EEG Seizure Detection Competition

BitTiger丨来自硅谷的终身学习平台

Rand Xie

Disclaimer   Any opinions and code here are my own, and in no way reflect that of MathWorks

Last Lecture

- Introduction to EEG dataset and Kaggle community
- Time series feature extraction
- From loading data to submission
- Introduction to feature selection
- Introduction to parameter tuning
- Boosting
- Ensemble learning

Recap: From loading data to submission

- Datawarehouse: A class to handle data exchange
  - Read and process data
  - Select features
  - Generate submission files
- Model: A class to store machine learning models
  - Training models
  - Cross validation

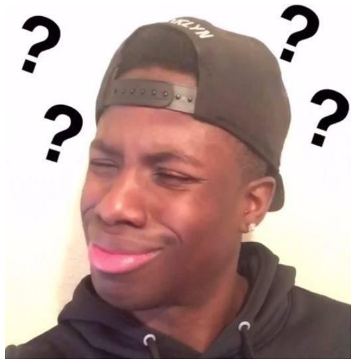
What happens to the private leaderboard

#	△1w	Team Name ✱ in the money	Kernel	Team Members	Score	Entries	Last
1	▲1	✱ Not-so-random-anymore			0.80701	260	3mo
2	▲35	✱ Areté Associates			0.79898	56	3mo
3	▲12	✱ GarethJones			0.79652	74	3mo
4	▲23	QingnanTang			0.79458	62	3mo
5	▲11	nullset			0.79363	119	3mo
6	▲14	tralala boum boum pouêt pouêt			0.79197	57	3mo
7	▲7	Medrr			0.79183	89	3mo
8	▲14	michaln			0.79074	48	3mo
9	▼8	DataSpring			0.79053	55	3mo
10	▼5	fugusuki			0.78773	82	3mo

Net LB Gain for top 10: **104**

Our result for L1-LR:

- Local: 0.93429
- Public LB: 0.63398
- Private LB: 0.65294



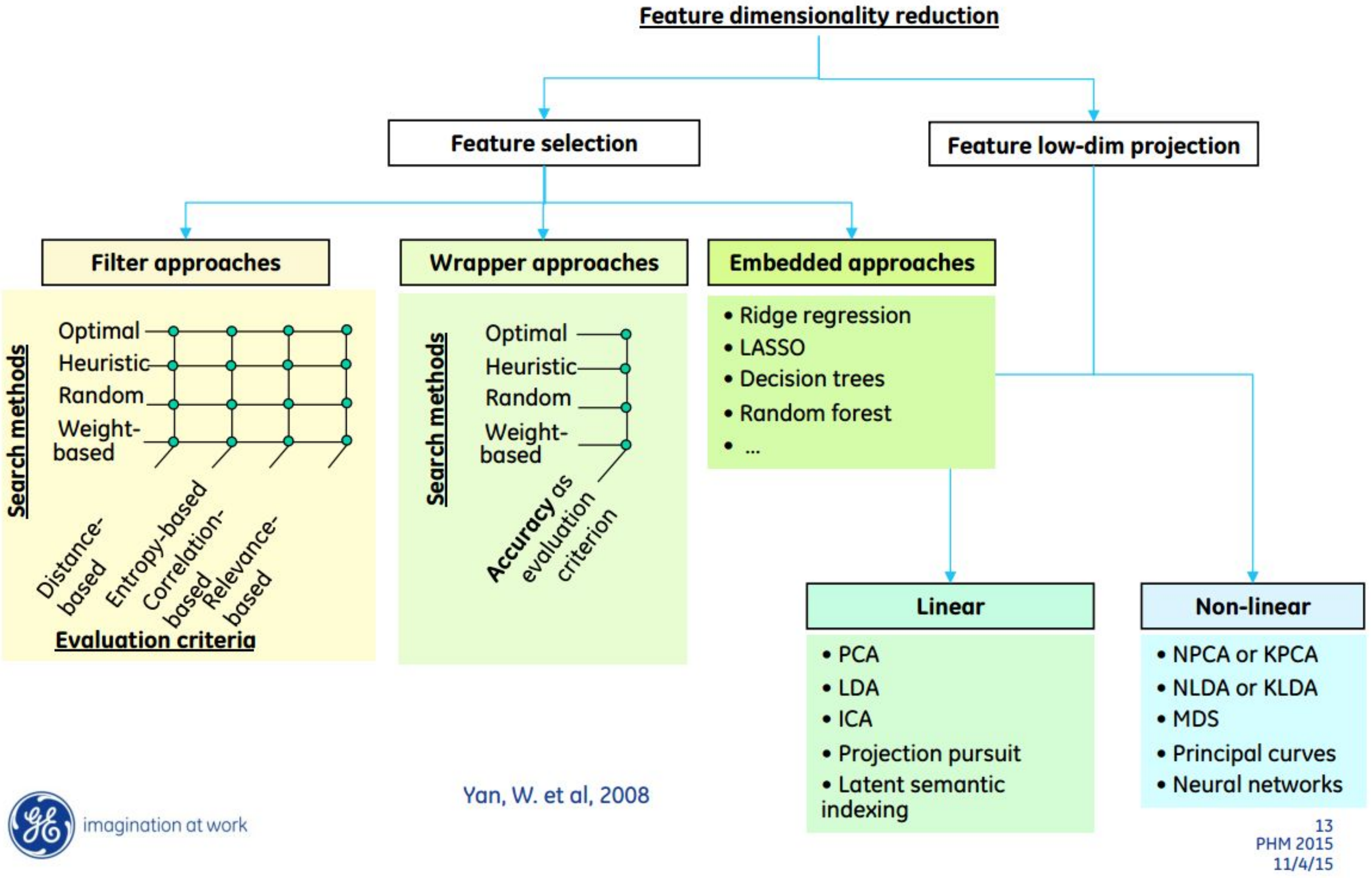
Cross Validation Methods

- K-fold
- Leave p out
- Based on group
- Sequential

Use group cross validation to close the gap

- Local: 0.79080
- Public LB: 0.63398
- Private LB: 0.65294

Feature Selection



Filter Approaches

`sklearn.feature_selection`: Feature Selection

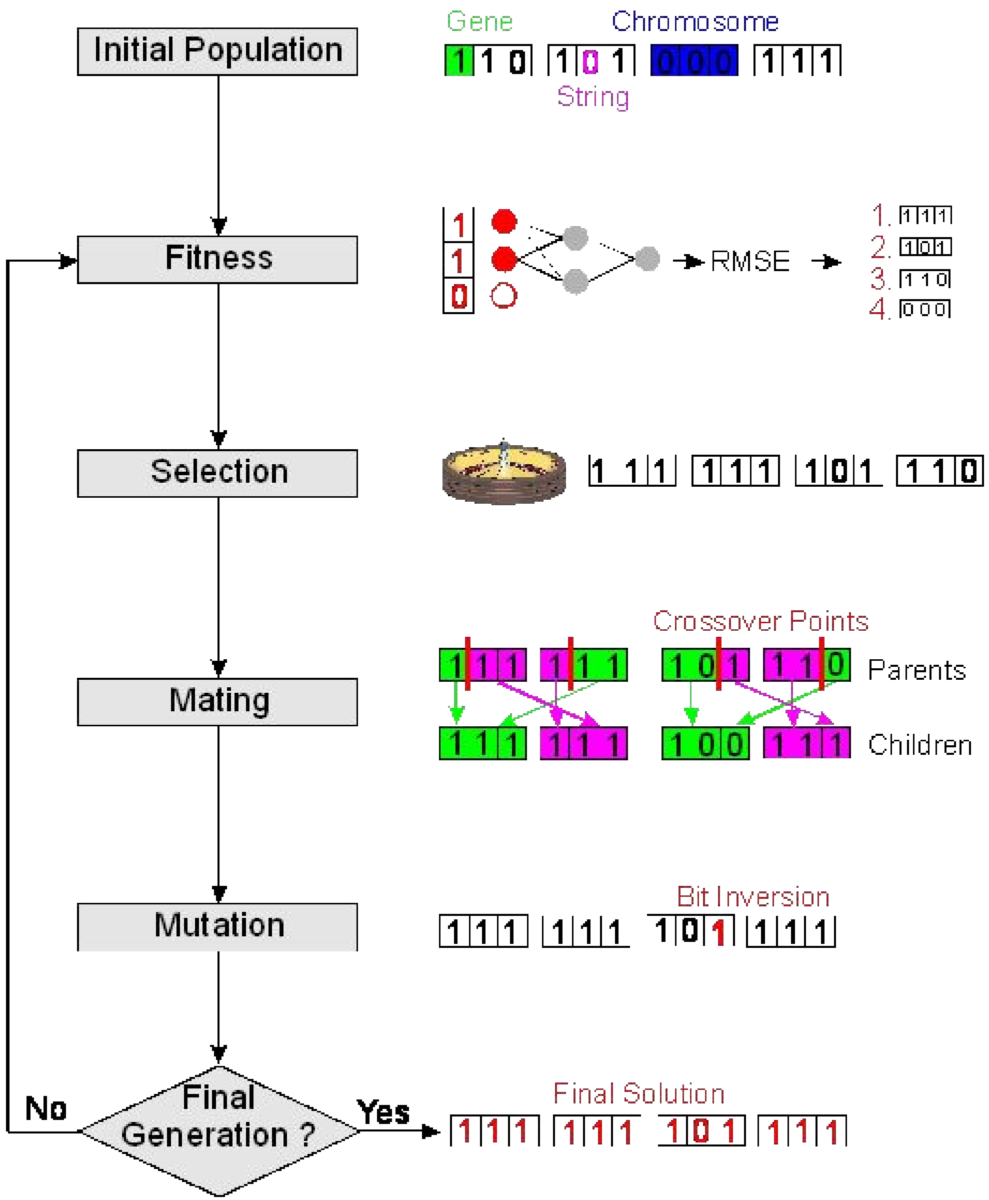
The `sklearn.feature_selection` module implements feature selection algorithms. It currently includes univariate filter selection methods and the recursive feature elimination algorithm.

User guide: See the Feature selection section for further details.

`feature_selection.GenericUnivariateSelect` ([...])	Univariate feature selector with configurable strategy.
`feature_selection.SelectPercentile` ([...])	Select features according to a percentile of the highest scores.
`feature_selection.SelectKBest` ([score_func, k])	Select features according to the k highest scores.
`feature_selection.SelectFpr` ([score_func, alpha])	Filter: Select the pvalues below alpha based on a FPR test.
`feature_selection.SelectFdr` ([score_func, alpha])	Filter: Select the p-values for an estimated false discovery rate
`feature_selection.SelectFromModel` (estimator)	Meta-transformer for selecting features based on importance weights.
`feature_selection.SelectFwe` ([score_func, alpha])	Filter: Select the p-values corresponding to Family-wise error rate
`feature_selection.RFE` (estimator[, ...])	Feature ranking with recursive feature elimination.
`feature_selection.RFECV` (estimator[, step, ...])	Feature ranking with recursive feature elimination and cross-validated selection of the best number of features.
`feature_selection.VarianceThreshold` ([threshold])	Feature selector that removes all low-variance features.
`feature_selection.chi2` (X, y)	Compute chi-squared stats between each non-negative feature and class.
`feature_selection.f_classif` (X, y)	Compute the ANOVA F-value for the provided sample.
`feature_selection.f_regression` (X, y[, center])	Univariate linear regression tests.
`feature_selection.mutual_info_classif` (X, y)	Estimate mutual information for a discrete target variable.
`feature_selection.mutual_info_regression` (X, y)	Estimate mutual information for a continuous target variable.

Wrapper Approaches - Genetic Algorithm

- Use binary vector to represent feature selection
- Randomized selection algorithm
- Computational Intensive



Embedded Approaches

Let’s try L1 regularized logistic regression

Comparison

Model	Local	Public	Private
L1 Regularized LR	0.79080	0.63398	0.65289
Random Forest	0.82348	0.75597	0.72346 (rank 66, bronze)
Random Forest with L1	0.82277	0.75811	0.72965 (rank 66, bronze)

Parameter Tuning

- Graduate student descent
- Grid search
- Random search (Genetic algorithm can also be used here)
- Bayesian optimization

Comparison

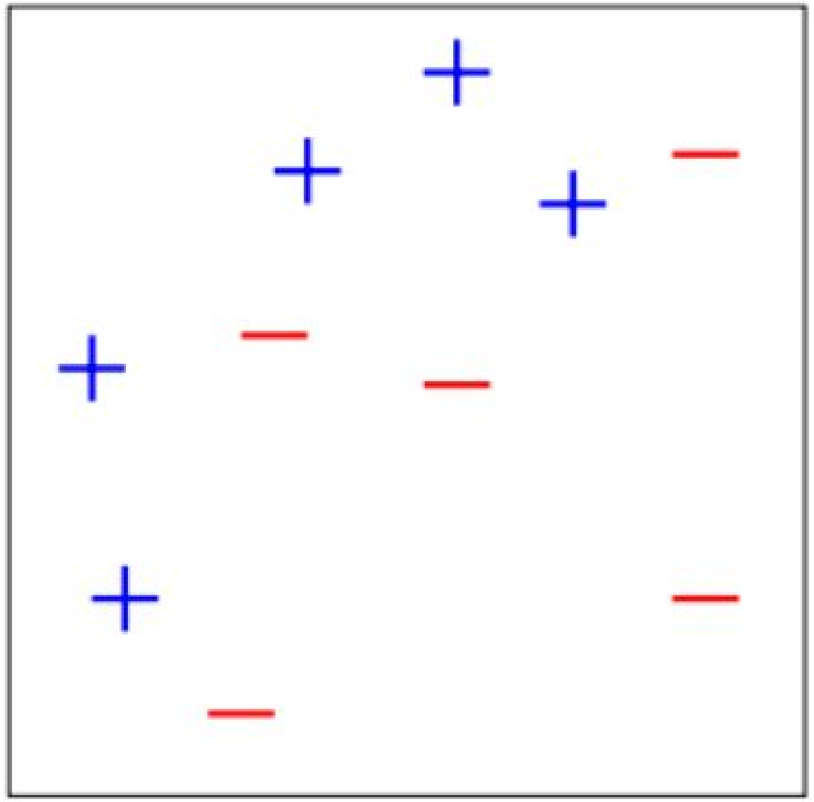
Model	Local	Public	Private
L1 Regularized LR	0.79080	0.63398	0.65289
L1 Regularized LR (C=0.465, 193 iteration)	0.79332	0.64986	0.65371
Random Forest	0.82348	0.75597	0.72346 (rank 66, bronze)
Random Forest with L1	0.82277	0.75811	0.72965 (rank 66, bronze)

Boosting

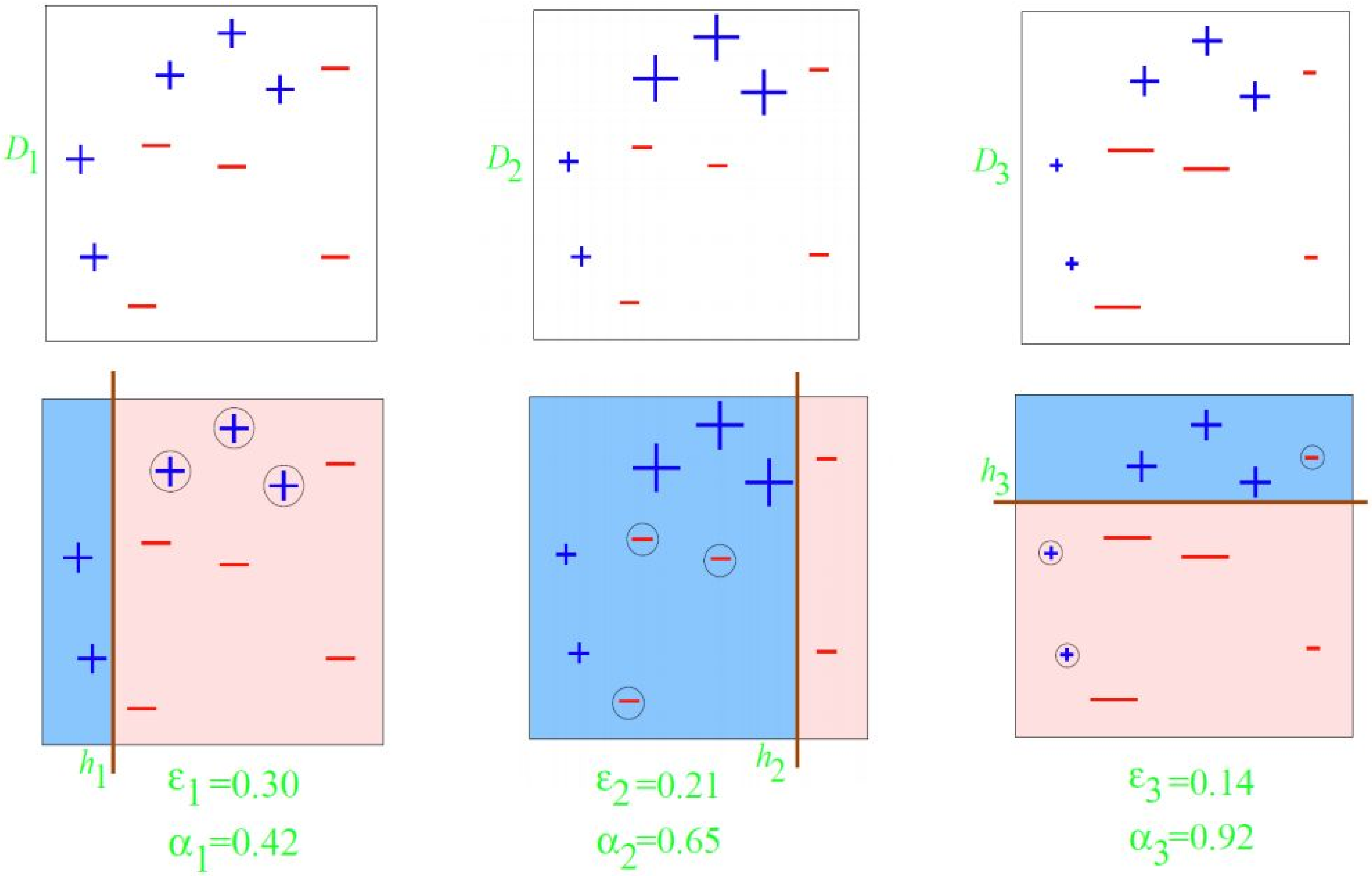
- Use several weak learners to approximate a strong learner
- Introduce adaptive boosting (adaboost) and gradient boosting

Adaboost

- How to classifier the following data?

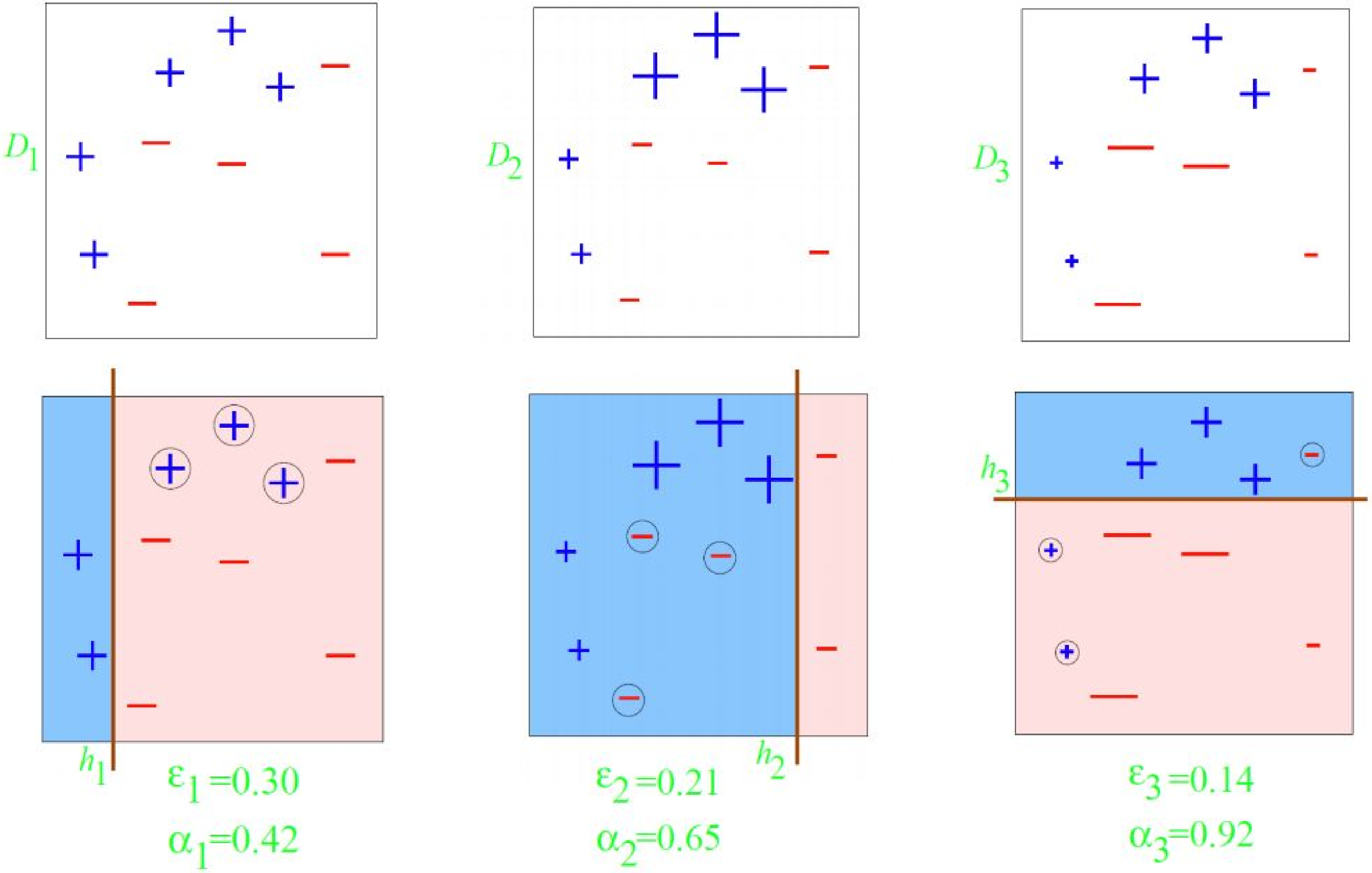


Adaboost



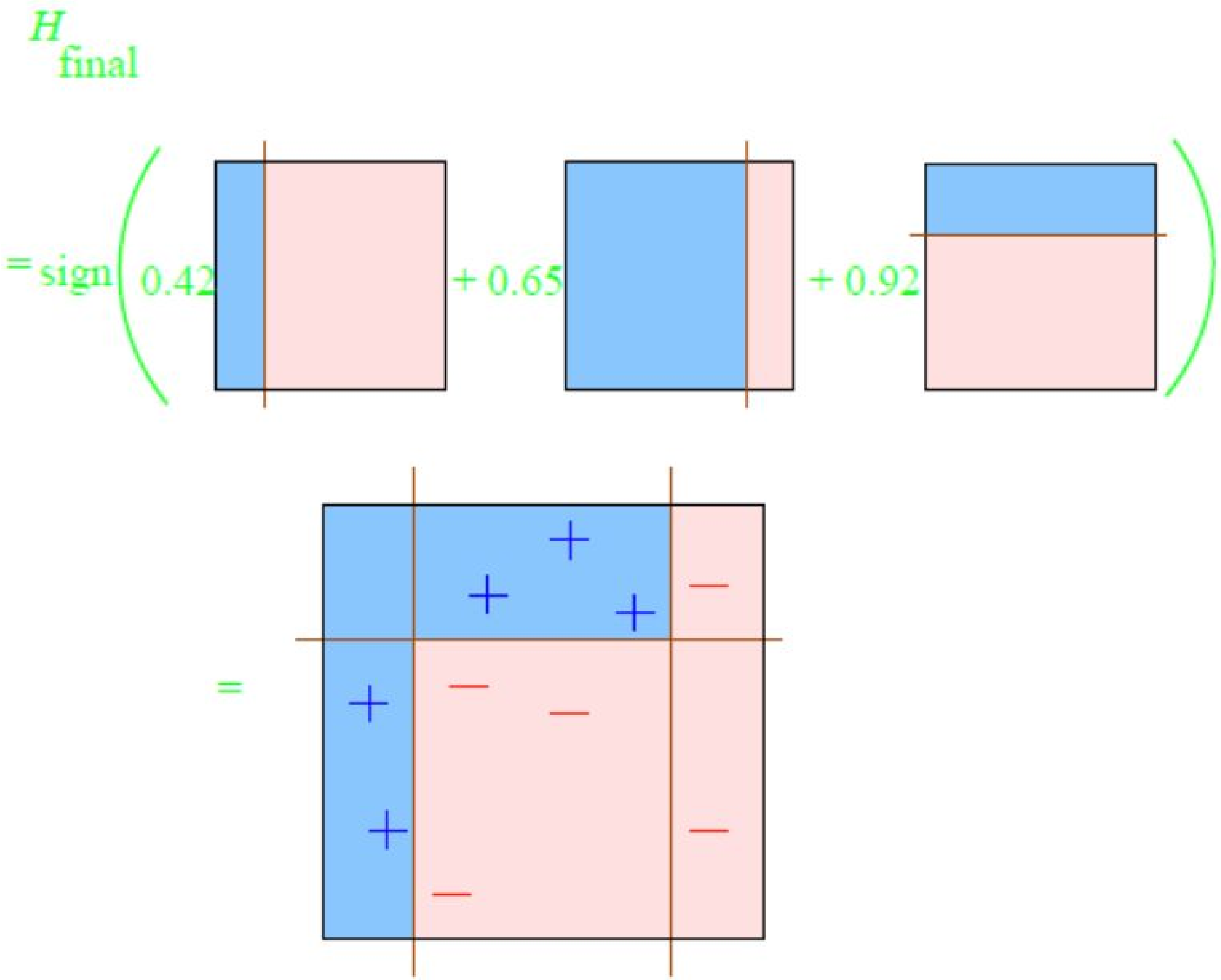
D_1
D_2
D_3
h_1
ε_1=0.30
α_1=0.42
h_2
ε_2=0.21
α_2=0.65
h_3
ε_3=0.14
α_3=0.92

Adaboost



D_1
D_2
D_3
h_1
h_2
h_3
ε_1=0.30
α_1=0.42
ε_2=0.21
α_2=0.65
ε_3=0.14
α_3=0.92

Adaboost



Adaboost

Given: $(x_1, y_1), \ldots, (x_N, y_N)$ where $x_i \in X$, $y_i \in \{-1, +1\}$
Initialize $D_1(i) = 1/N$.
For $t = 1, \ldots, T$:

- Train weak learner using training data weighted according to distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$.
- Measure "goodness" of h_t by its weighted error with respect to D_t:

$$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i: h_t(x_i) \neq y_i} D_t(i).$$

- Let $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final classifier:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

- Re-weight samples at each round
- Get improvement every round

Theorem $\epsilon_t = 1/2 - \gamma_t$ (error of h_t over D_t)

$$err_S(H_{final}) \leq \exp\left[-2 \sum_t \gamma_t^2\right]$$

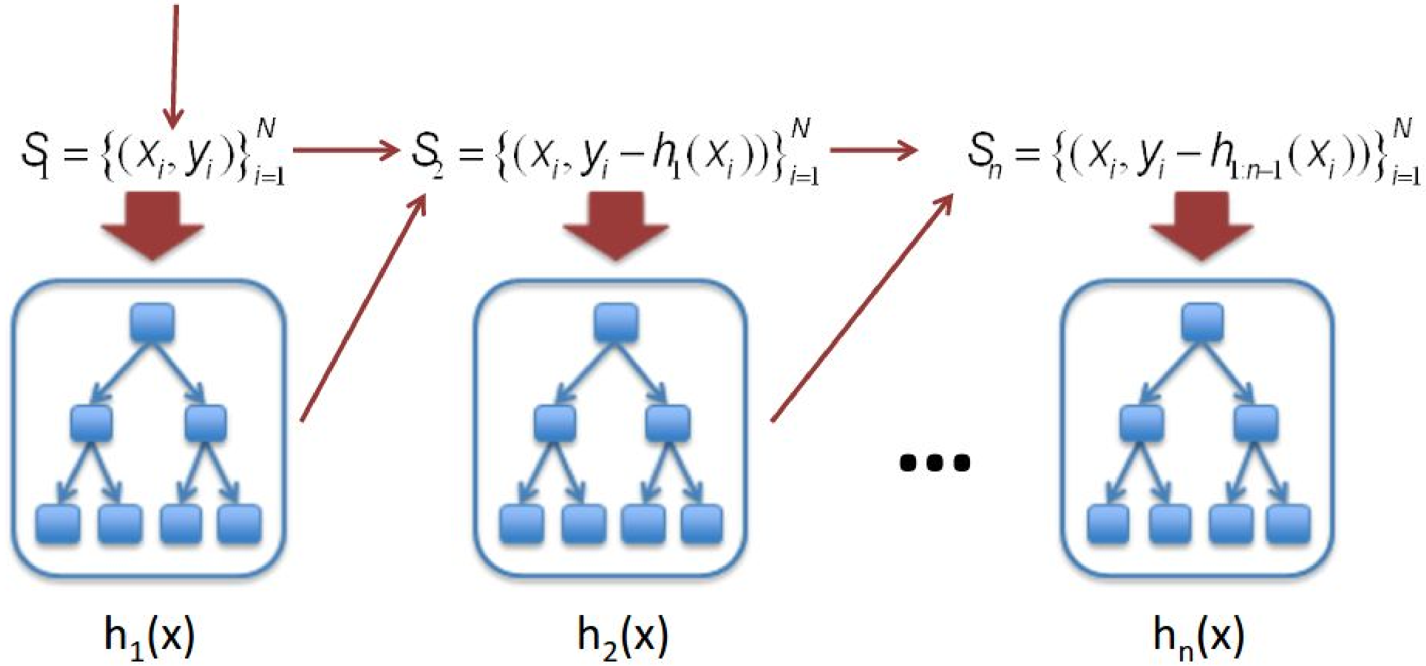
So, if $\forall t, \gamma_t \geq \gamma > 0$, then $err_S(H_{final}) \leq \exp[-2\gamma^2 T]$

Ref: https://www.cs.cmu.edu/~ninamf/courses/601sp15/slides/15_boosting_3-16-2015.pdf

Gradient Boosting

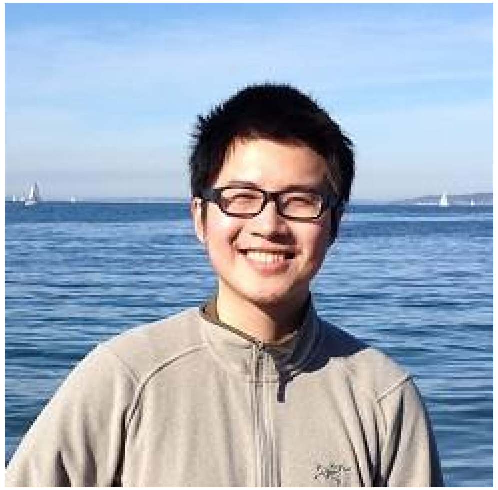
- Combine gradient descent and boosting
- Do not re-weight data points but to approximate residuals

$$S = \{(x_i, y_i)\}_{i=1}^{N} \qquad h(x) = h_1(x) + h_2(x) + \ldots + h_n(x)$$



Xgboost

- a very fast gradient boosting package
- have L1 and L2 regularization
- second order approximation to loss function



ref: http://xgboost.readthedocs.io/en/latest/model.html

Xgboost Parameters

• objective:	'binary:logistics', 'reg:linear', 'multi:softmax'
• max_depth:	maximum depth for each tree
• learning_rate:	used to shrink the feature weights
• reg_lambda:	L1 regularization on weights
• reg_alpha:	L2 regularization on weights
• n_estimators:	number of rounds
• subsample:	subset of data points used to train a tree
• colsample_bytree:	subset of features used to train a tree

ref: [1] https://github.com/dmlc/xgboost/blob/master/doc/parameter.md
[2] https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

Comparison

Model	Local	Public	Private
L1 Regularized LR	0.79080	0.63398	0.65289
L1 Regularized LR (C=0.465, 193 iteration)	0.79332	0.64986	0.65371
Random Forest	0.82348	0.75597	0.72346 (rank 66, bronze)
Random Forest with L1	0.82277	0.75811	0.72965 (rank 66, bronze)
Xgboost	0.84790	0.73464	0.73233 (rank 56, bronze)

Ensemble learning

- Three ways to improve scores
  - Feature engineering
  - Parameter tuning
  - Ensemble learning

Why ensemble works?

- Independent classifiers
  - how to create independence
    - Structural dissimilarity (empirical observation)
    - Different subsets of features (random forest)
    - Different random seeds
    - …… (more to explore)

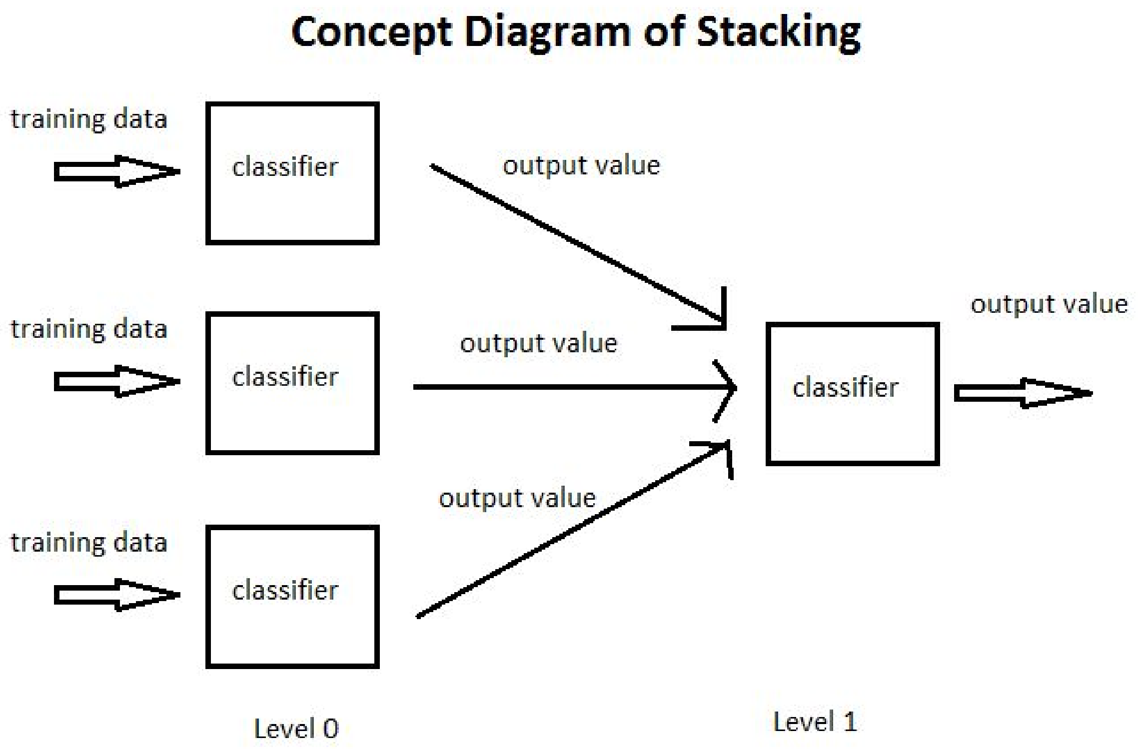
How to combine different models?

- Rank average
- Geometric mean
- Blending
- Stacking (we will do this together)
- …….

ref: http://mlwave.com/kaggle-ensembling-guide/

Stacking

- Get famous because it gives good result in Netflix competition



Comparison

Model	Local	Public	Private
L1 Regularized LR	0.79080	0.63398	0.65289
L1 Regularized LR (C=0.465, 193 iteration)	0.79332	0.64986	0.65371
Random Forest	0.82348	0.75597	0.72346 (rank 66, bronze)
Random Forest with L1	0.82277	0.75811	0.72965 (rank 66, bronze)
Xgboost	0.84790	0.73464	0.73233 (rank 56, bronze)
Ensemble xgboost, rf and extra-tree	N/A	0.75353	0.74200 (rank 46, silver)

To improve:

- Extract more features + Ensemble more models
- Train patient-specific models

Course Summary

- Overview of whole data processing workflow
- Commonly used techniques in Kaggle (xgboost, ensemble models)
- Similar techniques can be applied to other competitions
- Get involved in the Kaggle community