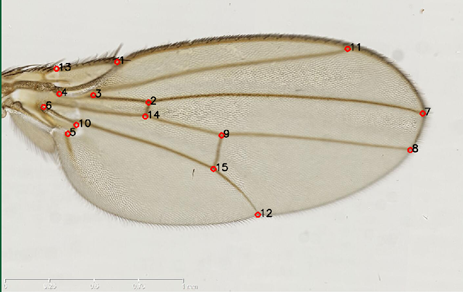
1. Follow the tutorial on using logistic regression with scikit learn

<https://towardsdatascience.com/logistic-regression-using-python-sklearn-numpy-mnist-handwriting-recognition-matplotlib-a6b31e2b166a>

1. Apply Naïve Bayes classifier and artificial neural network (MLPClassifier) with mnist dataset
2. Using various classifier to solve the landmark prediction problem. The problem is described as followed:

Given a data set of images of drosophila wings. Predict the location of the landmark point in the image.



An example of landmark point in the wing images.

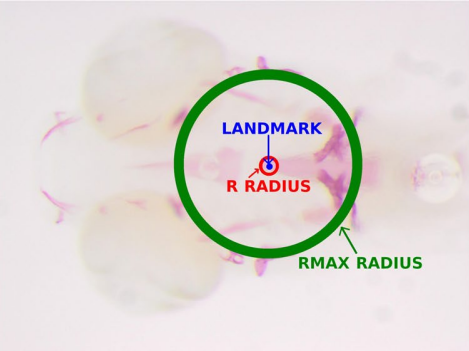
The framework includes two stages:

1. **Training Stage**

**The goal of the training stage is to build a classifier which can predict if a point in the image is a landmark point (positive) or not (negative)**

We need to build a classifier for each landmark point in the image. If the image contains 15 landmarks, we need to build 15 classifiers. The process of training the classifier for one landmark is described as follow:

* **Generating a list of positive points and negative points**. The positive/negative samples are extracted from sample images (the images with the positions of each landmark point). A positive point is the point which is very close to the true landmark. A negative point is the point which is far away to the true landmark. On the picture below, the true landmark is marked as blue dot, all the points inside the red circle (R radius) are considered as positive points. All the points outside the red circle and inside the green circle (RMAX radius) are considered as negative points.

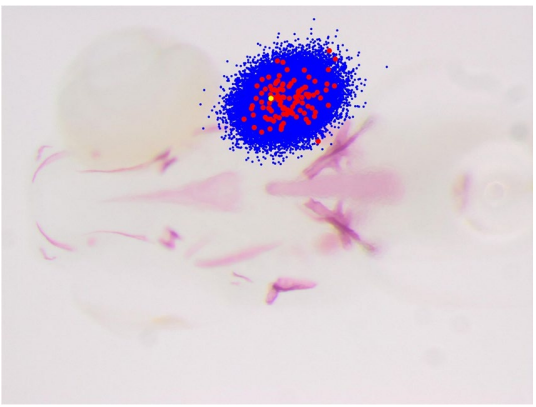


* **Feature extraction**. Each point (both negative and positive) is represented by a feature vector using a feature extraction method. Various feature extraction methods are supported: raw, usub, gsub, ssub, haar. The details of each feature extraction method can be found in the reference paper [1].
* **Training a classifier.** Training a classifier to classify if a feature vector is positive (landmark) or not.

1. **Prediction Stage**

The goal of prediction stage is the find the location of each landmark point in the image. The process of landmark prediction is described as follow:

* **Generate a list of candidate points**. For each landmark point, examine the sample data, determine a region where the landmarks should belong to, then randomly select Np points in this region as candidate points. Using feature extraction to extract a feature vector for each candidate point.



In red, the position of landmark 8 as observed in all the images of the ZEBRA dataset, overlaid on an image. In blue, the position of the corresponding 30,000 examples extracted during prediction according to our sampling strategy. In yellow, the real landmark position [1]

* **Prediction** Use the classifier trained in the training stage to predict if a candidate point is positive or negative. After predicting all candidate points, we achieve a list of positive points. The landmark we want to locate is computed as the center point of all positive points.

More information can be found in the reference paper [1].

In this exercise we focus only on the landmark number 9 only

**Your materials:**

* RAW feature vectors of some positive and negative points (mark as \_pos and \_neg in the file name). Each line in the file presents a feature vector. This is the output of feature extraction step in the training stage. The files are located in the “Features” folder
* Some images with their true landmark (the landmark points are stored in the .tps file with the same name as the images)
* A function to extract RAW feature vector. The function is located in line 46 in the file FeatureExtraction.py

computeRAW(listImgs, listPoints, W)

the example code to extract raw feature at point (504,304) is listed below:

*D = 5 # number of scale*

*W = 8 # window size is 2W+1*

*inputF = " egfr\_F\_R\_oly\_2X\_1.tif" #load the input image*

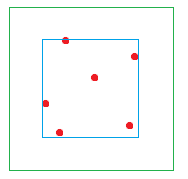
*listImgs = RescaleImage(inputF,D) #create multiscale images*

*listPoints = RescalePoint(504,343,D) # create the point location at each scale*

*raw = computeRAW(listImgs, listPoints, W) # raw is the output feature vector*

**Your task:**

* Load the RAW feature vector, train the classifier to classify if a feature vector is negative or positive.
* Determine the region in the image where the landmark 9 should belong to. This is done by locate the location of landmark 9 in each image. Create a rectangle bounding box to cover those landmark points. Adding a 50-pixels boundary.



Red: the location of landmark 9 in the various image

Blue: the rectangle bounding box

Greed: the bounding box adding 50- pixel boundary

* Randomly select 500 candidate points in the green region.
* Use the classifier trained in the training stage to predict if a point is positive or negative. The predicted landmark is computed as the center point of all positive points
* Calculate the mean absolute error over all the images.