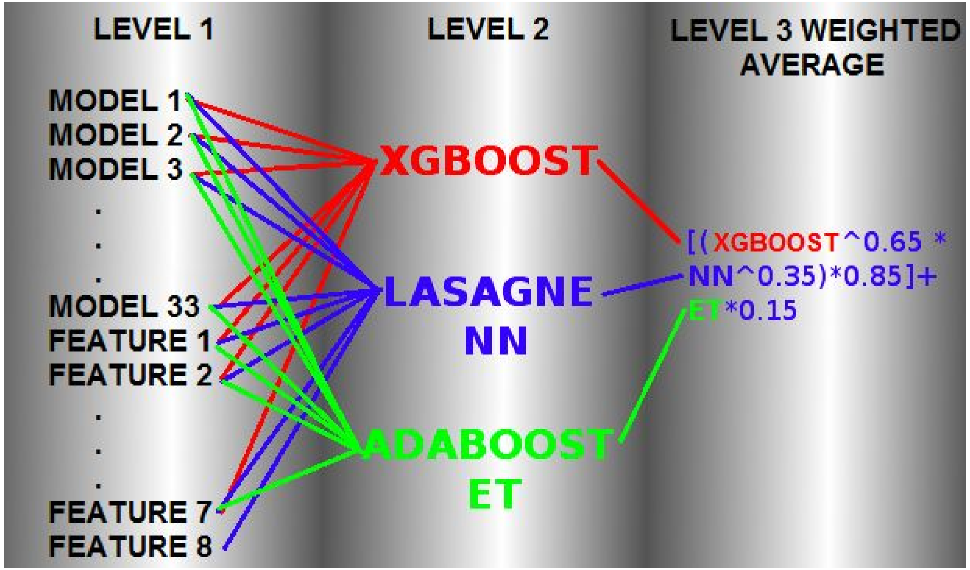
CDS
Cornell Data Science

Meta-Learning

Layers of Learning

Gilberto Titericz Junior (top-ranked user on [Kaggle.com](https://kaggle.com)) used this setup to win the $10,000 Otto Group Product Classification Challenge.



33 models???

3 levels???

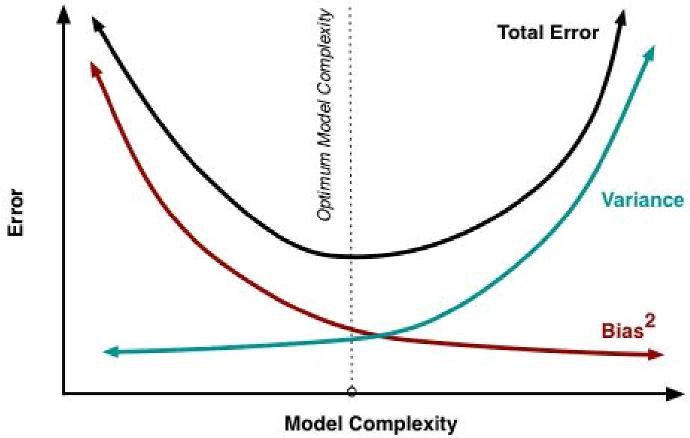
Lasagne NN??

Why so many models?

A single model on its own is often prone to bias and/or variance.

- **Bias** - Systematic or “consistent” error. Associated with underfitting.
- **Variance** - Random or “deviating” error. Associated with overfitting.

A tradeoff exists. We want to minimize both as much as we can.



Source

Ensembles and Hypotheses

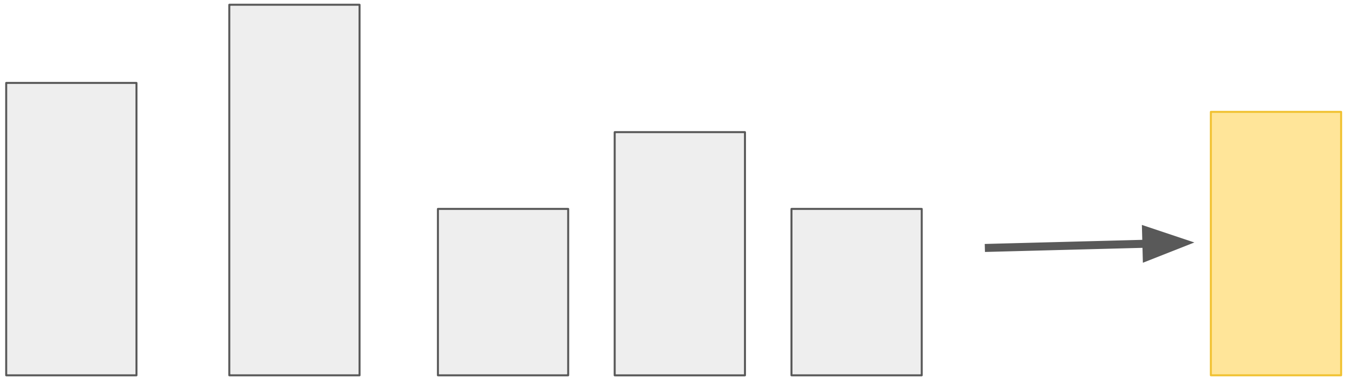
- Recall the definition of "hypothesis."
- Machine learning algorithms search the **hypothesis space** for hypotheses.
    - Set of mathematical functions on real numbers
    - Set of possible classification boundaries in feature space
- More searchers → more likely to find a "good" hypothesis

Source

Introduction: Ensemble Averaging

Basic ensemble composed of a **committee** of learning algorithms.

Results from each algorithm are averaged into a final result, reducing variance.



More Sophisticated Ensembles

Three important ensembles to know:

Boosting

[Source](#)

Bagging

[Source](#)

Stacking

[Source](#)

Boosting

A **sequential ensemble**. Models are applied one-by-one based on how previous models have done.

- Apply a model on a subset of data.
- Check to see where the model has badly classified data.
- Apply another model on a new subset of data, giving preference to data badly classified by the model.

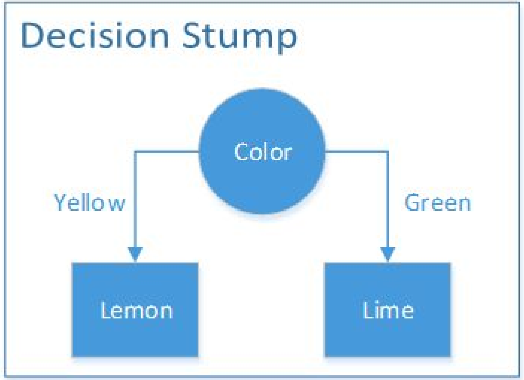
Boosting decreases **bias** and prevents **underfitting.**

Source

Weak Learners

Important concept in boosting.

Weak learners do only slightly better than the baseline for a given dataset. In isolation, they are not very useful.



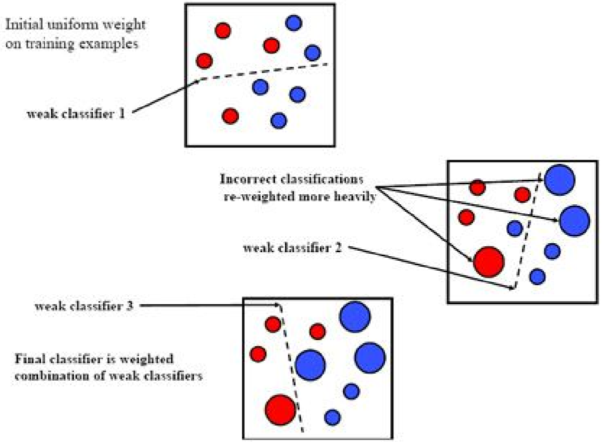
While boosting, we improve these learners sequentially to create hyper-powered models.

Source

AdaBoost

Short for **ada**ptive **boost**ing. Sequentially generates weak learners, adjusting newer learners based on mistakes of older learners

Combines output of all learners into weighted sum



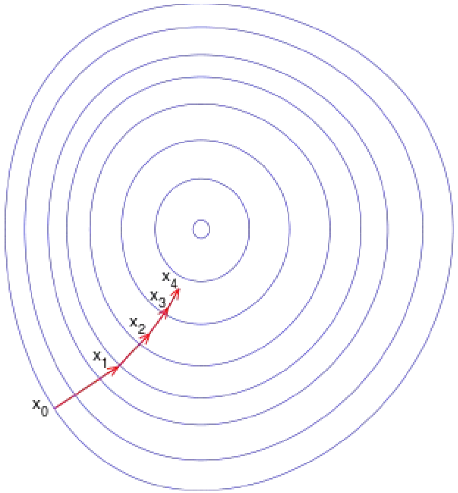
$$H(x) = sign(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$$

Source

XGBoost

Short for e**X**treme **G**radient **Boost**ing. Sequentially generates weak learners like AdaBoost

- Updates model by computing cost function
    - Computes gradient of cost function
    - Direction of greatest decrease = negative of gradient
    - Creates new learner with parameters adjusted in this direction



Source

Bagging

Short for <u>**b**</u>ootstrap <u>**agg**</u>regat<u>**ing**</u>.

A **parallel ensemble**. Models are applied without knowledge of each other.

- Apply each model on a random subset of data.
- Combine the output by averaging (for regression) or by majority vote (for classification)
- A more sophisticated version of ensemble averaging.

Bagging decreases **variance** and prevents **overfitting.**

Source

Random Forests

Designed to improve accuracy over CART.

Much more difficult to overfit

- Works by building a large number of CART trees
  - Each tree in the forest “votes” on outcome
  - Outcome with the most votes becomes our prediction

Source

Random Forests

- Random forest changes each tree's training data
    - Disadvantage: Makes model harder to understand and follow
- Each tree is trained on a random subset of the data
    - Example - original data: 1 2 3 4 5
    - New data:

2 4 5 2 1 ----> first tree

4 1 3 2 1 ----> second tree

3 5 1 5 2 ----> third tree

Random Forest Parameters

- Minimum number of observations in a branch
  - `nodesize` parameter, similar to `minbucket` in CART
  - Smaller the node size, more branches, longer the computation

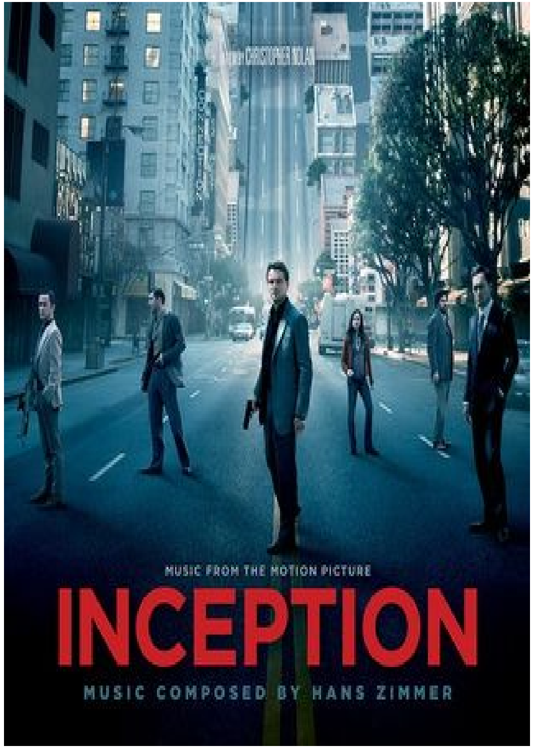
- Number of trees
  - `ntree` parameter
  - Fewer trees means *less accurate* prediction
  - More trees means *longer computation* time
  - Diminishing returns after a couple hundred trees

Stacking

Linear regression...

...on models.

Source

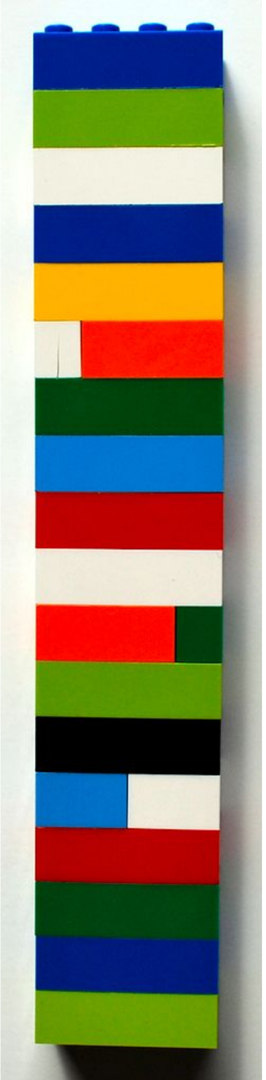


Source

Stacking pt. 1

Assumption: can improve performance by taking a **weighted average** of the predictions of models.

- Take a bunch of machine learning models.
- Apply these models on subsets of your data (how you choose them is up to you).
- Obtain predictions from each of the models.



Source

Stacking pt. 2

Once we have predictions from each individual model...

- Perform linear regression on the predictions.
  - This gives you the coefficients of the weighted average.
- Result: a massive blend of potentially hundreds of models.

Source

CDS Core Team Example: Stacking

CDS Kaggle Team (2017 March Madness Kaggle competition)

- Each member of the Kaggle team made a logistic regression model based on different features
- Combined these using a stacked model

Source

Coming Up

Your problem set: Project 2

Next week: Machine learning on text data

See you then!