MACHINE LEARNING FOR STATISTICIANS

Andy Liaw and Junshui Ma

P Public

Instructors Introduction

- **Dr. Andy Liaw**: Sr. Principal Scientist, *Biometrics Research, Merck* andy_liaw@merck.com
- **Dr. Junshui Ma**: Director, *Translational Oncology Statistics, Merck* junshui_ma@merck.com

Outline

- What is Machine Learning ? (Junshui Ma) ~ 45 minutes
- Supervised Learning Workflow (Andy Liaw) ~ 30 minutes
- Break (Special Q&A) ~ 10 minutes
- Supervised Learning Methods (Andy Liaw) ~ 60 minutes
- Break (Special Q&A) ~ 10 minutes
- Model Interpretation, including Feature Selection and Other Topics (Junshui Ma) ~ 45 minutes

Job Announcement

- An opening at **Early and Translational Oncology Statistics** Department, Merck
- A unique opportunity to work on both oncology **clinical studies** and **biomarker research**
- Flexible position level :
  - Fresh Ph.D. (Sr. Scientist) ,
  - 3+ years of related post-Ph.D. experience (Associate Principal Scientist), or
  - 7+ years of related post-Ph.D. experience (Principal Scientist).
- Requirements:
  - Excellent communication skills
  - In-depth understanding in statistical principles and methods
  - Programming capability
  - A self-motivated team player
- Contact Email: junshui_ma@merck.com

Machine Learning for Statisticians, Part I:
What is Machine Learning?
– Machine Learning vs. Statistics

Junshui Ma and Andy Liaw

Public

Outline

- Goals of This Course
- What is Machine Learning (ML)?
- ML vs. Statistics: Similarities and Dissimilarities
- Different areas of ML

“RELEASE OF LIABILITY” statements**:**

- **Lowering expectation**: I understand that machine learning experts were not made in 3.5 hours.
- **Resolving disagreement professionally**: I agree to hold the presenters harmless from any damages to my pride as a statistician.

Why Do You Want to Learn ML?

Goals of This Course

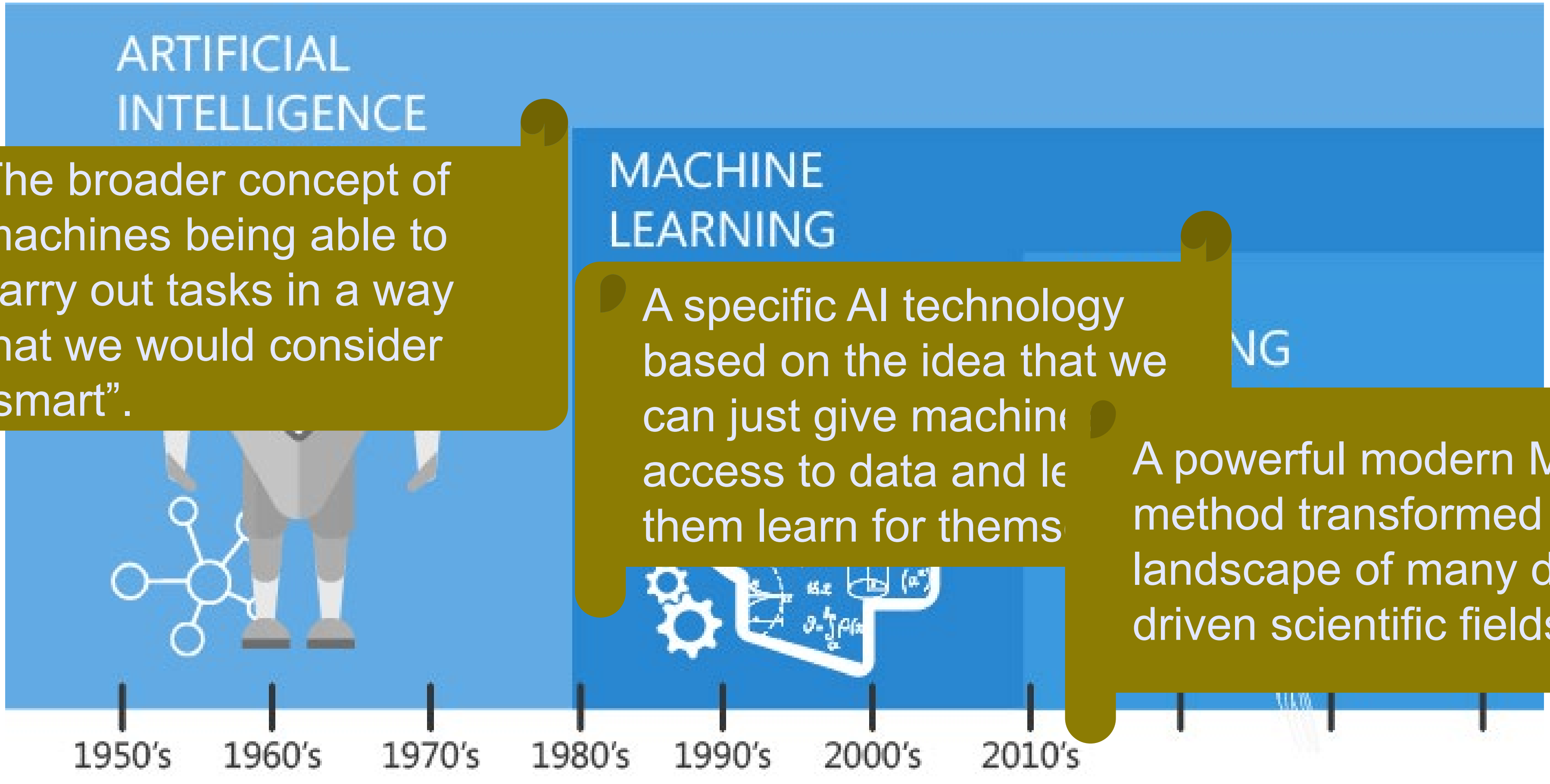
- To understand **similarities** and **differences** between ML and Statistics
- To learn general ML **concepts** and **methods**
- To be an **innovator** capable of identifying ML tasks in your daily work

WELCOME

TO THE

DARK SIDE

Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL)



Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.

The broader concept of machines being able to carry out tasks in a way that we would consider "smart".

A specific AI technology based on the idea that we can just give machin access to data and le them learn for thems

A powerful modern ML method transformed the landscape of many data-driven scientific fields

In the media,

AI ~= Machine Learning

or

AI ~= Deep Learning

https://www.linkedin.com/pulse/ai-machine-learning-evolution-differences-connections-kapil-tandon

What is **Machine Learning (ML)** ?

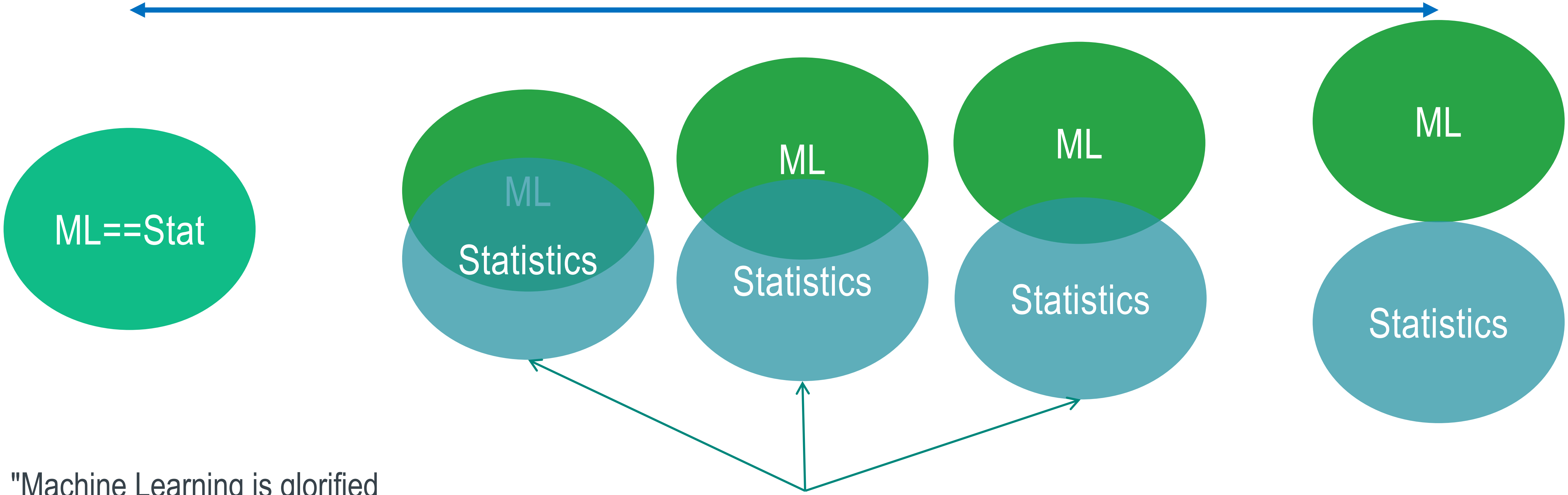
Two items on Wikipedia:

- "Machine Learning": a field that that gives computers the ability to *learn without* being explicitly *programmed*. (coined by an IBM Engineer, Arthur Samuel, in 1959)
- "Portal: Machine Learning": a scientific discipline that explores the construction and study of *algorithms* that can *learn* from *data*.

Regarding algorithms for computers to automatically learn from data

Machine Learning (ML) vs. Statistics

A Spectrum of Viewpoints



Our Viewpoints

"Machine Learning is glorified Statistics." -- Robert Tibshirani

(many others', e.g. **Targeted Learning** by Mark van der Laan et al.)

Public

Machine Learning (ML) vs. Statistics (Cont.)

Statistics is the science

- of *learning* from *data*, and
- of measuring, controlling, and communicating *uncertainty*.

* Davidian, M. and Louis, T.A. (2012), “Why Statistics?”, Science, 336(12). (like a statement from ASA)

Machine Learning (ML) vs. Statistics : **Similarities**

- The sciences and technologies of ***learning from data***
- Shared underlying math and computing machinery
  - general principles: sampling, overfitting, regularization, etc.
  - probability and stochastic theories
  - optimization theories and algorithms
  - computing languages and platforms
- Shared researchers and practitioners

  Many machine-learning methods were proposed by statisticians
- Fruitful marriage of two fields

  e.g. Hilbert Schmidt Independence Criterion, Targeted Learning

P Public

Machine Learning (ML) vs. Statistics: **Dissimilarities**

Fundamentally, different schools of thinking *

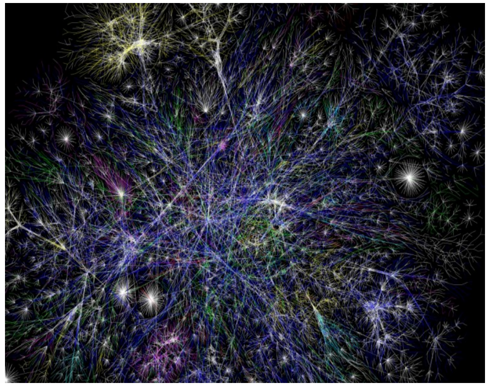
- Machine Learning

  "Algorithms" : Engineers' heuristic and practical approach to find solutions

- Statistics

  "Models" : Mathematicians' principled approach to figure out why.

How can I hack the system to get the job done? Truth? Who knows!



Assuming the model represents the system, I can figure how the system works!

Public

* Leo Breiman (2001), "Statistical Modeling: The Two Cultures", Statistical Science, 16(3).

Machine Learning (ML) vs. Statistics: **Dissimilarities (cont.)**

- Derived Differences:
  - Answering different questions

    ML: How does DrugA work on *John* vs. Placebo do? ⇔ Stat: Is DrugA better than Placebo?
  - Formulating the problem differently

    ML: An objective to optimize ⇔ Stat: A model to fit
  - Emphasizing *uncertainty* differently

    ML: Good to have ⇔ Stat: Inherent to have, and hardly acceptable not to have
  - Evaluating model and performance differently

    ML: separate datasets (e.g. cross-validation, etc). ⇔ Stat: same dataset (e.g. R^2, residual analysis, AIC, etc.)
  - Resultant models used differently

    ML: model prediction ⇔ Stat: inference based on model parameters

Public

Two Different Approaches on Linear Regression : y =$X\beta$

Statisticians:

Given data $\{X_i, yi\}_{i=1..N}$, try to fit a linear line (or plane) over the data

- Model Fitting

Assume $Y \sim N(\boldsymbol{X}\beta, \sigma^2 I)$

Therefore: $\hat{\beta} = (\boldsymbol{X}^t\boldsymbol{X})^{-1}\boldsymbol{X}^t Y$,

and $\hat{\beta} \sim \mathrm{Norm}(\beta, \sigma^2(\boldsymbol{X}^t\boldsymbol{X})^{-1})$.

- Model Evaluation

Evaluate **model fitting** using goodness-of-fit, residual check, and/or AIC etc.

- Model Utilization (or **Inference)**

Use **fitted model** to estimate and draw inference about β (e.g. Hypothesis Testing (HT), Conf. Int. (CI)).

Machine Learners:

Given training data $\{X_i, yi\}_{i=1..L}$, and test data $\{X'_i\}_{i=1..M}$, try to predict $\{y'_i\}_{i=1..M}$

- Predictive Model Learning/Training

Define an objective: minimizing $L(f) = \frac{1}{n}\sum\{f(X_i) - y_i\}^2$

Choose $f(X;\beta) = X\beta$ (i.e. a linear function)

Use an optimization procedure to find $\hat{\beta}$ to minimize $L(f)$

- Model Evaluation

Evaluate **trained predictive model** $f(X;\hat{\beta})$ by $L(f)$ on a held-out set.

- Model Utilization

Use **trained model** to predict y′ of $\boldsymbol{X}'$.

(Don't care much about the parameters themselves.)

Public

Two Different Paradigms

Statistics	Machine Learning
Specify a model that describes distribution of data (how the data were generated from a distribution)	Specify an objective and a functional class to related features X with outcome Y (no distribution specified)
Propose a principled way to fit the model (e.g. maximum likelihood)	Find a computational process (e.g. optimization) to find the functional parameters to optimize the objective
Emphasize population characteristics (CI, HT, etc.)	Emphasize prediction of individual subject
Assessment of uncertainty enabled by assumptions	Usually don't have uncertainty assessment
Model evaluation is usually based on AIC, BIC, etc.	Evaluate algorithmic performance on hold-out data
Assuming the model (at least close) to be true	No notion of a "true model"

Heavy focus on the model

Heavy focus on the computational algorithm

Public

Statistics and Machine Learning

“Neither Statistics nor Machine Learning is a subset of the other, and neither lays exclusive claim to a technique. They are like two pairs of old men sitting in a park playing two different board games. Both games use the same type of board and the same set of pieces, but each plays by different rules and has a different goal because the games are fundamentally different. Each pair looks at the other’s board with bemusement and thinks they’re not very good at the game.”

https://mllib.wordpress.com/2017/10/10/machine-learning-vs-statistics-he-texas-death-match-of-data-science-2/

Areas of Machine Learning

- **Supervised Learning**

  Training Data = { **X**, y } $\Rightarrow$ Predictive model for y

  – ***Semi-supervised Learning***

  Training Data = { **X**, some y } $\Rightarrow$ Predictive model for y

- **Feature Ranking, Selection, Engineering (Extraction)**

  X = {x_1[age], x_2[sex], x_3[region], x_4[ECOG], x_5[cancer stage], ……}

  Training Data = { **X**, y } $\Rightarrow$ Feature *ranking*, or a *feature subset*, or a derived *new feature* set

- **Unsupervised Learning**

  Data = { **X** } $\Rightarrow$ Representation model for **X**

- **Reinforce Learning**

- **Other** (e.g. Kernel Learning, **Deep Learning**, etc.)

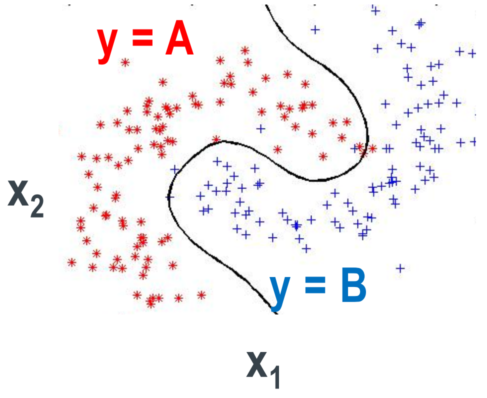
Supervised Learning: Training Data = { **X**, y } ⇒ Predictive model for y

- **Classification**

  y: a categorical variable

  e.g. {response, no-response},

  {cured, improved, unchanged, worse}

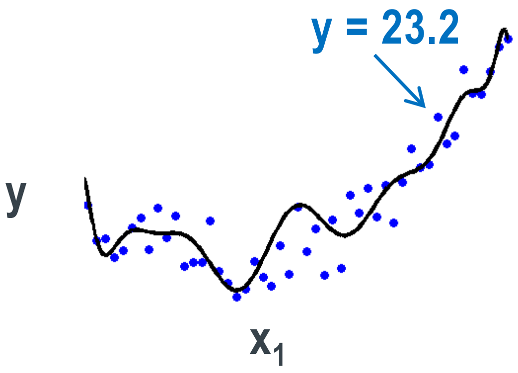


- **Regression**

  y: a *numeric* variable

  e.g. HIV virus load,

  Total sleep time



Semi-supervised Learning is just an extension of supervised learning

Public

Will John have delayed *response* to Treatment M?

Historical Data of Many People Treated by Treatment M ⇒ **Training Dataset**

y (response, dependent variable)

X (features, predictor, covariate, or independent variable)

YRegressN	YClass1C	AgeN	SexC	RaceC	BMIC	StageC	SurgLocC	Cond1C	Cond2C	DoseN	DurationN	FlagC	SysOrgan9C
5.208806	FALSE.	35	M	Asian	< 30	I	ENL	SU020	RO	0.79452055	6	0	0
5.508822	FALSE.	54	F	White	< 30	III_IV	Other	SU020	RO	0.60000000	2	0	0
5.106687	FALSE.	40	F	White	< 30	II	FE	SU040	RO	1.21328225	65	0	0
5.816826	TRUE.	70	F	White	< 30	III_IV	UR	SU020	RO	1.95000000	35	0	0

Supervised Machine Learning Method

John's X

AgeN	SexC	RaceC	BMIC	StageC	SurgLocC	Cond1C	Cond2C	DoseN
65	M	White	30 to <40	III_IV	Other	SU020	RO	0.60606

Trained Predictive Model

Predicted John's y

YRegressN
6.314168

Regression

YClass1C
TRUE.

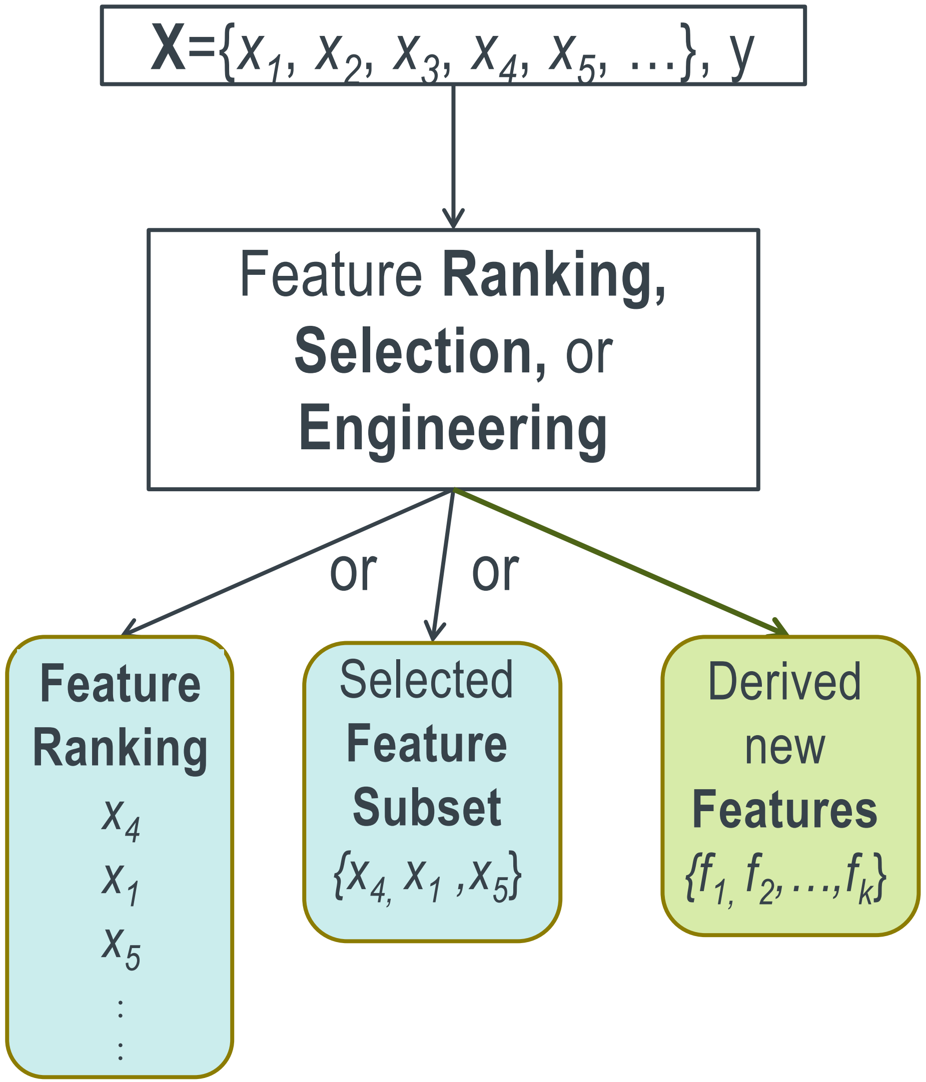
Classification

No p-Value / Confidence Interval

Question for Statistics: Does *Treatment M* have more delayed responders than *Standard Care does*?

Public

Feature Ranking, Selection, Engineering: Data { **X**, y } ⇒ Ranking, Subset, New

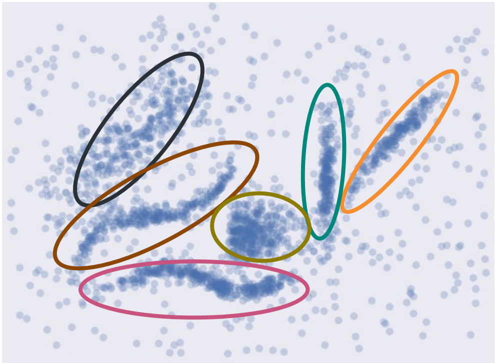


Example Questions:

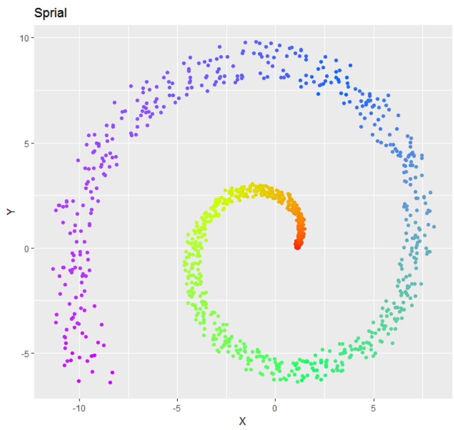
- How can I rank the importance of the risk factors for predicting a person to have breast cancer in 5 years? (*ranking*)
- Which genes are useful for differentiating responders from non-responders for Treatment M? (*selection*)
- Can I extract novel information from cancer tumor images to better predict a patient's survival? (*engineering*)

Unsupervised Learning: Data = { **X** } ⇒ Representation model of **X**

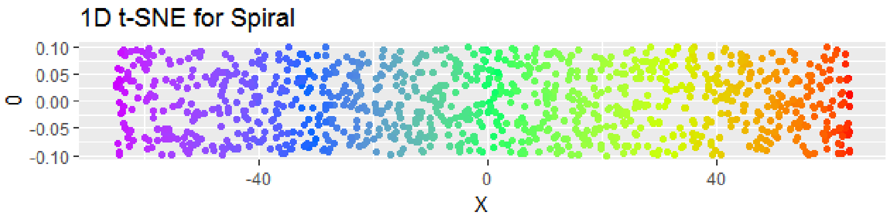
- **Clustering**



- **Dimension Reduction**



⇒



- **Association Rule**

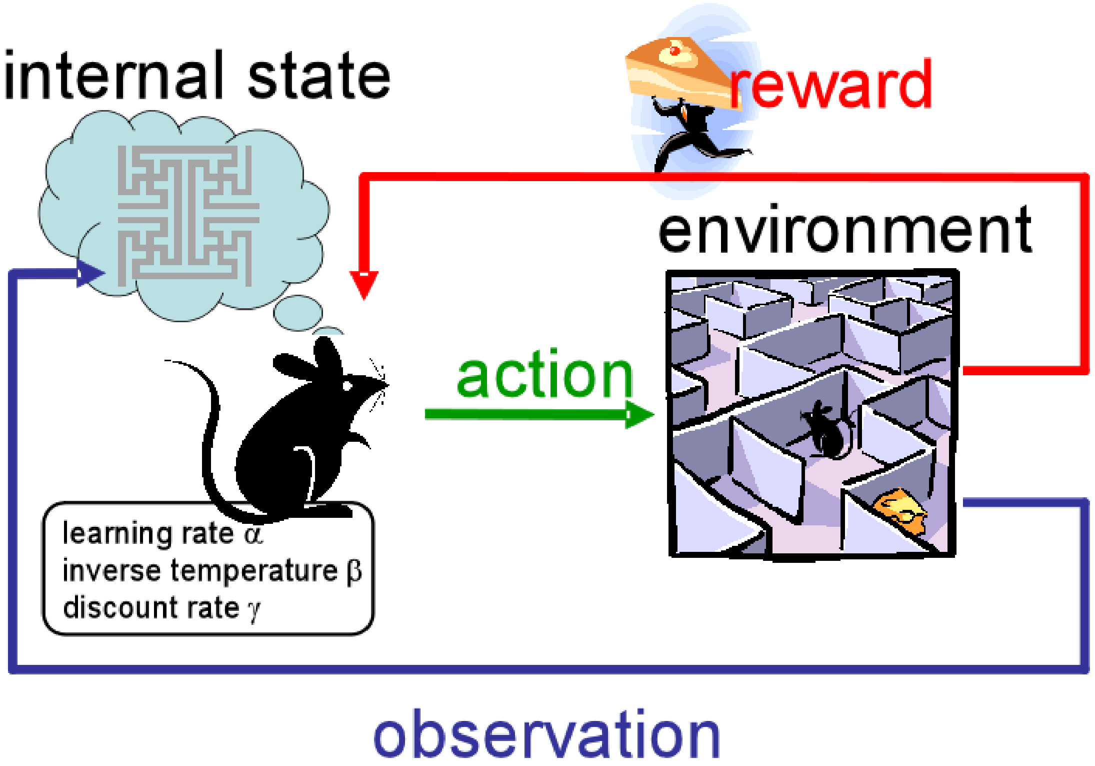
- ……

Example Questions:

- Can we group tumor size changes under Treatment M into some patterns?
- Can I somehow visualize all patient safety information in a plot?

Public

Reinforce Learning (Policy Learning)



Example Questions:

1. What is the best strategy to play this computer game and win?
2. How should the new antibiotic be promoted and used?

Policy : Pr(Action | Observation) for Max. Reward

Google's AlphaGo AI wins three-match series against the world's best Go player

Jon Russell May 25, 2017

Other (e.g. Kernel Learning, Deep Learning, etc.)

- A different way to categorize machine learning areas: not by problem to solve, but by **technology** to use.
- These learning technologies can be used at all the areas we mentioned before.
- **Deep Learning** is the reason we are talking about artificial intelligence and machine learning today.

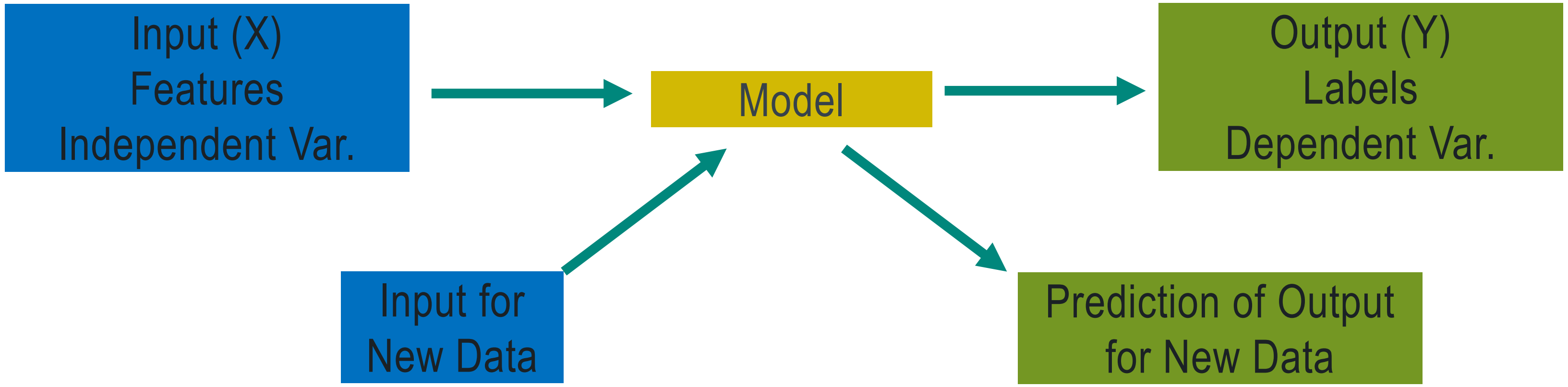
Public

Machine Learning for Statisticians, Part II: **Supervised Learning Workflow**

Andy Liaw

Public

Supervised Learning



- From a collection of input-output pairs, learn to predict what output should be given input
- Premise of supervised learning: data with **similar input** should have **similar output**
  - If we know what "similar" should mean for the task at hand; e.g., what variables are key in assessing similarity, the problem is easier
  - If we don't know what "similar" should mean, we have to rely on the algorithm to discover that

1.D. E. Patterson,, R. D. Cramer,, A. M. Ferguson,, R. D. Clark, and, L. E. Weinberger. Neighborhood Behavior: A Useful Concept for Validation of "Molecular Diversity" Descriptors. *Journal of Medicinal Chemistry* **1996,** *39* (16) , 3049-3059. https://doi.org/10.1021/jm960290n

Public

Data Splitting

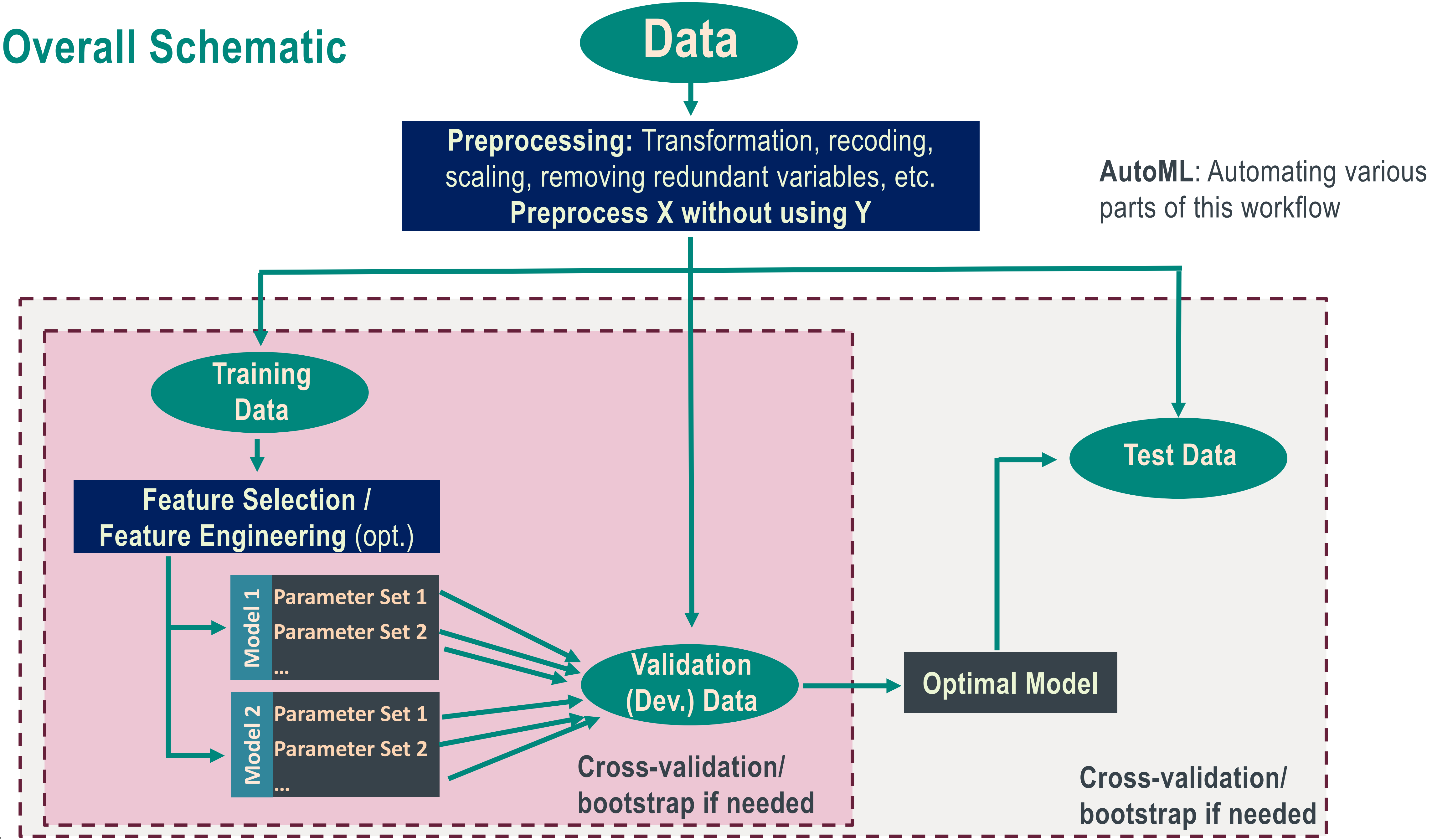
Recall one of the differences between Stats and ML:

- In Statistics data are assumed to be generated from a probability model. Everything stems from estimation of that probability model.
- In ML no probability models are assumed or even implied.

Consequence:

- In Statistics model performance is evaluated based on statistical principles. Typical method papers argue their superiority from theoretical grounds, and show performance on data only as example.
- In ML model evaluation is done empirically on independent data not used in model building. Typical method papers show no theoretical results that applies generally, but show performance on a battery of benchmark data sets.

Overall Schematic



Public

Workflow can Vary Depending on Goal

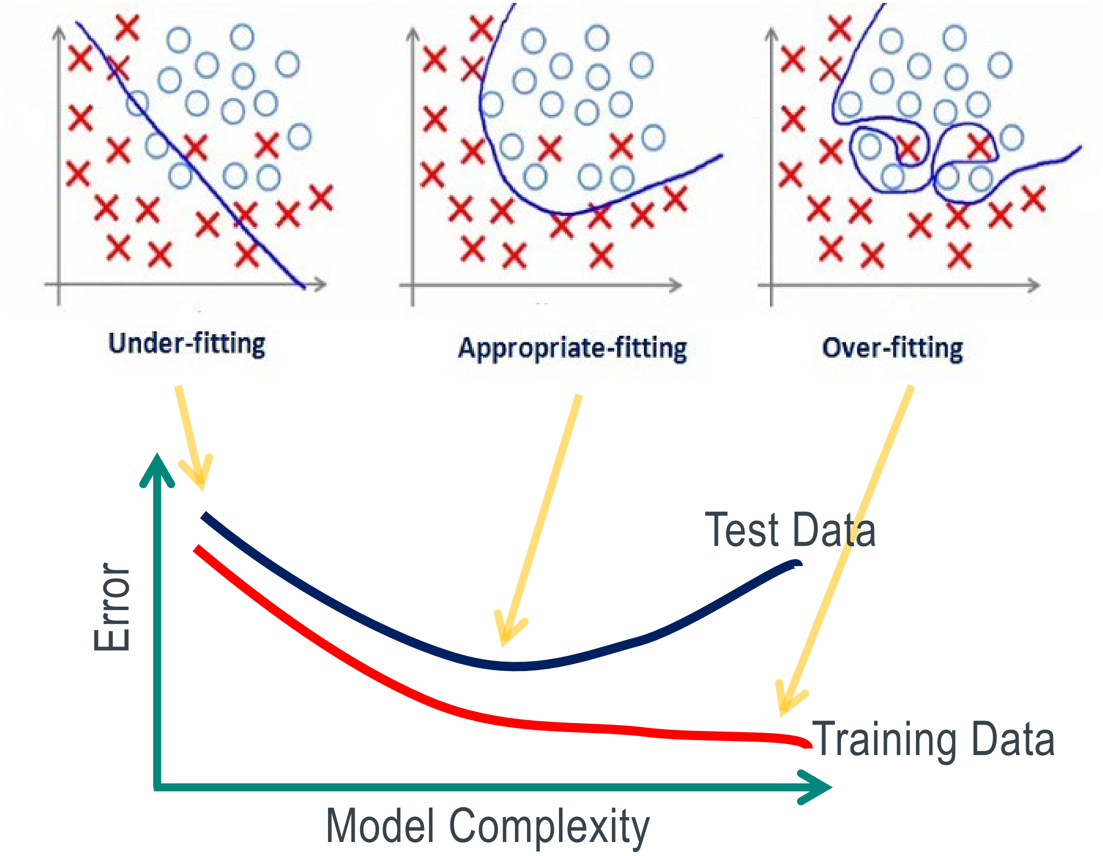
- If prediction of future data is the only utility of the model, feature selection may not be needed at all
- If the relationship between important predictor variables and the response is required, then one may need to pay closer attention to feature selection and perhaps also choice of methods
- Try to orient every part of the process toward the final objective, instead of optimizing each part with a different objective function
- Tradeoff between interpretability and performance: Generally highly flexible methods have capability for higher accuracy, but less interpretability, and vice versa
- If goal is interpretability, can still use flexible methods as performance benchmark

Data-Splitting is of Critical Importance

Training	Validation / Dev	Testing
Feature Selection Model fitting	Parameter Tuning Model Selection	Performance Evaluation
Use Majority of Data	Up to 20% of Data	Up to 20% of Data

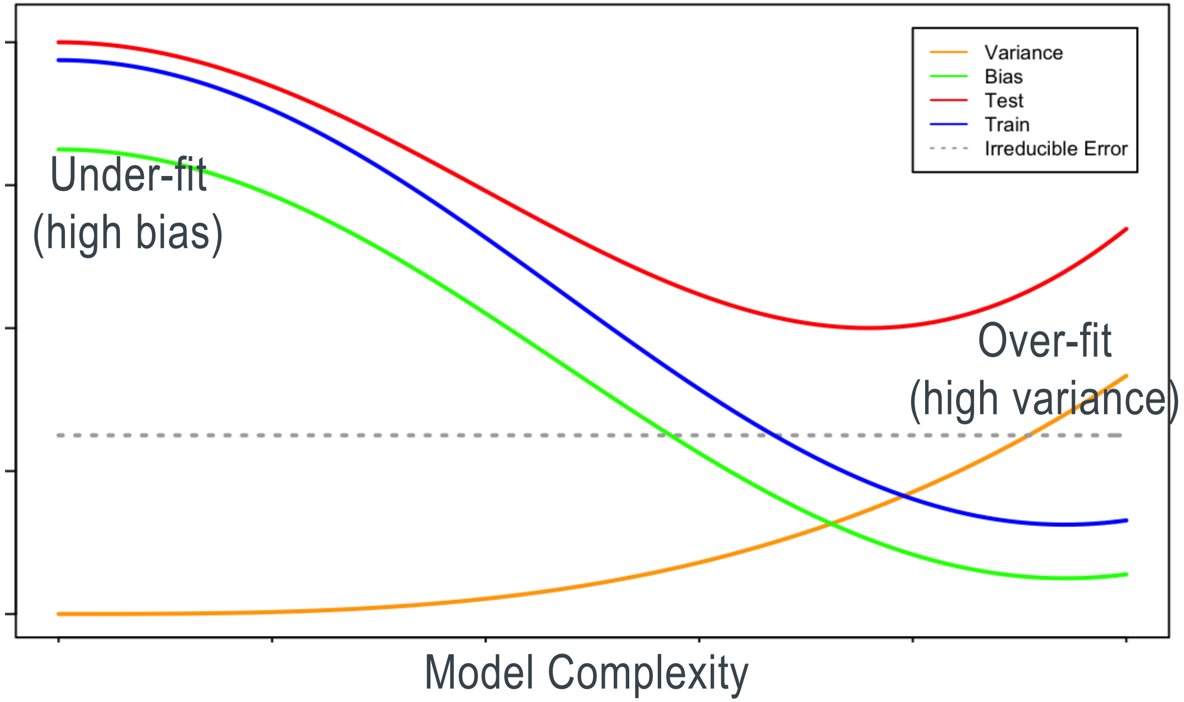
- The three stages of the workflow (train / dev / test) require separate subsets of data. The split should be done before any other data processing.
- Any fitting, selection, etc. can be seen as optimization, but only to the data operated on. Thus must be tested on separate data set. (Similar reason that data cannot be used for both hypotheses generation and confirmation.)
- Common pitfall: select features using all data, then split into training / dev / test sets; yielding overly optimistic result.
- If sample size is not sufficiently large, can split data randomly multiple times (cross-validation or bootstrap)

Training: Under- and Over-fitting



- ML is about finding repeatable patterns--- patterns seen in the training data that will repeat in the future
- Under-fitting is not picking up enough patterns (usually insufficient model complexity)
- Over-fitting is picking up patterns in the training data that will not occur in the future (usually excessive model complexity)
- Thus fitness is usually gauged as a function of model complexity

Bias-Variance Trade-off



Analogous to linear regression:

- Overly simple models (omitting variables with non-zero coefs) leads to biased prediction
- Including extraneous variables (those with 0 coefs) does not affect bias, but leads to increased variance

Optimal model finds the best compromise between the two

Regularization: Controlling Over-fitting

- One way to allow increased model complexity but still keep over-fitting in check is through regularization (a la ridge regression)
- Several strategies for regularization:
  - ❑ Add penalty to cost function (L1, L2, or combination of both)
  - ❑ Randomize certain aspect of optimization to control greediness (e.g., random feature selection, dropout)

Model Fitting and Parameter Tuning

- “Hyper-parameters” of an algorithm (modeling method) are user-selected parameters; e.g., shrinkage parameter in ridge regression
- Once hyper-parameters are set, an algorithm is executed to produce a model; usually by minimizing a loss function or maximizing a fitness function (e.g., likelihood)
- Hyper-parameters usually control the model complexity
- In ML, optimal model complexity is usually unknown and thus to be determined from data
- Lowest complexity: predict everything as mean of response; highest complexity: something like nearest neighbor, or interpolation of training data
- How well a hyper-parameter setting works need to be evaluated on data not used in model fitting
- Model / algorithm selection should be considered as part of the model optimization

Example: Logistic Regression

Given data like this
:

Y	AgeN	SexM	RaceBlack
0	-0.91	0	0
0	-0.21	0	0
1	1.59	0	0
0	1.46	1	0
0	-1.55	0	0
0	-0.72	0	0

"learn" a function $f()$ that can predict `Y` from `AgeN`, `SexM`, `RaceBlack`, …

For logistic regression (with parameters w):

$$f(x;w) = \{1 + \exp(-w^t x)\}^{-1},$$

Define the loss function

$$L(f(x;w),y) = y\log f(x;w) + (1-y)\log(1 - f(x;w))$$

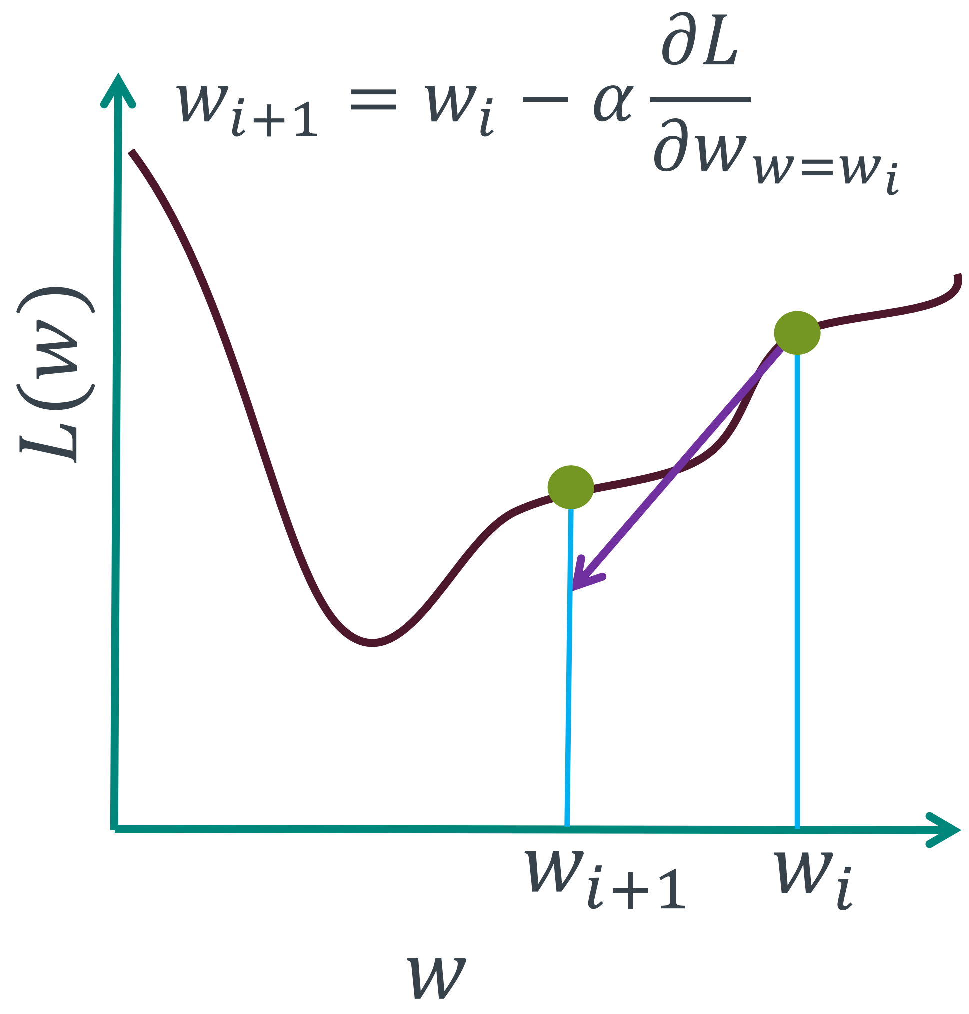
(This particular loss function is called *cross entropy*.)

Find w that minimizes average loss on the training data

Predict Y = 0 if f < 0.5; Y = 1 otherwise

Public

Minimizing the Loss with Gradient Descent



Loss: $\frac{1}{n}\sum\{y_i \log f(x_i; w) + (1 - y_i) \log(1 - f(f(x_i; w))\}$

Gradient: $\frac{\partial L}{\partial w} = \sum(f(x_i; w) - y_i)x_i$

Initialize w (usually with random numbers)

Set a learning rate α

Iterate: $w_{i+1} = w_i - \alpha \frac{\partial L}{\partial w}_{w=w_i}$

Stop when $|w_{i+1} - w_i| < \mathrm{tol}$

Regularized (or Penalized) Regression

For regularization, we add a term to the loss function that penalize solution with large values of $\boldsymbol{w}$

L2 penality (ridge): shrink $\boldsymbol{w}$ toward 0

$$\min_{\boldsymbol{w}} C \sum_{i=1}^{n} L(f(X_i; \boldsymbol{w}), Y_i) + \sum_{j=1}^{p} w_j^2$$

L1 penalty (LASSO): some $\boldsymbol{w}$ may be shrunken to 0

$$\min_{\boldsymbol{w}} C \sum_{i=1}^{n} L(f(X_i; \boldsymbol{w}), Y_i) + \sum_{j=1}^{p} |w_j|$$

C is the tuning parameter. For L1 (LASSO) it actually controls model complexity (due to selection).

Public

Hyper-parameter Tuning

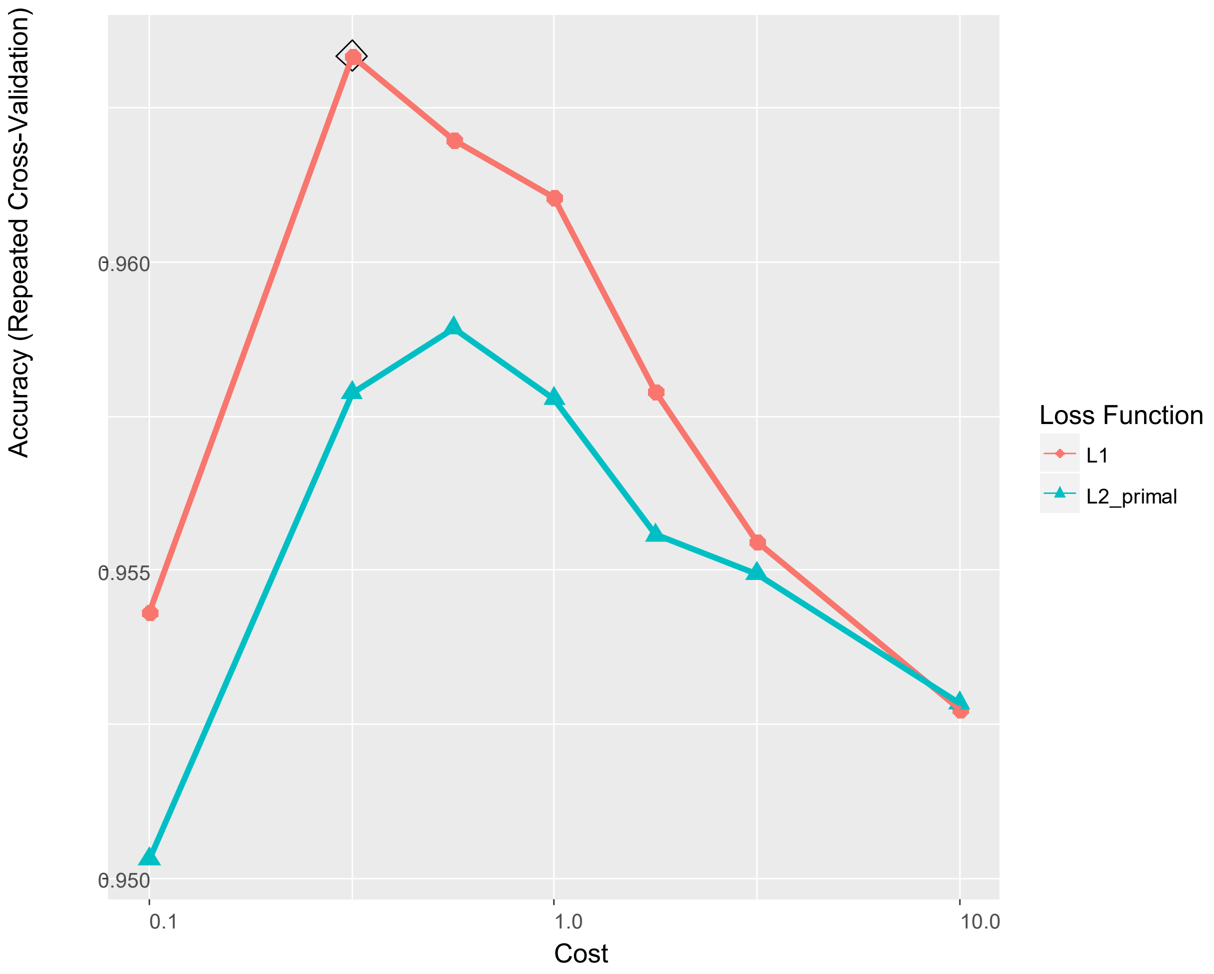
This is usually done by grid search: use a grid on each parameter and form all possible combinations

Range of each parameter may need to be chosen carefully (as some methods can be very sensitive)

Parallel/distributed computing would be very handy for this step

Less brute force method than grid search may be possible (e.g., Bayesian optimization)

This step usually requires more trial-and-error and intuition than theory; i.e., it's more art than science



Metrics for Classifier Performance

Confusion Matrix

	True 0	True 1
Predict 0	N_{TN}	N_{FN}
Predict 1	N_{FP}	N_{TP}

Accuracy: $(N_{TN} + N_{TP}) / N$

Error Rate: $(N_{FP} + N_{FN}) / N$

Sensitivity (Recall, TPR): $N_{TP} / (N_{FN} + N_{TP})$

Specificity: $N_{TN} / (N_{TN} + N_{FP})$

PPV (Precision): $N_{TP} / (N_{FP} + N_{TP})$

NPV: $N_{TN} / (N_{TN} + N_{FN})$

Youden's J: Sensitivity + Specificity – 1

AUROC (Area Under ROC) = transformed Mann-Whitney statistic

Public

Last Steps

- Test data should be used only for performance assessment, not comparison/selection
- If the goal is to obtain a model with best prediction performance, then we are done
- In drug discovery and development, that is rarely the case
- Further insights on relationship between predictor variables and response; e.g., variable importance, partial dependence, etc.
- If simpler models are needed either because of interpretation or feature selection:
  - ❑ Can build a more flexible, “black box” model possibly with better performance
  - ❑ The performance of the black box model can be the benchmark for the simpler model
  - ❑ Find simpler model that gets performance as close to the black box model as possible

Take Home

- ✓ Data splitting is crucial in preventing over-estimating model performance
- ✓ Feature selection should be considered part of the model training step
- ✓ Under-fitting (too much bias) vs. over-fitting (too much variance)
- ✓ Regularization provide more model flexibility while controlling over-fitting
- ✓ Performance estimates used for model / hyper-parameter selection are usually too optimistic
- ✓ Ultimate test of performance is on a completely independent test set, not random subset of available data
- ✓ Goal of model: prediction vs. interpretation

Public

Machine Learning for Statisticians, Part III: **Supervised Learning Methods**

Andy Liaw

Public

Outline

- Supervised Learning – classification and regression
- Classification and Regression Trees
- Ensemble Methods: Bagging, Random Forests, Boosting
- Support Vector Machines and Kernel Methods
- Artificial Neural Networks
- No Free Lunch!

Public

Supervised Learning: Classification and Regression

As if to confuse statisticians, some machine learners call this "hypothesis"!

Given training data $\{X, Y\}$, "learn" a function $f(X)$ that predicts Y

If Y is categorical (e.g., 0/1) then it's classification, and $f(X)$ can output either the label or probability

If Y is numeric (continuous) then it's regression

The form $f(X)$ takes on depends on the method

How $f(X)$ is fitted also depends on method, but typically involves minimizing a loss function

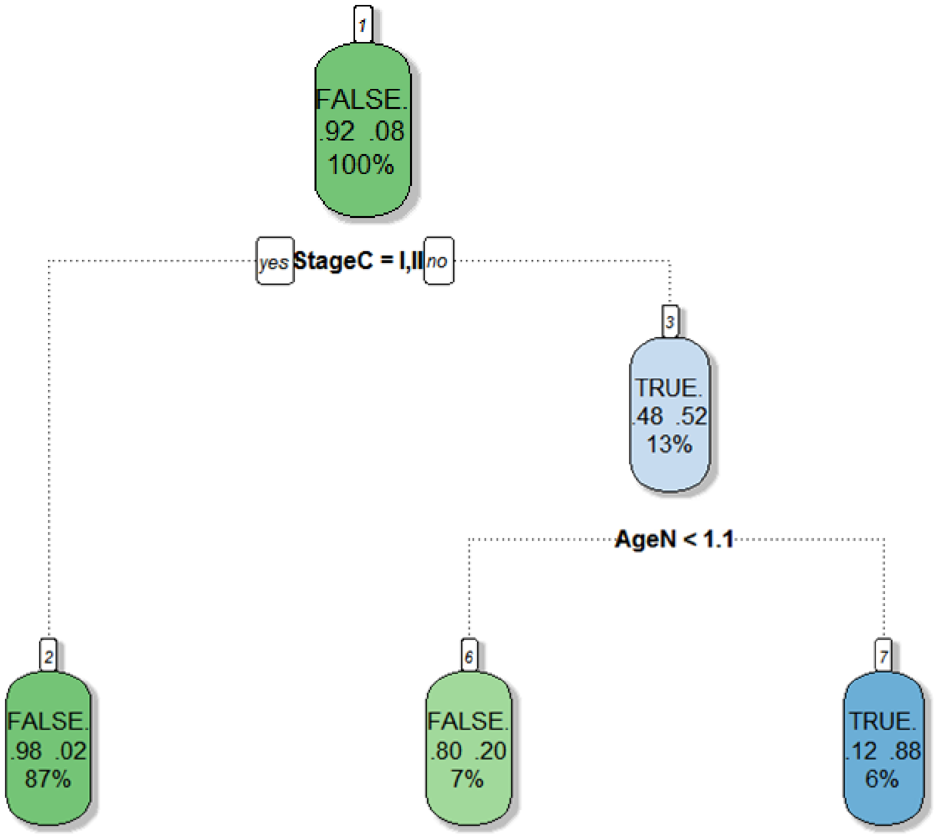
Loss (or cost) function $L(f(x), y)$ reflects the cost incurred by predicting y with $f(x)$

Typically, for classification, $L(f(x), y) = -\sum\{y \log f(x) + (1-y)\log(1-f(x))\}$

For regression, $L(f) = \sum\{y - f(x)\}^2$ (MSE)

Irreducible error (loss of the optimal f): For classification, the Bayes rate. For regression, the residual variance.

Classification And Regression Tree



Public

Classification And Regression Trees - Tuning

- Size of trees: too small can result in under-fitting; too large can lead to overfitting
- Minimum size of terminal nodes
- Node-splitting criterion
- Minimum improvement in splitting

Public

Ensemble Methods: General Idea

Every model need to be at least better than random guessing

Try to have different model make mistakes on different data

Stacking / Super Learner: More adaptive ensembles

Y	Model1	Model2	Model3	Model4	Model5	Aggregate
0	0	1	0	0	1	0
1	0	1	1	1	1	1
1	1	1	0	1	1	1
0	1	0	0	1	0	0
0	0	0	1	0	1	0
1	1	1	1	0	0	1
	67%	83%	67%	67%	50%	100%

CART

Boosting / Random Forest

Gradient Boosting

Original motivation: Incremental improvement

1. Initialize: $F_0(x) = n^{-1} \sum y_i$
2. Iterate $k = 1, \ldots, K$: $r_i = y_i - F_{k-1}(x_i)$. Fit a small tree model $T_k(x)$ to r_is and update $F_k(x) = F_{k-1}(x) + T_k(x)$

Final model is $F_K(x) = \sum_1^K T_k(x)$

Insight: residuals are gradients of the loss function, so the algorithm is basically doing gradient descent in the function space

Cannot "boost forever"; shrinkage helps

Base model should underfit the data; the sequence reduces bias while maintaining low variance

"Stochastic" gradient boosting: use random sample at each iteration

Stochastic Gradient Boosting - Tuning

- Number of iterations (too large can lead to overfitting)
- Shrinkage (multiplier for each base model before added to ensemble): small values can lead to better fit, but may need more iterations
- Bagging fraction: number of randomly chosen data points to use in each iteration; increase diversity of trees
- Size of trees: Controls model complexity; either one of:
  - Number of terminal nodes
  - Max depths of trees

B(ootstrap) AG(gregating)

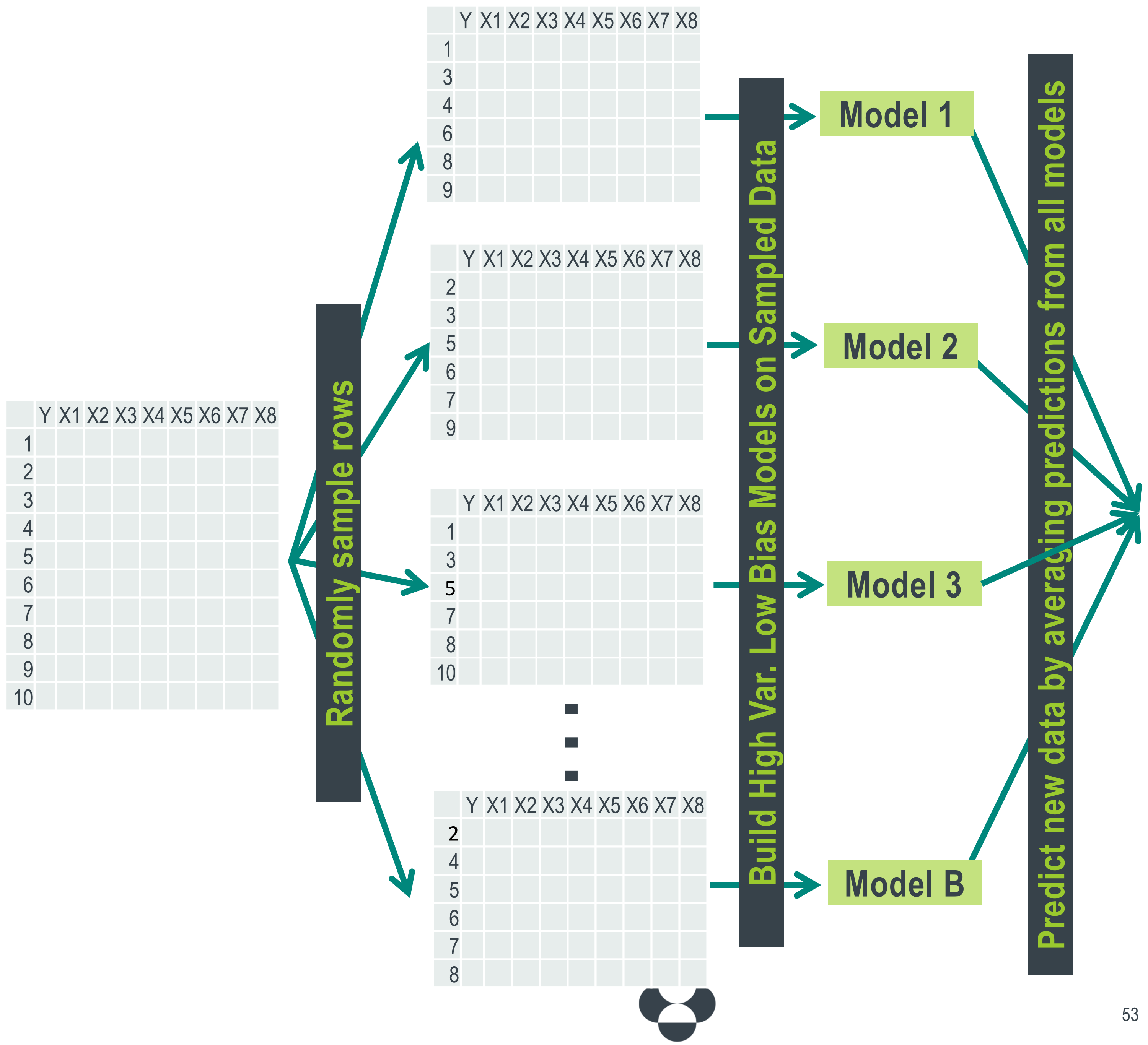
Bagging is a very simple yet effective way of creating ensembles; can be applied to any algorithm

Draw B random samples from training data

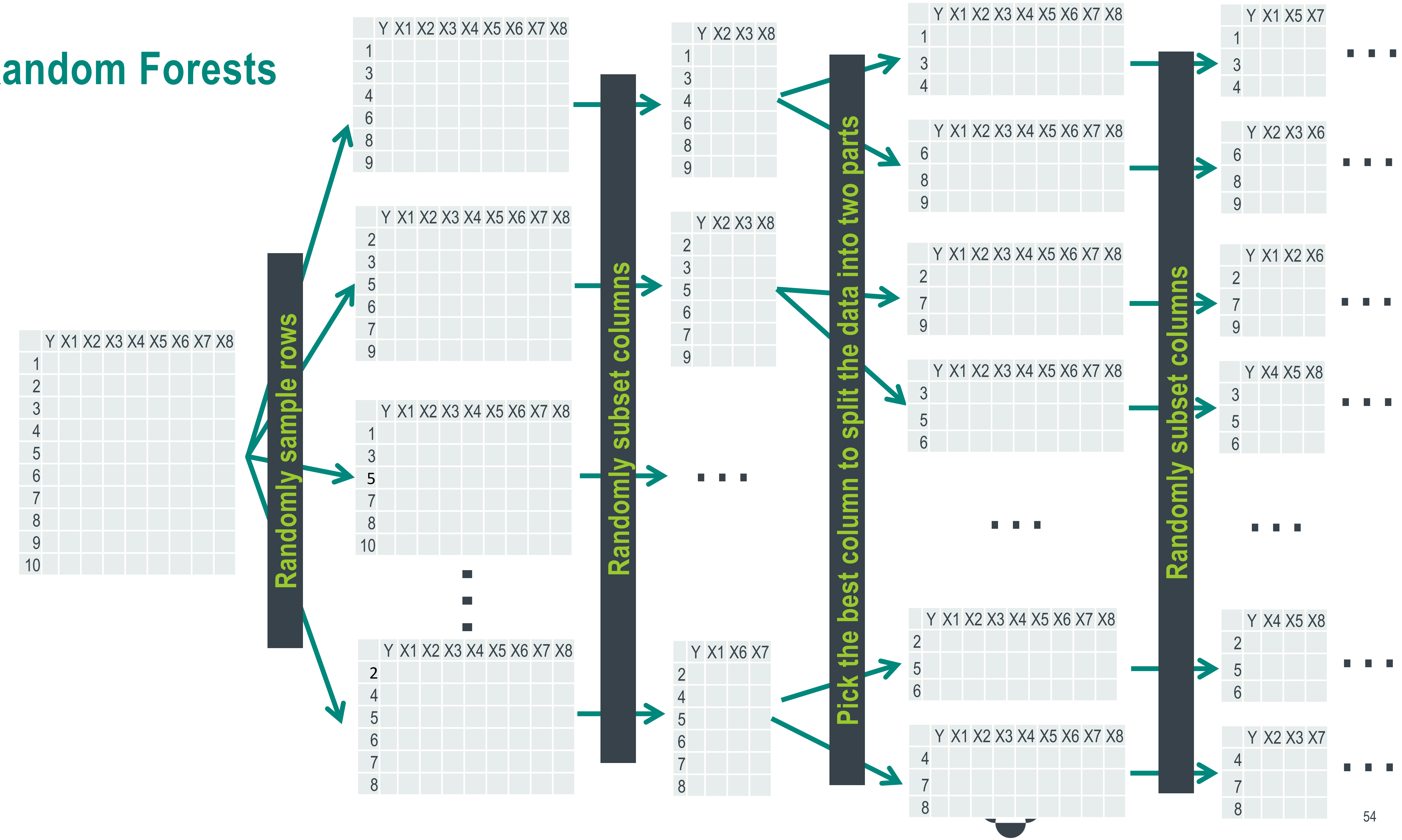
For each sample, fit a model (e.g., tree)

Prediction: average (or plural vote) from all models

The base models should overfit the data (low bias, high variance), then averaging reduces variance while maintaining low bias



Random Forests



Public

Out-of-Bag (OOB) Data

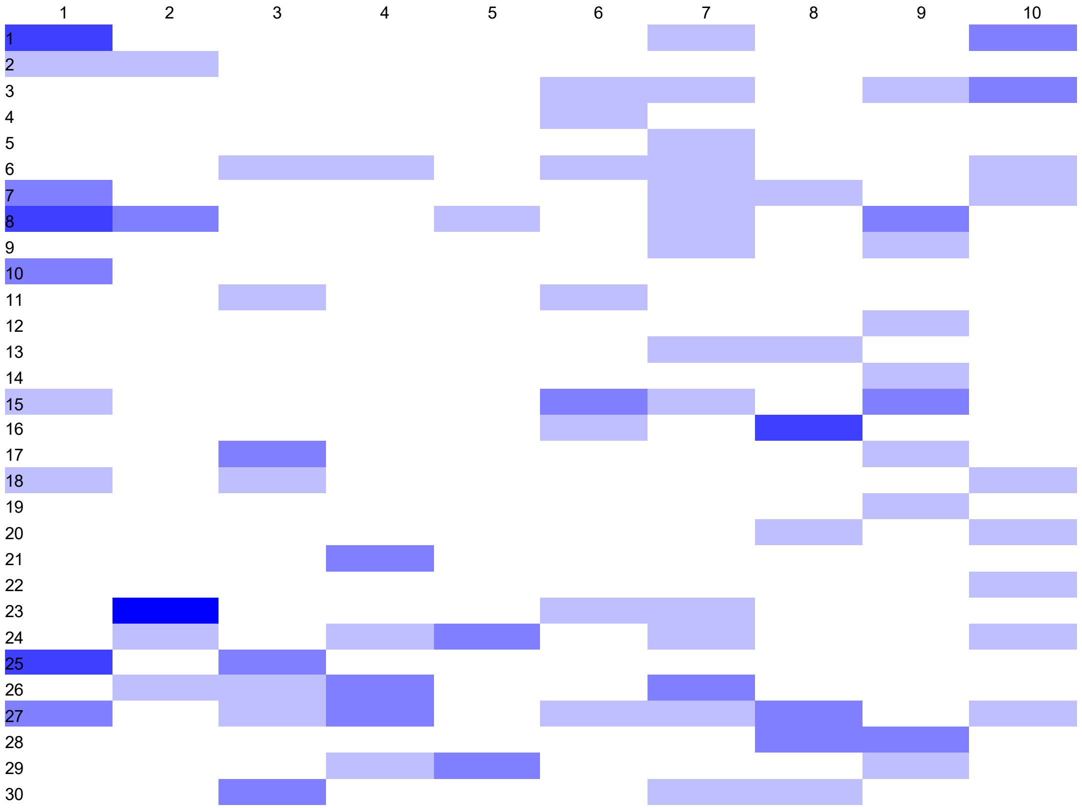
Each data point has probability $(1-\frac{1}{n})^n$ (approximately e^{-1} or about 37%) of being excluded in a bootstrap sample

These "out-of-bag" (OOB) data can be use as legitimate test data for the tree grown on the bootstrap data

Aggregating these OOB prediction, we can get very good out-of-sample prediction error estimates without external validation such as CV

They can be further manipulated to assess variable importance

Different Bootstrap Samples

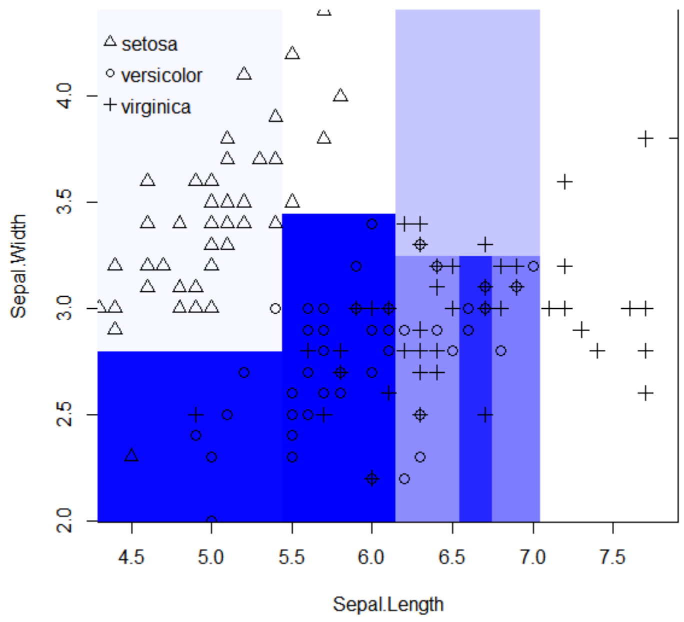


Nearest Neighbor Classifier

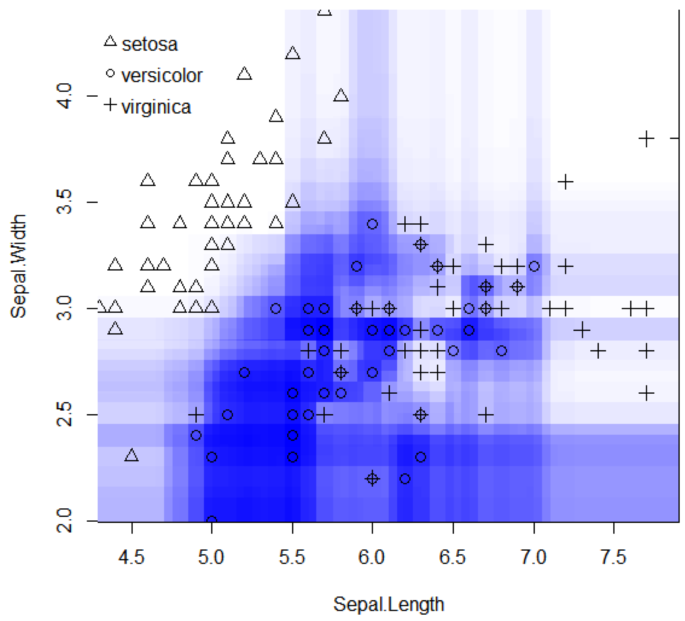
Terminal nodes in a decision tree represent groups of similar data, with sizes of neighborhoods decided from training data (hyper-rectangular regions)

RF averages terminal nodes from many trees, so neighborhoods are varied -- “smooth out” the crude neighborhoods of a single tree

CART



RF



Public

What Controls RF's Model Complexity?

- Viewed as adaptive weighted NN, increasing number of trees makes weights "smoother"
- Sizes of neighborhoods can also be an indicator of model complexity
- Given the same data, smaller trees ⇔ larger neighborhoods
- Implication on tuning RF:
  - Use **`mtry`** to balance correlation and strength
  - A larger **`nodesize`** forces the algorithm to produce smaller trees, thus larger neighborhoods
  - A smaller **`sampsize`** also induces smaller trees, also make trees more diverse (but should be used with larger number of trees)

Public

RF vs. Boosting

RF

- Trees are independently grown
- Use randomness to get diverse trees
- Usually grow large trees
- Number of trees is not a tuning parameter
- Model size can be huge due to large trees

Boosting

- Trees are grown sequentially
- Each tree tries to correct previous mistake
- Keep each tree relatively small
- Number of trees should be tuned
- Model size is usually small

Public

Tree-Based Methods: Summary

Single tree are intuitive and easy to interpret, but usually suffers in prediction accuracy

Ensemble methods can greatly improve prediction accuracy, but give up interpretability

Bagging and Random Forests combine independently generated models, reducing variance of low bias models

Boosting iteratively makes improvements, combining a sequence of models with low variance and reduce bias

RF is generally more robust (no worry of overfitting)

Boosting can be more efficient for very large data

Support Vector Machine: Maximum Margin Classifier

Given linearly separable data, there are ∞ solutions

$$f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^t \boldsymbol{x} + b)$$

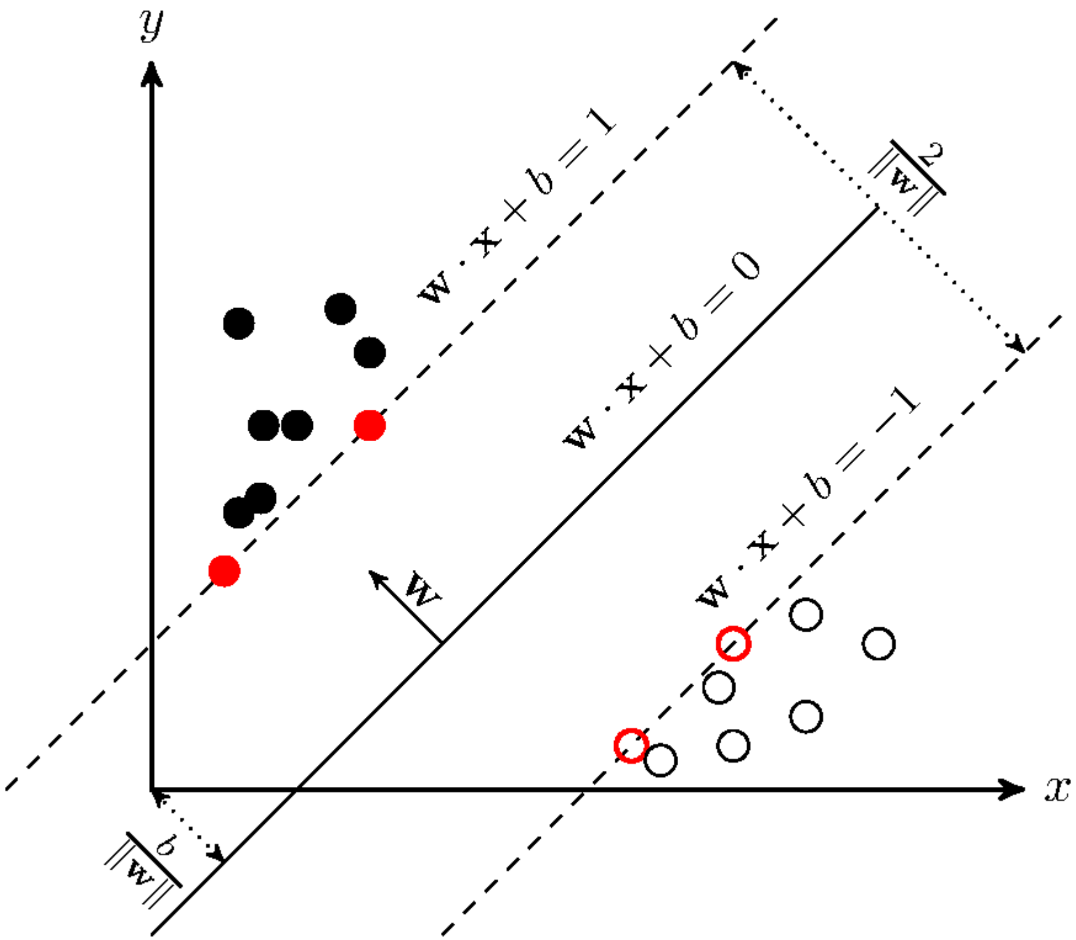
Margin is the distance around the decision plane to the nearest data points (the larger the better)

Purely mathematical formulation. Solution is the Support Vector Machine (SVM):

$$\hat{f}(\boldsymbol{x}) = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i^t \boldsymbol{x} + b)$$

The nearest data points to the decision plane have non-zero α_i and are called the **support vectors**

For non-linearly separable data, allow small fraction of data to be within the margin (soft margin classifier)



Nonlinear SVM: Kernel Trick

One strategy to cope with nonlinear structures in data: do nonlinear transformations and apply linear method on the transformed data

Linear SVM: $f(x) = \text{sign}(w^t x + b)$

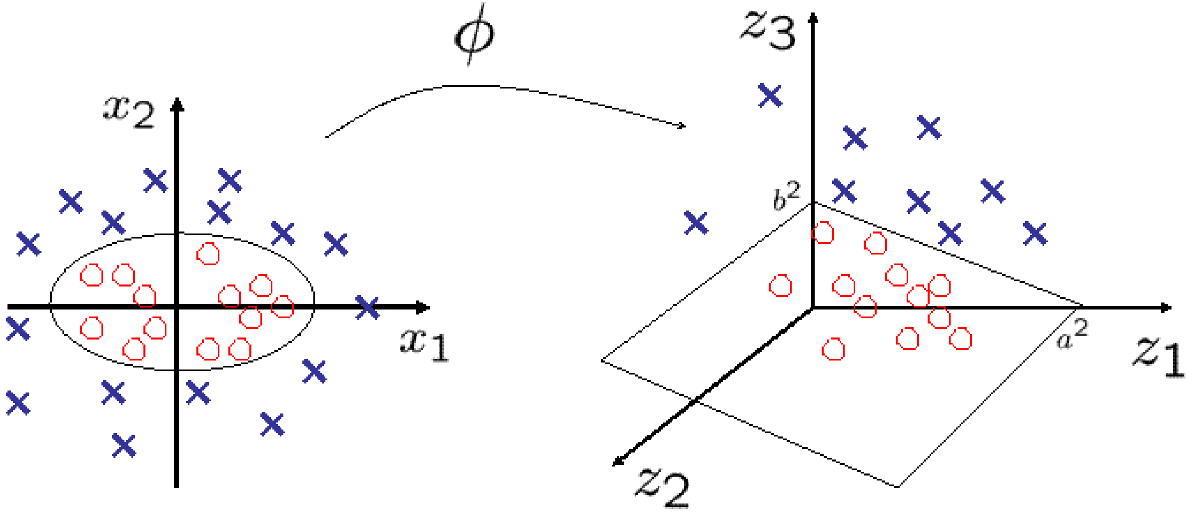
If $\phi(x)$ is a nonlinear function, a kernel is a function such that $K(w, x) = \langle \phi(w), \phi(x) \rangle$

Nonlinear SVM: $f(x) = \text{sign}(\langle \phi(w), \phi(x) \rangle + b)$

$$= \text{sign}(K(w, x) + b)$$

The "trick": we just need to know $K(w, x)$, not $\phi(x)$

RBF kernel: $K(w, x) = \exp(-\gamma \|w - x\|^2)$



$$\phi : (x_1, x_2) \longrightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 = 1 \longrightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = 1$$

Kernel Methods: Beyond SVM

The kernel trick is not tied to SVM--- it can be applied to basically any linear method

Re-express the objective function in terms of inner product, then "kernelize" by replacing them with kernel

Ex.: Ridge Regression:

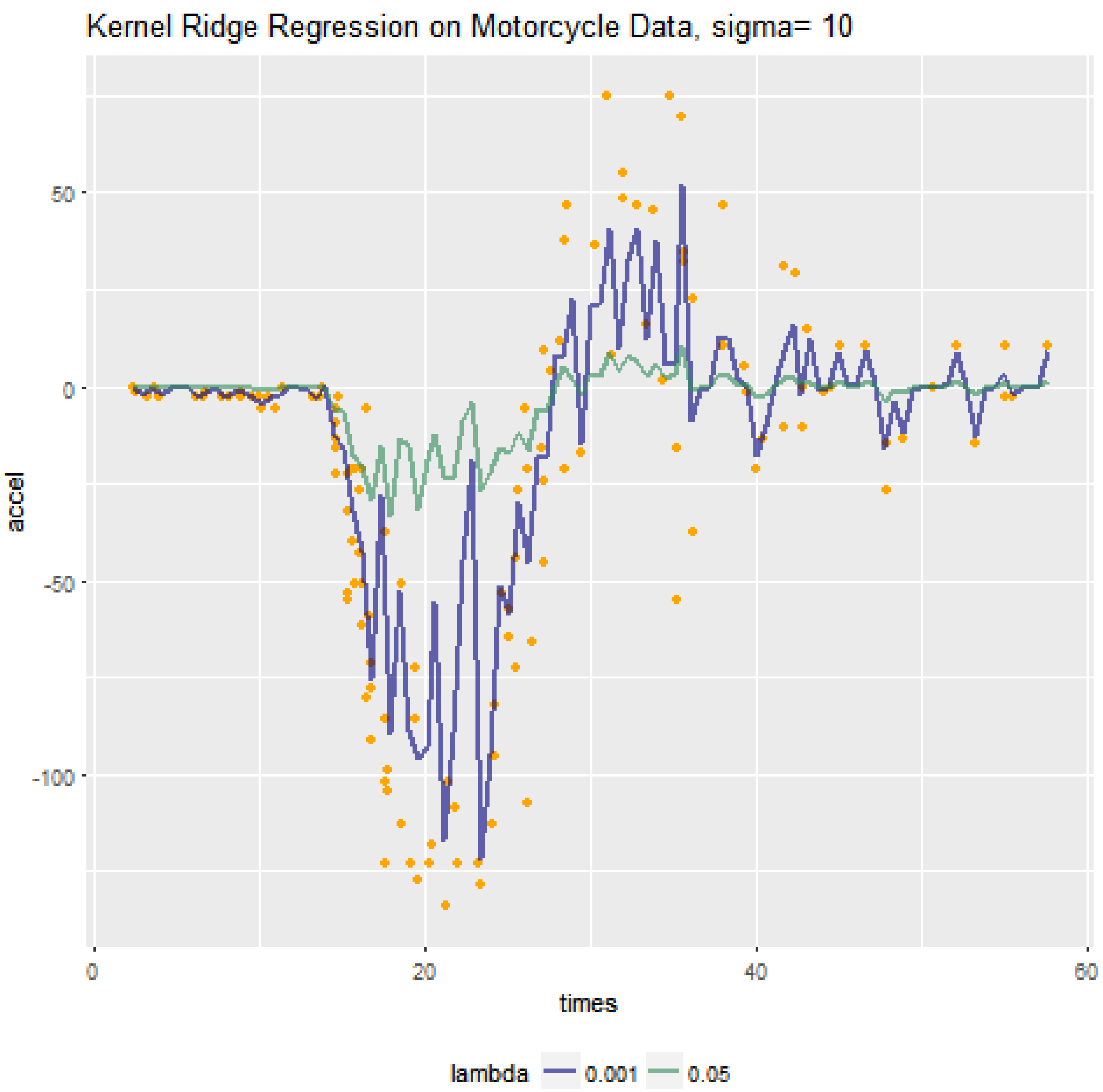
$$\hat{\beta} = (X^t X + \lambda I)^{-1} X^t y = X^t (X X^t + \lambda I)^{-1} y$$

For new data x, $\hat{y} = \hat{\beta}^t x = y^t (X X^t + \lambda I)^{-1} X^t x$

Let $K = \left[\phi\left(x_i, x_j\right)\right]_{n \times n}$ and $\kappa = [\phi(x_i, x)]_{n \times 1}$ where $\phi(u, v) = u^t v$, then $\hat{y} = y^t (K + \lambda I)^{-1} \kappa$

Applying the kernel trick and we get Kernel Ridge Regression

Same can be done with many other linear methods



Artificial Neural Networks

Model Y as nonlinear function of compositions of nonlinear functions of linear combinations of X

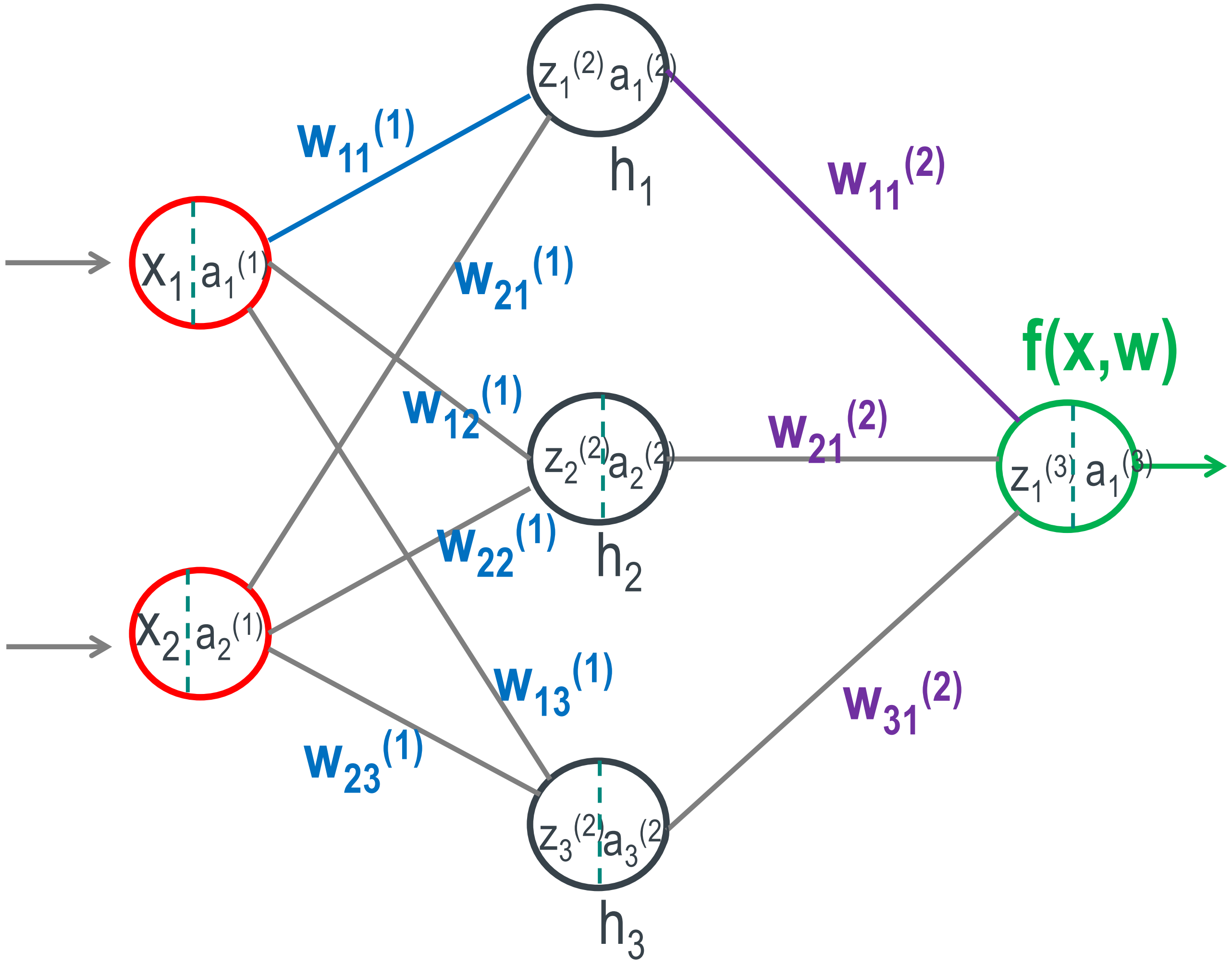
Connection from the predictor variables (inputs) to each hidden node is basically a logistic regression

From all nodes in the first hidden layer to each node in the next layer is again logistic regression

The logistic function is called the **activation function** (there are other functions commonly used)

Hornik (1991) proved that ANN with single hidden layer can approximate any continuous function arbitrarily well, given enough data and hidden nodes

(Logistic regression is a network with no hidden layer)



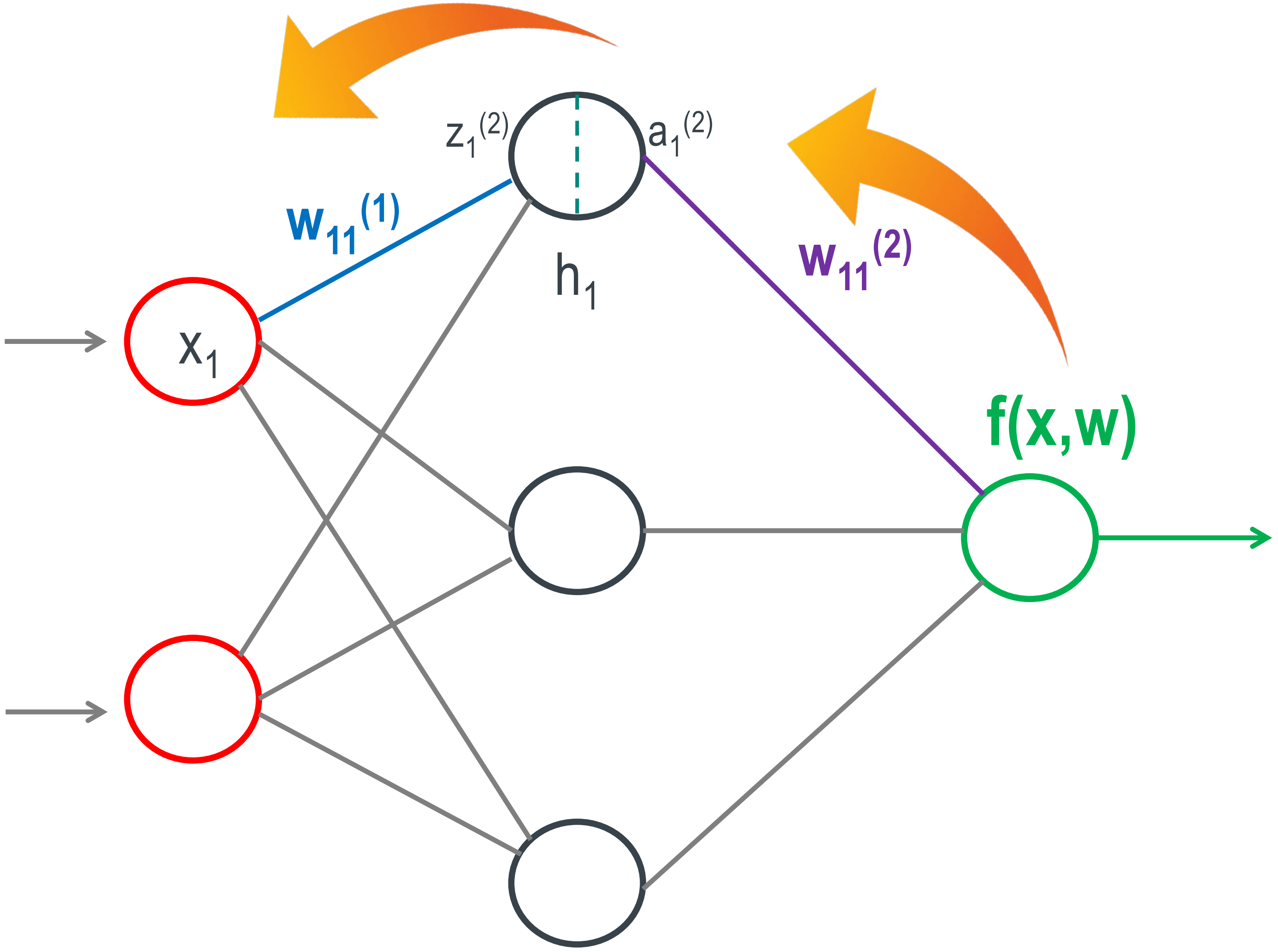
$$z_j^{(k)} = \sum W_{ij}^{(k-1)} a_i^{(k-1)}; \qquad a_j^{(k)} = h\left(z_j^{(k)}\right)$$

Training ANN: Back-Propagation

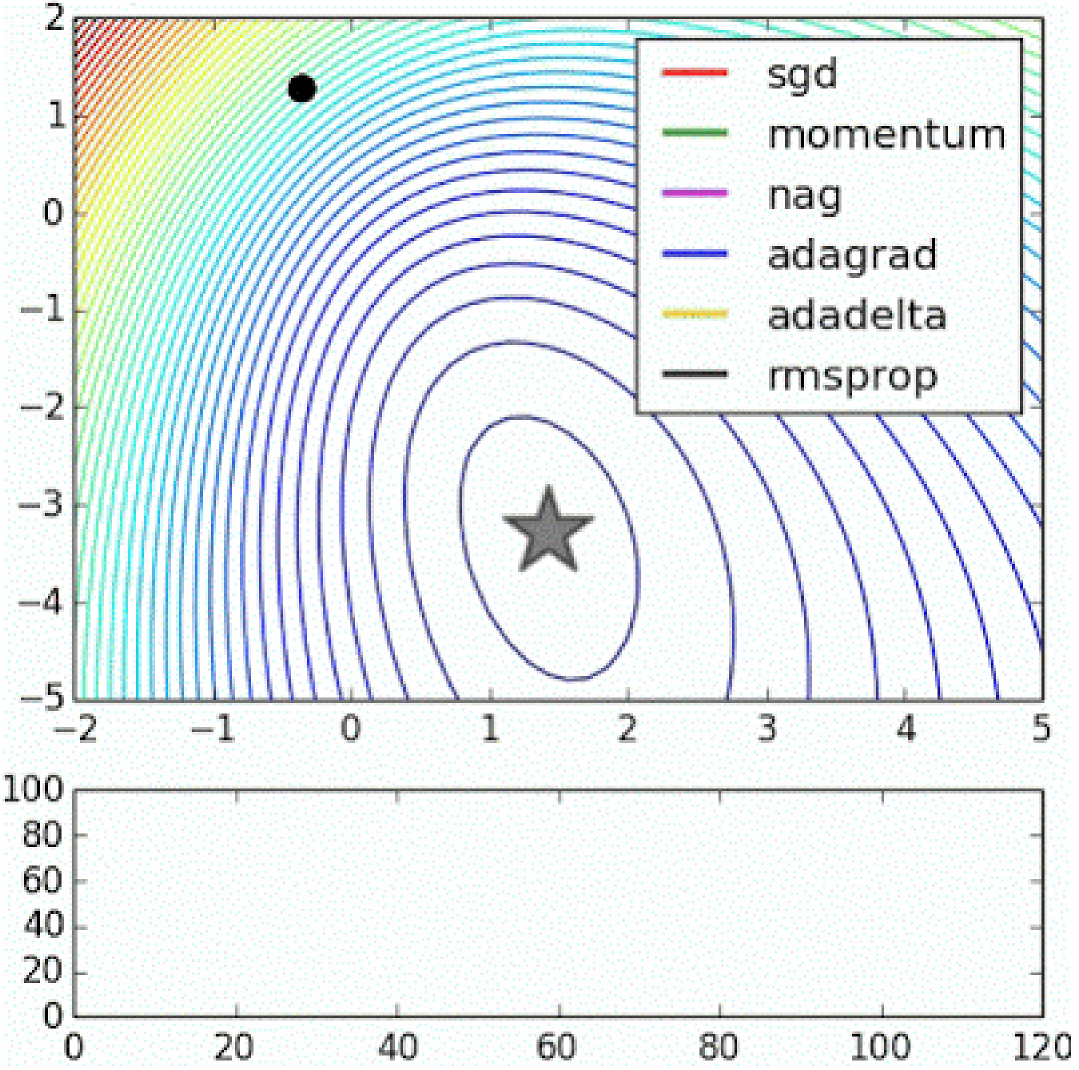
BP == chain rule for computing the derivative in composite function

$$\frac{\partial f(\mathbf{x},\mathbf{w})}{\partial w_{11}^{(1)}} = \frac{\partial f}{\partial a_1^{(2)}} \times \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \times \frac{\partial z_1^{(2)}}{\partial w_{11}^{(1)}}$$

Gradient descent with BP is done for a pre-determined number of iterations ("epochs")
"Convergence" is not a well-defined concept in this context. If more iterations are needed, the process can be re-started from where it left off.



"Modern" Neural Networks



So called "Deep Learning", for the most part, is just a collection of tricks that improve on the old ANN:

Minibatch (Stochastic Gradient Descent): feed only a small subset of data at each update

Dropout: randomly give 0 updates for some parameters at each step (regularization)

ReLU activation: alleviate vanishing gradient

Batch normalization: allow faster learning

Momentum: give weights to gradient from previous step

Many variations on gradient descent

Deep Learning (ANN with >1 hidden layer): made possible/easier with the "tricks" above

<u>These toy examples can be instructive</u>

Public

Tuning Neural Networks

- Number of hidden layers
- Number of nodes in each layer
- Choice of activation function
- Regularization (Weight Decay)
- Dropout / batch normalization
- Minibatch size
- Number of epochs
- Optimization algorithm and its parameters (e.g., learning rate, momentum, etc.)

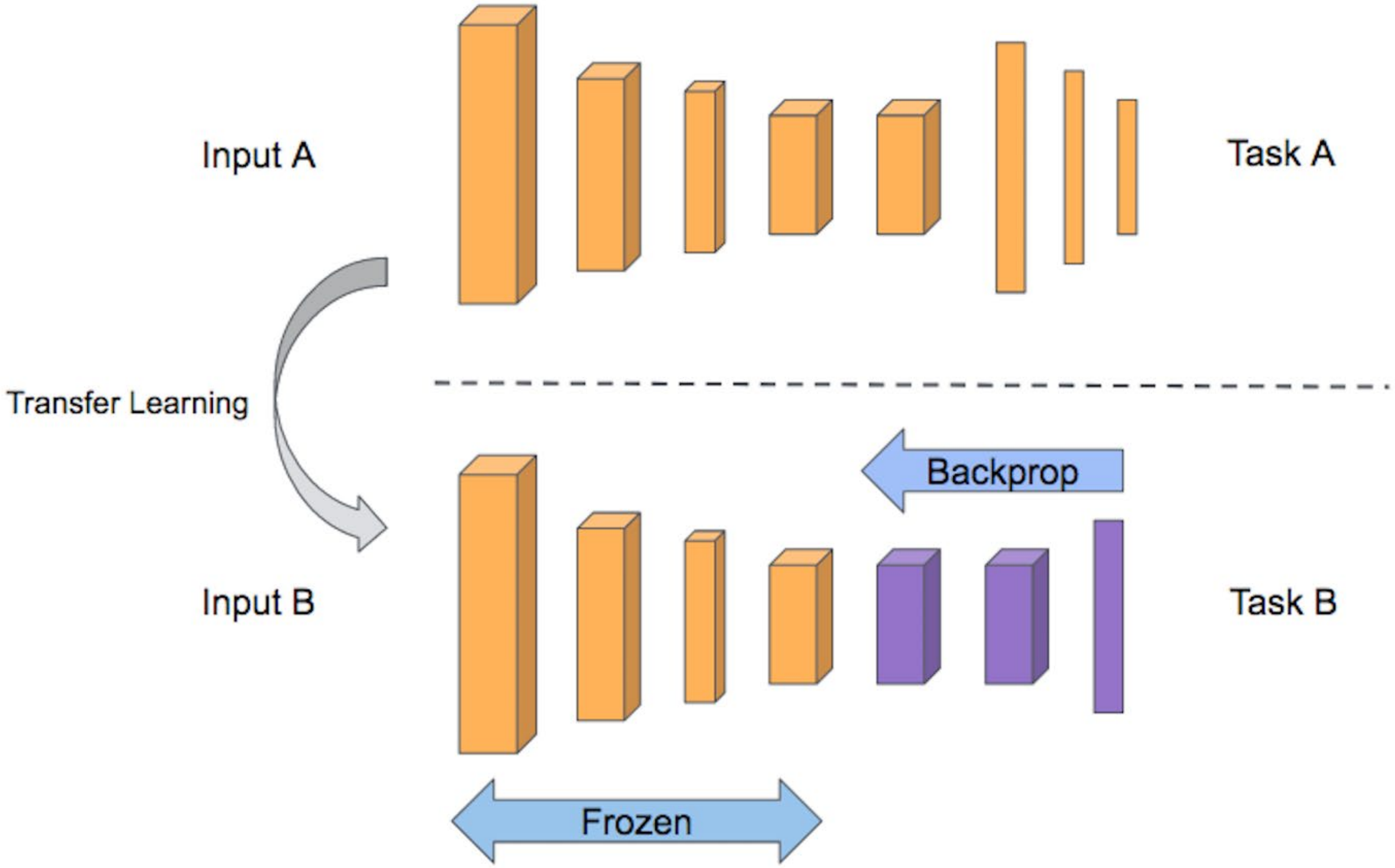
ANNs are highly flexible models with too many tuning parameters to make grid search and cross-validation practical or even feasible.

Due to the flexibility (especially with more hidden layers), they have large appetite for data. They are unlikely to perform better than other methods with less than enormous amount of data.

Early stopping (monitor performance of validation set as training is underway and stop when it starts to degrade) is a good alternative.

Public

Transfer Learning: Update a Well-Trained Model with New Data



How to Choose Methods

How much time do you have?

“Small” datasets are unlikely to be amenable to highly flexible methods

Flexible methods may require significant computational resource on “large” datasets

The need to cross-validate can exacerbate the problem

More knowledge of the data and some understanding of the methods can help guide the choice

For large data, experiment on smaller subset of the training set to narrow down choice of methods

No Free Lunch (NFL) Theorem (Wolpert, 1996): No method (algorithm) is best for all possible data sets

Corollary: For every method (algorithm) there exists a dataset for which the method is the best

Lessons Learned from Running Hundreds of Kaggle Competitions

https://mlconf.com/sessions/lessons-learned-from-running-hundreds-of-kaggle-co/

Public

Translation of Terms

Machine Learning	Statistics
Train an algorithm	Fit a model
Input, features	Independent (predictor) variables
Output	Response, outcome
Weights	Model parameters
Bias	Intercept
Epochs	1 epoch = # of Iterations to use all training data for update
Inference	Prediction
Sigmoid function	Logistic function
Cross entropy loss	Negative log likelihood

Software

MachineLearning CRAN Task View

Framework: caret, parsnip, mlr, mlr3, rattle, h2o

Deep learning: keras/tensorflow (call Python)

Framework: scikit-learn

Boosting: xgboost, lightgbm, ...

Deep learning: Tensorflow/keras, PyTorch, CNTK, Mxnet

Machine Learning for Statisticians, Part IV: **Model Inference and Interpretation**

Junshui Ma

Public

Outline

- What is Machine Learning ? (Junshui Ma) ~ 45 minutes
- Supervised Learning Workflow (Andy Liaw) ~ 30 minutes
- Break ~ 10 minutes
- Supervised Learning Methods (Andy Liaw) ~ 60 minutes
- Break ~ 10 minutes
- Model Interpretation, including Feature Selection and Other Topics (Junshui Ma) ~ 45 minutes

Outline

- Model Multiplicity

- Use of Trained Models ("Inference")
  - ➢ To predict the outcome Y of **new samples**
  - ➢ To better understand the relationship between **X** (predictors) and Y (outcome)
    - – Feature ranking or selection
    - – Partial dependence between y (outcome) and x_i (one of the predictors)

- Interpretability

Notes:

– Models in this section are by default models obtained via *supervised learning*.

– {"Learning", "Training", "Fitting"}, {"Feature" , "Variable", "Predictor"} are interchangeable in this section.

Model Multiplicity

- The concept of "Model Multiplicity"
  - Based on goodness-of-fit test and/or residual check, etc., Model 1, 2, and 3 can fit the data equally good.
  - However, Model 1, 2, and 3 produce very different interpretations of the relationship between **X** and Y.

X

Model 1: $Y = x_1 + x_2 + x_3 x_4 + x_5$

Model 2: $Y = x_1 + x_2 x_3 + x_4 + x_5$

Model 3: $Y = x_1 x_2 + x_3 + x_4 + x_5$

Y

$\Rightarrow$ Opinion Leaders

- How ML community handles this issue?
  - Models are generally complex with (very) large number of parameters
  - The joint effect of all parameters matters, instead the value of each parameter, because ML does not assume that a "true model" exists.
  - The models that best predicts Ys of new Xs are **the best models to use.**

$\Rightarrow$ Competition Winners

Use of Trained Models ("Inference")

- To predict the outcome Y of **new Xs**

- To better understand the relationship between **X** (predictors) and Y (outcome)
  - Feature ranking or selection
  - Partial dependence between y (outcome) and x_i (one predictor)

Public

Prediction with Trained Models

Machine Learning: Prediction

Training Data $\{X_i, y_i\}_{i=1 \ldots n}$

New Data $\mathbf{X}_{new}$ → Trained Predictive Model → Predicted y_{new}

Statistics : Model Inference

Data $\{X_i, y_i\}_{i=1 \ldots n}$

non-negligible assumptions

$Y = f(\beta_1 \mathbf{x}_1 + \ldots + \beta_n \mathbf{x}_n)$

From β_1, effect of $\mathbf{x}_1$ is … with a p-value of …

Notes to statisticians:

- Individual model parameter is not used directly
- It is "personal"
  - Both $\mathbf{X}_{new}$ and y_{new} are about a case/patient/sample/...
  - It is possible to calculate group effect from personal results *numerically*.
- Uncertainty
  - No distribution assumption ⇒ No inherent parametric uncertainty measurements (e.g. p-value, 95% CI)
  - It is possible to get uncertainty measure using some non-parametric procedures

Feature Ranking/Selection Overview

- Why feature ranking/selection ?
  - Improving the prediction performance

    e.g. Improving the generalization by de-selecting noisy/non-informative features before training
  - Producing faster and more cost-effective prediction capability
  - Better understanding of the underlying natural process

    e.g. Understanding a treatment's Mechanism of Action
  - Finding biomarkers

- An area claimed by both Machine Learning and Statistics
  - Both communities contributed ideas and methods
  - The same method can be implemented differently between these two communities

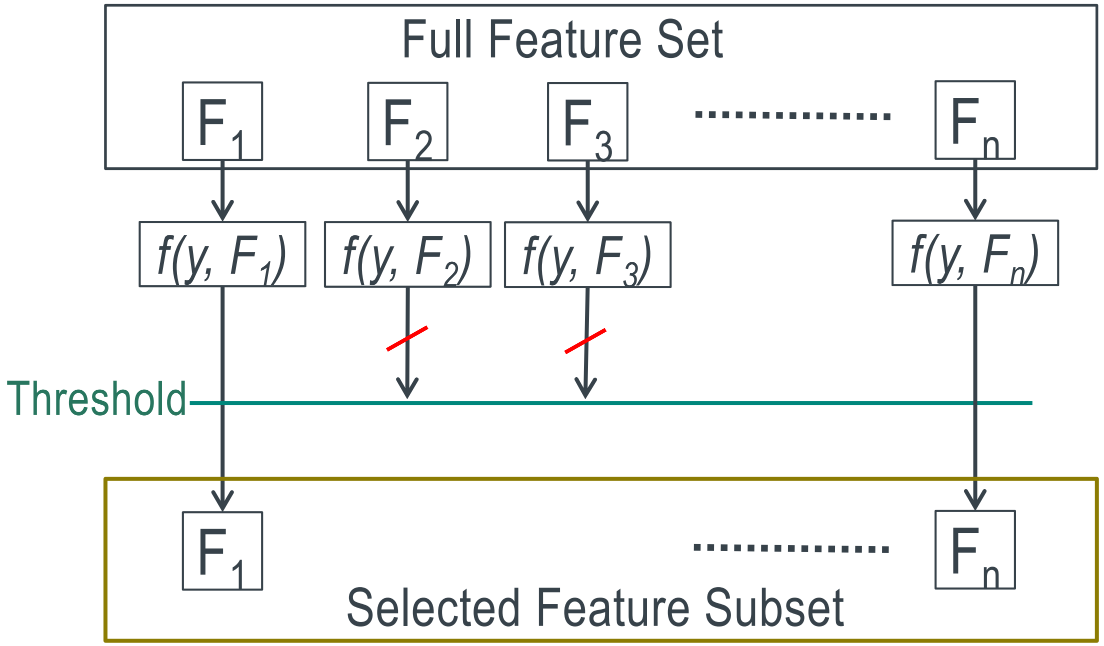
Feature Ranking/Selection Methods

- **Filter Methods:** Ranking features by their individual measures/scores
- **Wrapper Methods:** Formulating feature selection as an optimal feature subset search
- **Embedded Methods:** Ranking or selecting features while a multiple (generalized) linear model is built
- **Add-on Methods:** Ranking feature based on the specific properties of predictive models

Note:
Filter Method is listed here for completeness and the convenience of teaching. They are generally done before the target predictive models were trained.

P Public

Feature Ranking and Selection: Filter Methods

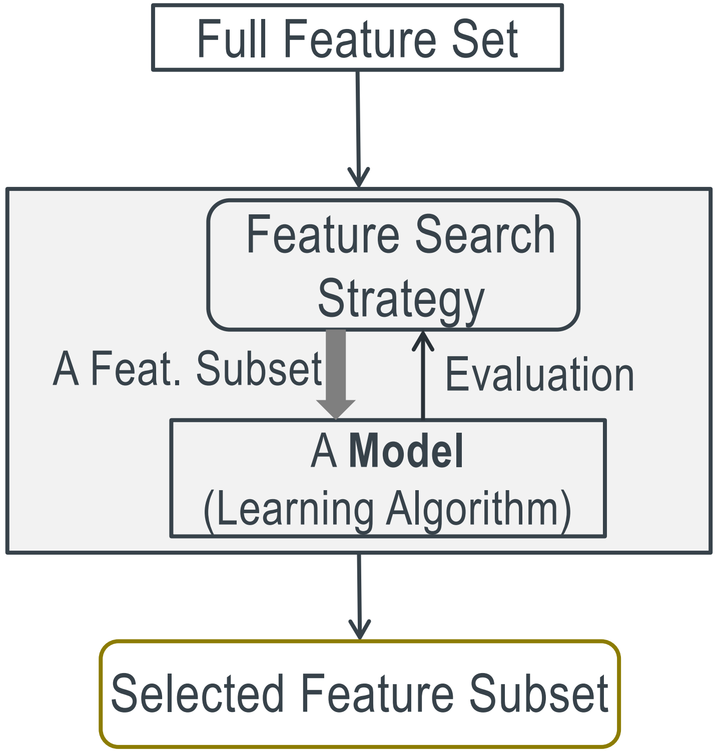


- Each feature can be evaluated by a measure/score, $f(y, F_i)$, which can be
  - correlation coefficient,
  - odd ratio,
  - information gain (IG)*,
  - Hilbert-Schmidt Independent Criterion (HSIC) **, etc.
- ***Ranking*** features by $f(y, F_i)$ or ***selecting*** features by thresholding $f(y, F_i)$.
- No model or only univariate model needed.
- Should be done in the ***training dataset***.

* IG(y, *Fi*) = $H(y) - \sum_{Fi} H(y|Fi), where\ H(y) = -\sum_y p(y)\log(p(y))$

** HSIC(*F*) : a measure using kernel trick to quantify independence of 2 random variables

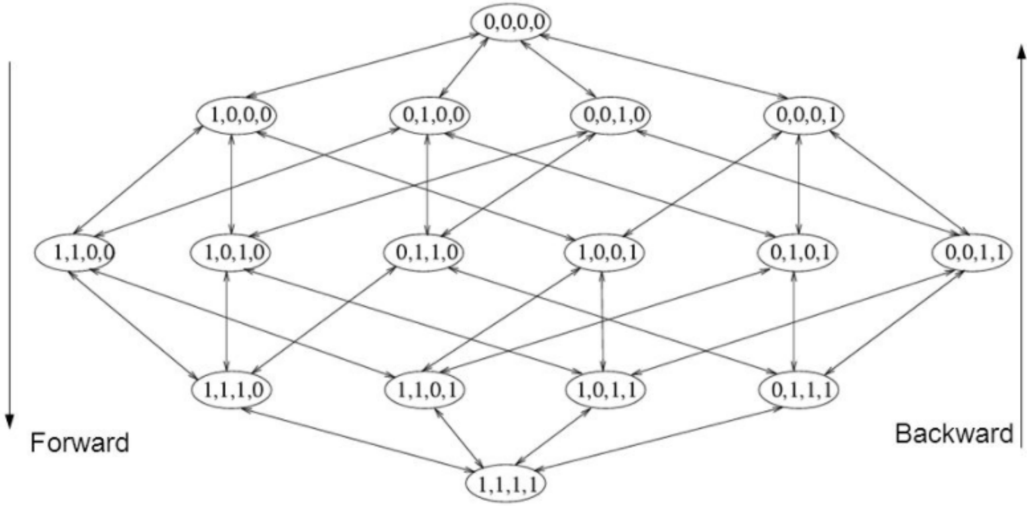
Feature Selection: Wrapper Methods



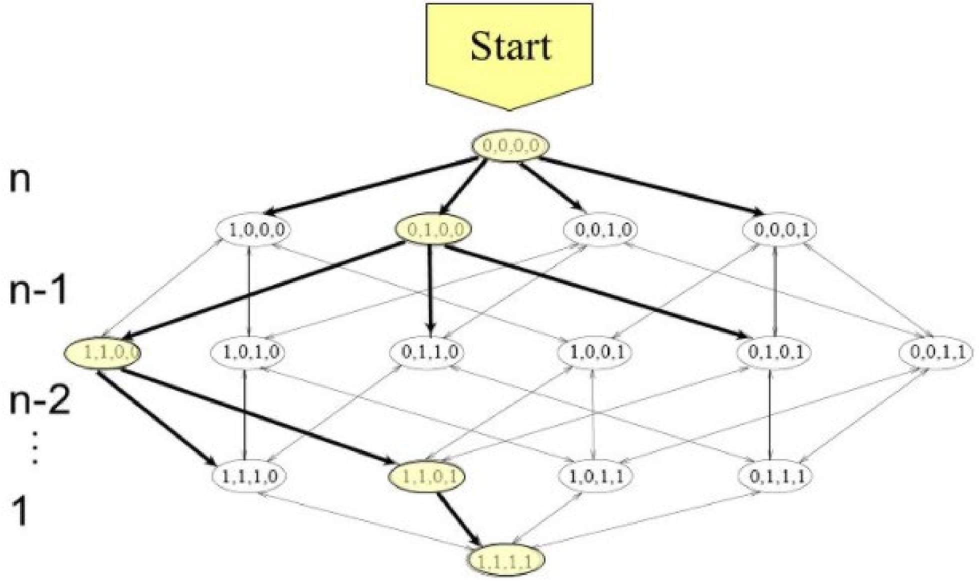
- Formulating feature selection as an **optimal search**, and "wrap" the selection around a model for evaluating different sets of features.
- Example: Best-subset Selection, Forward-stepwise Selection, etc.

Exhaustive Search for the Best Subset:

n Features $\Rightarrow$ 2^n possible Subsets (e.g. 2^{30} = 1,073,741,824)



Forward-Stepwise Greedy Search:



Public

Feature Selection: Forward-stepwise Selection

- Both ML and Statistics communities claim this method with a similar process
  - Input:
    - A set of candidate features (covariates) $\boldsymbol{S}=\{F_1, F_2, \ldots, F_n\}$, and
    - Data : $\boldsymbol{X}=[F_1, F_2, \ldots, F_n]$ and y,
    - A model $\boldsymbol{f}(\{F_{i1}, F_{i2}, \ldots, F_{ik}\})$: $\boldsymbol{X'} \rightarrow y$, where $\boldsymbol{X'}=[F_{i1}, F_{i2}, \ldots, F_{ik}]$
  - Procedure:
    1. Start with an empty model $\boldsymbol{f}(\{\})$, and a candidate feature set $\boldsymbol{S}$ with all features.
    2. For each feature F_j in the candidate feature set $\boldsymbol{S}$,
       a) Add that feature to the existing model $\boldsymbol{f}(\{F_{i1}, \ldots, F_{im}\})$ to train a new model of $\boldsymbol{f}(\{F_{i1}, \ldots, F_{im}, F_j\})$
       b) Evaluate the new model performance using a pre-specified model evaluation criterion
    3. Record the feature F_s, which produces a model $\boldsymbol{f}(\{F_{i1}, \ldots, F_{im}, F_s\})$ with the best performance
    4. If model $\boldsymbol{f}(\{F_{i1}, \ldots, F_{im}, F_s\})$ is better than existing model $\boldsymbol{f}(\{F_{i1}, \ldots, F_{im}\})$,

          Update existing model to $\boldsymbol{f}(\{F_{i1}, \ldots, F_{im}, F_s\})$, and remove feature F_s from candidate feature set $\boldsymbol{S}$,
          Repeat from Step 2.

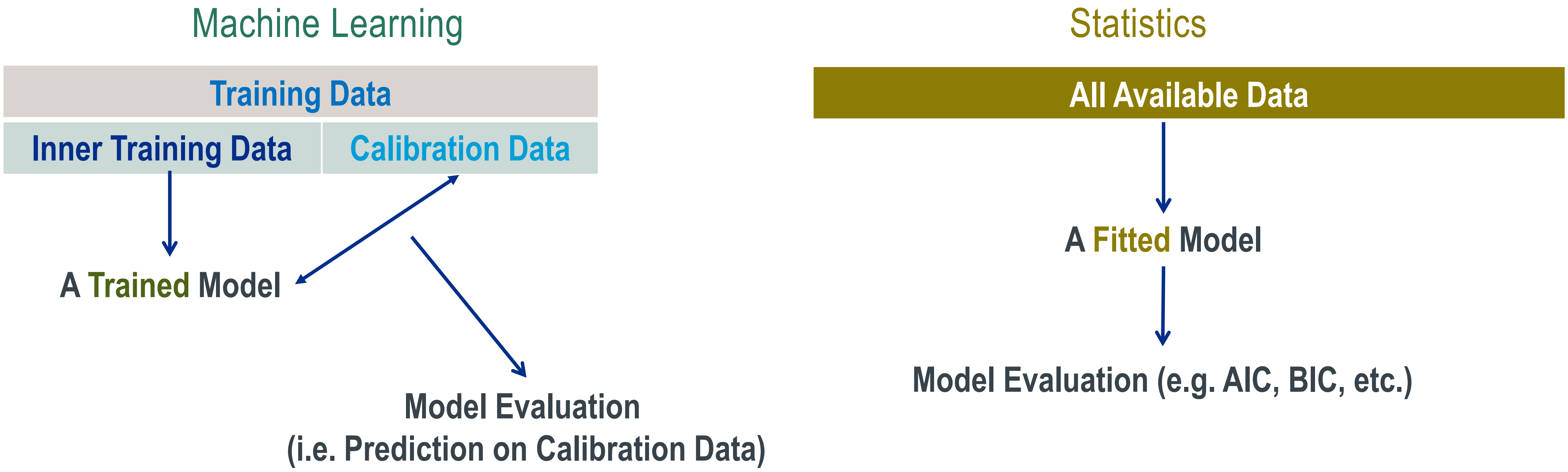
       else

          Quit.

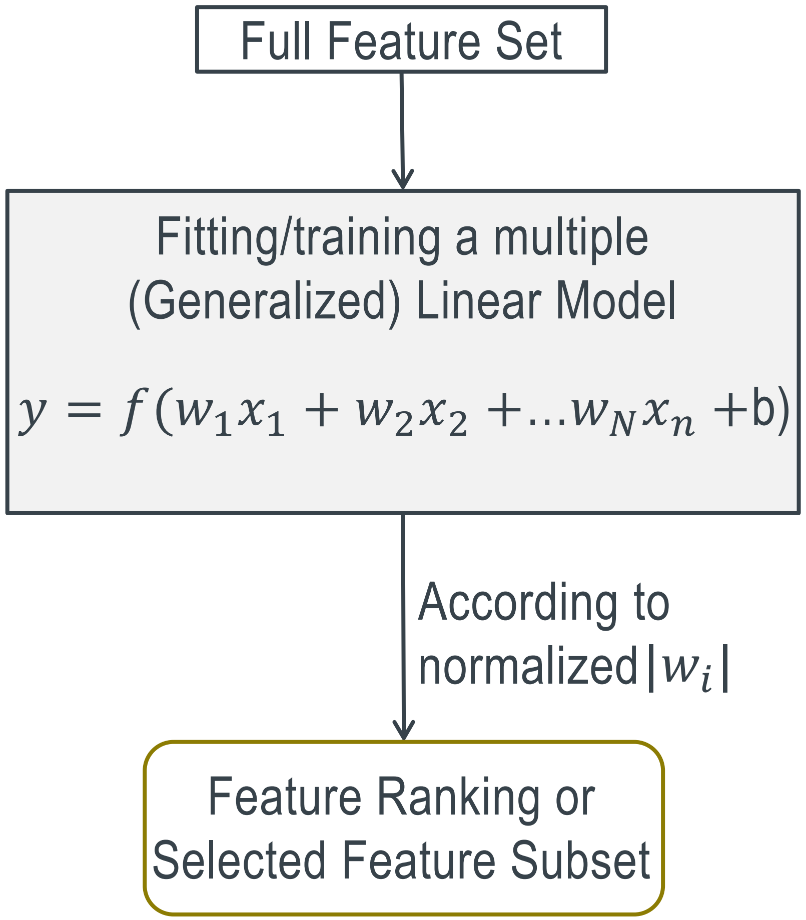
Public

Feature Selection: Forward-stepwise Selection (Cont.)

- ML and Statistics are different in two related aspects
  - Model evaluation criteria
    ML: prediction on new data (e.g. cross-validation) ⇔ Stat: AIC, BIC, Adjusted R^2, etc.
  - How to Use Dataset



Feature Selection: Embedded Methods



- Ranking or selecting features while a multiple (generalized) linear model is built.
- Example models: (Generalized) Linear Regression, LASSO, Relaxed LASSO, Group LASSO, Elastic Net, Ridge Regression, linear Support Vector Machine, etc.
- This category of methods has more statistical flavor.

Public

Embedded Methods : Regarding Feature Selection Using “p-values”

- General practice:

  For $\mathbf{E}(y) = f(w_1x_1 + \cdots)$, select factor x_1 if the p-value of coefficient $w_1 < 0.05$.

- What is p-value? (*Am. Stat. Assoc. 6 Statements on p-values (2016)*)

  #1: “*indicates how incompatible the data are with a specified statistical model*”

  #6: “*does not provide a good measure of evidence regarding a model or hypothesis*”

- Argument: p-value is just as a statistic of a normalized $|w_1|$ (e.g. ~$|w_1|/SE(w_1)$)
- Counter-argument: Why don’t directly using the underlying statistics (e.g. t-statistic, z-score)?
  - Forcing a careful selection of threshold (not automatically 0.05!)
  - Advocating a proper use of the p-value concept

Public

Wrapper Methods vs. Embedded Methods

- Forward-stepwise Selection ≈ best-subset selection
- *LASSO* = a convex relaxation of *best-subset problem*

LASSO: $$\underset{\beta}{\text{minimize}} \frac{1}{2}\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \quad \text{s.t. } \|\beta\|_1 \leq t$$

Best-subset: $$\underset{\beta}{\text{minimize}} \frac{1}{2}\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \quad \text{s.t. } \|\beta\|_0 \leq k$$

- Embedded method ≈ LASSO-like w/ different regularization

Wrapper Methods + Embedded Methods ➔ Multiple Methods

Feature Selection: Add-on Methods

- Ranking feature based on some specific forms/properties of the predictive models (i.e. model-dependent)
- Examples: **Random Forest variable importance**, **Tree-based feature ranking**, etc.
- This category of methods demonstrate more ML spirit.

Public

Feature Ranking: Variable/Feature Importance Ranking with Random Forest

- Out-of-Bag (OOB) samples of each tree in Random Forest

2 9 6 12 7 original data

bootstrap sample ↓

2 2 12 7 7 | 9 6 OOB

- Idea: If feature F_i is important, "messing-up" with it degrades prediction.

- Variable/Feature Importance defined by Permutation Test:

$$\text{Importance}(F_i) = \text{Performance}(\{F_1,\ldots,F_i^{(p)},\ldots,F_n\},y) - \text{Performance}(\{F_1,\ldots,F_i,\ldots,F_n\},y)$$

where $F_i^{(p)}$ is a *permutation* of F_i

- **Performance($\{F_1,\ldots,F_i^{(p)},\ldots,F_n\},y$)**

Given a trained Random Forest,

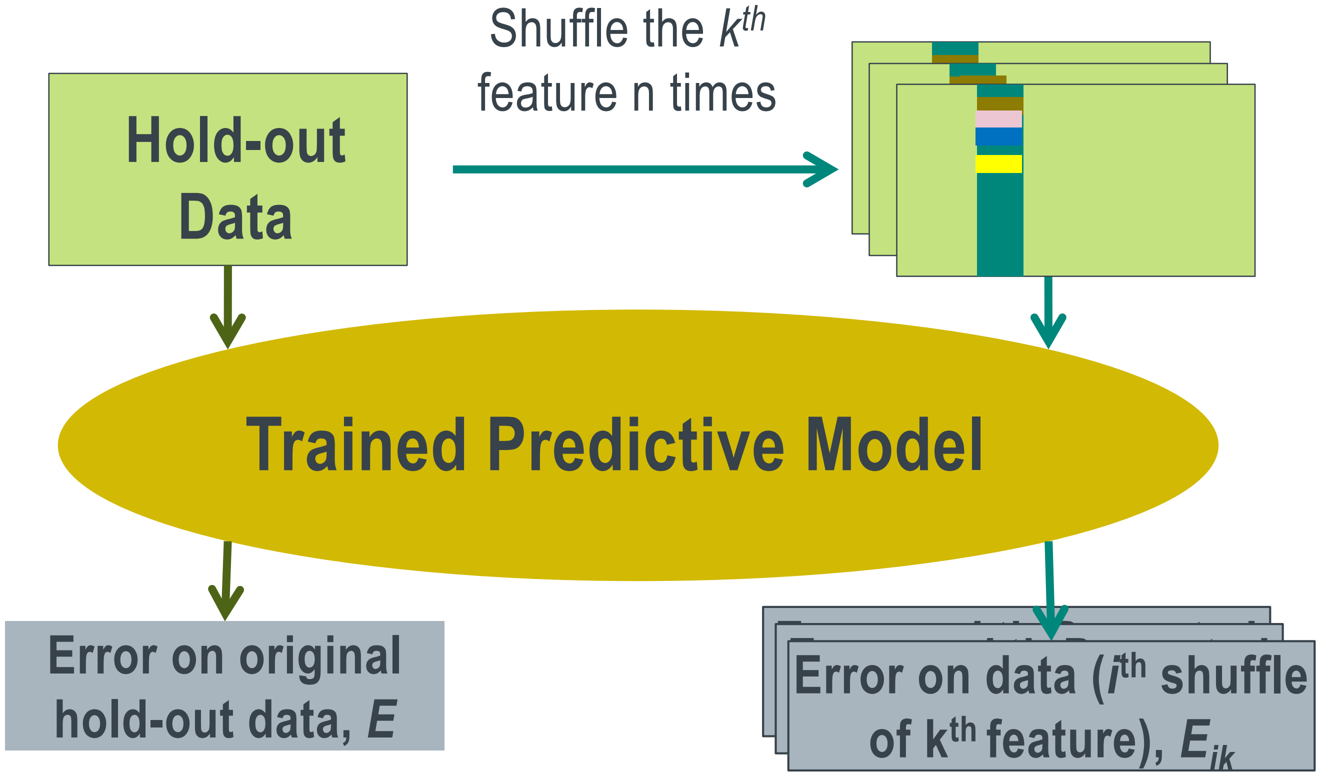
1) For each tree, calculate the prediction error of this tree on the OOB samples with the permuted feature $\mathbf{F_i^{(p)}}$,
2) Average the prediction errors of all trees as the **performance** of Random Forest**($\{F_1,\ldots,F_i^{(p)},\ldots,F_n\},y$)**.

* This approach, i.e. Mean Decrease in Accuracy (MDA), is one of several methods to rank features in Random Forest.

Public

Feature Ranking: Variable Importance Ranking (Cont.)

The Variable Importance (VarIm) idea is a generic one, and applicable to other predictive models



For the k^{th} feature at the i^{th} shuffle:

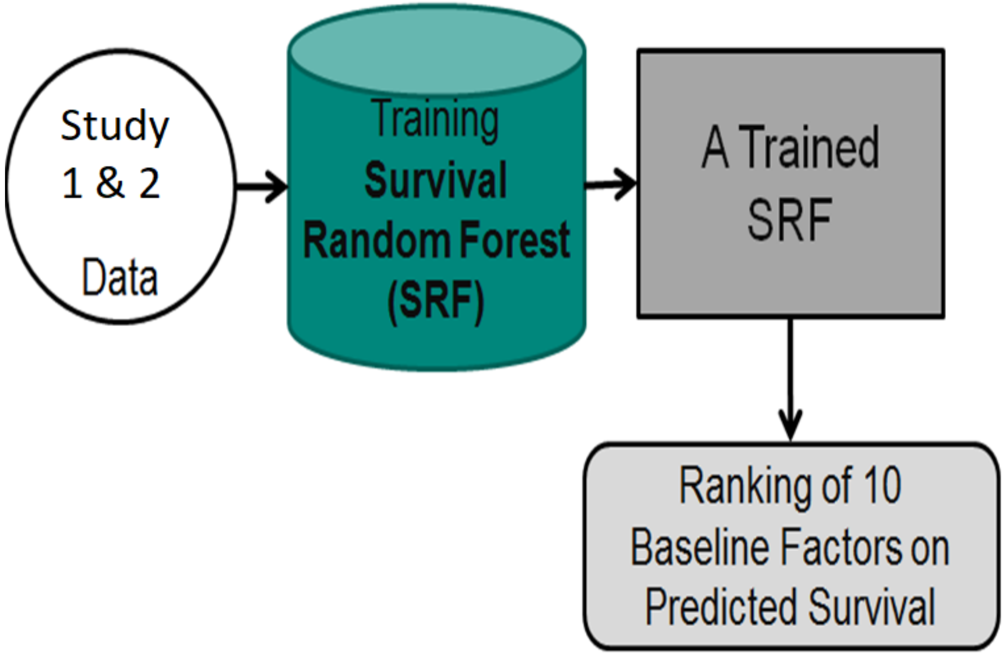
$$d_{ik} = E_{ik} - E,$$

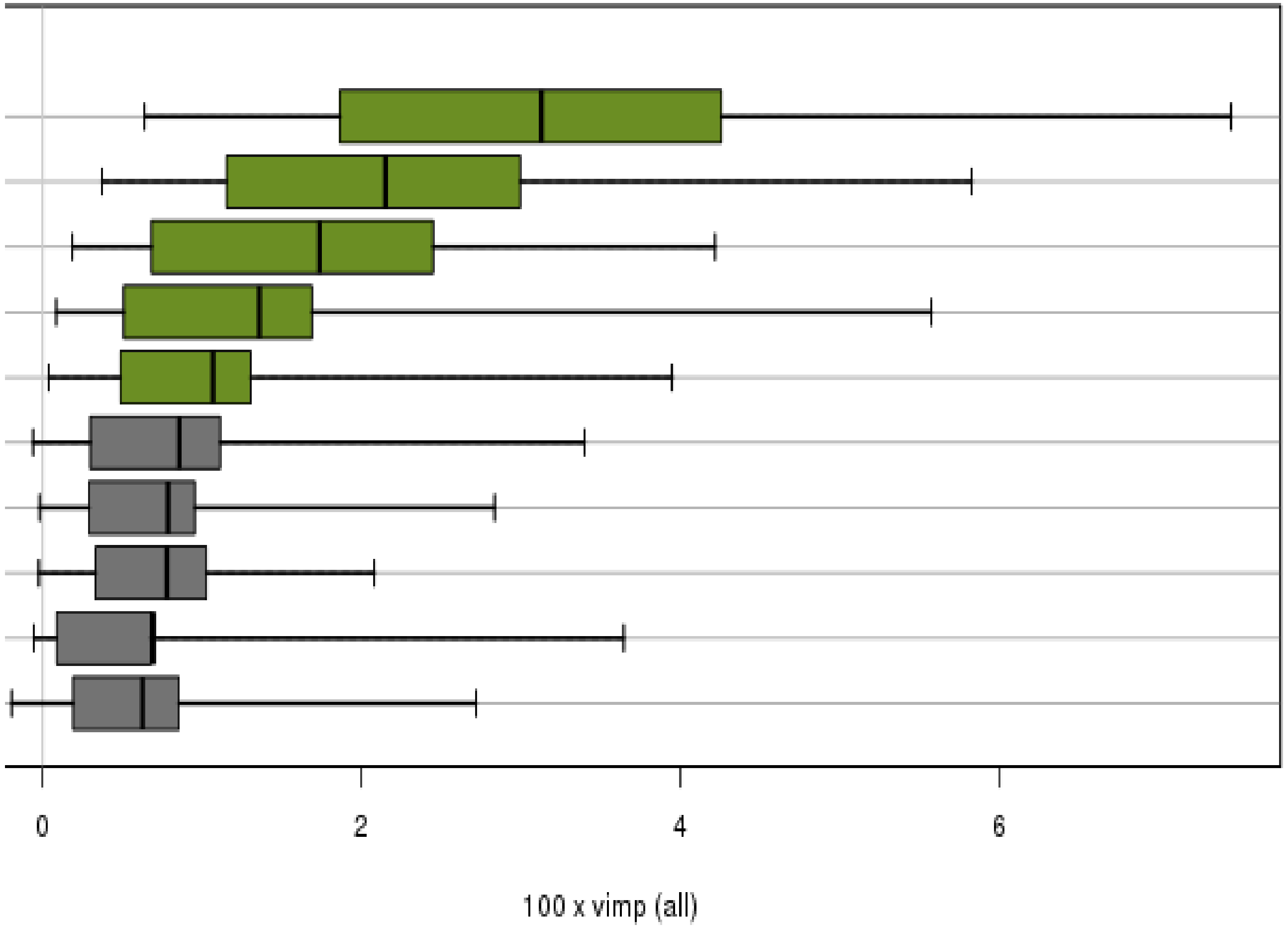
$$\mathrm{VarIm}(F_k) = \frac{mean(d_{ik})}{SD(d_{ik})}$$

Public

Feature Ranking: Variable Importance Ranking with Random Forest (Example)

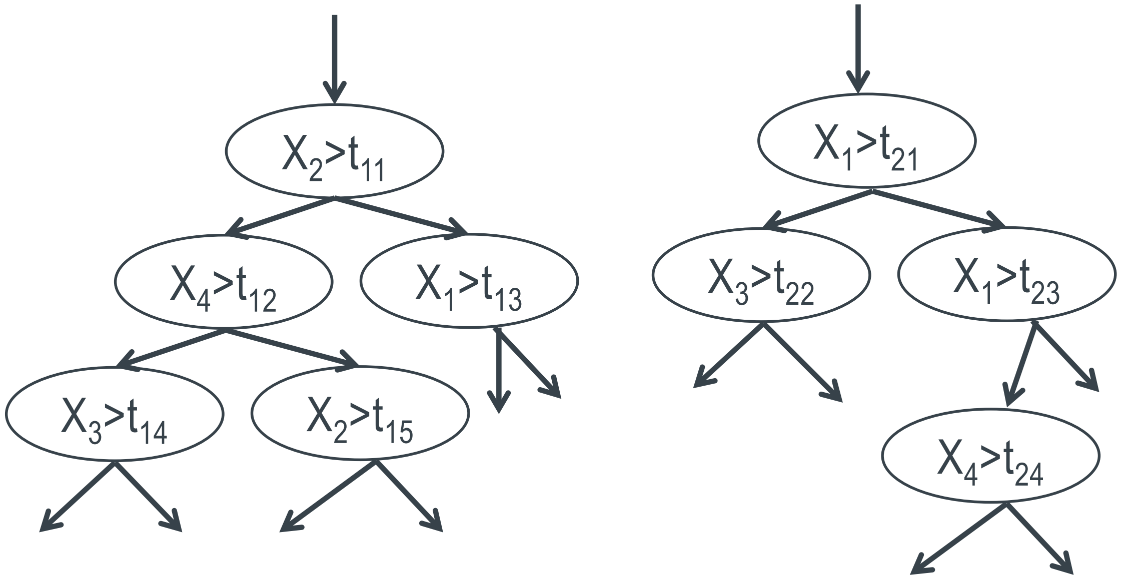
Importance Ranking of 10 Baseline Variables using a Random Forest Trained with Study 1 & Study 2 OS Data





Other Feature Ranking Methods by Tree-based Methods

- Tree-based Methods: Ensemble models with trees as the building blocks. e.g.
  - Random Forest : big and deep trees
  - Gradient Boosting Trees : small and shallow trees
- Different Approaches:
  - **Minimal Depth (Random Forest):**
    Features used at top layers of the trees are more important, because they are
    (1) selected early, and
    (2) influence more samples.
  - **Feature Importance Measurements (Gradient Boosting Trees):**
    ✓ Gain: relative % of local improvement in accuracy after splitting on feature X
    ✓ Cover: relative % of samples influenced by feature X
    ✓ Frequency: relative % of number of times feature X used



Remember: Check your models' prediction capability first (better than random guess?)

Public

Partial Dependence: *y* (outcome) vs. x_i (one of the predictors)

- Every predictive model represents a function with multiple variables $\mathrm{y} = f(x_1, \ldots, x_p)$
- The marginal relation between y and a particular variable/predictor x_i can be examined using ***Partial Dependence*** (proposed by Friedman 1999).

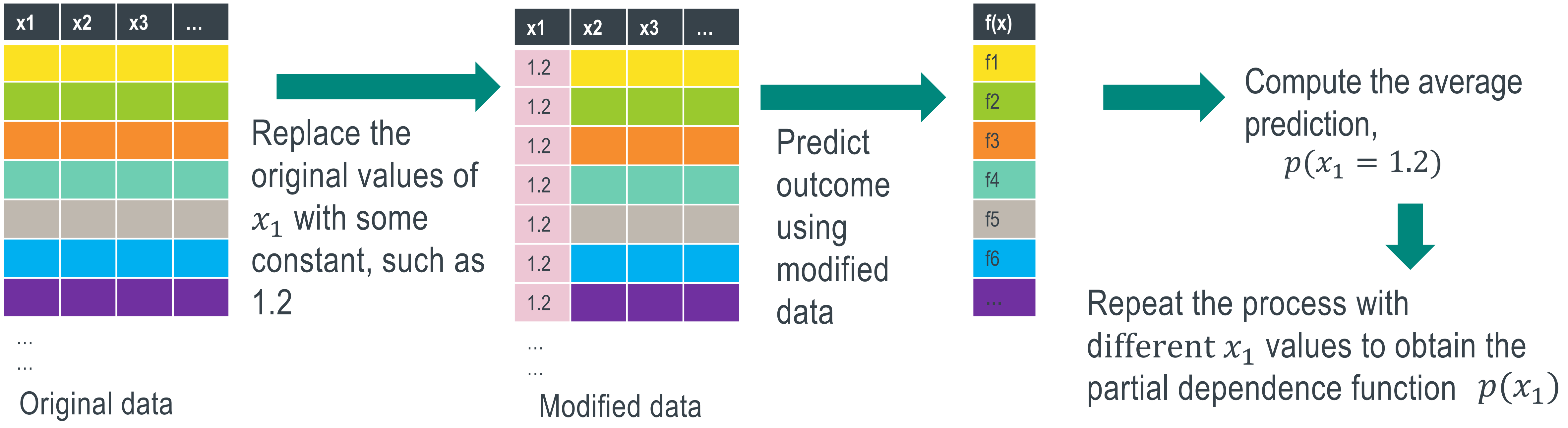
Example:
Assuming a model with 2 predicators, $\mathrm{y} = f(x_1, x_2)$,
The **partial dependence** on x_1: $\boldsymbol{p}(\boldsymbol{x_1}) = \int f(x_1, x_2)\, dx_2$
i.e., all remaining variables are integrated out

Public

Computing Partial Dependence

With actual data, the integral is approximated by averaging predictions

When calculating the partial dependence of variable x_1 for model $y = f(x_1, \ldots, x_p)$



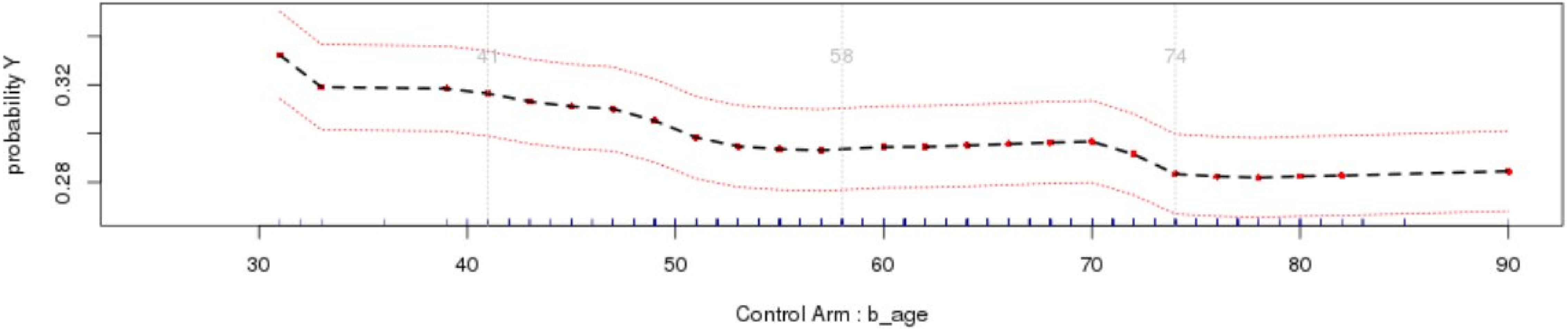
Note: A R package “pdp” for this

Partial Dependence: Example – Age vs. Response

A trained Random Forest represents the relationship between many ***baseline variables***, including ***age***, vs. ***probability of having a treatment response***

$$\Pr(Response) = f(x_1, \ldots, age, \ldots x_p)$$

The partial dependence of ***age*** vs. ***Response***, $\Pr(Response) = p_s(age)$, is shown below



Interpretability : regarding “ML Models are hard to interpret”

- Is it about data (observational vs. experimental), instead of models?
- Is it about model complexity, instead of the areas of the models (statistics vs. ML) used?
- Many complex models are somewhat interpretable, if you know how to.
  - ✓ Interpretability of complex models is a research area– Interpretable ML

Session Review

- Model multiplicity
- Using trained ML models to predict the outcomes of new cases
- Feature ranking and selection
  - 4 categories of methods
  - Jointly owned by ML and Statistics
- Partial dependence
- Interpretability of ML models

Public

Finally

- **What we learned**
  - Machine Learning vs. Statistics
  - Machine Learning concepts and methods:
    - General workflow
    - Supervised learning (RF, Boosting, SVM, Neural Network…)
    - How to use learned models (prediction, features, partial dependent function, …)

- **How to make a difference tomorrow**
  - **Innovator** = the person capable of looking at old things from a new perspective
  - **Question**: Can this be done with a Machine Learning approach?

Public