supports or disproves our ideas, we change our model of the world.

The basic procedure for implementing Bayesian Linear Regression is:

* Specify priors for the model parameters (normal distribution)
* Create a model mapping train\_X to train\_Y
* Markov Chain Monte Carlo (MCMC) algorithm that draws samples from the posterior distribution for the model parameters **(questo perchè solo nella teoria si può lavorare con una distribuzione continua)**.

The first plots show the approximations of the posterior distributions of model parameters. These are the result of 1000 steps of MCMC, meaning the algorithm drew 1000 steps from the posterior distribution.

|  |
| --- |
|  |
|  |

However, while we can use the mean as a single point estimate, we also have a **range of possible values for the model parameters**. As the number of data points increases, **this range will shrink and converge one a single value representing greater confidence in the model parameters**.

When we want show the linear fit from a Bayesian model, instead of showing only estimate, we can draw a range of lines, with each one representing a different estimate of the model parameters. As the number of datapoints increases, the lines begin to overlap because there is less uncertainty in the model parameters.

There is much more variation in the fits when using fewer data points, which represents a greater uncertainty in the model. With all of the data points, the OLS and Bayesian Fits are nearly identical because the priors are washed out by the likelihoods from the data.

When predicting the **output** for a single datapoint using our Bayesian Linear Model, we also do not get a single value but a **distribution**.

|  |
| --- |
|  |

We see that the probability of the number of calories burned peaks around 89.3, but the full estimate is a range of possible values.

In problems where we have limited data or have some prior knowledge that we want to use in our model, the Bayesian Linear Regression approach can both incorporate prior information and show our uncertainty. Bayesian Linear Regression reflects the Bayesian framework: we form an initial estimate and improve our estimate as we gather more data.

The model parameters are assumed to come from a distribution as well.

**Bayesian regression techniques** can be used to include **regularization parameters** in the estimation procedure: the regularization parameter is not set in a hard sense but tuned to the data at hand.

As we’ve seen, it can be done by introducing **uninformative priors** over the **hyper parameters of the model.**

Uninformative priors: In Bayesian parameter estimation, uninformative priors are a way of making minimal assumptions about the model. They are commonly chosen to be invariant to certain transformations, such as translation or scaling. While uninformative priors are often improper, they can still lead to proper posterior distributions, and thereby be usable in posterior inference (they are used when we don’t have a prior)

Hyperparameter model: configuration that is external to the model and whose value cannot be estimated from data.

* They are often used in processes to help estimate model parameters.
* They are often specified by the practitioner.
* They can often be set using heuristics.
* They are often tuned for a given predictive modelling problem.

The ℓ2 regularization (used as upper bound for the BIAS) in [Ridge regression and classification](https://scikit-learn.org/stable/modules/linear_model.html#ridge-regression) is equivalent to finding a maximum a posteriori estimation p(y|B,X) under a Gaussian prior over the coefficients w with precision λ−1. **Instead of setting lambda manually, it is possible to treat it as a random variable to be estimated from the data (we estimate lambda as well).**

To obtain a fully probabilistic model, the output y is assumed to be Gaussian distributed around Xw:

|  |
| --- |
|  |

where α (standard deviation?) is again treated as a random variable that is to be estimated from the data

[**BayesianRidge**](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.BayesianRidge.html#sklearn.linear_model.BayesianRidge)**estimates a probabilistic model of the Bayesian regression.**

* The prior for the coefficient w is given by a spherical Gaussian(with μ = 0):

|  |
| --- |
| * The priors over α and λ are chosen to be gamma distributions |

**Spherical Gaussian:** (Non ci serve)

<https://mynameismjp.wordpress.com/2016/10/09/sg-series-part-2-spherical-gaussians-101/>

A spherical Gaussian still works as a Gaussian, except that it lays on the surface of a sphere instead of on a line or a flat plane.

|  |  |
| --- | --- |
| 1d Gaussian | Spherical gaussian |

“x-b” term refers of the cartesian distance between a given point and the center of the Gausian, chich can be trivially extended into 2D, using the distance formula.

To make this work on a sphere, we must instead make our Gaussian a function of the angle between two unit direction vectors. In practice we do this by making an SphGauss a function of the cosine of the angle between two vectors, which can be efficiently computed using a dot product like in fig 2

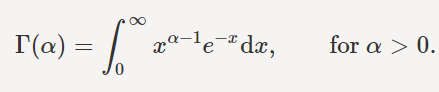
* μ, which is the axis, or direction of the lobe.
* λ, which is the sharpness of the lobe. As this value increases, the lobe will get “skinnier”, meaning that the result will fall off more quickly as you get further from the lobe axis.
* a is the amplitude or intensity of the lobe. It corresponds to the height of the lobe at its peak. The amplitude can be a scalar value

**Gamma distributions:**

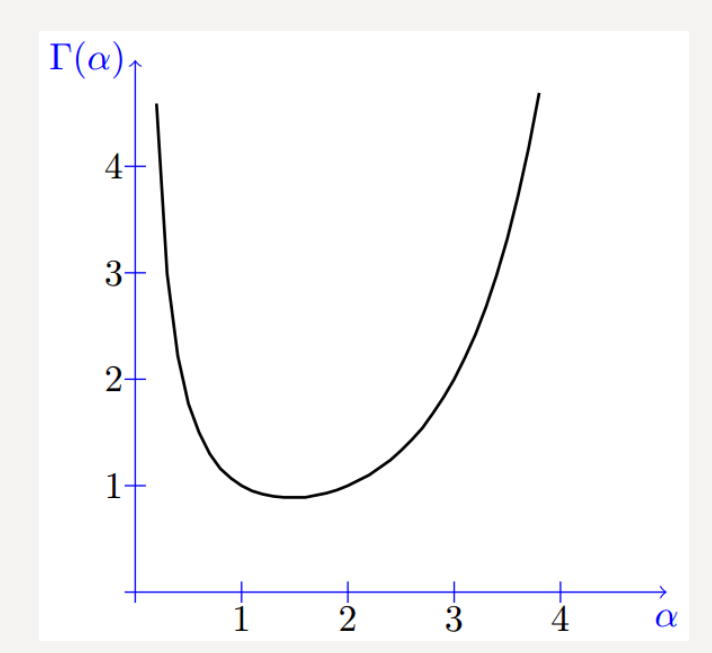
Gamma function: The gamma function [[10](https://www.probabilitycourse.com/bibliography.php" \l "wiki-gamma-distr)], shown by Γ(x)Γ(x), is an extension of the factorial function to real (and complex) numbers. Specifically, if n∈{1,2,3,...}n∈{1,2,3,...}, then

Γ(n)=(n−1)!

More generally, for any positive real number α, Γ(α) is defined as

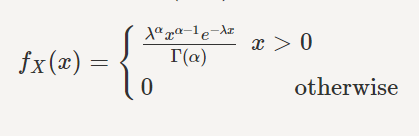


The figure shows the gamma function for positive real values.

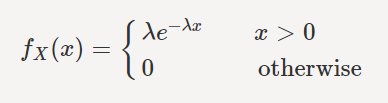


<https://www.probabilitycourse.com/chapter4/4_2_4_Gamma_distribution.php>

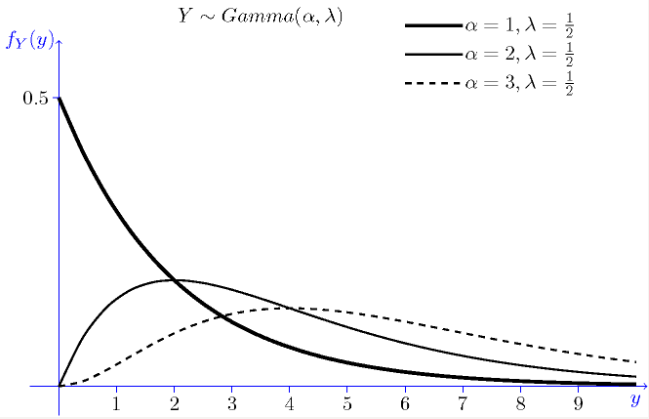
A continuous random variable X is said to have a gamma distribution with parameters α>0 ,λ>0 , shown as X∼Gamma(α,λ), if its PDF is given by



If we let α=1, we obtain



Thus, we conclude Gamma(1,λ)=Exponential(λ). More generally, if you sum n independent Exponential(λ random variables, then you will get a Gamma(n,λ) random variabl. The gamma distribution is also related to the normal distribution as will be discussed later. Figure 4.10 shows the PDF of the gamma distribution for several values of α.



Going back to Bayesian ridge regression:

The parameters w, α and λ are estimated jointly during the fit of the model, the regularization parameters α and λ being estimated by maximizing the log marginal likelihood.

There are four more hyperparameters, α1, α2, λ1 and λ2 of the gamma prior distributions over α and λ. These are usually chosen to be non-informative.

By default α1=α2=λ1=λ2= 10^−6.

We are not going further in Ridge Regression because what we are really interest in is the ARD which is similar to ridge Regression**.**

**Automatic Relevance Determination - ARD**

In the literature ARD is known as *Sparse Bayesian Learning* and *Relevance Vector Machine.*

Similar to ARD, but it leads to sparser coefficients w (this because the gaussian is not spherical anymore) but the distribution over w is assumed to be axis-parallel, elliptical **Gaussian** distribution.

This means each coefficient

wi is drawn from a Gaussian distribution, centered on zero and with a precision λi:



With diag(A) = λ = { λ1, ..., λ p}

each coordinate of wi has its own standard deviation λi. The prior over all λi is chosen to be the same gamma distribution given by hyperparameters λ1 and λ2.

Compared to the OLS (ordinary least squares) estimator, the coefficient weights are slightly shifted toward zeros, which stabilises them.

The histogram of the estimated weights is very peaked, as a sparsity-inducing prior is implied on the weights.

The estimation of the model is done by iteratively maximizing the marginal log-likelihood of the observations.

<https://www.slideshare.net/FlorianWilhelm2/explaining-the-idea-behind-automatic-relevance-determination-and-bayesian-interpolation-59498957>

<https://www.youtube.com/watch?v=2gT-Q0NZzoE> (video con spiegazione delle slide sopra, 39 minuti)



Our minimization problem in ARD (Automatic Relevance Determination)

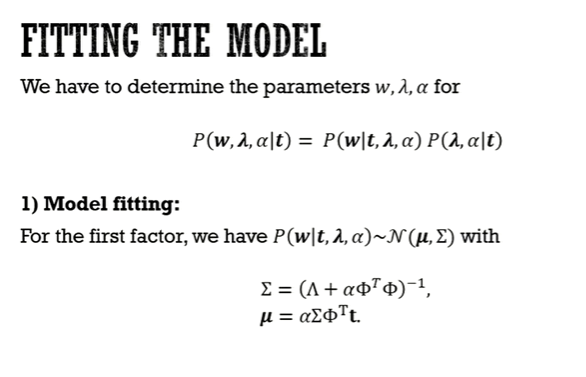


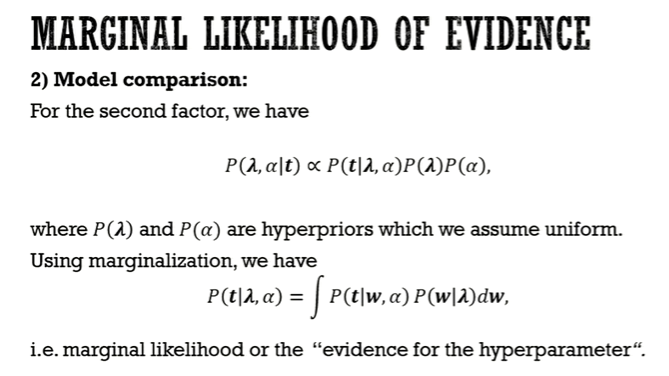
**Pruning:** If precision λ1 of a feature h is high, its weight is very likely to be close to zero and is therefore pruned (se una action unit ha peso vicino allo zero, non la consideriamo).

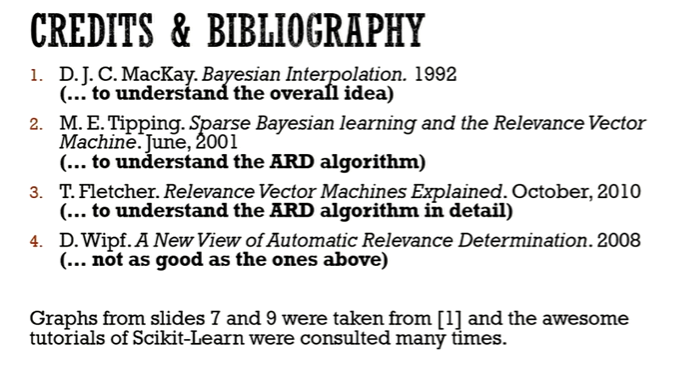
ARD va bene quando hai tante feature e pochi dati ossevati, noi avremo le action units sull’asse x. Arousal e Valence? Secondo me no perchè non è autoregressione in questo caso, credo. (lo dico perchè non voglio i dati passati)

Prendiamo un sottoinsieme, chiedere a Erio come penderlo

|  |
| --- |
|  |
|  |







Non leggere fletcher per primo (ma questi li vedo più io in dettaglio per MSA al limite)

<https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ARDRegression.html>

fit the weights of a regression model, using an ARD prior. The weights of the regression model are assumed to be in Gaussian distributions. Also estimate

* Parameter λ is precisions of the distributions of the weights
* Parameter alfa, precision of the distribution of the noise

he estimation is done by an iterative procedures (Evidence Maximization)

Attributi che ci interessa estrarre dalle action unit.

**coef\_*array-like of shape (n\_features,)***

Coefficients of the regression model (mean of distribution)

**alpha\_*float***

estimated precision of the noise.

**lambda\_*array-like of shape (n\_features,)***

estimated precisions of the weights.

Metodi da applicare sul modello:

* Fit
* Get\_params
* Predict

**Esempio su ARD**

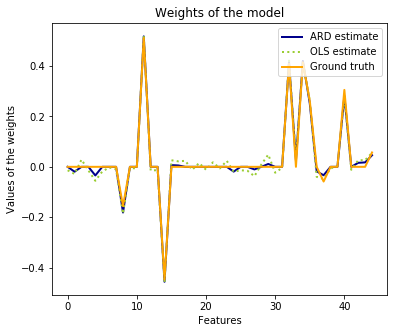
Sulla base di cosa prendo le feature rilevanti? Lui lo fa prima di ARD prendendone 10 casuali dicendo a priori che quelle si suppone siano rilevanti, noi abbiamo le 17 action unit, ma non sappiamo quale peso assegnare loro in partenza (dovrebbe essere una supposizione, idea, partiamo da quelle di bocca e occhi che soon quelle più significative)

Nel training fa:

Y^ = w\*X + noise

Io ho anche Y (adatto il modello per avere Y in risultato)

Plot:



Ground truth: i PESI delle feature che abbiamo supposto noi essere rilevanti, assegnati andando a prendere i valori da una distribuzione gaussiana, generata casualmente

* Precision of the noise alpha = 50
* Precision on weights, lambda = .4 (standard)

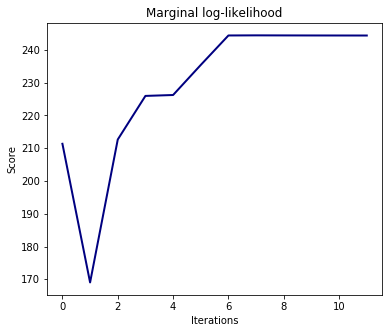
ARD estimated : pesi stimati fittando il modello con ARD (quello che interessa a noi)

OLS estimated: pesi stimati con una regressione lineare semplice OLS.

|  |  |
| --- | --- |
|  |  |

Da questo grafico andiamo a vedere l’istogramma dei i pesi stimati (blu) e lo confrontiamo con i pesi w che indicano i pesi stabiliti da noi per le feature rilevanti.

(Per fare una verifica più utile ci conviene andare direttamente a stampare le feature con i pesi ARD superiori ad una certa soglia)



**scores\_*float***

if computed, value of the objective function (to be maximized)

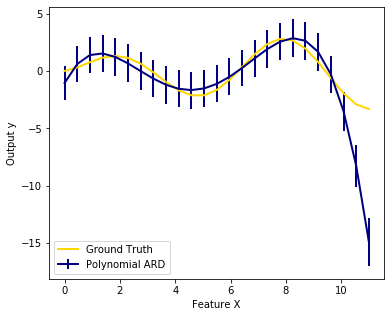
dalla sesta iterazione ottengo una stabilità nella marginal likelihood (o evidence del modello)

* Score è un attributo messo a disposizione da ARD regression

Questa devo vedere a cosa serve. A occhio direi che stabilisce quante iterazioni fa la fit prima di convergere e stopparsi.

L’iterazione si ferma quando c’è una convergenza di MLE, a farlo però è il modello ARD nella fit. È tutto incorporato lì dentro.

Test con Polynomial ARD da studiare.



**Maximum-likelihood-estimation**

<https://towardsdatascience.com/a-gentle-introduction-to-maximum-likelihood-estimation-9fbff27ea12f>

MLE in a nutshell helps us answer this question:

Which are the best parameters/coefficients for my model?

*P(β∣y) = P(y∣β) x P(β) / P(y) 🡪*

*posterior = likelihood \* prior / evidence(or marginal likelihood)*

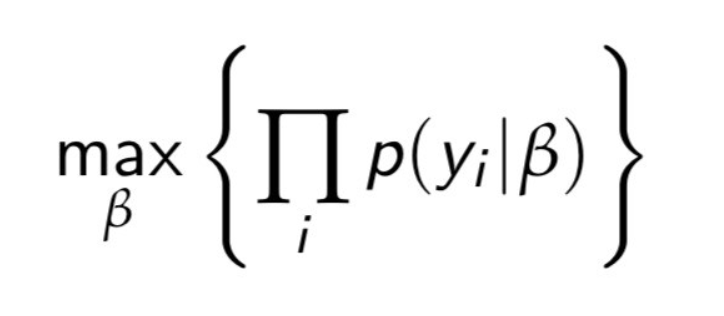
The distinction between probability and likelihood is fundamentally important: Probability attaches to possible results; likelihood attaches to hypotheses.

*p(y|β) is equivalent to L(β|y) 🡪 p(y\_1,y\_2,...,y\_n|β) is equivalent to L(β|y\_1,y\_2,...,y\_n) .*

Also, remember that we can multiply independent probabilities:

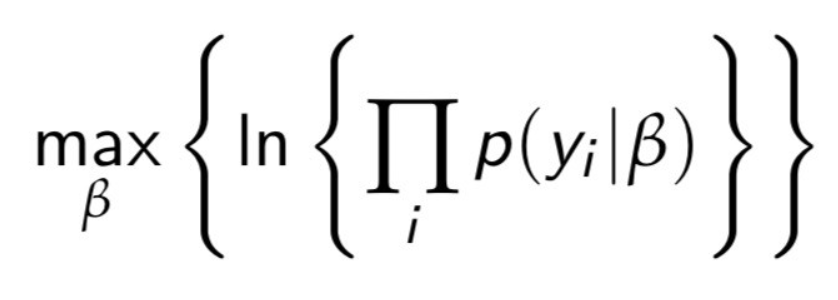
*L(β|y1,y2,…,yn) = p(y1|β)p(y2|β),…,p(yn|β) = ∏p(yi|β)*

*∏p(yi|β)* looks like something we can maximize:



We want to find the parameter *β* which maximize the likelihood

We can use Logs, that are [monotonic transformations](https://en.wikipedia.org/wiki/Monotonic_function), so we’ll simplify our computation but maintain our optimal result.



Our final cost function looks like this:

