## Elastic-Net regression

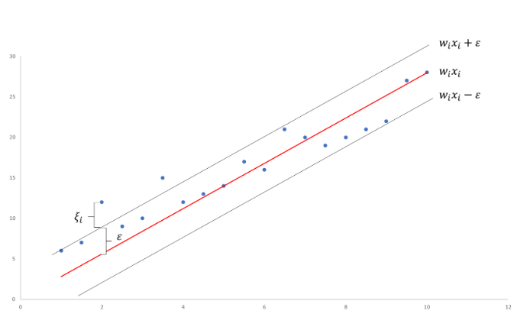
Elastic-Net’s loss function contains Ridge and Lasso’s penalty terms. The advantage of combining them is to smoothen Lasso’s severe feature selection and still obtain Ridge regularization characteristics. **It is particulary useful when there are multiple correlated features**. The algorithm has two hyperparameters: l1\_ratio (phi) and alpha. Alpha is the weight the penalty term, the higher alpha the more the penalty term has influence. The l1\_ratio is the scaling between l1 and l2 penalty:

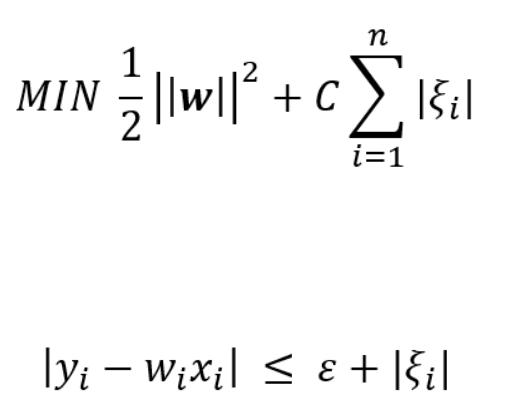
* L1\_ratio: 0 is an l2 regression
* L1\_ratio: 1 is an l1 regression

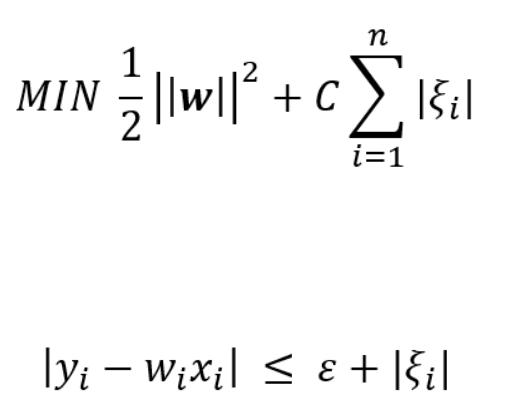


## Support vector Regression (**not able to perform this, took too long**)

The objective of a Support Vector Regression (SVR) algorithm is to find a hyper plane that is able to predict the target value at is best. To obtain such hyper plane, the algorithm searches two boundary lines, each at distance e of the hyperplane, such that the majority of support vectors lies within the surface created by these boundaries. During construction, two hyperparameters are taken into account, i.e. Epsilon and C. Epsilon represents the amount of support vectors oustide the boundary lines and C indicates the amount of tolerance for the support vectors outside the boundaries.

Mathematically, this comes down to the following minimalisation problem. We want to find these coefficients (wi) such that the error term is minimized. The epsilons are the amount of points outside the boundaries we tolerate and c is the penalization for each of these points.





<https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2>

<https://medium.com/coinmonks/support-vector-regression-or-svr-8eb3acf6d0ff>

There are three different implementations of Support Vector Regression: [**SVR**](https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html#sklearn.svm.SVR), [**NuSVR**](https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVR.html#sklearn.svm.NuSVR) and [**LinearSVR**](https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVR.html#sklearn.svm.LinearSVR). [**LinearSVR**](https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVR.html#sklearn.svm.LinearSVR) provides a faster implementation than [**SVR**](https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html#sklearn.svm.SVR) but only considers linear kernels, while [**NuSVR**](https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVR.html#sklearn.svm.NuSVR) implements a slightly different formulation than [**SVR**](https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html#sklearn.svm.SVR) and [**LinearSVR**](https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVR.html#sklearn.svm.LinearSVR).

## Nearest Neghbors Regression

The algorithm search for each observation the k nearest neighbors, the label assigned to the observation is the mean of its k nearest neighbors. Under some circumstances, it can be advantageous to weight points such that nearby points contribute more to the regression than faraway points. This can be accomplished through the weights keyword. The default value, weights = 'uniform', assigns equal weights to all points. weights = 'distance' assigns weights proportional to the inverse of the distance from the query point