**Label Spreading.**

**Label Spreading allows for each labeled point to accept information from its neighbors which may lead to a deviation from its original label.**

**Label spreading**, which offers a slight better stability when the dataset is very noisy or dense. In these cases, standard label propagation might suffer a loss of precision due to the closeness of points with different labels. Conversely, label spreading is more robust because the Laplacian is normalized and abrupt transitions are more heavily penalized using weights.

Label propagation computes a similarity matrix between samples and uses a KNN-based approach to propagate samples, while label spreading takes a similar approach but adds a regularization to be more robust to noise. This is done by hyperparameter **gamma.**

High gamma extends the influence of each individual point wide, hence creating a smooth transition in label probabilities. Meanwhile, low gamma leads to only the closest neighbors having influence over the label probabilities.

We have another parameter that is the main difference with label propagation, called soft clamping concept, control by parameter α (alpha) the clamping factor. A value in (0, 1) that specifies the relative amount that an instance should adopt the information from its neighbors as opposed to its initial label. alpha=0 means keeping the initial label information; alpha=1 means replacing all initial information.

The key idea of the two methods is essentially the same. The difference lies in the design of the transition matrix. Label propagation uses the graph Laplacian while Label spreading uses the normalized graph Laplacian.

However, to get the best results, it is often beneficial to combine these two sets of data. Such a situation is an excellent example of where we would want to use a Semi-Supervised Learning approach, with the Label Spreading algorithm being one of our options.

**Symmetric normalized Laplacian vs. random walk normalized Laplacian**

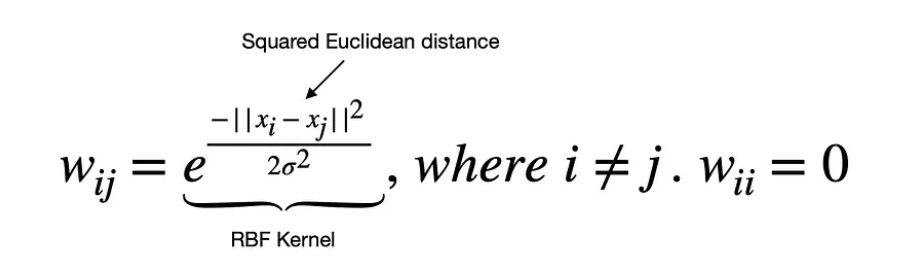
**The Label Spreading algorithm uses symmetric normalized graph Laplacian matrix in its calculations, while Label Propagation employs a random walk normalized Laplacian.**

However, note that the two matrices are similar and that one can be derived from the other. Hence, from the perspective of this article, it is not crucial for us to understand the nuances of these two matrices.

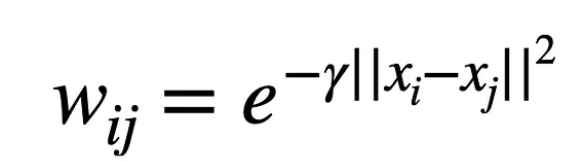
# **A brief explanation of how Label Spreading works**

Four steps describe how the Label Spreading algorithm operates.

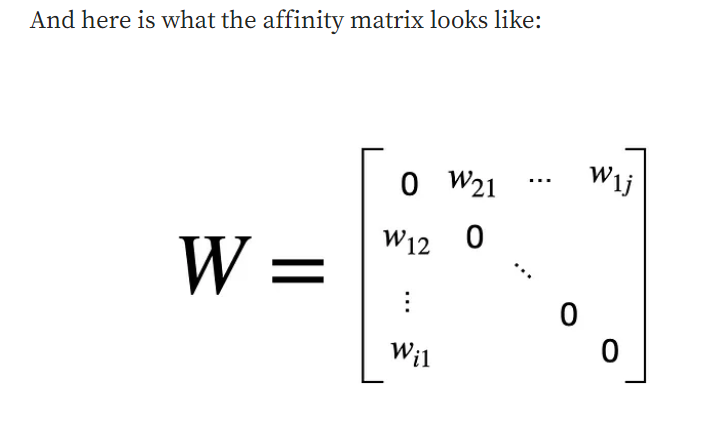
1. Define a pairwise relationship between points, called affinity matrix **W**. The matrix is created with the help of a Radial Basis Function kernel (a.k.a. RBF kernel), which is used to determine edge weights. Note that matrix **W** contains 0’s in the diagonal since no edge connects a point to itself.



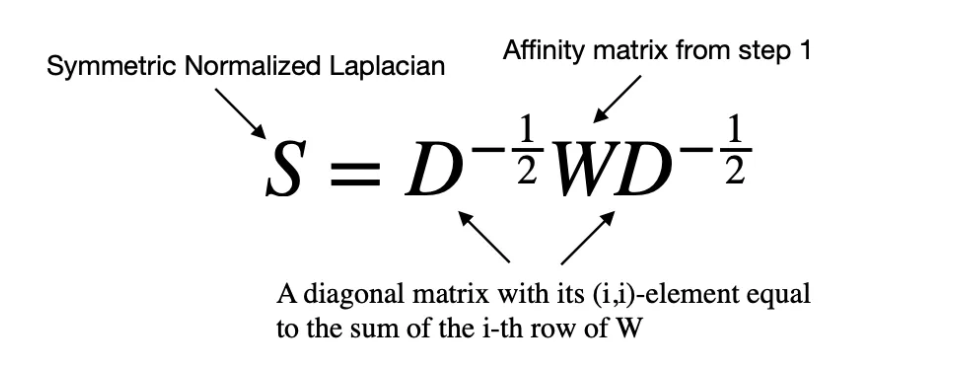
Note, sklearn's implementation of RBF kernel looks slightly different as it replaces **1/2sigma^2** with a hyperparameter **gamma**. The effect is the same as it allows you to control the smoothness of the function. High gamma extends the influence of each individual point wide, hence creating a smooth transition in label probabilities. Meanwhile, low gamma leads to only the closest neighbors having influence over the label probabilities.



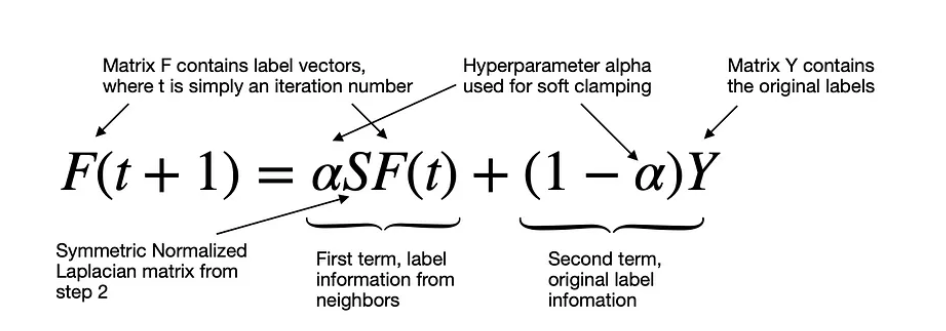
Sklearn’s implementation of the RBF kernel.



**2**. Create a symmetric normalized graph Laplacian matrix. This step takes affinity matrix **W** and normalizes it symmetrically, which helps with the convergence in step 3.

****

3 The third step is iterative, which uses matrix multiplication to spread information from labeled points to unlabeled points.



An iterative process to find the labels.

**Soft clamping concept: Each point receives the information from its neighbors (first term) and also retains its initial information (second term). The parameter α (alpha) enables soft clamping by controlling the proportion of information received from neighbors vs. the initial label. Alpha close to 0 keeps all the initial label information (equivalent to hard clamping), while alpha close to 1 allows most of the initial label information to be replaced.**

Note that F(0)=Y, so the iterative process starts with the initial label information.

4. After the process in step 3 converges or reaches the specified maximum number of iterations, we arrive at the final step of assigning the labels.

**Matrix F contains label vectors, representing the probabilities of each point belonging to a specific class (i.e., having a particular label). The final label is then chosen using argmax operation, meaning that the algorithm assigns the label with the highest probability.**

**Parameters:**

<https://scikit-learn.org/stable/modules/generated/sklearn.semi_supervised.LabelSpreading.html>

**kernel*{‘knn’, ‘rbf’} or callable, default=’rbf’***

String identifier for kernel function to use or the kernel function itself. Only ‘rbf’ and ‘knn’ strings are valid inputs. The function passed should take two inputs, each of shape (n\_samples, n\_features), and return a (n\_samples, n\_samples) shaped weight matrix.

**gamma*float, default=20***

Parameter for rbf kernel.

**n\_neighbors*int, default=7***

Parameter for knn kernel which is a strictly positive integer.

**alpha*float, default=0.2***

Clamping factor. A value in (0, 1) that specifies the relative amount that an instance should adopt the information from its neighbors as opposed to its initial label. alpha=0 means keeping the initial label information; alpha=1 means replacing all initial information.

**max\_iter*int, default=30***

Maximum number of iterations allowed.

**tol*float, default=1e-3***

Convergence tolerance: threshold to consider the system at steady state.

**n\_jobs*int, default=None***

The number of parallel jobs to run. None means 1 unless in a **[joblib.parallel\_backend](https://joblib.readthedocs.io/en/latest/generated/joblib.parallel_backend.html" \l "joblib.parallel_backend" \o "(in joblib v1.4.dev0))** context. -1 means using all processors. See [Glossary](https://scikit-learn.org/stable/glossary.html#term-n_jobs) for more details.