Exercise 06: The Hebb's rule and the perceptron rule

Let us take, as always, into account the classification problem for $N = 40$.

The perceptron is given by

$$\sigma(\vec{\xi}^{\mu}, \vec{J}) = \text{sign}(\vec{\xi}^{\mu} \cdot \vec{J}).$$

In the **training**, we initialise the student perceptrons at random. They will attempt learning with a **learning rule**.

In Exercise 03 we used the following rule:

$$\vec{J}(t+1) = \vec{J} + \frac{1}{\sqrt{N}} \sigma_T^{\mu} \vec{\xi}^{\mu} (1 + r)$$

which is called the **randomised perceptron rule**. This time, we are asked to do the same but with another learning rule, namely the **Hebb's rule**, given by

$$\vec{J}^H = \frac{1}{\sqrt{N}} \sum_{\mu=1}^{P} \vec{\xi}^{\mu} \sigma_T^{\mu}$$

which is not anymore an *iterative rule*. So this operation alone will account for the training, given a training set, and we will not have to look for a convergence.

Since we previously did the computation for a teacher which was taken to be the perfect perceptron, we might start with the same use case. But keep in mind that the choice of the teacher is arbitrary and does not influence the performance of the learning.

Step 1, 2 and 3: Apply Hebb's rule, perform train test loop and compare with analytical results

We do the same we did for the randomised perceptron rule with the Hebb's rule: we perform 1000 loops of training and tests, with varying values of training set size P and with $P_{\text{test}} = 1000$.

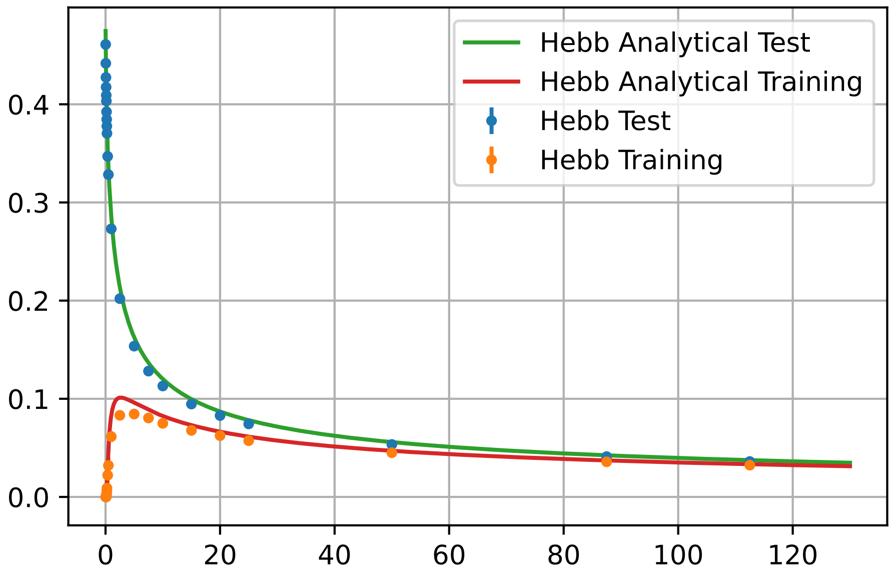
Then, we superimpose the numerical results from the computation of the analytical functions

$$\epsilon(\alpha) = \frac{1}{\pi}\arccos(\sqrt{\frac{2\alpha}{2\alpha+\pi}}) \qquad \epsilon_{\text{train}}(\alpha) = 2\int_0^{+\infty} dx\; H(\frac{1}{\sqrt{\alpha}} + \sqrt{\frac{2\alpha}{\pi}}x)$$

with

$$H(u)\int_u^{+\infty} \frac{dy}{2\pi}\; \exp(-y^2/2).$$

We get the following plot:



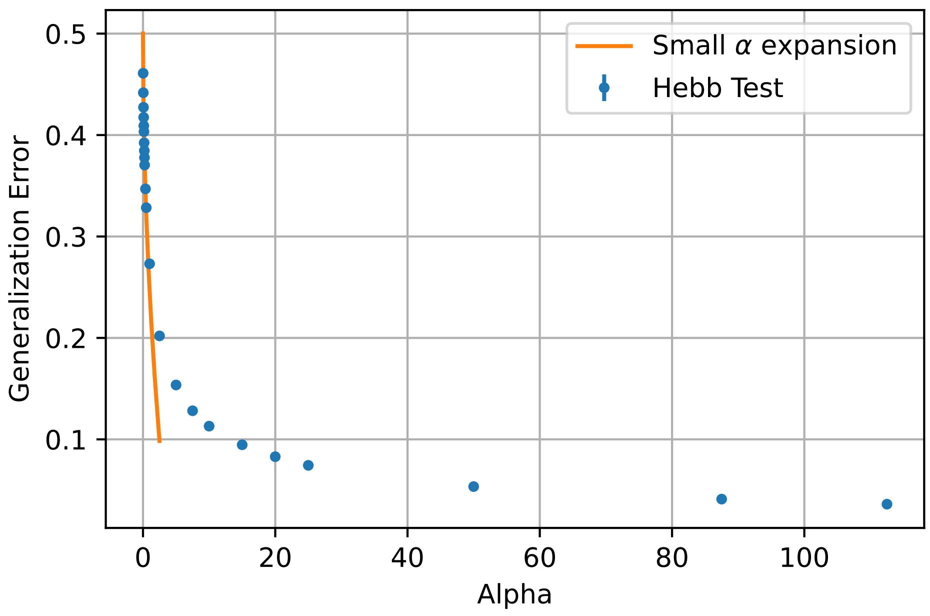
Step 4: Small and large behaviour of ϵ, ϵ_{train} and their difference

The behaviour of ϵ and ϵ_{train} are given by the plot above. We can observe that:

- the training error is not null (this doesn't happen, for instance, in the case of the perceptron rule since we aim for convergence to zero training error)
- the downward slope for small values of α is much steeper with respect to that of the perceptron learning.
- since a finite training error is present, for large P values the generalization error converges towards the curve of the training error and hence seems to decrease less rapidly with respect to the case of the perceptron learning.

From the analytic expectation, we can try to fit it with the α-small or large limit expansion and see whether we get the correct prefactors. For small alpha, we get that

$$\frac{1}{2} - \epsilon \sim \frac{\sqrt{2}}{\pi^{3/2}} \sqrt{\alpha}$$



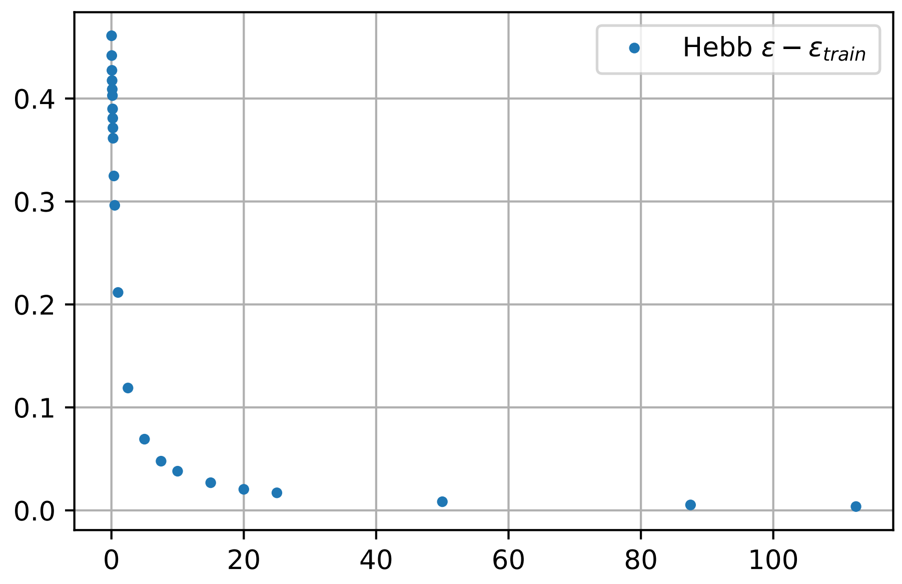
We can then compute the slope between $\frac{1}{2} - \epsilon$ vs. $\sqrt{\alpha}$ by fitting the function. We get the following result:

```
Slope of the fit given by the first 10 points is 0.258155019585139
```

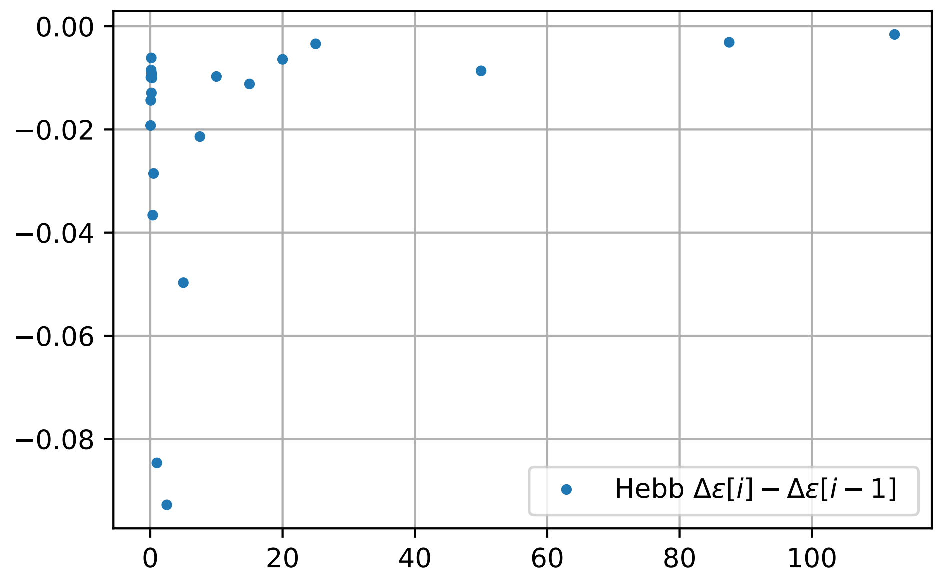
which is compatible with the expected result:

$$\frac{\sqrt{2}}{\pi^{3/2}} \sim 0.25397.$$

The difference $\epsilon - \epsilon_{\text{train}}$ at large α values behaves like the following:



We see that at large α values, the difference between the generalization and training error gets flatter and flatter. We expect that at large values they will behave in the same manner. We can check this by looking at finite differences between successive datapoints:



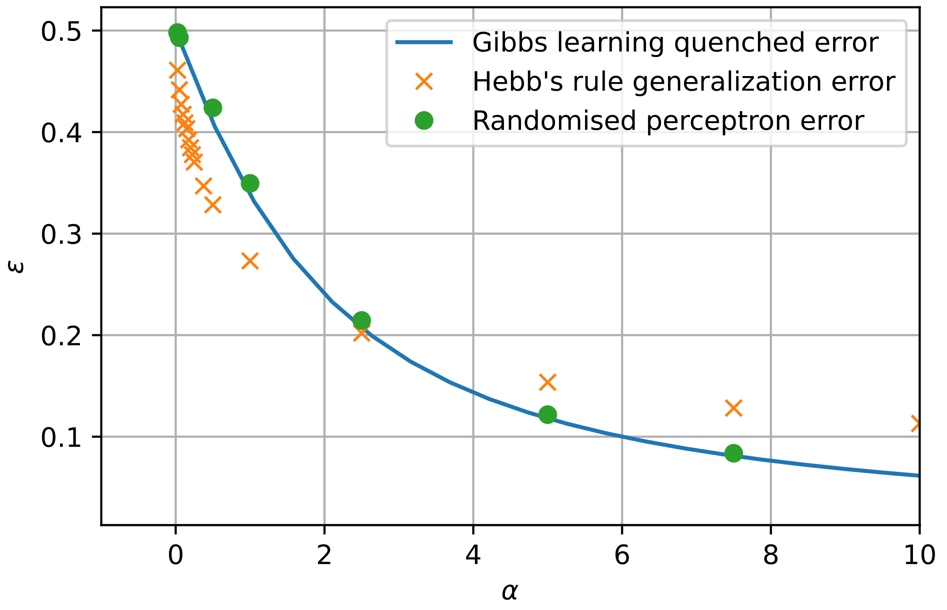
The finite successive differences have a good behaviour, since they converge to value approximately zero. This means that we expect our derivative to get to zero, meaning that the difference between the two errors becomes constant. Since

$$\epsilon_t \sim \epsilon \sim \frac{1}{\sqrt{2\pi\alpha}}$$

is the behaviour we expect, the two results are compatible for, say, $\alpha > 20$.

Step 5: compare Hebbs, randomised perceptron and Gibbs result

We understood in Exercise 05 that Gibbs learning is best represented by the quenched computation. Then let us compare those results with the simulations of the randomised perceptron rule (Exercise 03) and Hebb's rule.

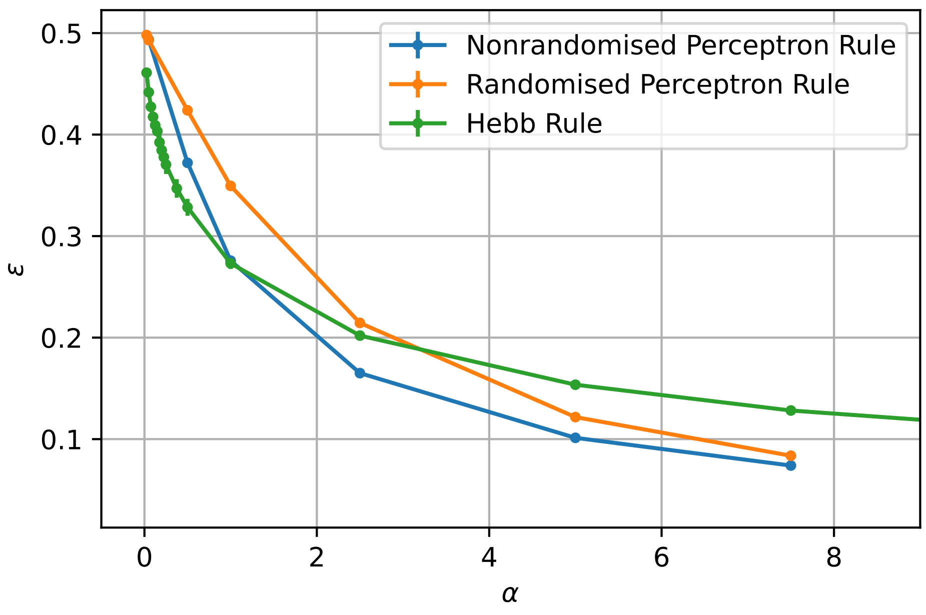


We can notice that at low values of α the generalization error in Hebb's rule decreases much faster with respect to the behaviour of the error in the case of Gibbs rule. The results from the randomised perceptron error lie, as we previously saw, on the error curve of the quenched computations for the Gibbs learning.

Step 6, 7 and 8: apply the (nonrandomised) perceptron rule and discuss the results

We will now perform the same type of analysis that we performed for the Hebbs rule and Gibbs rule (both in the annealed and quenched scenarios) by iterating 1000 times the train-test cycle over multiple values of P and evaluating the resulting behaviour of the generalization error.

```
For P=2, eps=(0.49 +- 0.0), took 0s
For P=20, eps=(0.37 +- 0.0), took 1s
For P=40, eps=(0.28 +- 0.0), took 1s
For P=100, eps=(0.17 +- 0.0), took 2s
For P=200, eps=(0.1 +- 0.0), took 5s
For P=300, eps=(0.07 +- 0.0), took 8s
```



We can clearly see that:

- as pointed out in Step 5, the Hebb's rule performs well at low α's and worse at high α's with respect to both of the other perceptron rule results
- the **nonrandomised** perceptron rule generalization error is *lower* than the randomised one over higher and higher values of α.

We can hence conclude that the nonrandomised perceptron rule yields to better results in terms of generalization error because the noise term does not slow down the learning process, leading to a faster convergence to the zero training error over the dataset.

However, without the noise term the comparison with results from statistical physics is not possible anymore, because computations with the saddle point method are possible thanks to the presence of noise that allows a better sampling over the configuration space.