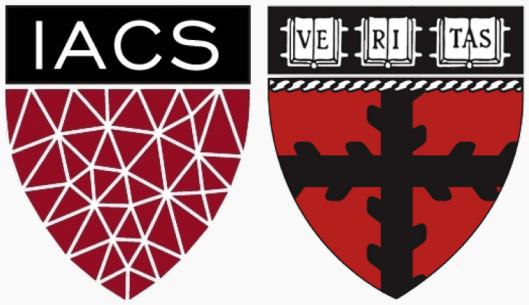
Lecture 21: Stacking

CS109A Introduction to Data Science

Pavlos Protopapas and Kevin Rader



Outline

- General Review of Methods
- Stacking

Module 1: Regression Methods

When is it appropriate to perform a regression method? What regression models have we learned?

1. Linear Regression (simple, multiple, polynomial, interactions, model selection, Ridge & Lasso, etc...)
2. *k*-NN
3. Regression Trees

What is the main difference between these two types of models (advantages and disadvantages)? When should you use each method?

Module 2: Classification Methods

When is it appropriate to perform a classification method? What classification models have we learned?

1. Logistic Regression: same details as linear regression apply
2. *k*-NN
3. Discriminant Analysis: LDA/QDA
4. Classification Trees
5. SVM

What is the main difference between these two types of models (advantages and disadvantages)? When should you use each method?

Module 3: Ensemble Methods

What does it mean for a model to be an ensemble method?

1. Bagging Trees
2. Random Forests
3. Boosting Models
4. Neural Networks
5. Stacking Models (coming today)

What approach does each model take to improve prediction accuracy?

Bags and Forests of Trees (cont.)

Bagging:

- create an ensemble of full trees, each trained on a bootstrap sample of the training set;
- "average" the predictions

Random forest:

- create an ensemble of full trees, each trained on a bootstrap sample of the training set;
- in each tree and each split, randomly select a subset of predictors, choose a predictor from this subset for splitting;
- average the predictions

Note that the ensemble building aspects of both method are embarrassingly parallel!

Bags and Forests of Trees (cont.)

Boosting:

- Iteratively build a model from lots of little models.
- Each subsequent model predicts the residuals from the previous model, overweighting the large residuals.

Neural Networks:

- Build layers of models based on overly simple “neurons” of models.
- Uses back-propagation to efficiently communicate between output of models to update earlier models.

These methods are not as easily parallelizable.

Stacking

Motivation for Stacking

For each of our ensemble methods so far (besides Neural Networks), we have:

- Fit the base model on the same type (regression trees, for example).
- Combined the predictions in a naïve way.

Stacking is a way to generalize the ensembling approach to combine outputs of various types of model, and improves on the combination as well.

Motivation for Stacking

Recall that in boosting, the final model T, we learn is a weighted sum of simple models, T_h,

$$T = \sum_h \lambda_h T_H$$

where λ_h is the learning rate. In AdaBoost for example, we can analytically determine the optimal values of λ_h for each simple model T_h.

On the other hand, we can also determine the final model T implicitly by ***learning any model, called meta-learner, that transforms the outputs of T_h into a prediction***.

Stacked Generalization

The framework for ***stacked generalization*** or ***stacking*** (Wolpert 1992) is:

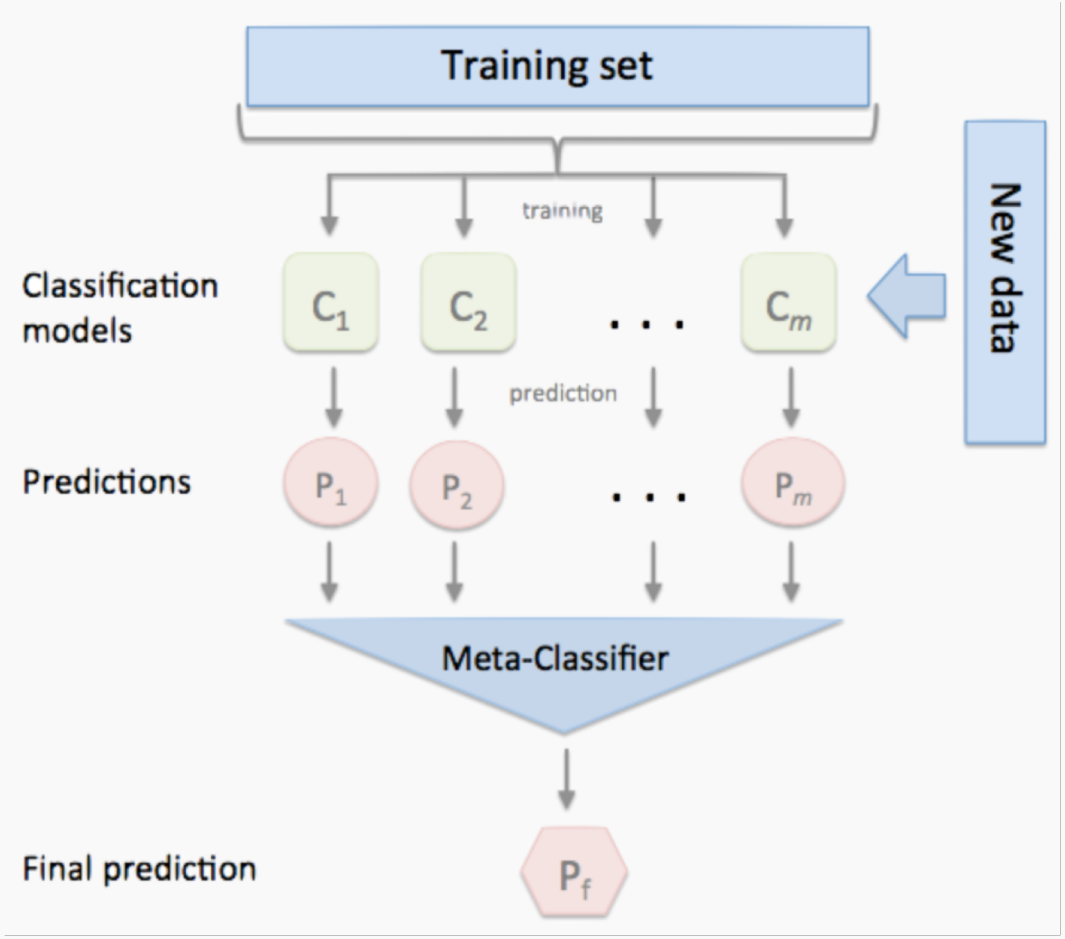
- train L number of models, T_l on the training data

$$\{(x_1, y_1), ..., (x_N, y_N)\}$$

- train a meta-learner $\tilde{T}$ on the predictions of the ensemble of models, i.e. train using the data

$$\{(T_1(x_1), ..., T_L(x_1), y_1), ..., (T_1(x_N), ..., T_L(x_N), y_N)\}$$

Stacking: an Illustration



Stacked Generalization

Stacking is a very general method,

- the models, T_l, in the ensemble can come from different classes. The ensemble can contain a mixture of logistic regression models, trees, random forests, etc.
- the meta-learner, T, can be of any type. **Note:** we want to train T on the ***out of sample*** predictions of the ensemble. For example we train T on

$$\{(T_1(x_1), ..., T_L(x_1), y_1), ..., (T_1(x_N), ..., T_L(x_N), y_N)\}$$

where $T_l(x_n)$ is generated by training T_l on

$$\{(x_1, y_1), ..., (x_{n-1}, y_{n-1}), (x_{n+1}, y_{n+1}), ...(x_N, y_N)\}$$

Stacking: General Guidelines

The flexibility of stacking makes it widely applicable but difficult to analyze theoretically. Some general rules have been found through empirical studies:

- models in the ensemble should be diverse, i.e. their errors should not be correlated.
- for binary classification, each model in the ensemble should have error rate < 1/2.
- if models in the ensemble outputs probabilities, it's better to train the meta-learner on probabilities rather than predictions.
- apply regularization to the meta-learner to avoid overfitting.

Stacking: Subsemble Approach

We can extend the stacking framework to include ensembles of models that specialize on small subsets of data (Sapp et. al. 2014), for de-correlation or improved computational efficiency:

- divide the data into *J* subsets
- train models, T_j , on each subset
- train a meta-learner $\tilde{T}$ on the predictions of the ensemble of models, i.e. train using the data

$$\{(T_1(x_1), ..., T_J(x_1), y_1), ..., (T_1(x_N), ..., T_J(x_N), y_N)\}$$

Again, we want to make sure that each $T_j(x_i)$ is an out of sample prediction.

Stacking in sklearn

Unfortunately, Python does not have stacking algorithms implemented for you ☹

So how can we do it?

We can set it up by 'manually' fitting several base models, take the outputs of those models, and fitting the meta model on the outputs of those base models.

It's a model on models!