**English version**

A new version of nnetsauce, version 0.4.0, is now available on Pypi and for **R**. As usual, you can install it on **Python** by using the following commands (command line):

pip install nnetsauce

And if you’re using **R**, it’s still (R console):

library(devtools)

library(nnetsauce)

This new release is accompanied by a few goodies:

1) **New features**, detailed in the changelog.

2) A refreshed web page containing all the information about package installation, use, interface’s work-in-progress documentation, and contribution to package development.

3) A specific RSS Feed related to nnetsauce on this blog (there’s still a general feed containing everything).

4) A working paper related to Bayesianrvfl2Regressor, Ridge2Regressor, Ridge2Classifier, and Ridge2MultitaskClassifier :

Among nnetsauce’s new features, there’s a new model class called MultitaskClassifier, briefly described in the first paper from point 4). It’s a **multitask** classification model based on regression models, with shared covariates. **What does that mean?** We use the figure below to start the explanation:

Imagine that we have 4 fruits at our disposal, and we would like to classify them as avocados (is an avocado a fruit?), apples or tomatoes, by looking at their color and shapes. What we called **covariates** before in model description are color and shape, also known as explanatory variables or predictors. The column containing fruit names in the figure – on the left – is a so-called **response**; a variable that MultitaskClassifier must learn to classify (which is typically much larger). This raw response is transformed into a one-hot encoded one – on the right.

Instead of one response vector, we now have three different responses. And instead of one classification problem on one response, three different two-class classification problems on three responses: **is this fruit an apple or not? Is this fruit a tomato or not? Is this fruit an avocado or not?** All these three problems share the same covariates: color and shape.

MultitaskClassifier can use any regressor (meaning, a statistical learning model for continuous responses) to solve these three problems; with the same regressor being used for all three of them – which is *a priori* a relatively strong hypothesis. Regressor’s predictions on each response are interpreted as raw probabilities that the fruit is either one of them or not.

We now use MultitaskClassifier on breast cancer data, for AdaBoostClassifier, to illustrate how it works. The **R** version for this code would be almost identical, replacing “.”’s by “$”’s.

**Import packages:**

import nnetsauce as ns

import numpy as np

from sklearn.datasets import load\_breast\_cancer

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

from time import time

**Model fitting on training set:**

breast\_cancer = load\_breast\_cancer()

Z = breast\_cancer.data

t = breast\_cancer.target

# Training/testing datasets (using reproducibility seed)

np.random.seed(123)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(Z, t, test\_size=0.2)

# Linear Regression is used (can be anything, but must be a regression model)

regr = LinearRegression()

fit\_obj = ns.MultitaskClassifier(regr, n\_hidden\_features=5,

n\_clusters=2, type\_clust="gmm")

# Model fitting on training set

start = time()

fit\_obj.fit(X\_train, y\_train)

print(time() - start)

# Model accuracy on test set

print(fit\_obj.score(X\_test, y\_test))

# Area under the curve

print(fit\_obj.score(X\_test, y\_test, scoring="roc\_auc"))

These results can be found in nnetsauce/demo/. MultitaskClassifier’s accuracy on this dataset is 99.1%, and other indicators such as precision are equal to 99% on this dataset too. Let’s visualize the **missclassification results**,

In this case with MultitaskClassifier, and no advanced hyperparameter tweaking, there is one patient out of 114 who is missclassified. A robust way to understand MultitaskClassifier’s accuracy on this dataset using these same parameters, could be to repeat the same procedure for multiple random reproducibility seeds (see code, the training and testing sets randomly change when we change the seed, and we change MultitaskClassifier’s seed too).

We obtain the results below for 100 reproducibility seeds. The accuracy is always at least 90%, mostly 95% and quite sometimes, higher than 98% (with no advanced hyperparameter tweaking).