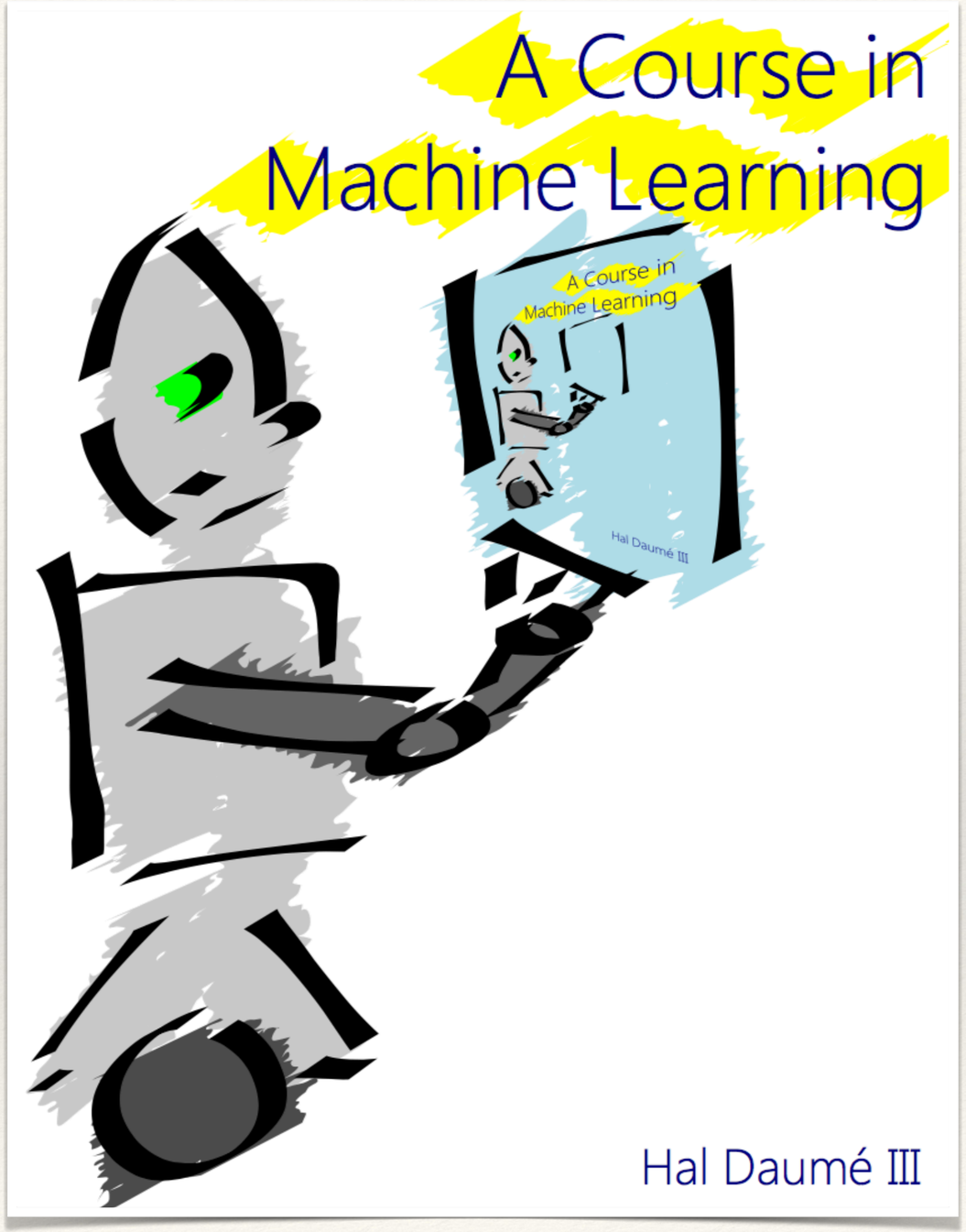
CropS_545 Statistical Genomics 2018

Introduction to Machine Learning and Ensemble Methods

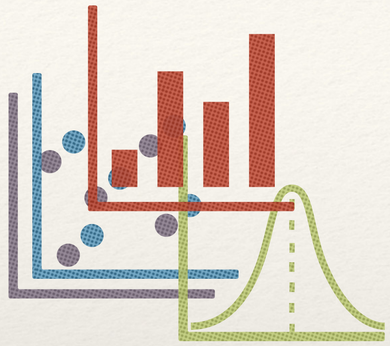
James Chen



http://ciml.info

What is Machine Learning?

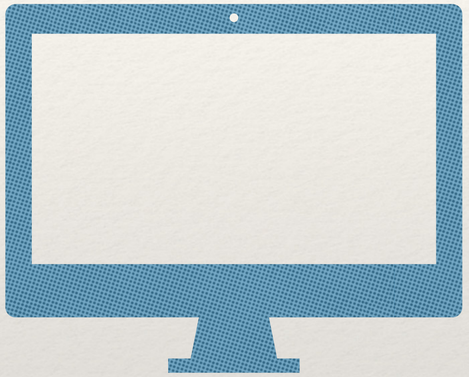
Traditional Programming



Input

```
If x > y
  return x*3 + 5
else
  return 1
```

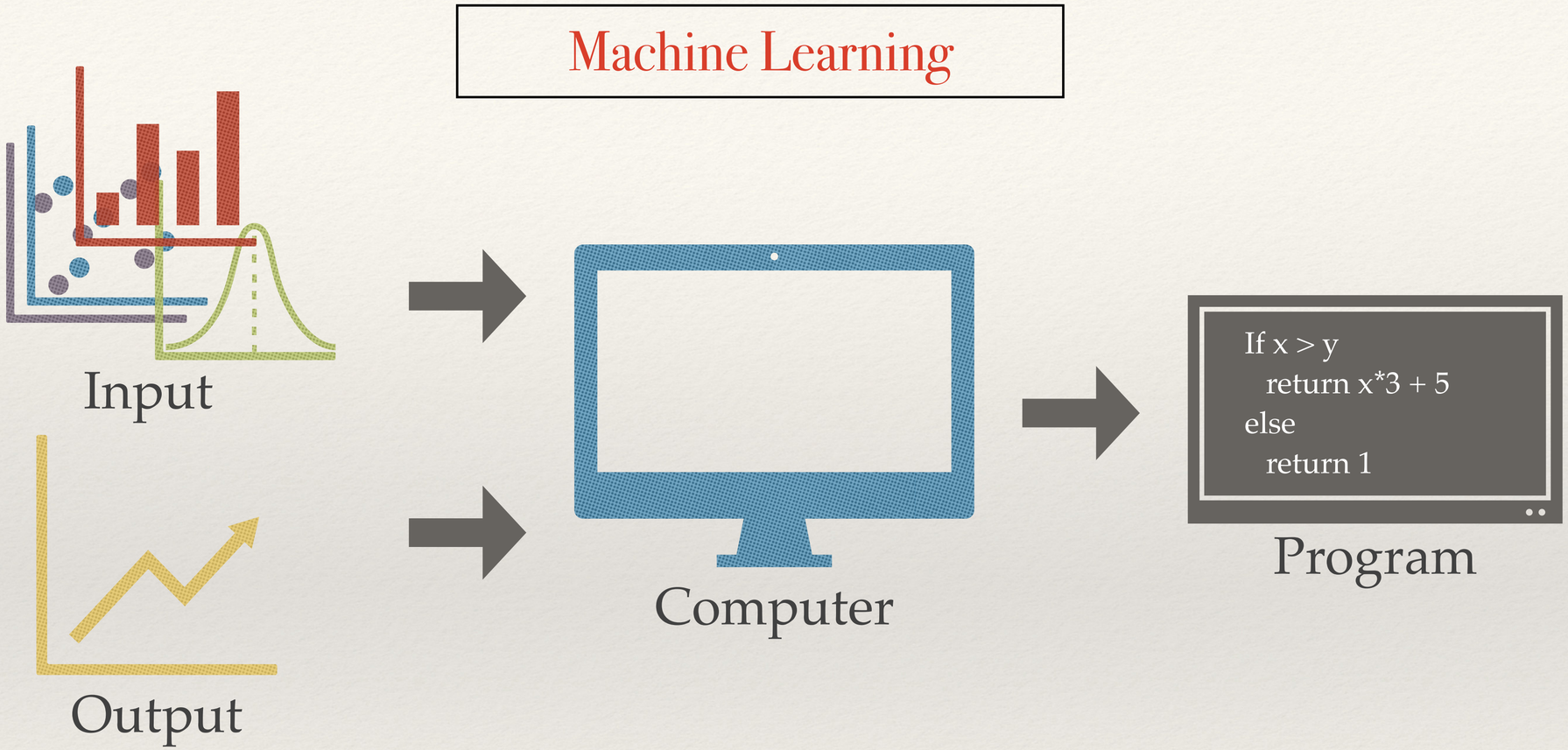
Program



Computer

Output

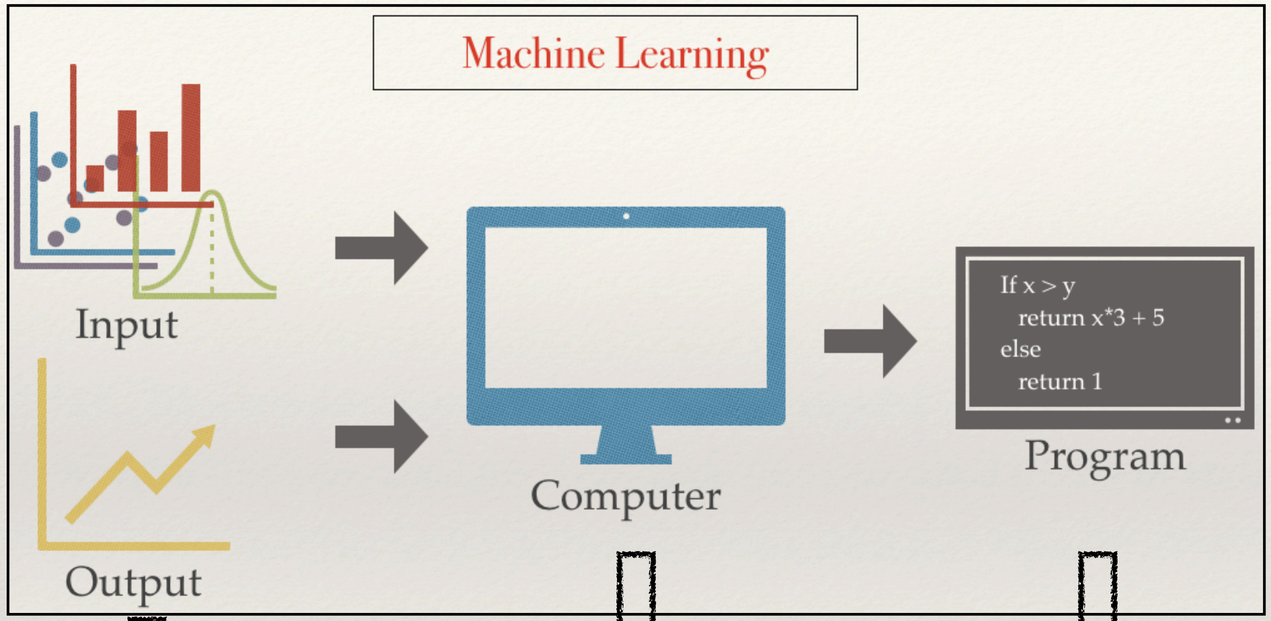
What is Machine Learning?

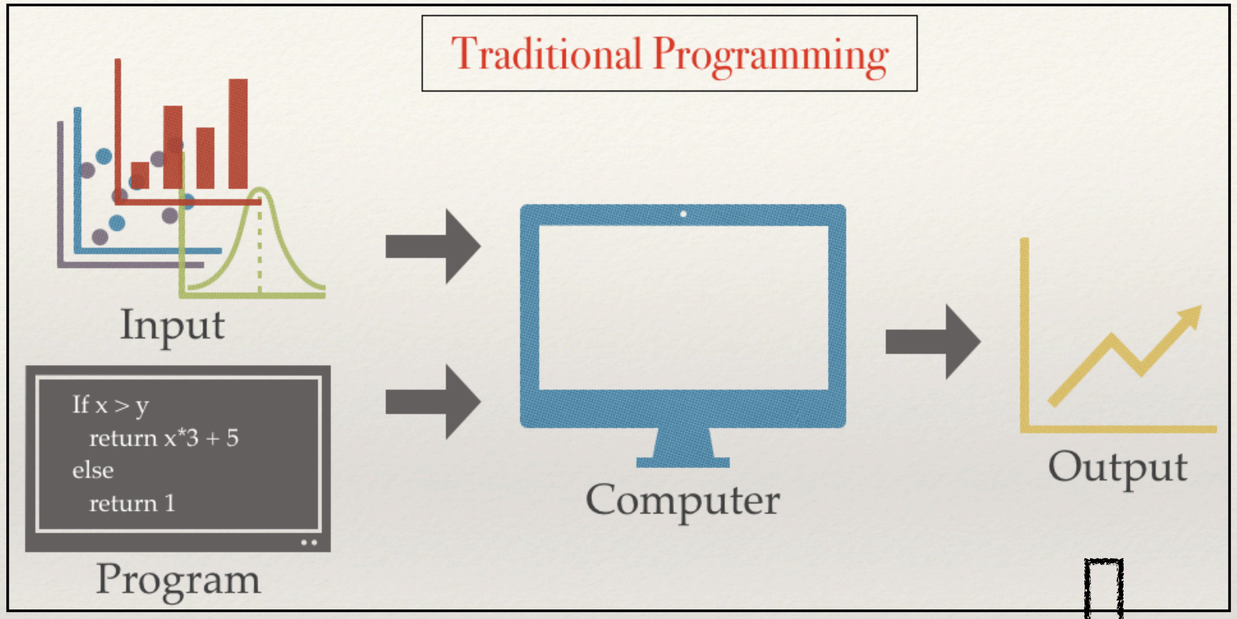


Machine learning is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed.

https://en.wikipedia.org/wiki/Machine_learning#cite_note-1

We're already doing machine learning all the time!





Provide Genotype And Phenotype

Solve the model, find marker effects

Marker effects

Apply marker effects on new population

GEBVs

Terminology (Supervised Learning Problems)

$$Y = X\beta + \epsilon$$

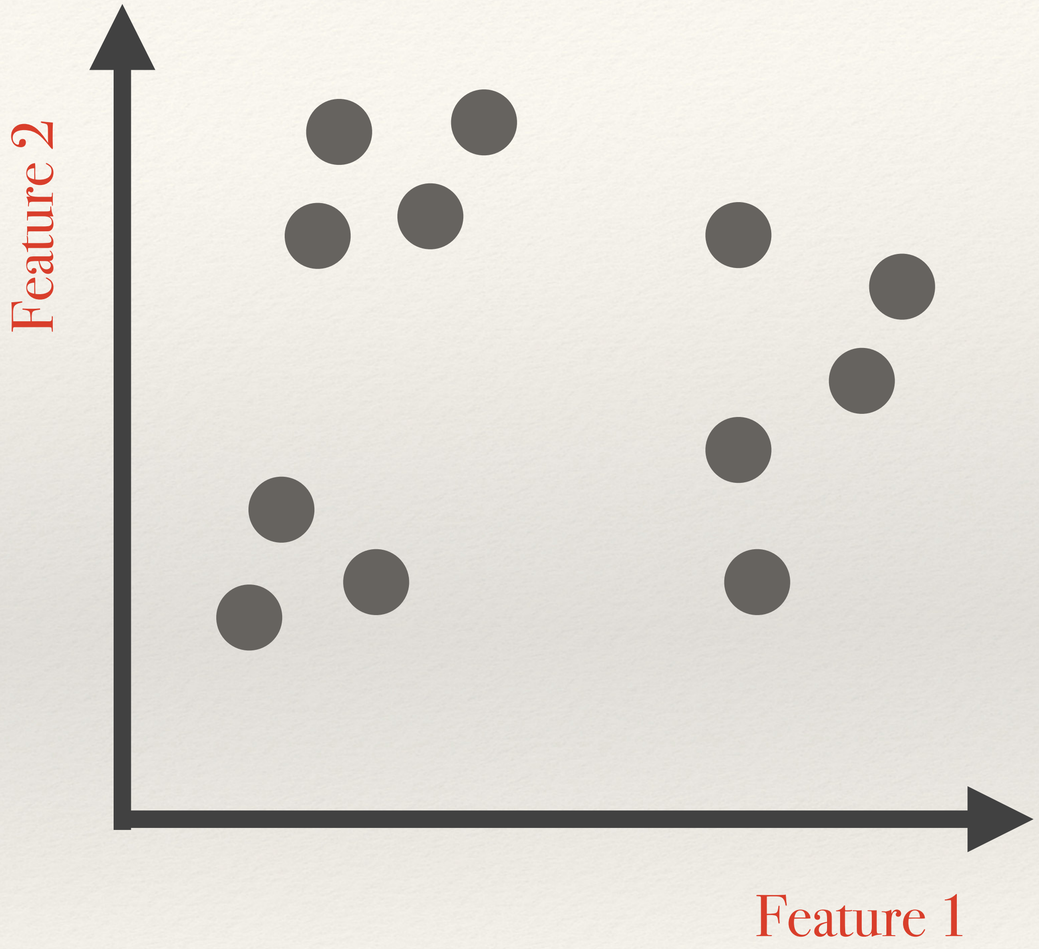
Statistical Model	Machine Learning	Functional Genomic Studies
Response Variables (Y)	Label	Phenotypic value (Observation)
Independent Variable (X)	Predictor, Feature, Attribute (vector)	Genotype for markers
Coefficient (β)	Weight, Solution, Training Results	Marker effects
Error, Residual (ϵ)	Error	Error, Residual

Paradigms of Machine Learning

Paradigms	Key Properties
Supervised Learning	Given **labeled** data, predict labels by the predictors (Classification, Regression)
Unsupervised Learning	Given **unlabeled** data, predict labels by the predictors (Clustering)
Reinforcement Learning	Software agent learns what action to take given the environment (Strategy development)

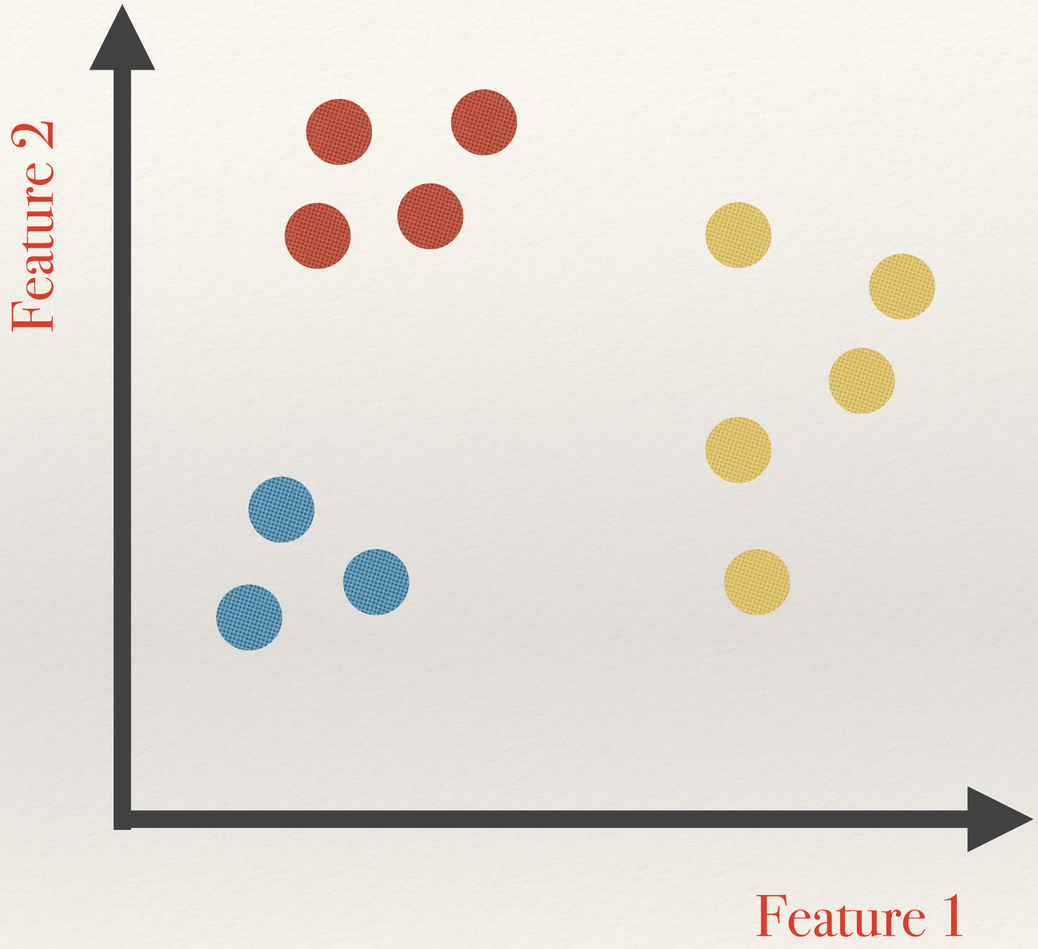
Unsupervised Learning

Feature 1	Feature 2
39.4	51.1
96.4	16.3
19.3	9.8
54.6	32.6
2.5	85.4
94.7	46.1
90.1	96.2
3.9	65.3



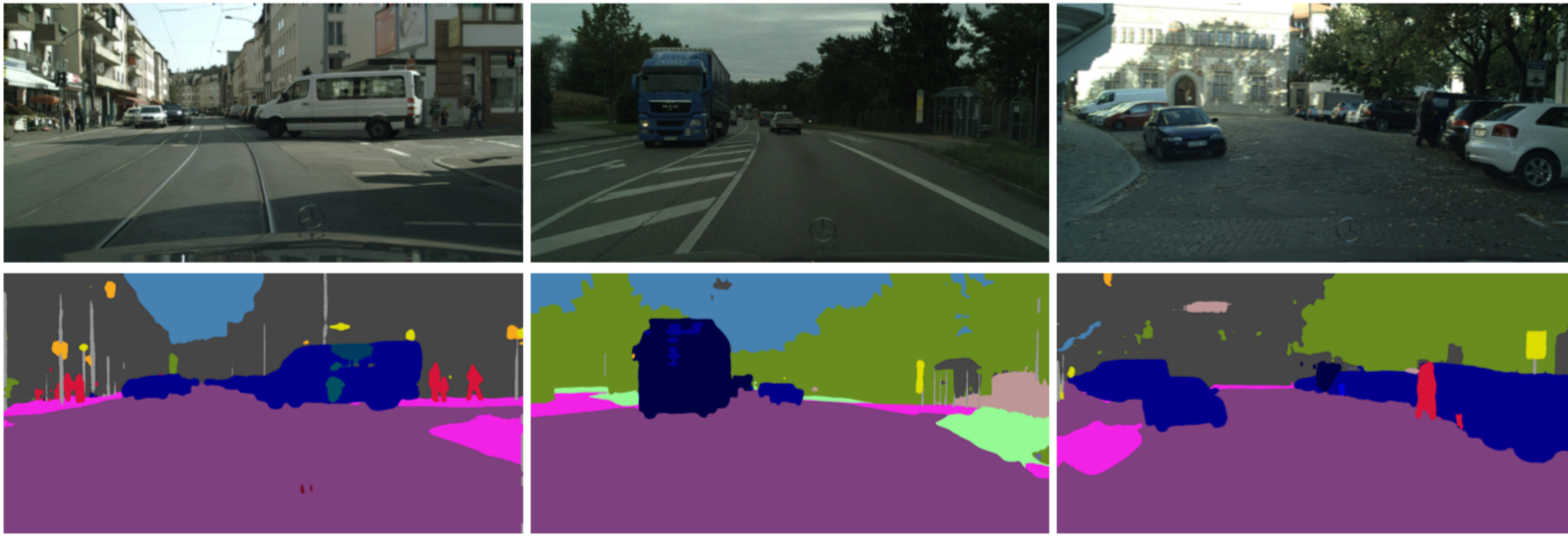
Unsupervised Learning

Feature 1	Feature 2	Cluster
39.4	51.1	A
96.4	16.3	B
19.3	9.8	B
54.6	32.6	C
2.5	85.4	A
94.7	46.1	B
90.1	96.2	B
3.9	65.3	C



Clustering algorithm : K-mean algorithm, KNN algorithm

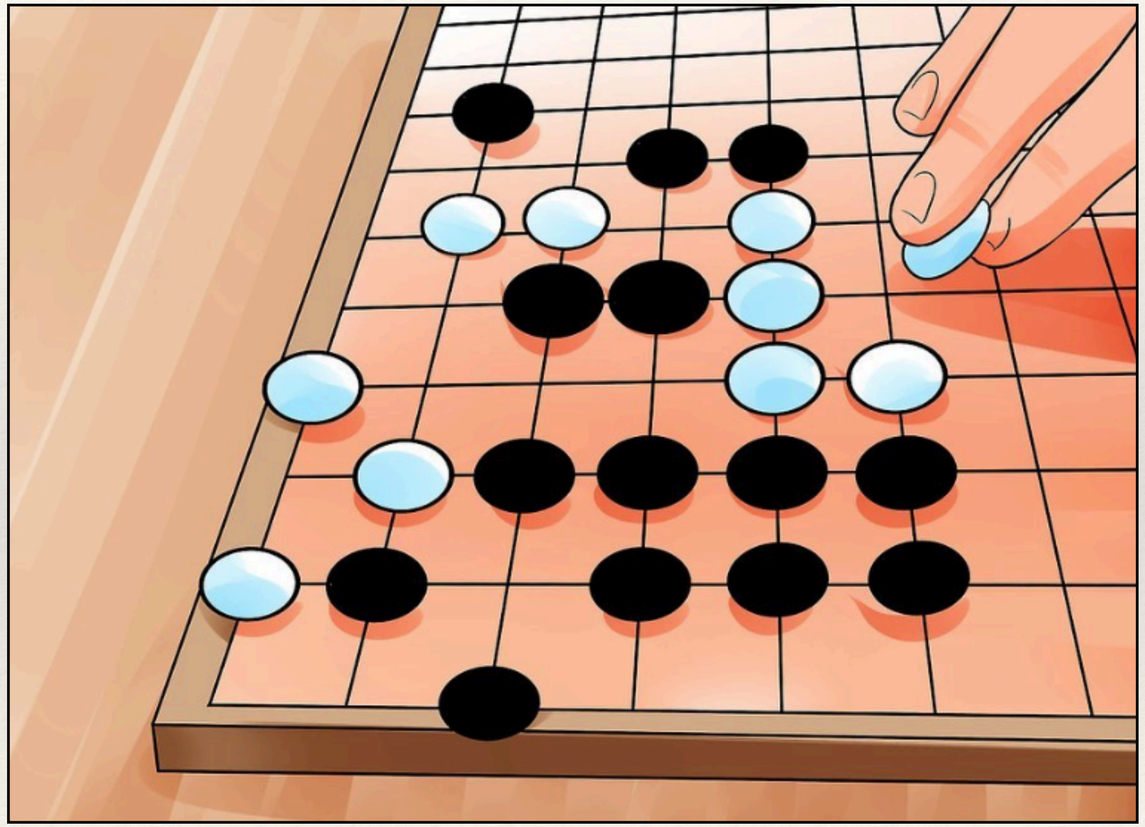
Unsupervised Learning (example)



Feature space optimization for semantic video segmentation (Kundu et al., 2016)

Reinforcement Learning

How do you define these problems?

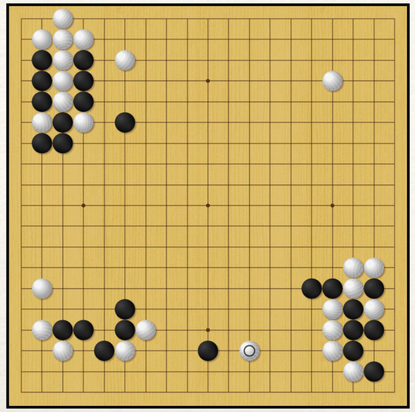


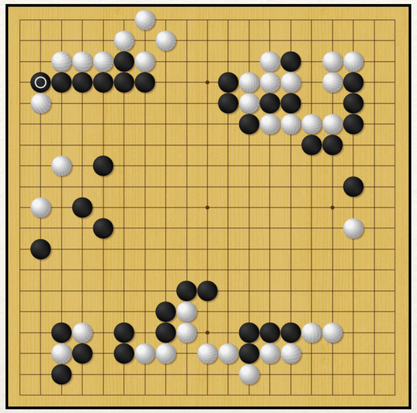
https://www.wikihow.com/Play-Go#/Image:Play-Go-Step-11-Version-2.jpg

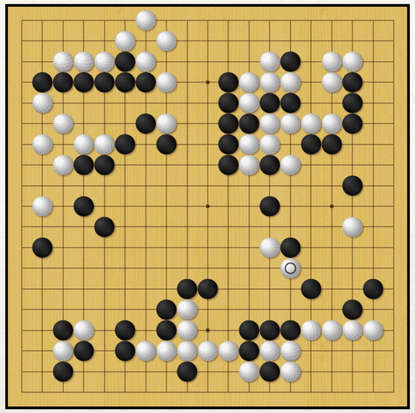
https://www.technobuffalo.com/wp-content/uploads/2016/02/chess.jpg

Reinforcement Learning (MDP : Markov Decision Process)

State







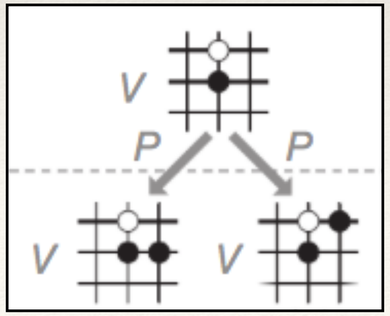
Action

Where to put the stone

Reward

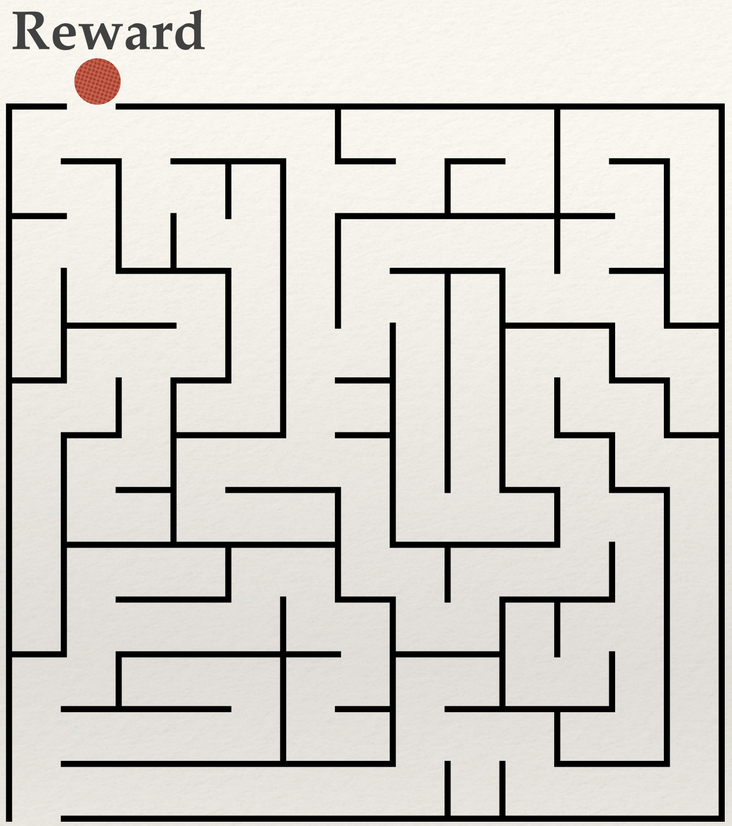
How likely will computer agent win the game given the state

Transition Function



Mastering the game of Go without human knowledge. (Silver et al., 2017)

Reinforcement Learning (MDP : Markov Decision Process)

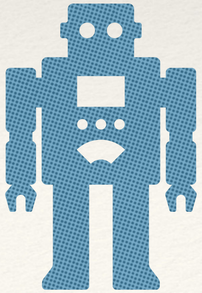


State — Position in the maze : (x, y)

Action — Move up, down, right or left

Reward — Probability of gaining reward given the state

Transition Function — $P(s' \mid s, a)$



Agent

$$V^k(s) = R(s) + \max_a \sum_{s'} T(s,a,s') \cdot V^{k-1}(s')$$

Basic Form of Supervised Learning

$$\mathcal{L} = l(\hat{y}, y) + \Omega(f(X))$$

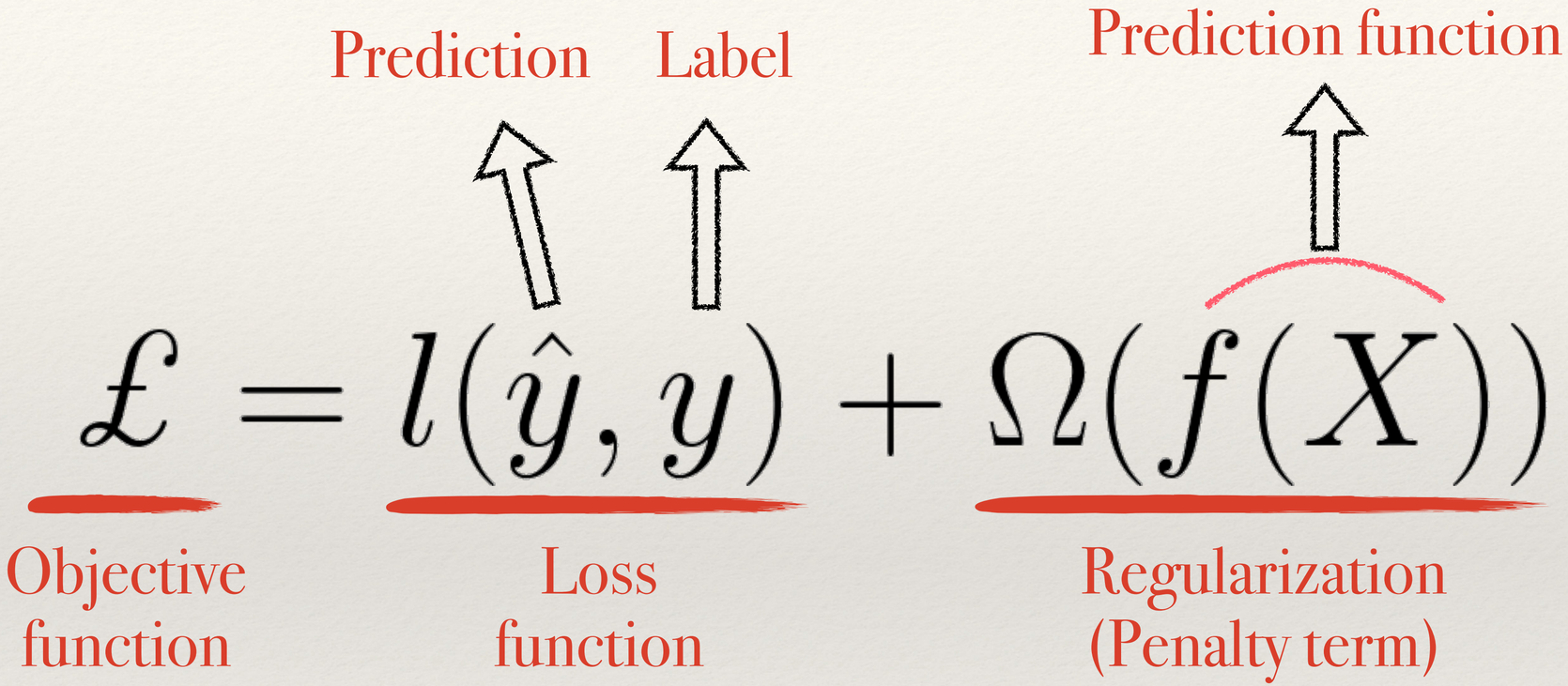
Objective function

Loss function

Regularization (Penalty term)

There're two functions need to be minimized

Basic Form of Supervised Learning



There're two functions to be minimized

Basic Form of Supervised Learning : Loss Function

Loss Function

$$l(\hat{y}, y)$$

$$y = X\beta + \varepsilon$$

$$\Downarrow$$

$$\varepsilon = y - X\beta$$

$$\Downarrow$$

$$\mathcal{L}(\beta) = \min_{\beta}(y - X\beta)^2$$

square error

Basic Form of Supervised Learning : Loss Function

$$\mathcal{L}(\beta) = \min_{\beta} \frac{(y - X\beta)^2}{n}$$

Mean Square Error (MSE)

$$\mathcal{L}(\beta) = \min_{\beta} \left| \frac{y - X\beta}{n} \right|$$

Mean Absolute Error (MAE)

$$L^p(f; \mathbf{x}, \mathcal{L}) = \sum_{s=1}^{n-1} \sum_{i=1, l(i)<l(s)}^{n} \phi\big(f(x_s) - f(x_i)\big)$$

Pairwise rank learning

Ranking Measures and Loss Functions in Learning to Rank (Chen et al.)

Basic Form of Supervised Learning : Regularization (Shrinkage)

$$\mathcal{L} = l(\hat{y}, y) + \Omega(f(X))$$

A way to limit the model complexity

https://www.merriam-webster.com/assets/mw/images/article/art-wap-article-main/shrink-3623-bda0784d7e14101f173c4121b21b3eb1@1x.jpg

Basic Form of Supervised Learning : Regularization (Shrinkage)

$$\mathcal{L} = l(\hat{y}, y) + \Omega(f(X))$$

A way to limit the model complexity

$$\Omega(f) = \frac{1}{2}\lambda \|\beta\|^2 + \alpha \|\beta\|$$

L2 term L1 term

L1 regularization term :

Reduce the **number** of parameters

L2 regularization term :

Reduce the **length** of parameters

Most common way is to shrink the solution

https://www.merriam-webster.com/assets/mw/images/article/art-wap-article-main/shrink-3623-bda0784d7e14101f173c4121b21b3eb1@1x.jpg

Large size of solution

β

3.42

99.3

0.0043

Large size of solution

β

$\frac{1}{2}\lambda \|\beta\|^2$

L2 term

3.42	3.12
99.3	6.8
0.0043	0.0027

https://onlinecourses.science.psu.edu/stat857/node/158

Large size of solution

β	$\frac{1}{2}\lambda \|\beta\|^2$ L2 term		$\alpha \|\beta\|$ L1 term	
3.42		3.12		3.12
99.3	→	6.8	→	6.8
0.0043		0.0027		0

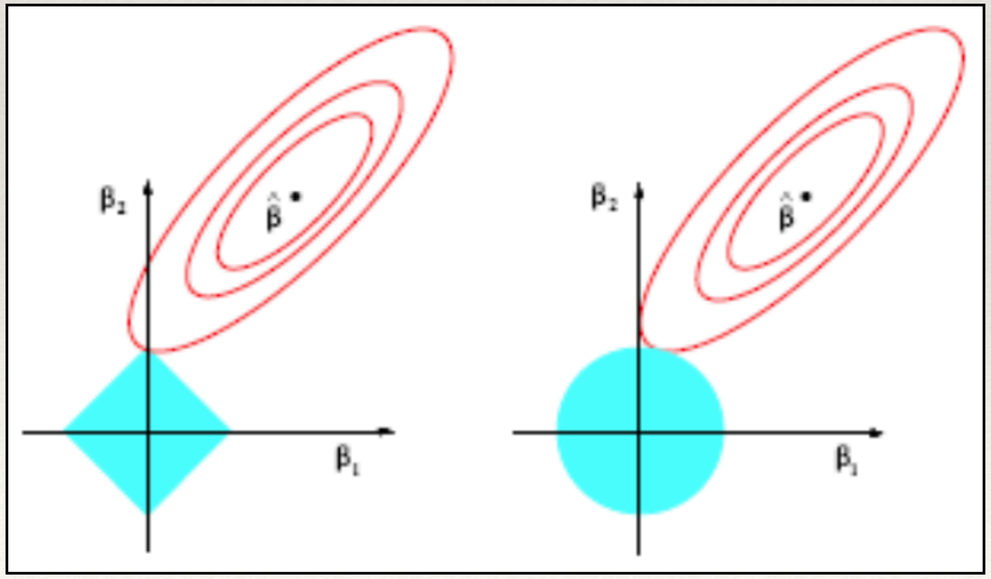
https://onlinecourses.science.psu.edu/stat857/node/158

Large size of solution

β

$\frac{1}{2}\lambda \|\beta\|^2$ (L2 term)	$\alpha \|\beta\|$ (L1 term)
3.42	3.12
99.3	6.8
0.0043	0.0027

$\alpha \|\beta\|$ (L1 term)
3.12
6.8
0



Red ellipses : **Error function**

Blue solid shape : **Regularization**

https://onlinecourses.science.psu.edu/stat857/node/158

Large size of solution

Ridge Regression

$$\sum_{i=1}^{n}(y_i-\hat{y}_i)^2+\lambda\sum_{j}^{m}\beta_j^2$$

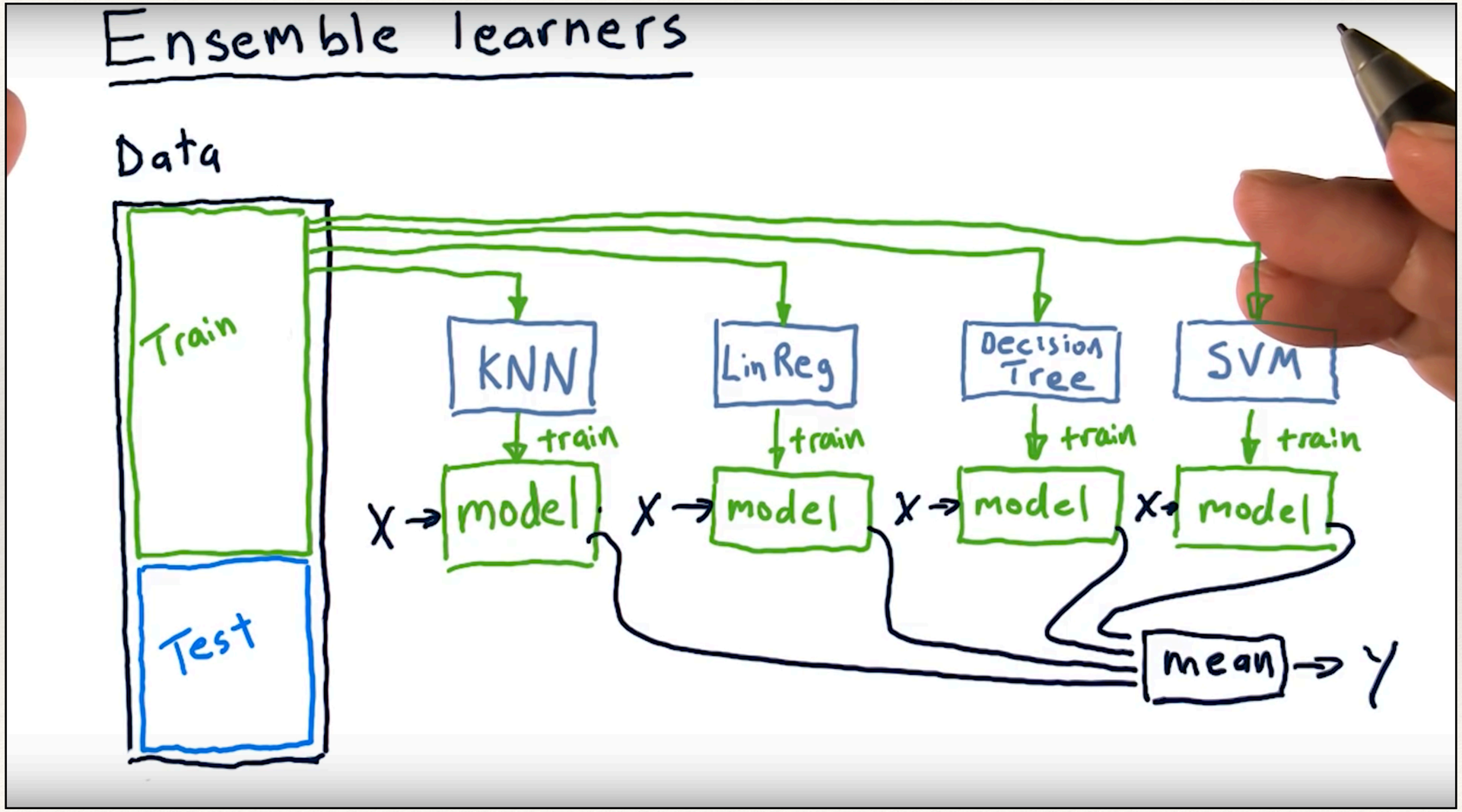
LASSO Regression

$$\hat{\beta}^{\text{lasso}}=\arg\min_{\beta}\sum_{n=1}^{N}\frac{1}{2}(y_n-\beta x_n)^2+\lambda\sum_{i=1}^{p}|\beta_i|$$

http://www.thefactmachine.com/ridge-regression/

https://www.datasciencecentral.com/profiles/blogs/intuition-behind-bias-variance-trade-off-lasso-and-ridge

Ensemble methods

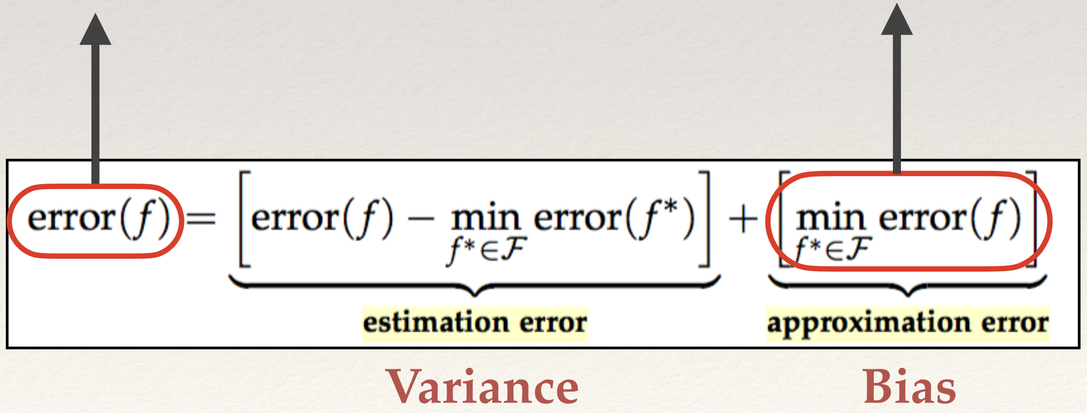


https://www.youtube.com/watch?v=Un9zObFjBH0

Why Ensemble? (Bias-Variance Tradeoff)

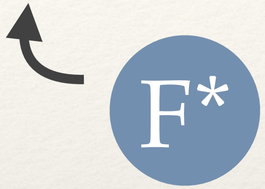
Deviation between the **observation** and the **prediction**

Deviation between the **samples** and the **prediction**



Why Ensemble? (Bias-Variance Tradeoff)

Prediction function from the unlimited sampling space (No sampling error)



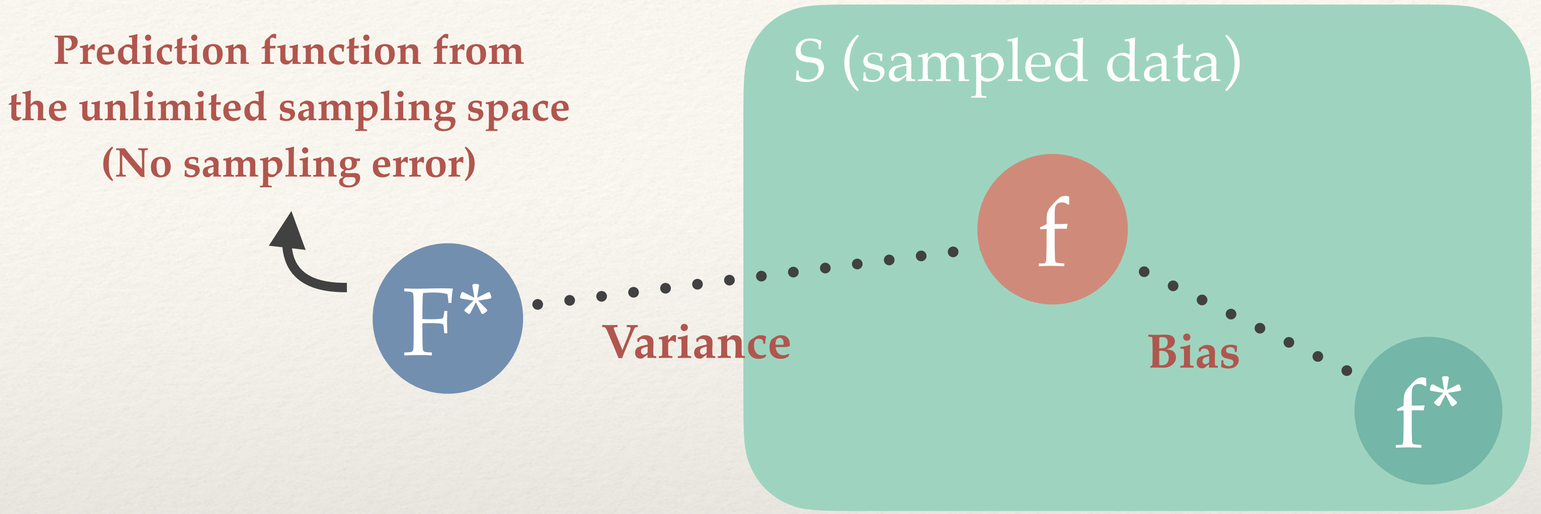
Deviation between the **observation** and the **prediction**

Deviation between the **samples** and the **prediction**

$$\text{error}(f) = \underbrace{\left[\text{error}(f) - \min_{f^* \in \mathcal{F}} \text{error}(f^*)\right]}_{\text{estimation error}} + \underbrace{\left[\min_{f^* \in \mathcal{F}} \text{error}(f)\right]}_{\text{approximation error}}$$

Variance **Bias**

Why Ensemble? (Bias-Variance Tradeoff)



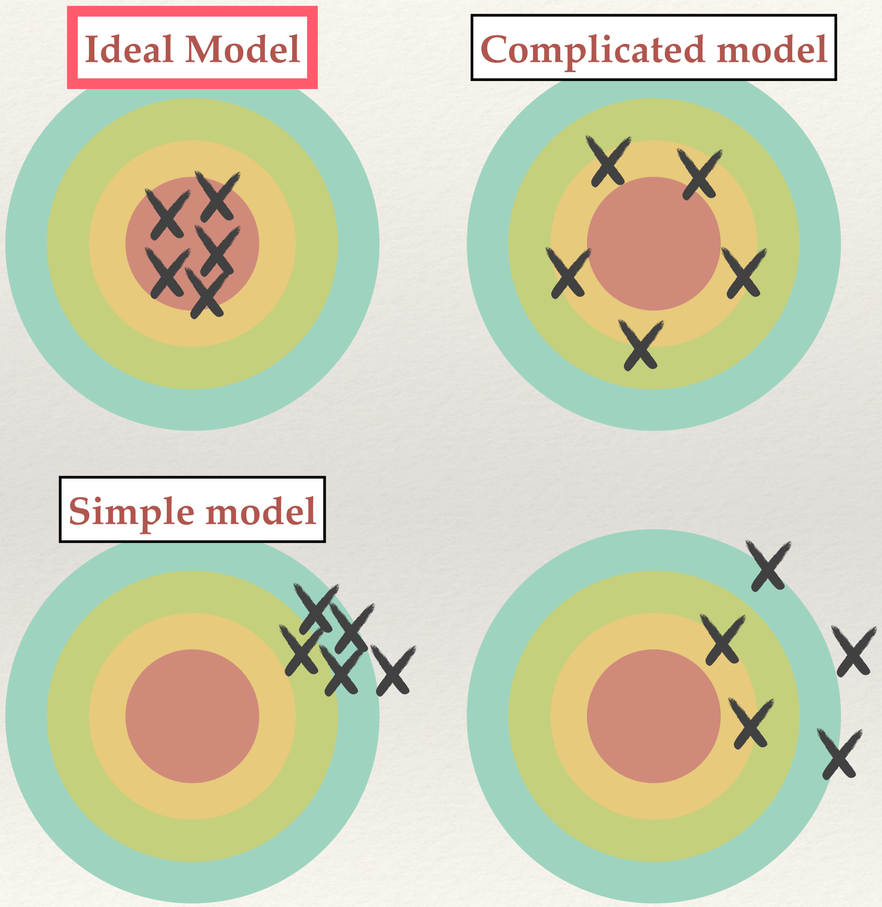
Deviation between
the **observation** and
the **prediction**

Deviation between
the **samples** and
the **prediction**

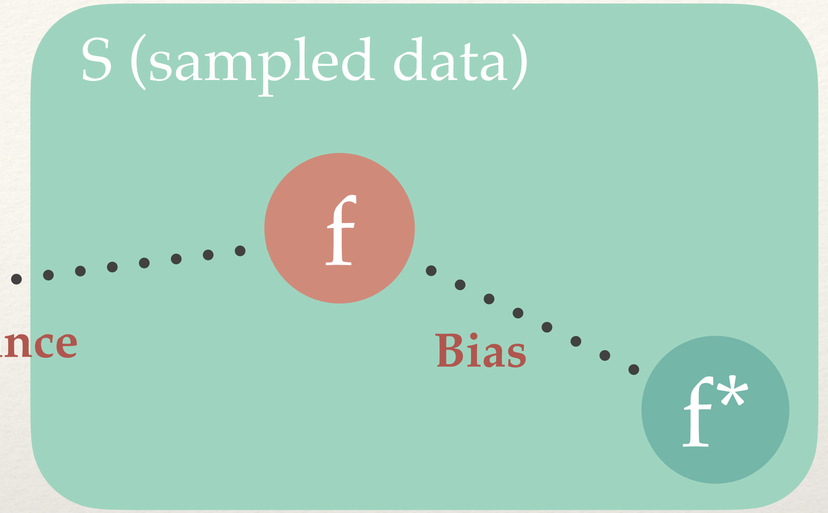
$$\text{error}(f) = \underbrace{\left[\text{error}(f) - \min_{f^* \in \mathcal{F}} \text{error}(f^*)\right]}_{\text{estimation error}} + \underbrace{\left[\min_{f^* \in \mathcal{F}} \text{error}(f)\right]}_{\text{approximation error}}$$

Variance **Bias**

Why Ensemble? (Bias-Variance Tradeoff)



Prediction function from the unlimited sampling space (No sampling error)



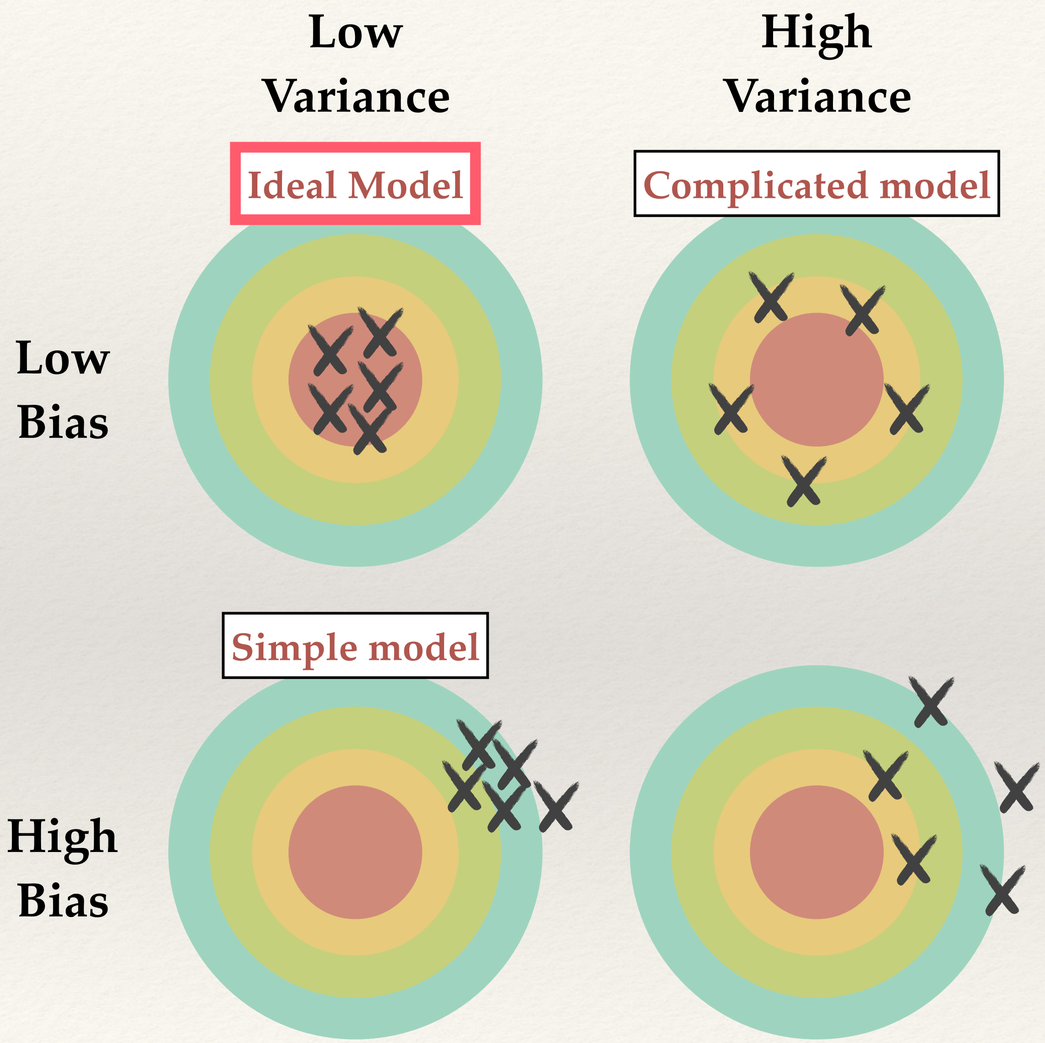
Deviation between the **observation** and the **prediction**

Deviation between the **samples** and the **prediction**

$$\text{error}(f) = \underbrace{\left[\text{error}(f) - \min_{f^* \in \mathcal{F}} \text{error}(f^*)\right]}_{\text{estimation error}} + \underbrace{\left[\min_{f^* \in \mathcal{F}} \text{error}(f)\right]}_{\text{approximation error}}$$

Variance Bias

Why Ensemble? (Bias-Variance Tradeoff)

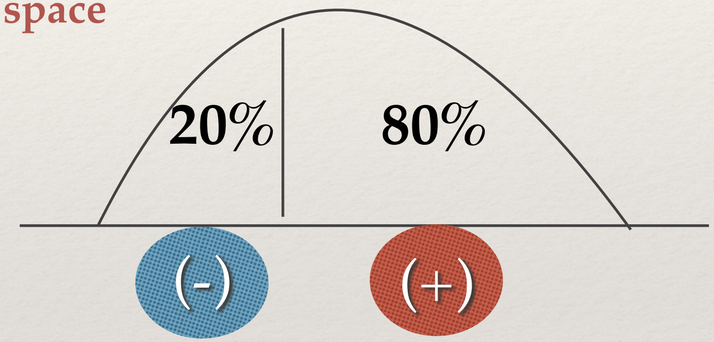


Naive prediction function (Will always make the same prediction)

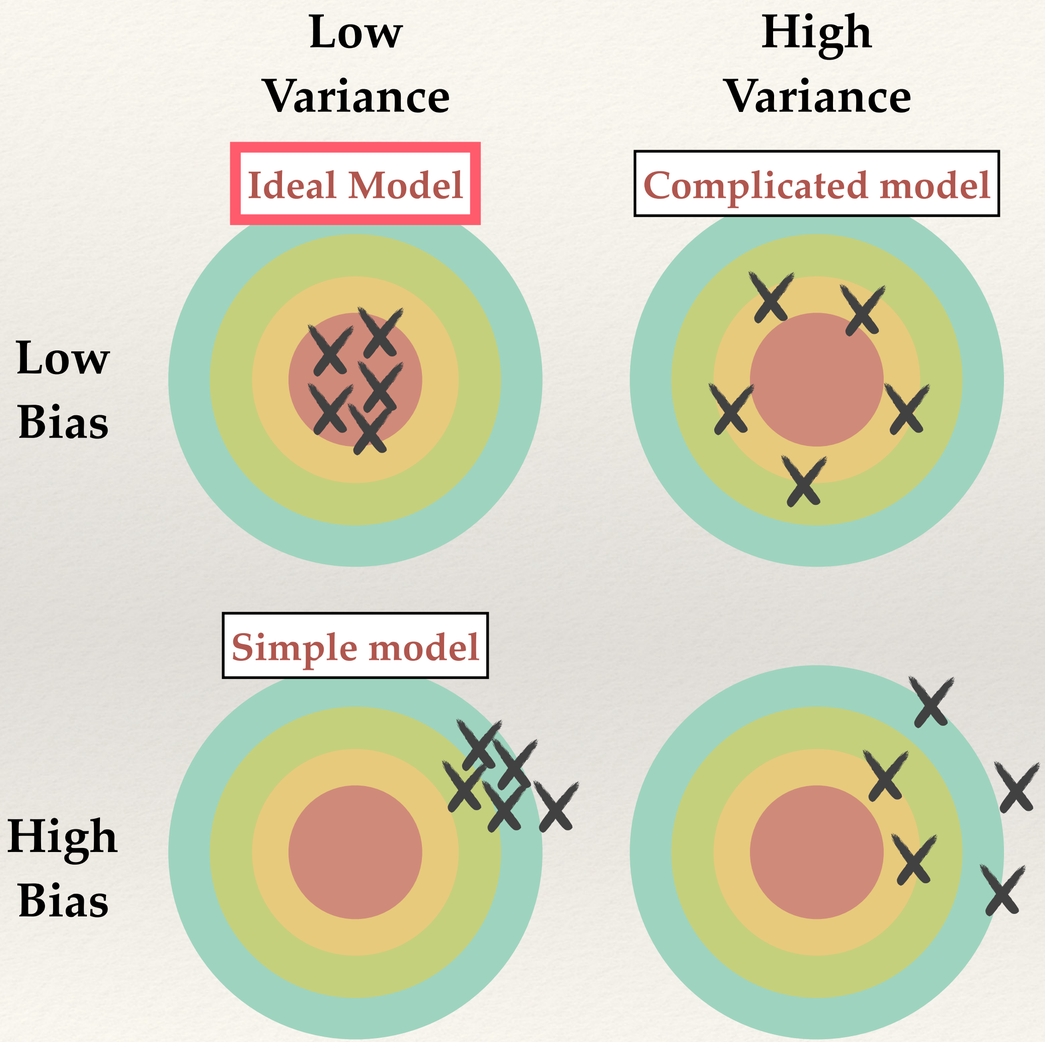
f_1 Always Negative (-)

f_2 Always Positive (+)

Sampling space



Why Ensemble? (Bias-Variance Tradeoff)

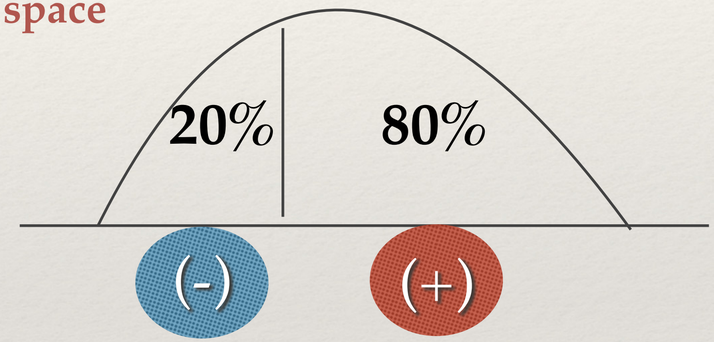


Naive prediction function (Will always make the same prediction)

f_1 Always Negative (-)

f_2 Always Positive (+)

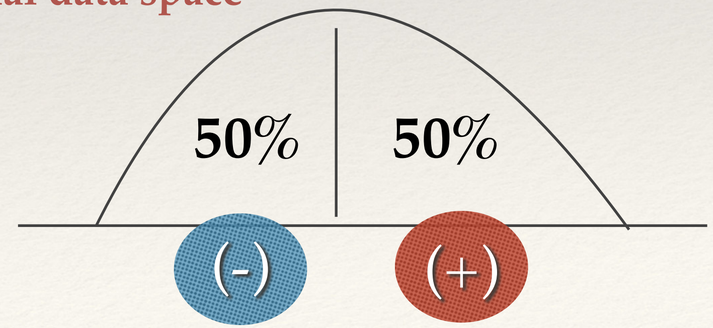
Sampling space

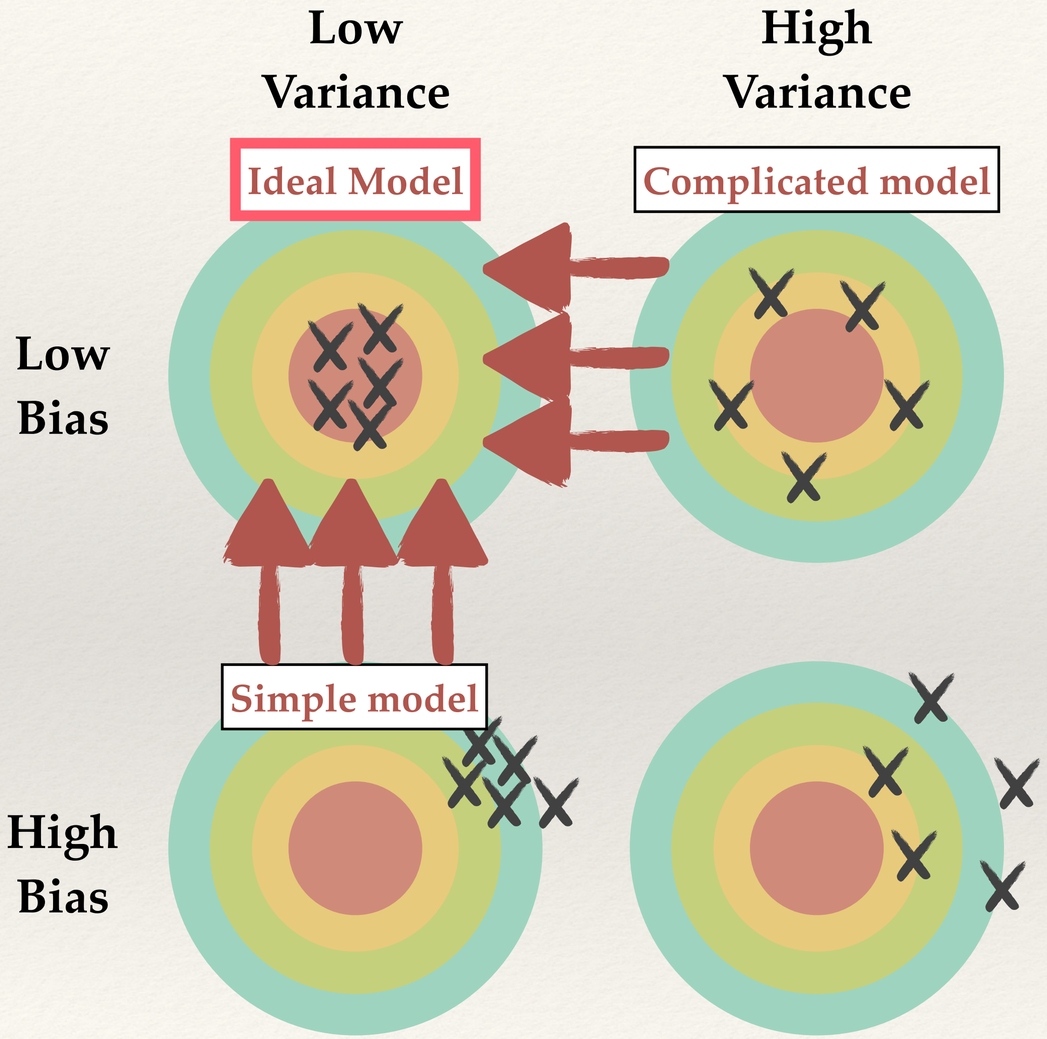


Function learned

f_2 Always Positive (+)

Actual data space





H : Sampled space

h1, h2, h3 : Hypothesis (Prediction function)

f : Ideal prediction

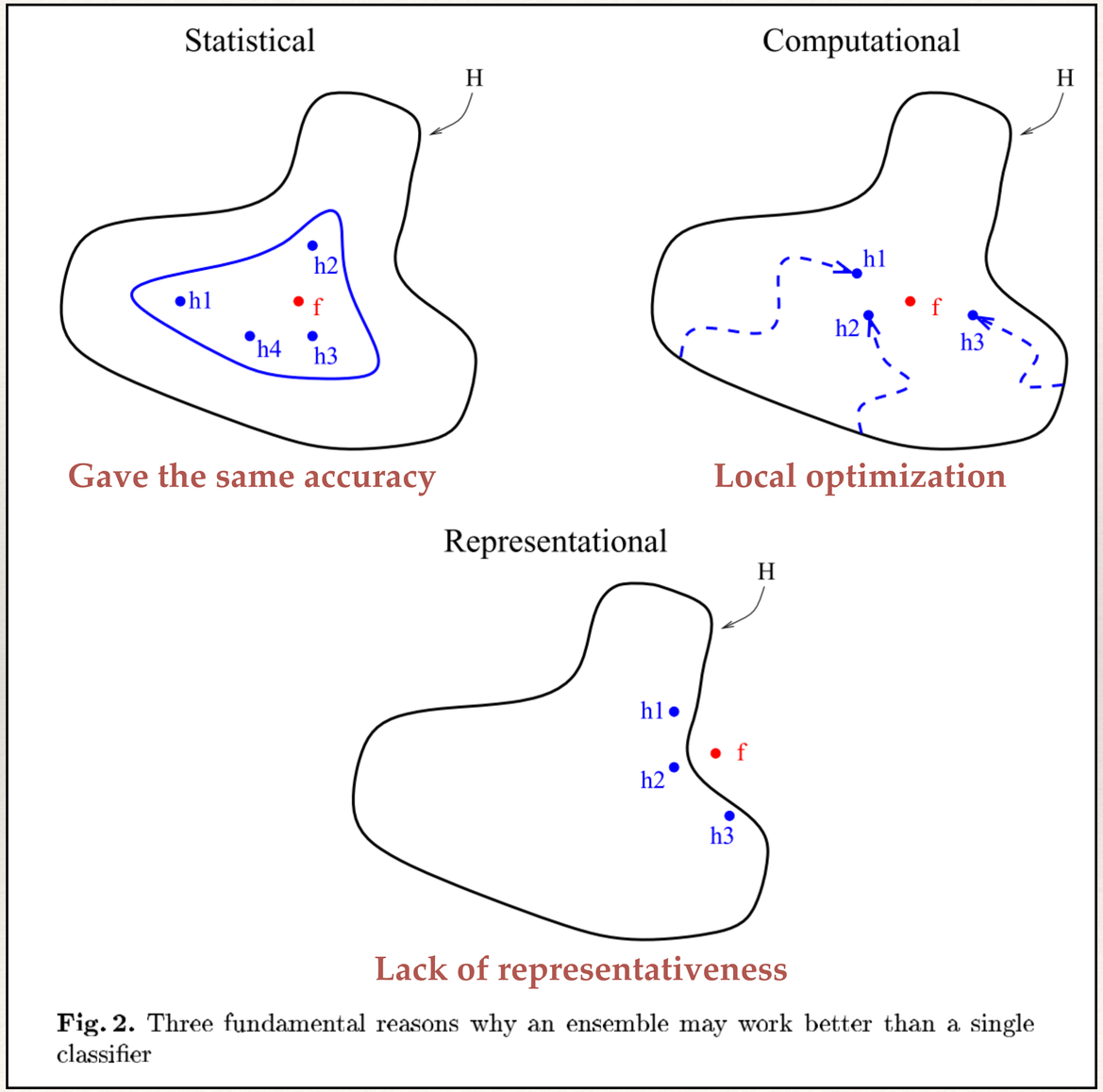


Fig. 2. Three fundamental reasons why an ensemble may work better than a single classifier

Ensemble Methods in Machine Learning (Dietterich, 2000)

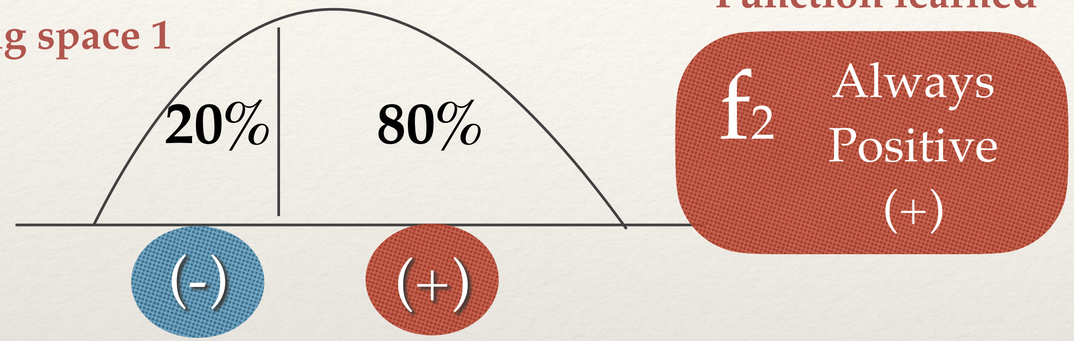
Naive prediction function (Will always make the same prediction)

f_1 Always Negative (-)

f_2 Always Positive (+)

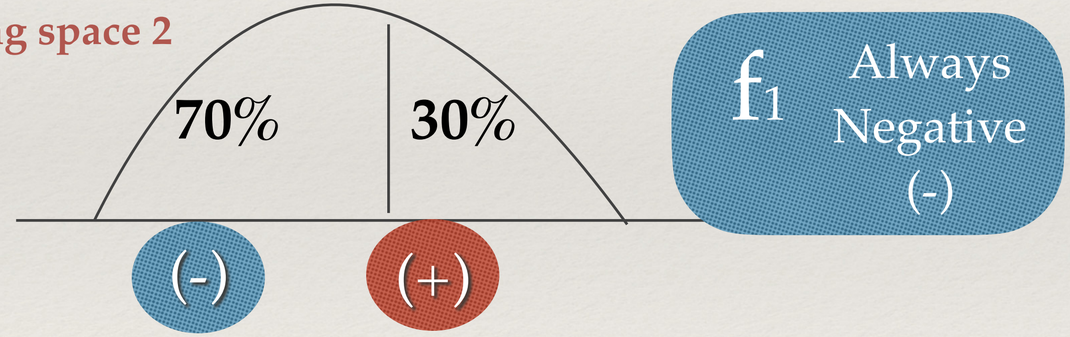
Function learned

Sampling space 1



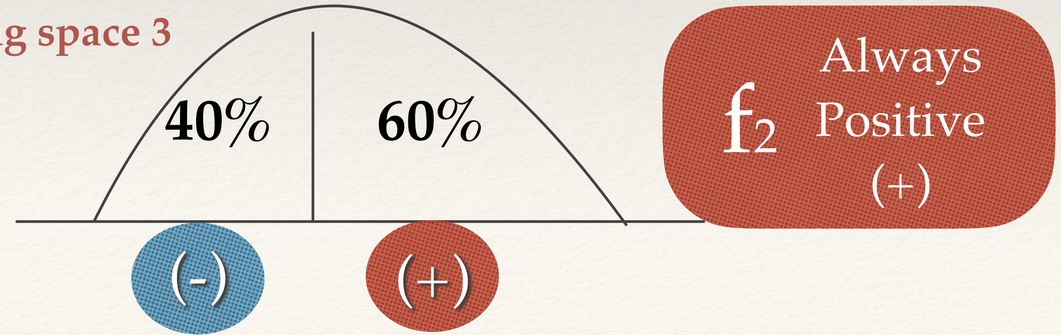
f_2 Always Positive (+)

Sampling space 2



f_1 Always Negative (-)

Sampling space 3



f_2 Always Positive (+)

H : Sampled space

h1, h2, h3 : Hypothesis (Prediction function)

f : Ideal prediction

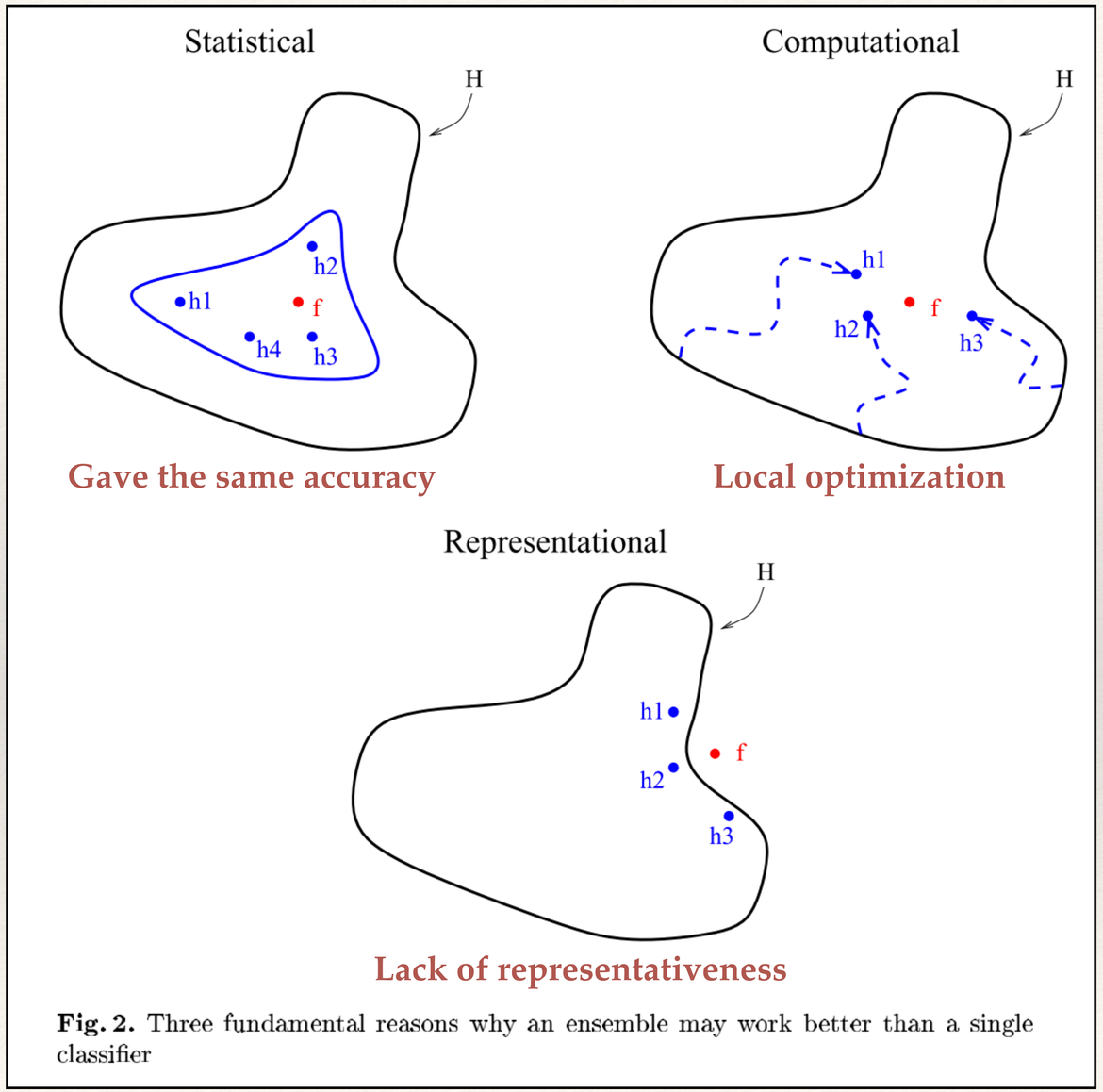


Fig. 2. Three fundamental reasons why an ensemble may work better than a single classifier

Ensemble Methods in Machine Learning (Dietterich, 2000)

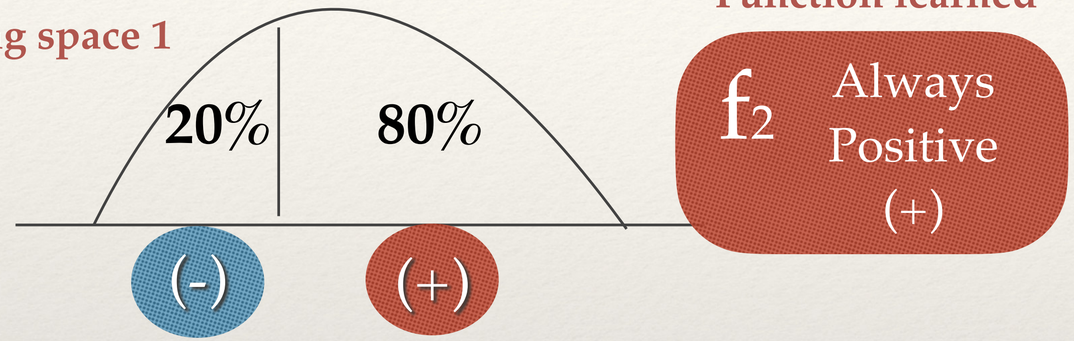
Naive prediction function (Will always make the same prediction)

f_1 Always Negative (-)

f_2 Always Positive (+)

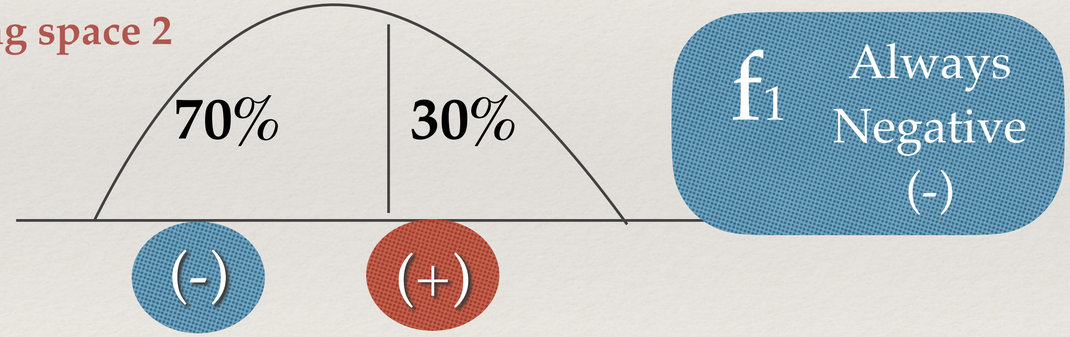
Function learned

Sampling space 1



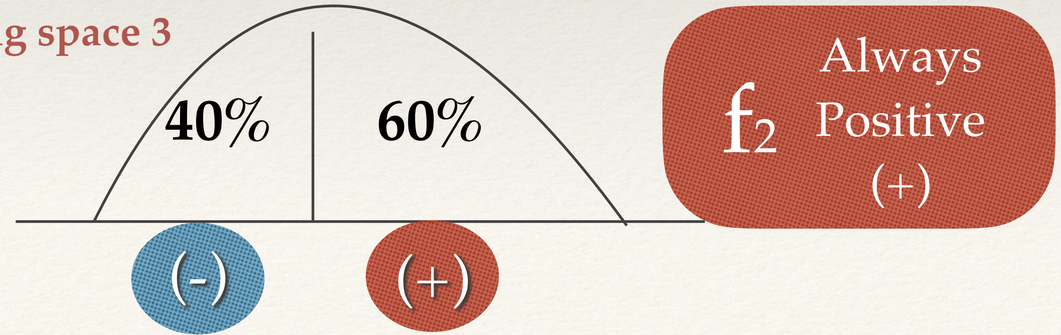
f_2 Always Positive (+)

Sampling space 2



f_1 Always Negative (-)

Sampling space 3



f_2 Always Positive (+)

One single model

f Positive (+) — 100%

Ensemble method

f Positive (+) — 66.6%

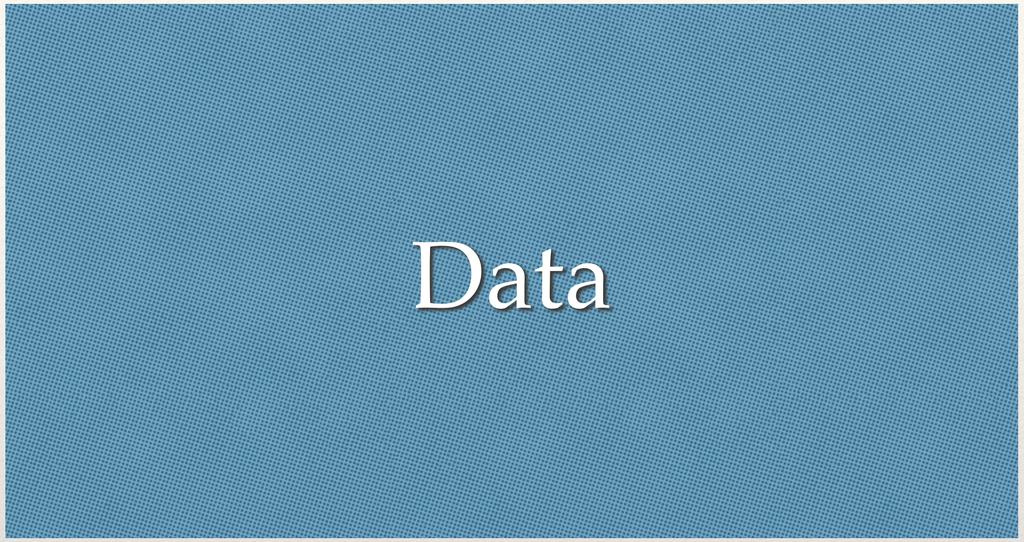
f Negative (-) — 33.3%

More flexible prediction

Randomness in ensemble methods

m : # of markers

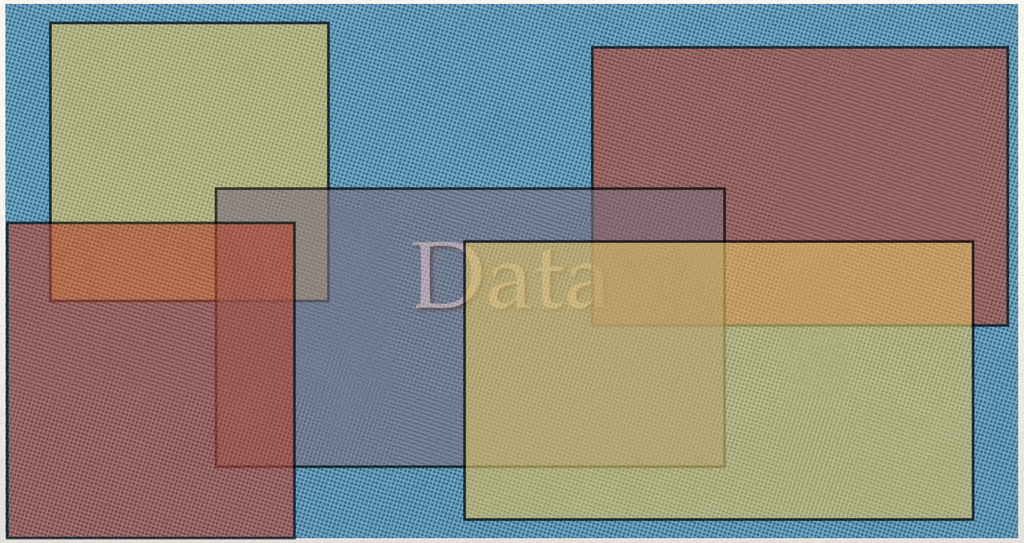
n : sample size



Randomness in ensemble methods

m : # of markers

n : sample size



Couple of prediction functions

$f_1(X)$ $f_2(X)$ $f_3(X)$ $f_4(X)$ $f_5(X)$

Ensemble : Bagging

Prediction Functions

$$f_1(X) \quad f_2(X) \quad f_3(X) \quad f_4(X) \quad \cdots\cdots \quad f_s(X)$$

Make Prediction

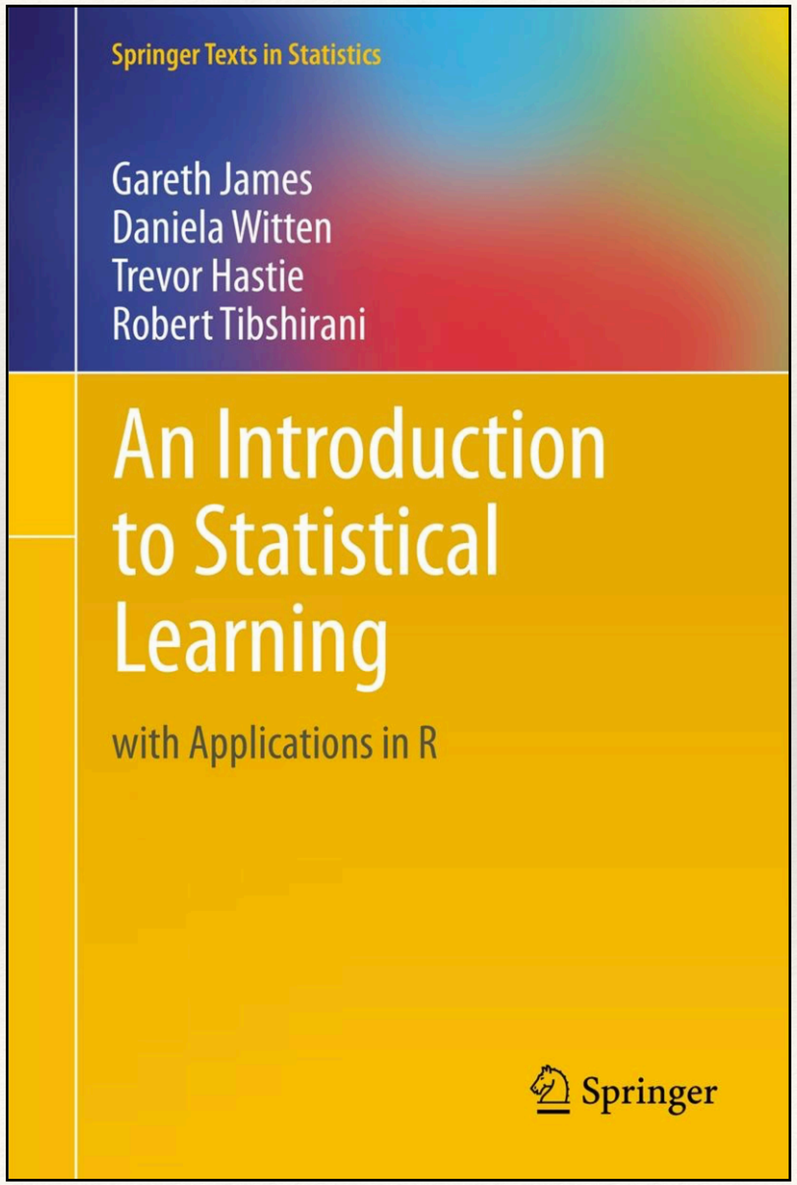
$$\hat{y}_1 \quad \hat{y}_2 \quad \hat{y}_3 \quad \hat{y}_4 \ldots \hat{y}_s$$

Average Results

$$\frac{\sum_{j=1}^{s} \hat{y}_j}{s}$$

Final prediction

Ensemble : Boosting



Algorithm 8.2 *Boosting for Regression Trees*

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.

2. For $b = 1, 2, \ldots, B$, repeat:

   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).

   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:

   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$

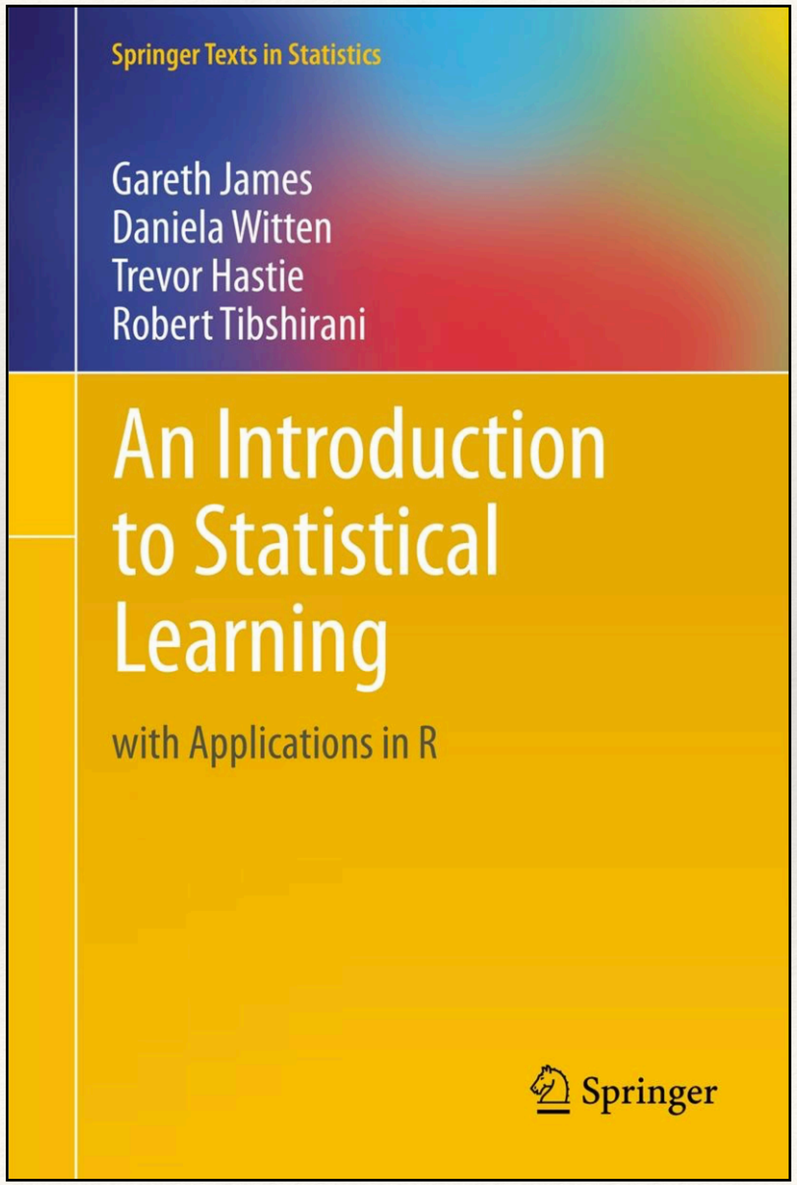
   (c) Update the residuals,

   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

Ensemble : Boosting



Algorithm 8.2 *Boosting for Regression Trees*

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.

2. For $b = 1, 2, \ldots, B$, repeat:

   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).

   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$

   Add a new function (x) at each iteration

   (c) Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$

   Update error

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

Ensemble : Boosting

$$l(\hat{y}_i, y_i)$$

Breakdown
estimated y (y.hat)

New model

$$l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i))$$

Ensemble : Boosting

$$l(\hat{y}_i, y_i)$$

Breakdown estimated y (y.hat)

New model

$$l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i))$$

$$l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i))$$

$$l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i))$$

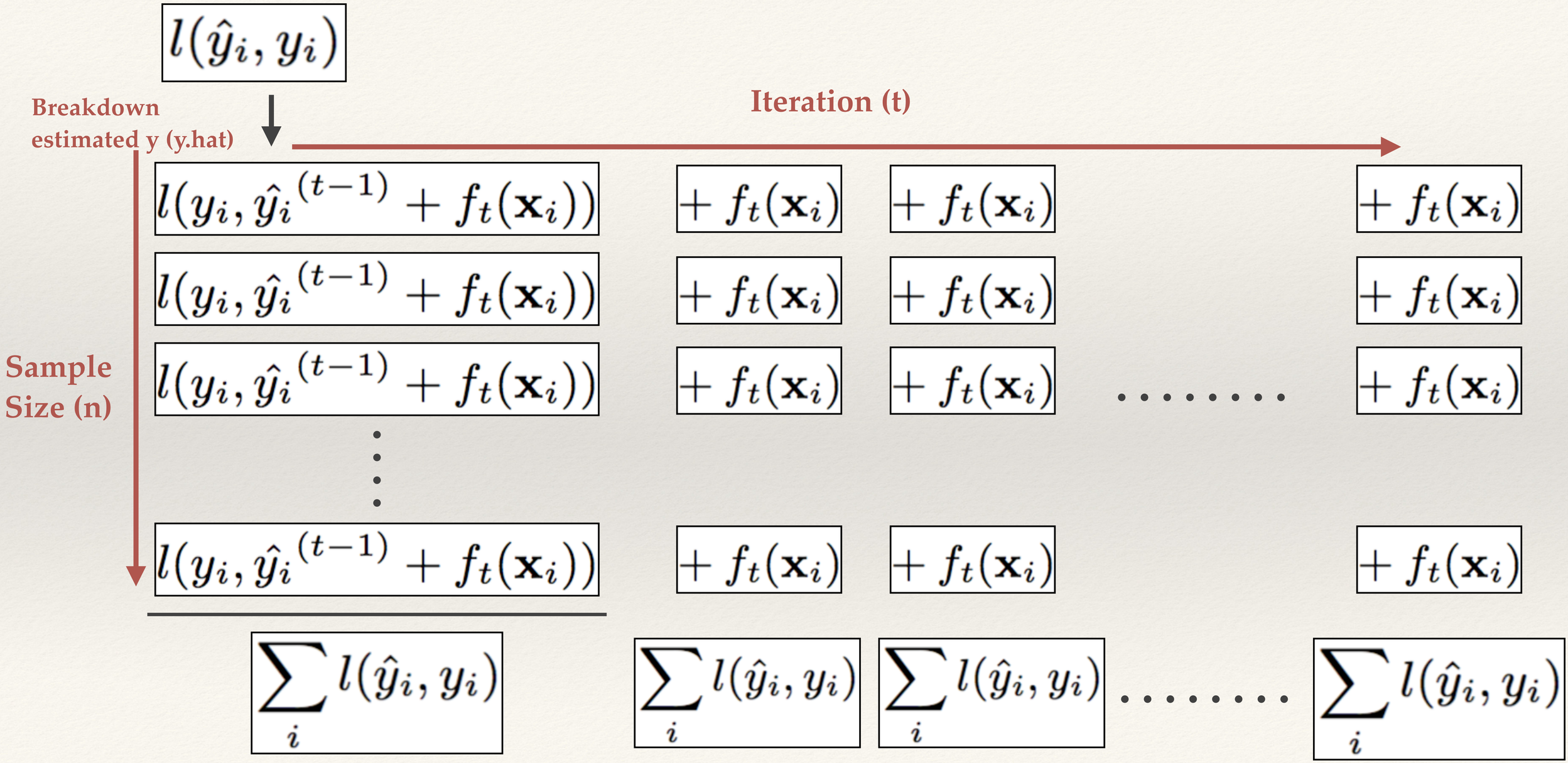
Sample Size (n)

$$\vdots$$

$$l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i))$$

$$\sum_i l(\hat{y}_i, y_i)$$

Ensemble : Boosting



Boosting Linear Model

$$l(y_i, \hat{y_i}^{(t-1)} + f_t(\mathbf{x}_i))$$

$$\rightarrow \; l(y_i, \hat{y}_i + X_1 b_1 + X_2 b_2 + X_3 b_3)$$

N x 3 3 x 1 N x 5 5 x 1 N x 2 2x 1

Boosting Linear Model

$$l(y_i, \hat{y_i}^{(t-1)} + f_t(\mathbf{x}_i))$$

$$\rightarrow l(y_i, \hat{y}_i + X_1 b_1 + X_2 b_2 + X_3 b_3) \rightarrow l(y_i, \hat{y}_i + Xb)$$

N x 3 3 x 1 N x 5 5 x 1 N x 2 2x 1

N x 9 9x 1

Step-wise GLM!!

Boosting Linear Model

$$l(y_i, \hat{y_i}^{(t-1)} + f_t(\mathbf{x}_i))$$

$$\rightarrow\ l(y_i, \hat{y}_i + X_1 b_1 + X_2 b_2 + X_3 b_3) \rightarrow\ l(y_i, \hat{y}_i + Xb)$$

N x 3 3 x 1 N x 5 5 x 1 N x 2 2x 1

N x 9 9x 1

Step-wise GLM!!

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

$$\Omega(f) = \frac{1}{2}\lambda \|w\|^2 + \alpha \|w\|$$

Step-wise Ridge Regression or LASSO

Practical Studies

Guzzetta *et al. BMC Bioinformatics* 2010, **11**(Suppl 8):S3
http://www.biomedcentral.com/1471-2105/11/S8/S3

BMC Bioinformatics

RESEARCH **Open Access**

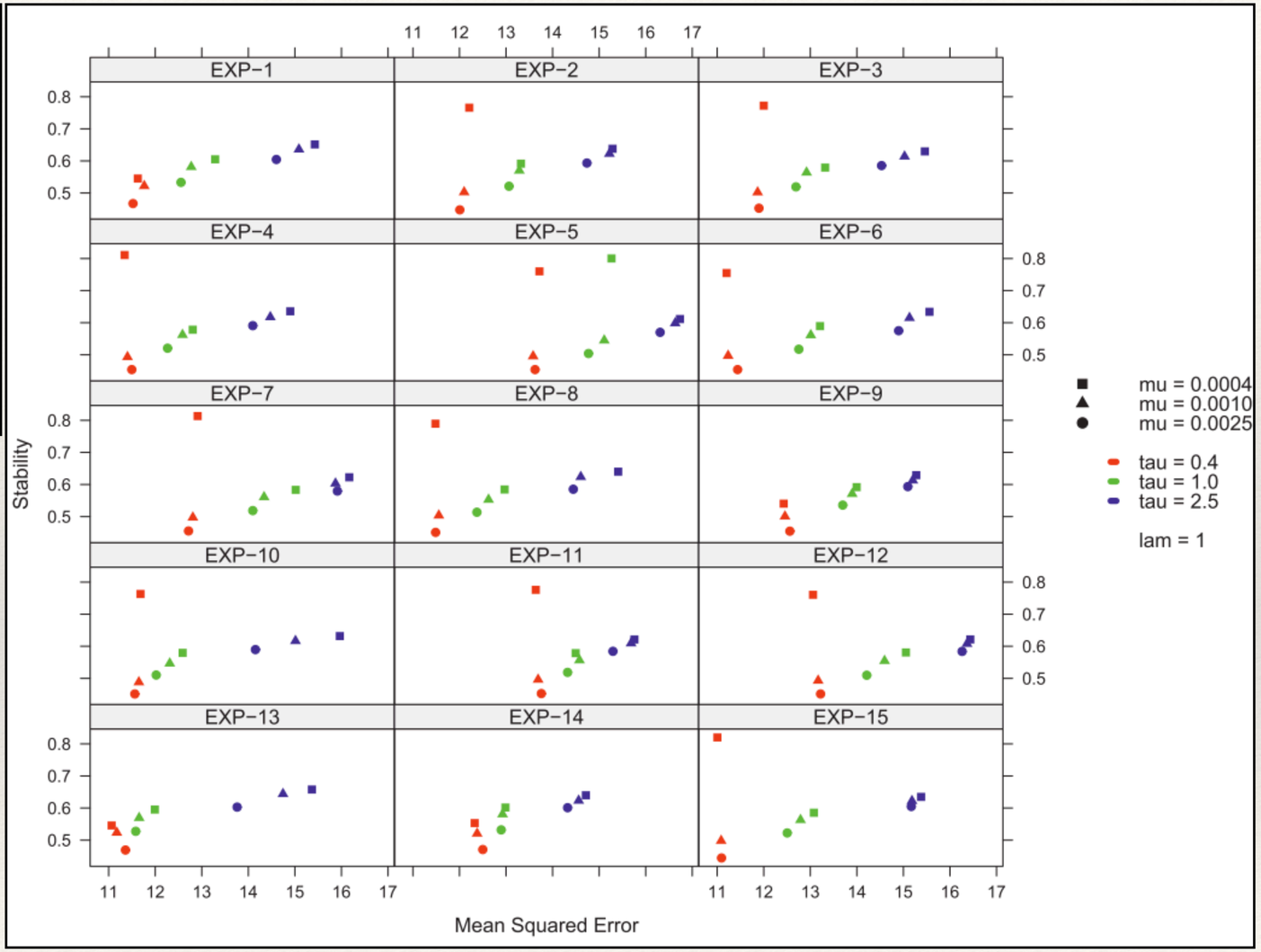
A machine learning pipeline for quantitative phenotype prediction from genotype data

Giorgio Guzzetta[1,2], Giuseppe Jurman[1], Cesare Furlanello[1*]

From Machine Learning in Computational Biology (MLCB) 2009
Whistler, Canada. 10-11 December 2009

$$\mathrm{w}_{\tau,\mu} = \arg\min_{\mathrm{w}} \left(\frac{1}{n} \sum_i (y_i - \mathrm{w} \cdot \mathrm{x}_i)^2 + \tau \|\mathrm{w}\|_1 + \mu \|\mathrm{w}\|_2^2 \right) \quad (1)$$

Find the best combination
To regularize models



Practical Studies

Squared loss L_1 penalty L_2 penalty

$$\sum_{i=1}^{n}(y_i-\hat{y}_i)^2+\lambda_1\sum_{j=1}^{p}|w_j|+\lambda_2\sum_{j=1}^{p}w_j^2. \qquad (2)$$

OPEN ACCESS Freely available online

PLOS | GENETICS

Review

Regularized Machine Learning in the Genetic Prediction of Complex Traits

Sebastian Okser[1,2], Tapio Pahikkala[1,2], Antti Airola[1,2], Tapio Salakoski[1,2], Samuli Ripatti[3,4,5], Tero Aittokallio[2,4]*

1 Department of Information Technology, University of Turku, Turku, Finland, **2** Turku Centre for Computer Science (TUCS), University of Turku and Åbo Akademi University, Turku, Finland, **3** Hjelt Institute, University of Helsinki, Helsinki, Finland, **4** Institute for Molecular Medicine Finland (FIMM), University of Helsinki, Helsinki, Finland, **5** Wellcome Trust Sanger Institute, Hinxton, United Kingdom

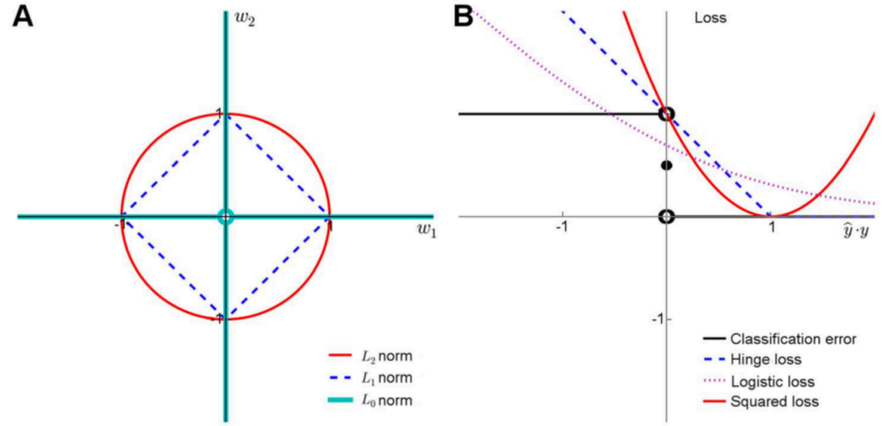
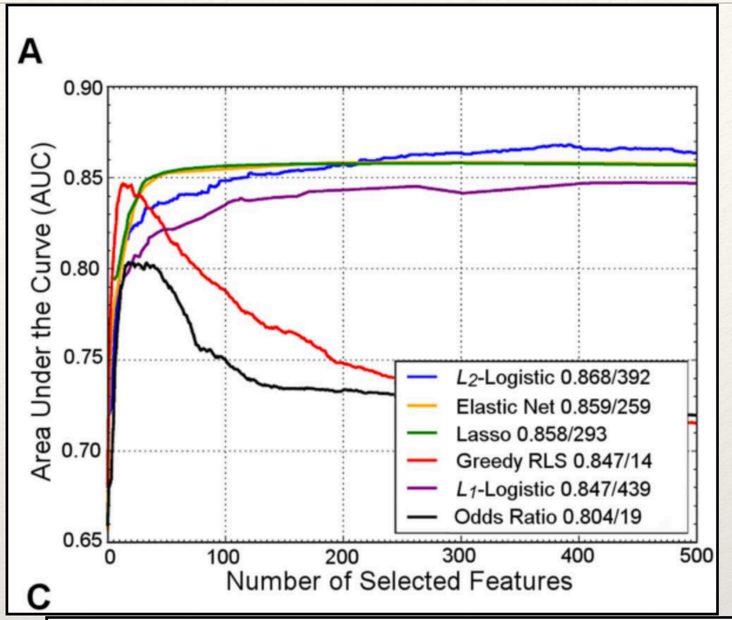
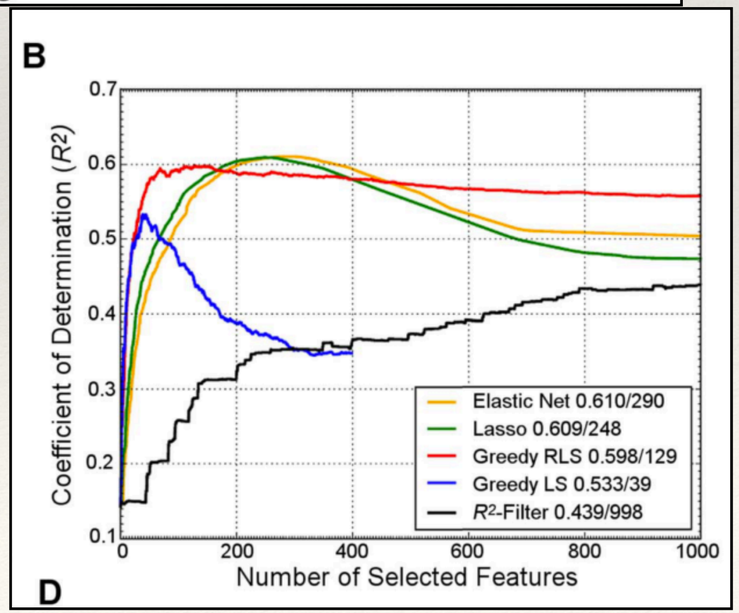
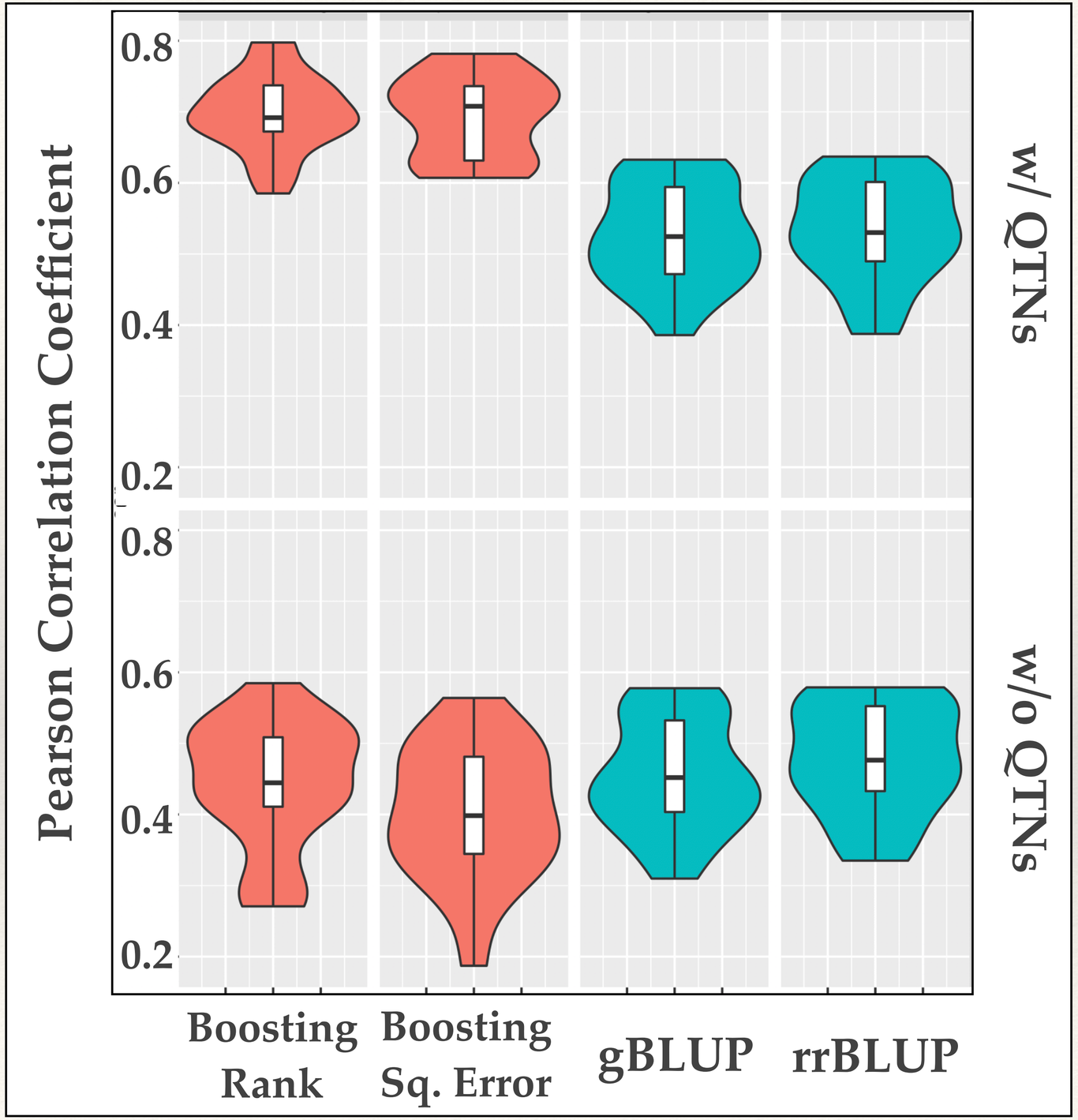


Figure 2. Penalty terms and loss functions. (A) Penalty terms: L_0-norm imposes the most explicit constraint on the model complexity as it effectively counts the number of nonzero entries in the model parameter vector. While it is possible to train prediction models with L_0-penalty using, e.g., greedy or other types of discrete optimization methods, the problem becomes mathematically challenging due to the nonconvexity of the constraint, especially when other than the squared loss function is used. The convexity of the L_1 and L_2 norms makes them easier for the optimization. While the L_2 norm has good regularization properties, it must be used together with either L_0 or L_1 norms to perform feature selection. (B) Loss functions: The plain classification error is difficult to minimize due to its nonconvex and discontinuous nature, and therefore one often resorts to its better behaving surrogates, including the hinge loss used with SVMs, the cross-entropy used with logistic regression, or the squared error used with regularized least-squares classification and regression. These surrogates in turn differ both in their quality of approximating the classification error and in terms of the optimization machinery they can be minimized with (Text S1).
doi:10.1371/journal.pgen.1004754.g002







Data

3093 markers

281 maize association inbred lines.

Simulation

Heritability : 0.75

# of QTNs : 30

5-fold validation for 20 iterations

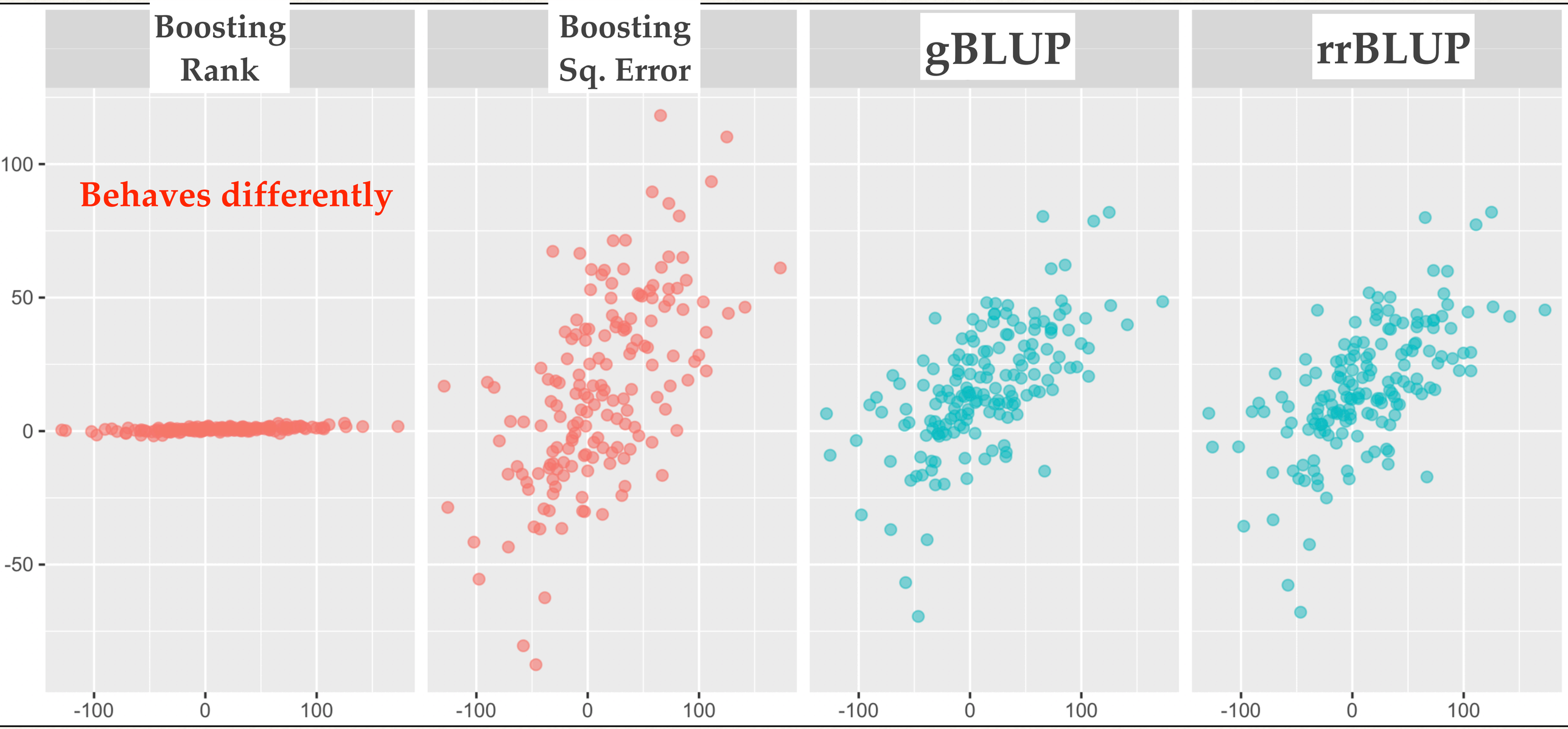
Evaluation

Pearson Correlation Coefficient

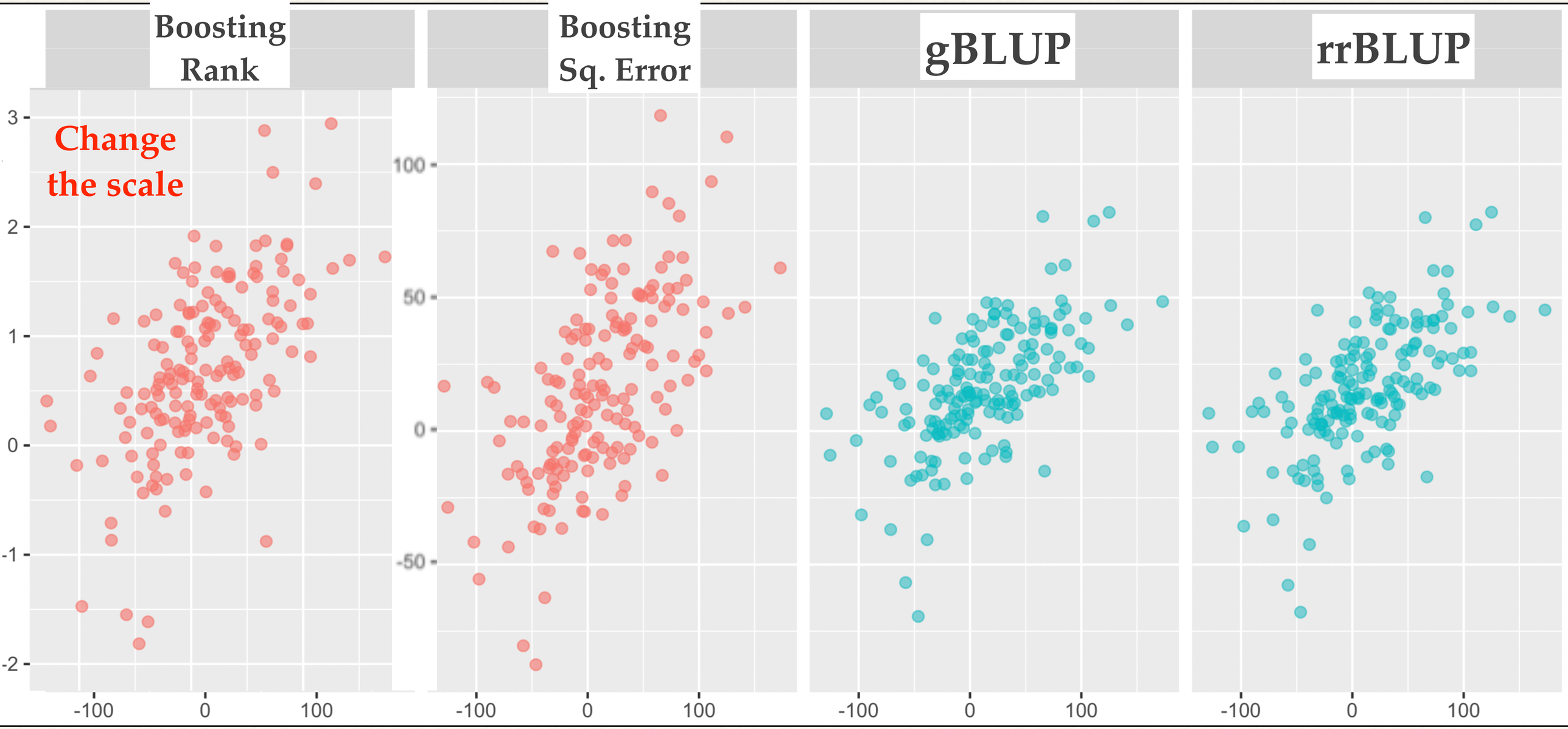
Boosting

Stats Model

Observation vs. Prediction plot



Observation vs. Prediction plot



Available API in R : XGBoost

dmlc XGBoost

- Flexible API
- Work with boosted models

Parameters for Linear Booster

- lambda [default=0, alias: reg_lambda]
  - L2 regularization term on weights, increase this value will make model more conservative. Normalised to number of training examples.
- alpha [default=0, alias: reg_alpha]
  - L1 regularization term on weights, increase this value will make model more conservative. Normalised to number of training examples.
- updater [default='shotgun']
  - Linear model algorithm
    - 'shotgun': Parallel coordinate descent algorithm based on shotgun algorithm. Uses 'hogwild' parallelism and therefore produces a nondeterministic solution on each run.
    - 'coord_descent': Ordinary coordinate descent algorithm. Also multithreaded but still produces a deterministic solution.

Regularized terms

- objective [default=reg:linear]
  - "reg:linear" –linear regression
  - "reg:logistic" –logistic regression
  - "binary:logistic" –logistic regression for binary classification, output probability
  - "binary:logitraw" –logistic regression for binary classification, output score before logistic transformation
  - "gpu:reg:linear", "gpu:reg:logistic", "gpu:binary:logistic", gpu:binary:logitraw" –versions of the corresponding objective functions evaluated on the GPU; note that like the GPU histogram algorithm, they can only be used when the entire training session uses the same dataset
  - "count:poisson" –poisson regression for count data, output mean of poisson distribution
    - max_delta_step is set to 0.7 by default in poisson regression (used to safeguard optimization)

Objective functions

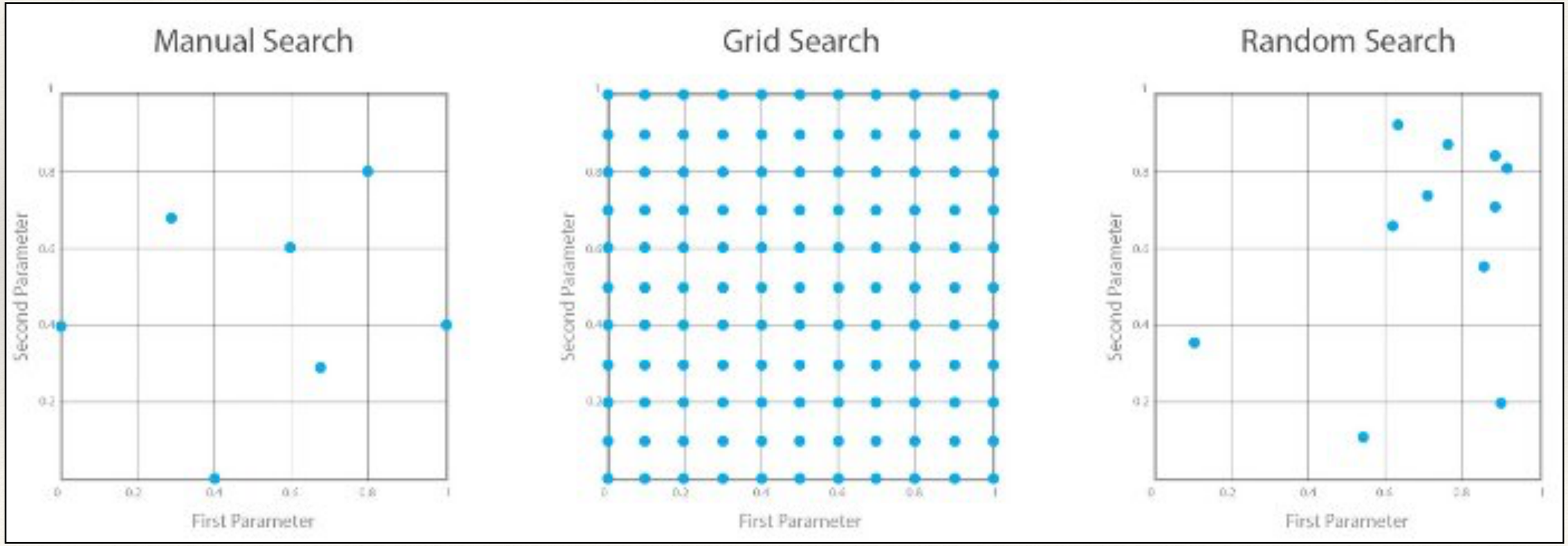
- max_delta_step [default=0]
  - Maximum delta step we allow each tree's weight estimation to be. If the value is set to 0, it means there is no constraint. If it is set to a positive value, it can help making the update step more conservative. Usually this parameter is not needed, but it might help in logistic regression when class is extremely imbalanced. Set it to value of 1-10 might help control the update
  - range: [0,∞]
- subsample [default=1]
  - subsample ratio of the training instance. Setting it to 0.5 means that XGBoost randomly collected half of the data instances to grow trees and this will prevent overfitting.
  - range: (0,1]
- colsample_bytree [default=1]
  - subsample ratio of columns when constructing each tree.
  - range: (0,1]
- colsample_bylevel [default=1]
  - subsample ratio of columns for each split, in each level.
  - range: (0,1]

Subsampling parameters

Tuning Hyperparameters

Tuning Hyperparameters

Grid Search -> Bayesian Optimization



1st parameter

2nd parameter

2nd parameter

2nd parameter

https://www.kdnuggets.com/2016/08/automating-data-science.html

Tuning Hyperparameters : Grid Search

Genomic Selection in Plant Breeding: A Comparison of Models

Nicolas Heslot, Hsiao-Pei Yang, Mark E. Sorrells, and Jean-Luc Jannink*

CROP SCIENCE, VOL. 52, JANUARY–FEBRUARY 2012

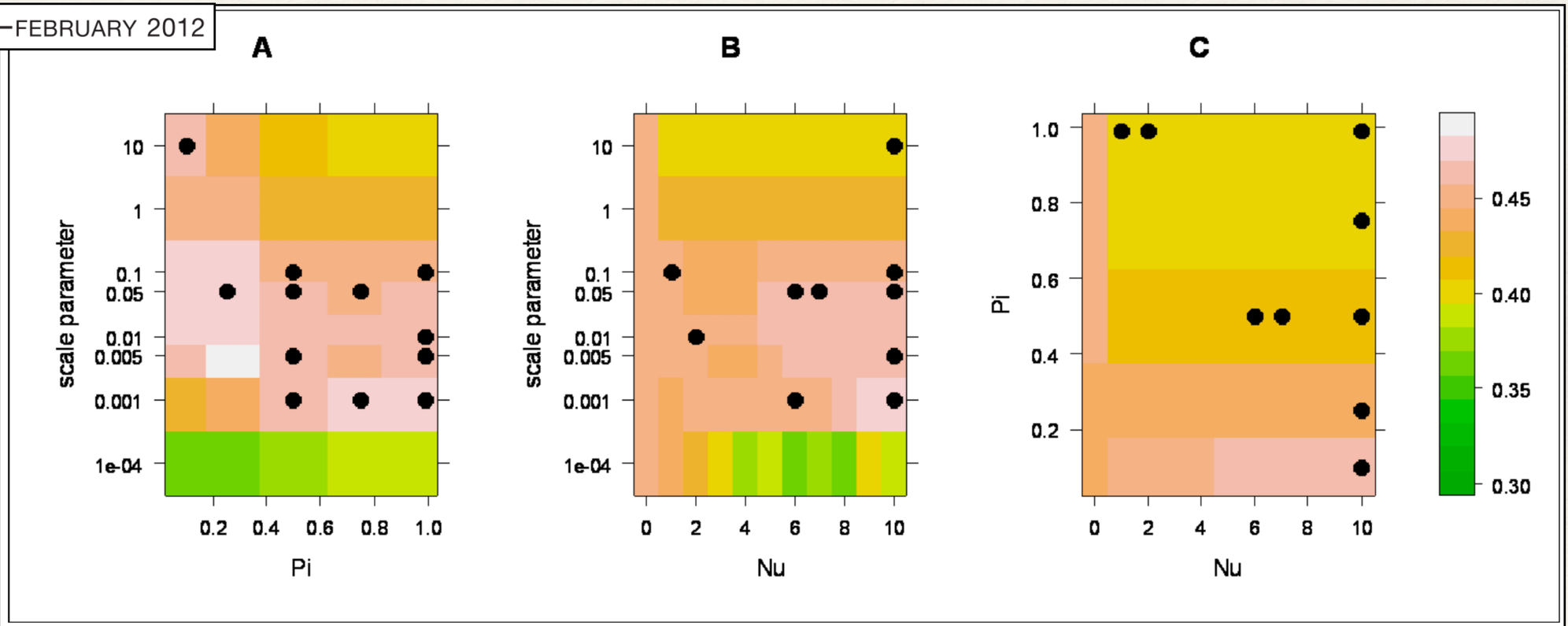


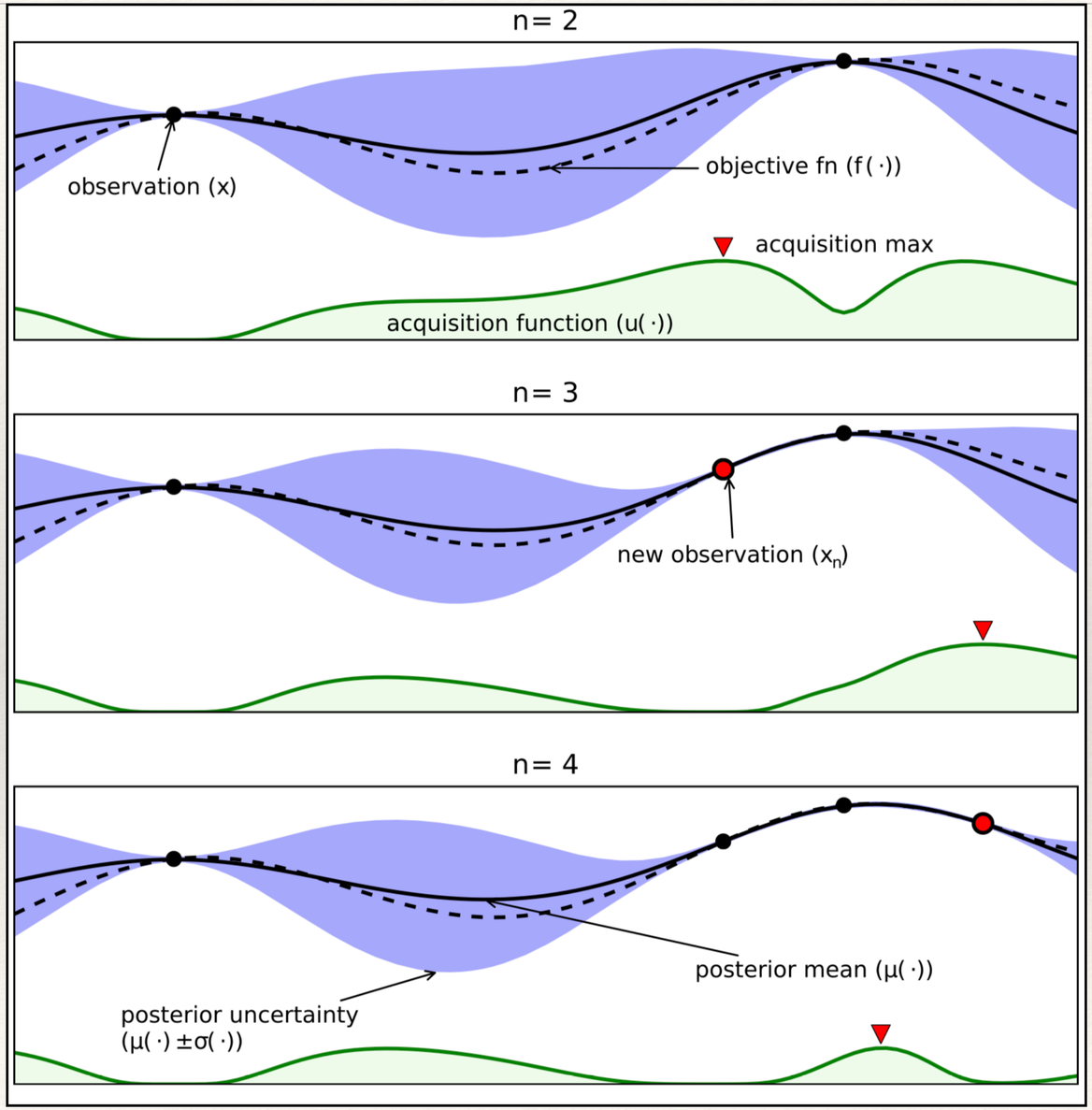
Figure 2. Heat maps summarizing accuracies for the grid search on weighted Bayesian shrinkage regression (wBSR) prior parameters. The accuracies were centered across the 600 triplets of parameters for each trait and averaged. The axes of the heat map correspond to the prior parameters for the marker effect variance. Nu, the degree of freedom, and the scale parameter are the parameters of the

Tuning Hyperparameters : Bayesian Optimization

Iteration = 2

Iteration = 3

Iteration = 4



→ **Performance**

→ **Acquisition function**

- Large variance
- High performance

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7352306&tag=1

Summary

- Different paradigms of machine learning and their examples
- Basic form of supervised machine learning
- Why do we do regularization, and how?
- What is ensemble method, and why? (Variance-bias tradeoff)
- How does bagging work?

Thanks for your attention