Machine learning principles and communications for material scientists

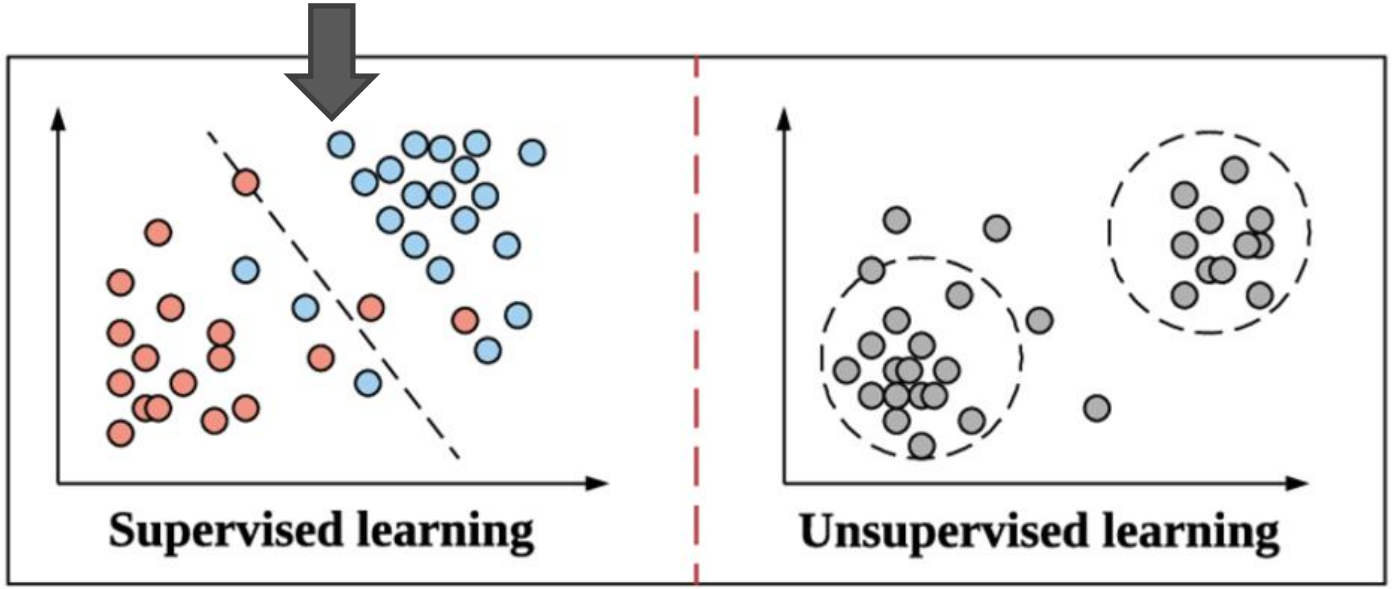
Lecture 3: Supervised learning

September 19, 2022

Sira Sriswasdi, PhD

- Research Affairs
- Center of Excellence in Computational Molecular Biology (CMB)
- Center for Artificial Intelligence in Medicine (CU-AIM)

Machine learning paradigms

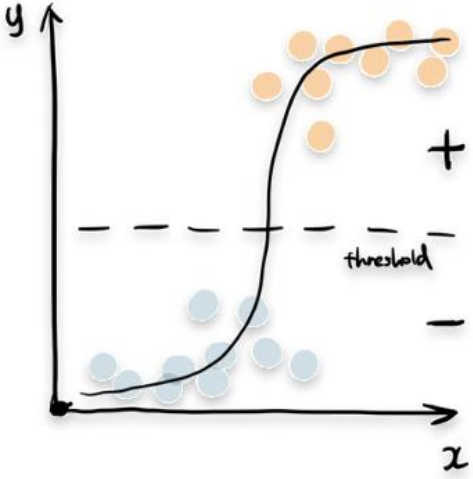


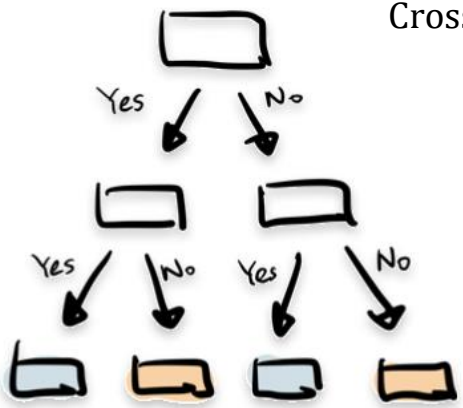
Qian, B. et al. "Orchestrating the Development Lifecycle of Machine Learning-Based IoT Applications: A Taxonomy and Survey"

- Identify robust patterns that can be generalized to new data

The cores of supervised learning

Model





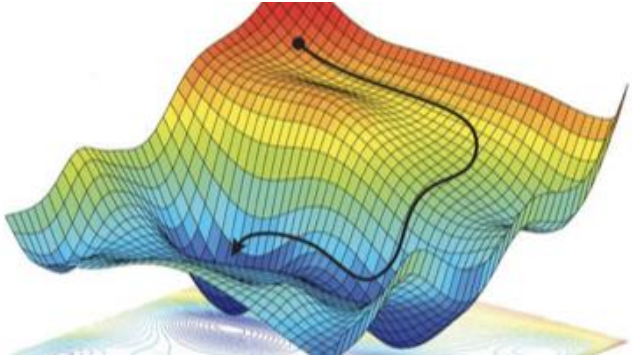
https://towardsdatascience.com/top-machine-learning-algorithms-for-classification-2197870ff501

Objective / Loss Function

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad \text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \times 100$$

$$\text{Crossentropy} = -\frac{1}{n}\sum_{i=1}^{n} y_i \ln(\hat{y}_i) + (1 - y_i)\ln(1 - \hat{y}_i)$$

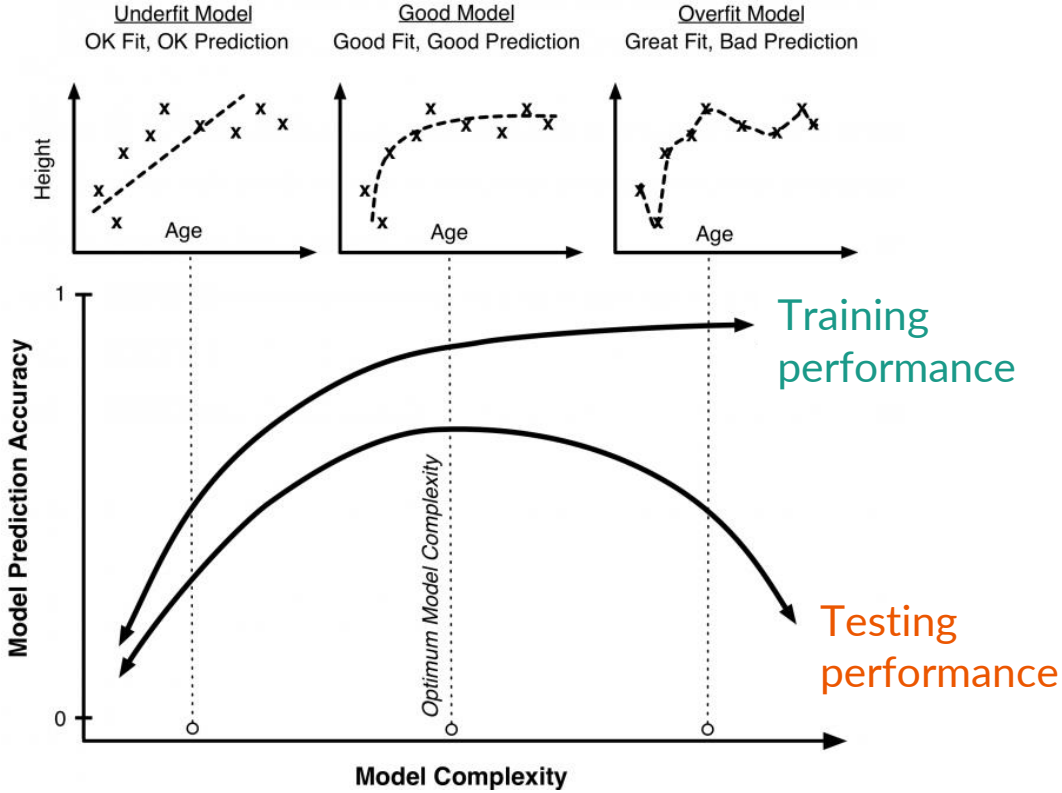
Learning Algorithm



https://medium.com/analytics-vidhya/gradient-descent-b0dc1af33517

Supervised learning is all about control

https://en.wikipedia.org/wiki/Bull_riding



https://realkm.com/2018/04/23/optimization-and-complexity-the-cost-of-complexity-systems-thinking-modelling-series/

Likelihood

- **Likelihood**: Probability of observing data x from a model with parameters θ

- Probability of getting two heads in a row, given a fair coin
  - P(HH | $p_H = 0.5$) = 0.5 x 0.5 = 0.25

- Probability of getting two heads in a row, given a biased coin
  - P(HH | $p_H = 0.8$) = 0.8 x 0.8 = 0.64

- **Maximum Likelihood Estimate (MLE)**: find θ that maximize the likelihood

Statistical control of overfitting

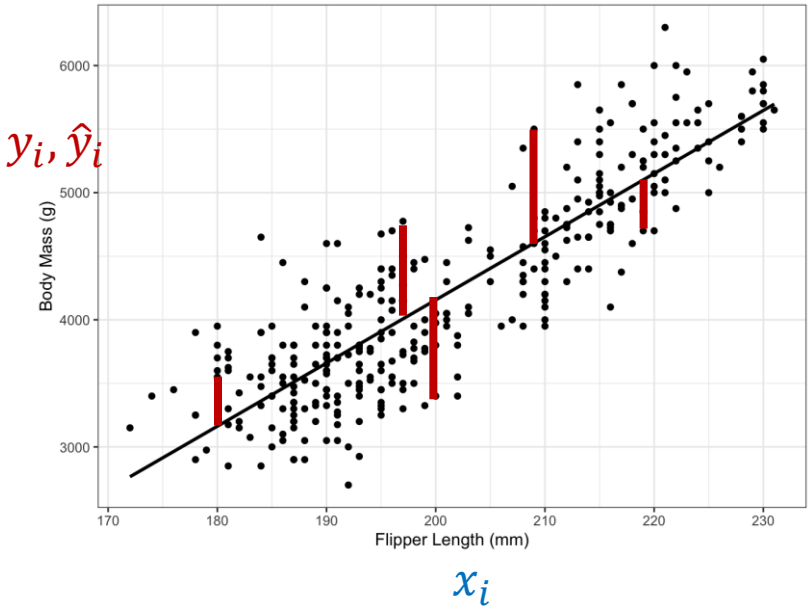
- Better model achieves higher likelihood
- Complex model has more parameters

- Information Criterion
  - Akaike (AIC) $= 2k - 2 \cdot \ln(\hat{L})$, where $\hat{L}$ is the likelihood
  - Bayesian (BIC) $= \ln(n)\, k - 2 \cdot \ln(\hat{L})$, where n is the sample size

- Nested model testing
  - Simple model has n parameters, fit the data with likelihood $\widehat{L_1}$
  - Complex model has $m > n$ parameters, fit the data with likelihood $\widehat{L_2} > \widehat{L_1}$
  - Is the improvement $\frac{\widehat{L_2}}{\widehat{L_1}}$ worth the increase in $m - n$ parameters?

Linear and logistic regression

Linear regression (Ordinary Least Square)



- Model: $\hat{y}_i = b_0 + b_1 x_i$
- Minimize MSE: $\frac{1}{n}\sum_i (y_i - [b_0 + b_1 x_i])^2$

- $\frac{\delta MSE}{\delta b_o} = -2\sum_i y_i - 2b_1 \sum_i x_i - 2nb_0$
- $\frac{\delta MSE}{\delta b_1} = -2\sum_i x_i y_i - 2b_1 \sum_i x_i^2 - 2b_0 \sum_i x_i$

- $b_0 = \frac{\sum xy \sum x - \sum x^2 \sum y}{(\sum x)^2 - n\sum x^2}$
- $b_1 = \frac{\sum y \sum x - n\sum xy}{(\sum x)^2 - n\sum x^2}$

Ordinary Least Square interpretation

- Observed value = True value + Normally-distributed noise
- **Assumption**: Noises are identical and independent across samples

- Model: $(y_i - \hat{y}_i) \sim N(0, \sigma^2)$
- Density: $\mathrm{P}(y_i - \hat{y}_i = \varepsilon_i \mid \sigma^2) \propto e^{\frac{-\varepsilon_i{}^2}{2\sigma^2}}$
- Likelihood: $\prod_i \mathrm{P}(y_i - \hat{y}_i = \varepsilon_i \mid \sigma^2) \propto e^{\frac{-\sum_i \varepsilon_i{}^2}{2\sigma^2}}$

- MSE: $\frac{1}{n}\sum_i (y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_i \varepsilon_i{}^2$
- Minimizing MSE is the same as maximizing likelihood

Logistic regression

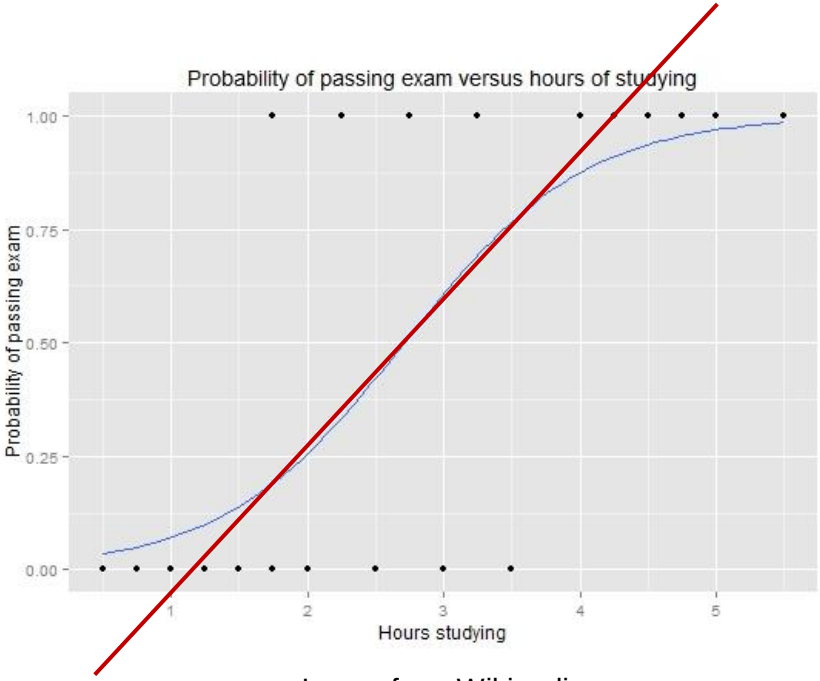


Image from Wikipedia

- Classification output = 0 or 1
- Linear regression outputs $-\infty$ to ∞

- Probability of success p
- Log odd: $\ln\left(\frac{p}{1-p}\right)$
  - $\ln\left(\frac{p}{1-p}\right) \to -\infty$ as $p \to 0$
  - $\ln\left(\frac{p}{1-p}\right) \to \infty$ as $p \to 1$

- Transform linear regression output with log odd!

Logistic regression

- Model: $\ln\left(\frac{\widehat{y_i}}{1-\widehat{y_i}}\right) = f(x_i) = b_0 + b_1 x_{i,1} + \cdots + b_n x_{i,n}$
- $\widehat{y_i} = \frac{e^{b_0+b_1 x_{i,1}+\cdots+b_n x_{i,n}}}{1+e^{b_0+b_1 x_{i,1}+\cdots+b_n x_{i,n}}}$
  - When $x_i \to \infty$, $\widehat{y_i} \to 1$
  - When $x_i \to -\infty$, $\widehat{y_i} \to 0$

- Can we keep using MSE as the loss function?
  - Brier score $= \frac{1}{N}\sum_i (y_i - \widehat{y_i})^2$
  - But this does not interpret logistic output as probability

Likelihood for logistic regression

- Likelihood: $\mathrm{P}(y_i \mid x_i) = \hat{y}_i^{\,y_i}(1-\hat{y}_i)^{1-y_i}$
  - y_i is either 0 or 1
  - When y_i is 0, the likelihood is $1-\hat{y}_i$
  - When y_i is 1, the likelihood is $\hat{y}_i$

- Log likelihood: $y_i \ln(\hat{y}_i) + (1-y_i)\ln(1-\hat{y}_i)$
  - This is the cross-entropy loss function!
  - Maximizing likelihood is the same as minimizing cross-entropy

Impact of large coefficients

- **Model:** $\widehat{y_i} = b_0 + b_1 x_{i,1} + \cdots + b_n x_{i,n}$

- If there are two models (two sets of $b_j's$) that achieve similar performance, we should prefer the model with smaller magnitudes of $b_j's$

- Prevent overfitting to an input feature
  - Magnitude of b_k = influence of the kth input feature on the model

- Robustness to future inputs
  - Measurement error of 1 unit in $x_{i,k}$ will be amplified to b_k units in $\widehat{y_i}$

Regularization of linear model

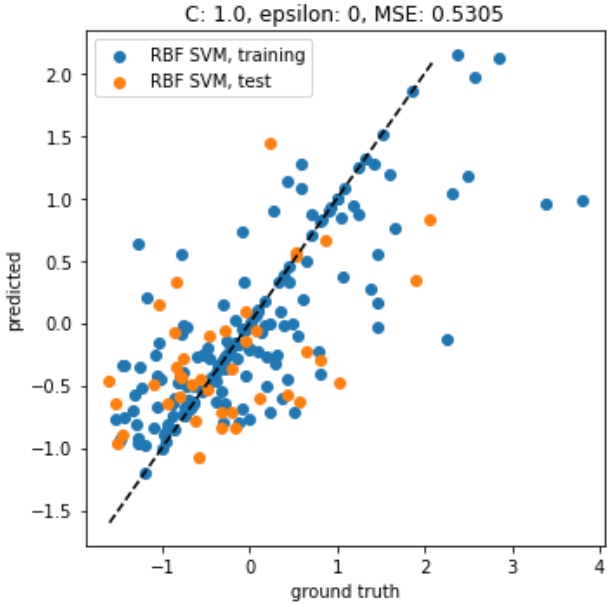
- L1 regularization (LASSO): MSE + $\alpha \sum_k |b_k|$
- L2 regularization (Ridge): MSE + $\alpha \sum_k {b_k}^2$

- α is the hyperparameter that controls the regularization strength

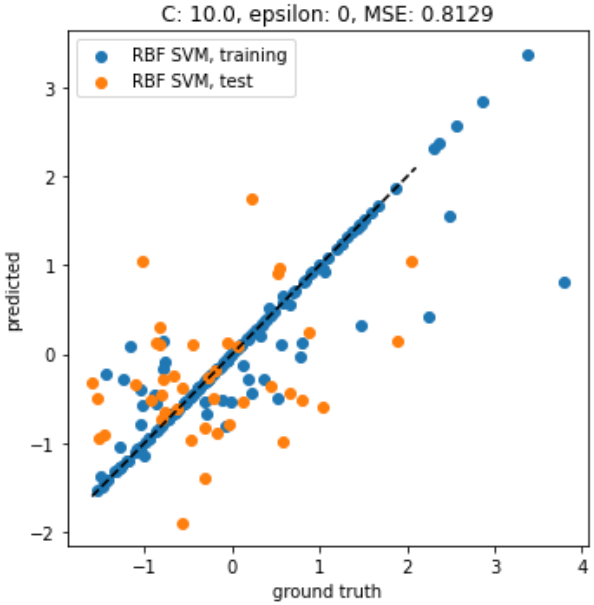
- Hyperparameter must be tuned
  - Split data into **Training-Validation-Test**
  - Try many values of α while training the model on the **Training** set
  - Select α that results in highest performance on the **Validation** set
  - Report model performance with selected α on the **Test** set

Tuning regularization strength

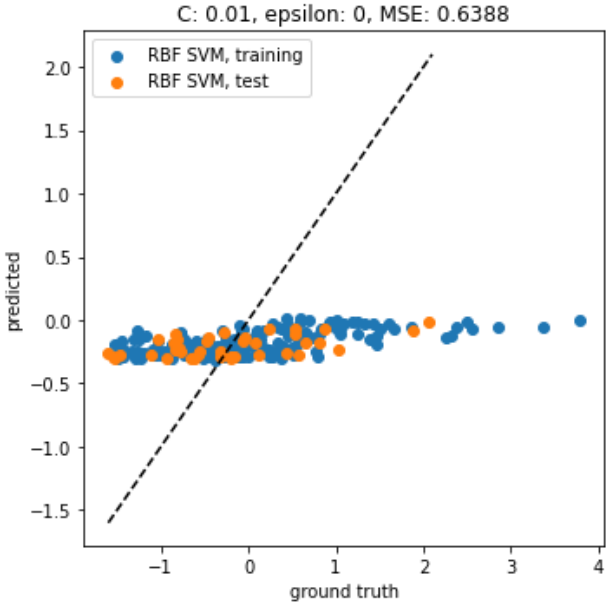
Just right



Too weak

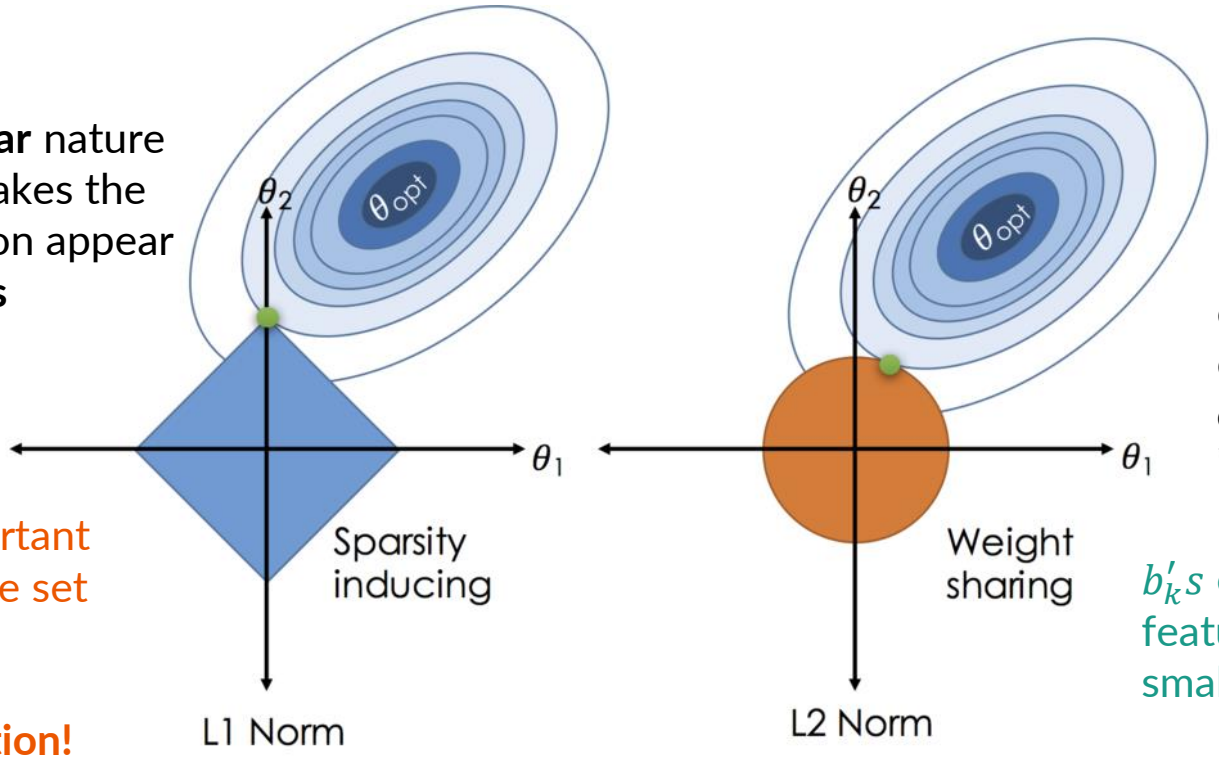


Too strong



L1 versus L2 regularization

The **rectangular** nature of $|b_0 + b_1|$ makes the optimal solution appear on the **corners**



For $(b_0 + b_1)^2$, the optimal solution can be **anywhere on the boundary**

$b_k's$ of unimportant features will be set to zeroes

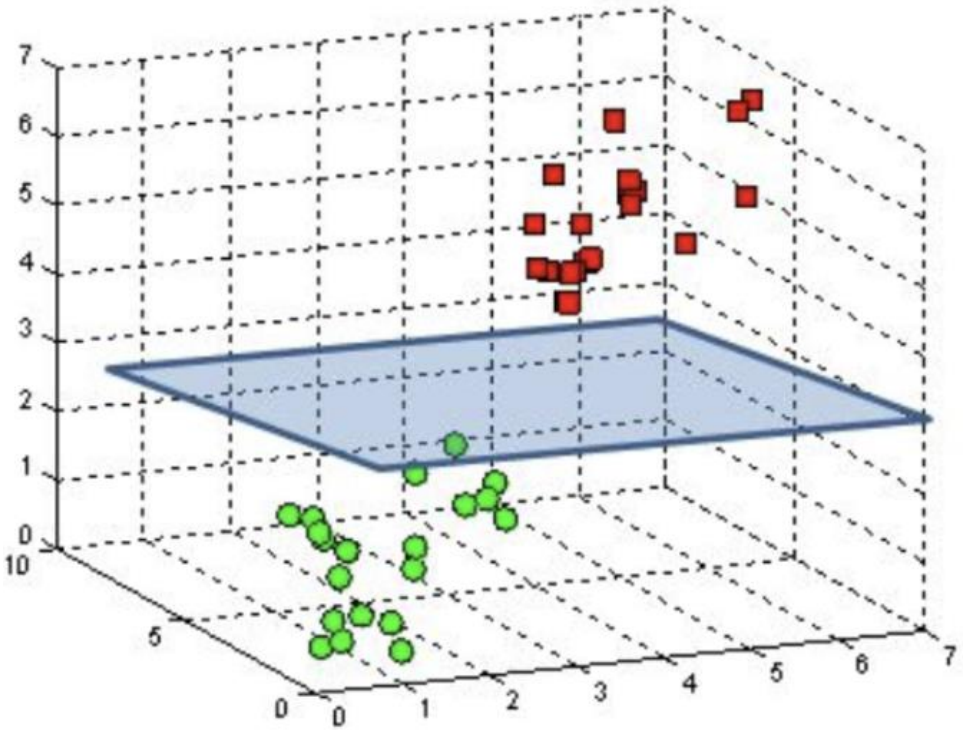
Feature selection!

$b_k's$ of unimportant features will be small but not zeroes

https://www.linkedin.com/pulse/regularization-episode-1-amr-mahmoud/

Support vector machine

Separating hyperplane

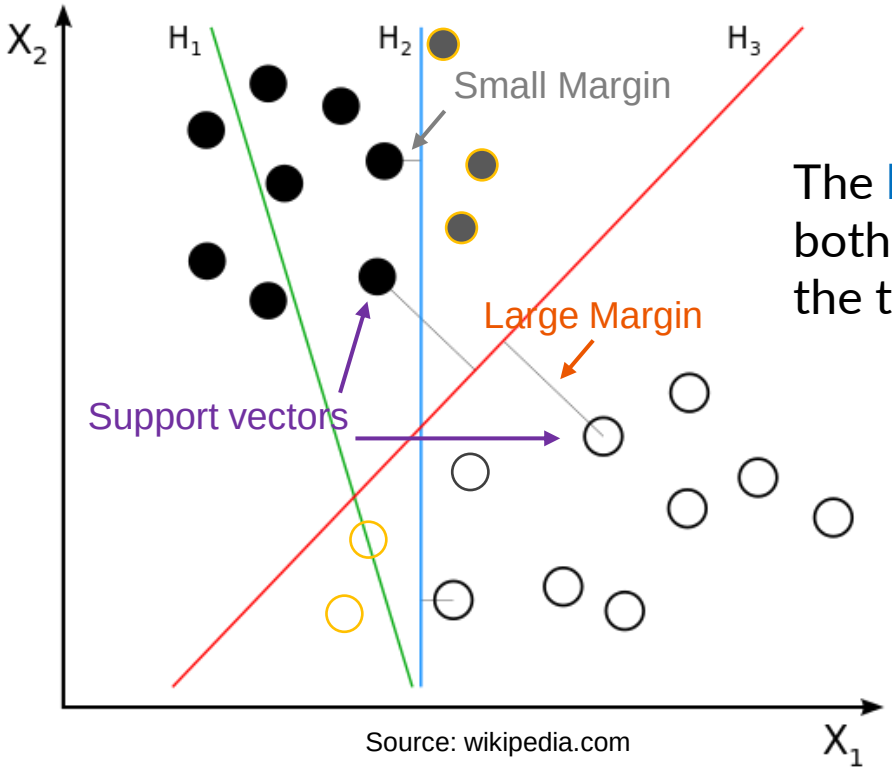


- *n*-dimensional data points from two classes
- Find the best (*n*–1)-dimensional hyperplane that separate the classes

Image from developer.xilinx.com

Support vector machine

The green line is bad at distinguishing black data points

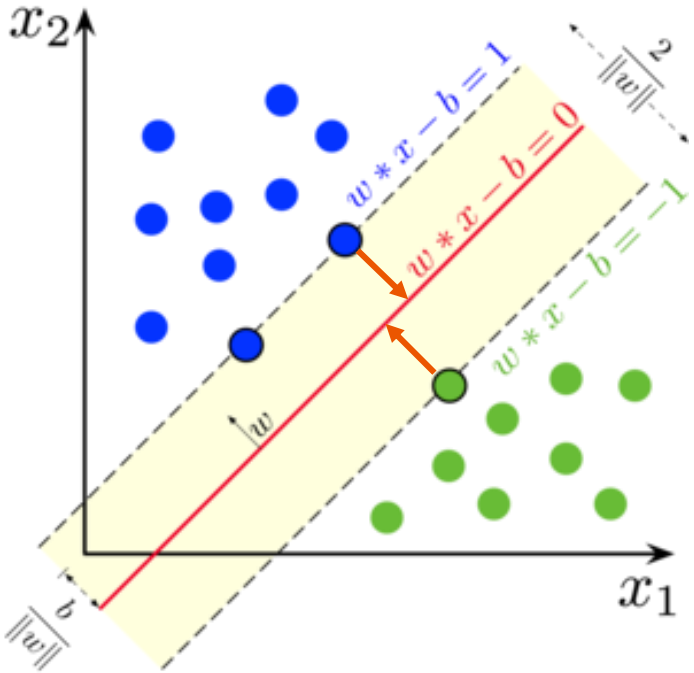


Source: wikipedia.com

The blue and red line both perfectly distinguish the two classes

Prefer model that maximizes the margin

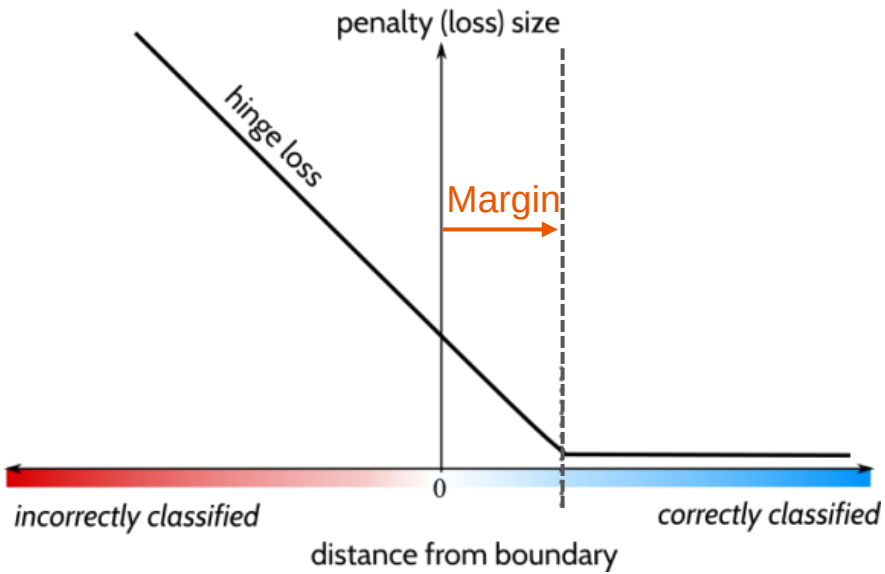
Margin and L2 regularization



Source: wikipedia.com

- Hyperplane H: $w_1 x_{i,1} + \cdots + w_n x_{i,n} - b = 0$
- Margin from a point on the hyperplane $w_1 x_{i,1} + \cdots + w_n x_{i,n} - b = \pm 1$ to H is equal to $\frac{1}{\sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}}$
- Maximizing margin is the same as minimizing L2 regularization!

Hinge loss for SVM

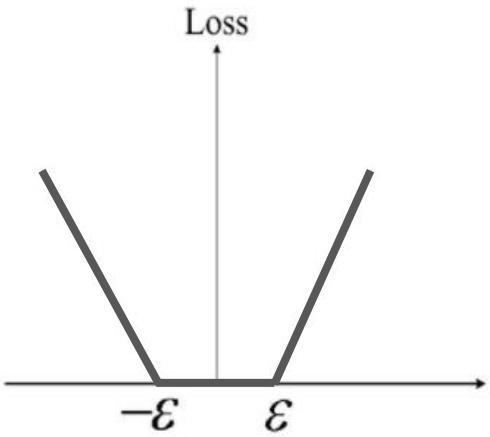


Source: towarddatasciences.com

- Penalize correctly classified data points that lie within the margin
- **Hinge loss**: $\max(0, 1 - y_i(w \cdot x_i - b))$
  - y_i and $w \cdot x_i - b$ have the same sign for correctly classified data points
- SVM loss:

$$C \sum \max(0, 1 - y_i(w \cdot x_i - b)) + \frac{1}{2}\|w\|^2$$

Support vector regressor (SVR)



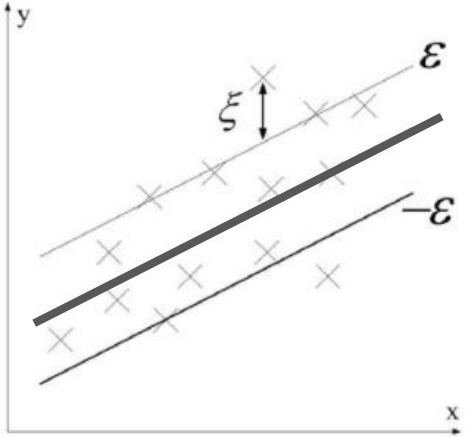


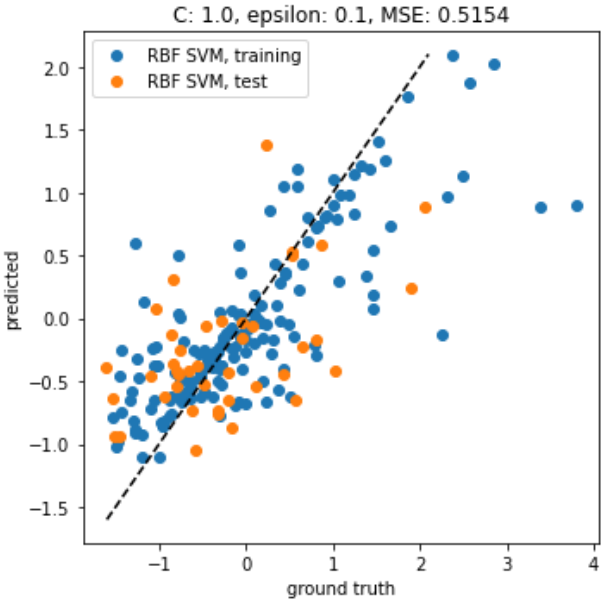
Image from https://slideplayer.com/slide/15044351/

- Penalize only data points with regression error $> \varepsilon$
  - Reverse Hinge
- SVM loss:

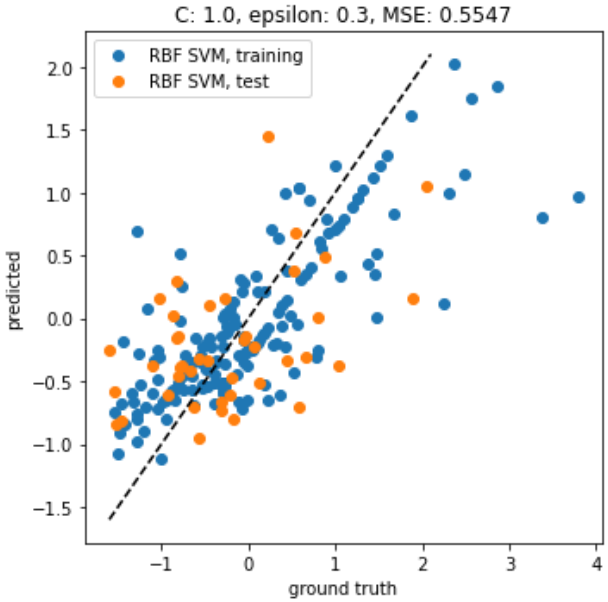
$$C \sum \max(0, |y_i(w \cdot x_i - b)| - \varepsilon) + \frac{1}{2}\|w\|^2$$

Tuning epsilons for SVR

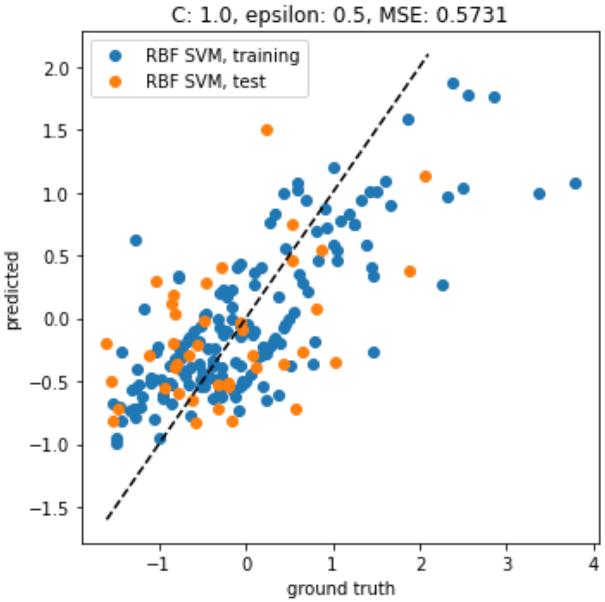
Small epsilon



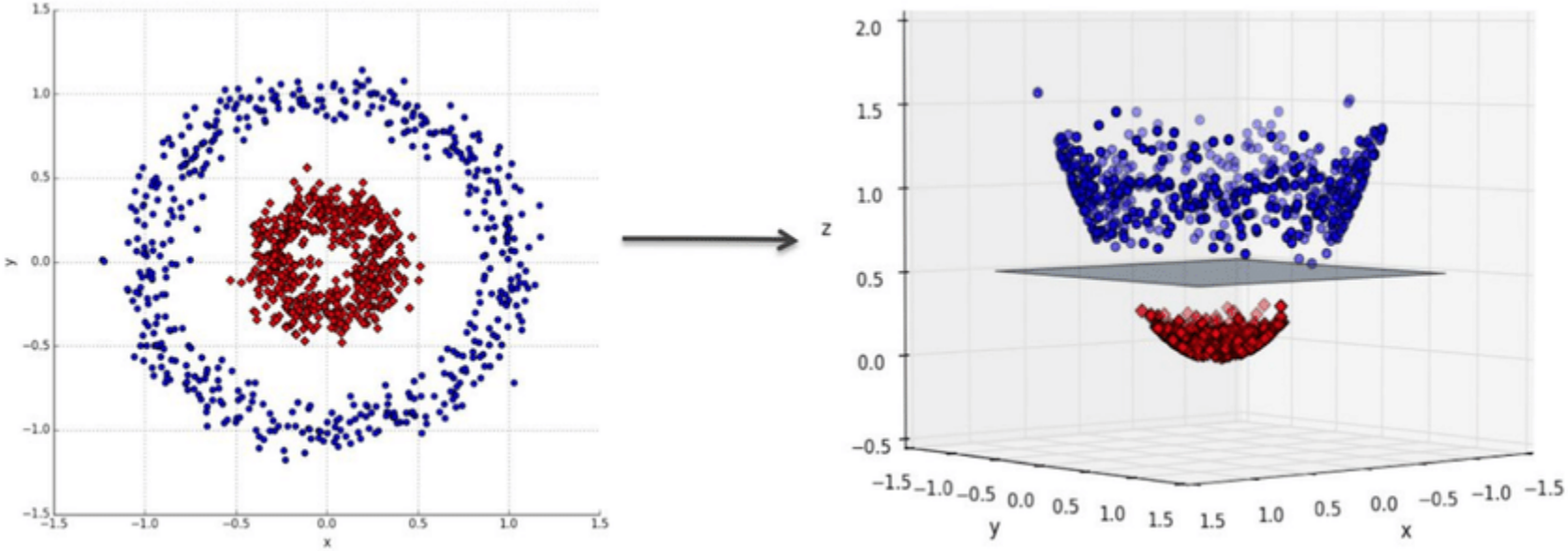
Medium epsilon



High epsilon

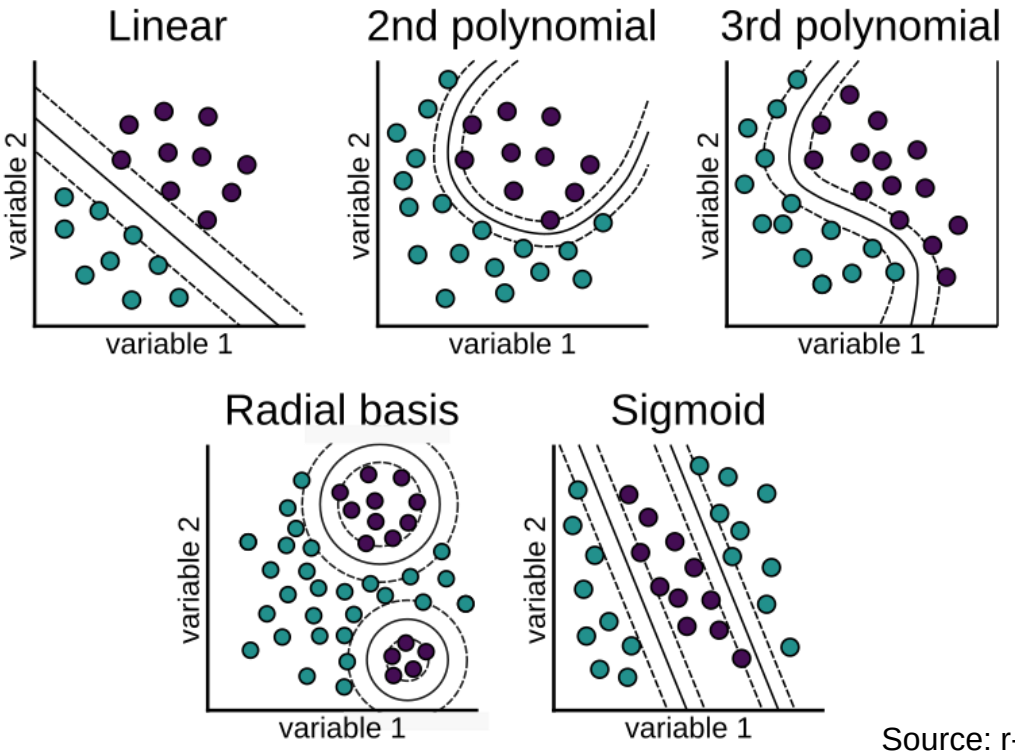


Feature transformation



- Some problem can be solved linearly by transforming input features

Kernel tricks for SVM



- Feature transformation costs computing resource
- $x \rightarrow \phi(x) = (x, x^2, x^3, \dots, x^m)$
- Kernel: $k(x, y) = \phi(x) \cdot \phi(y)$
- Suffice to train SVM
- Enable nonlinear feature transformation on a linear model

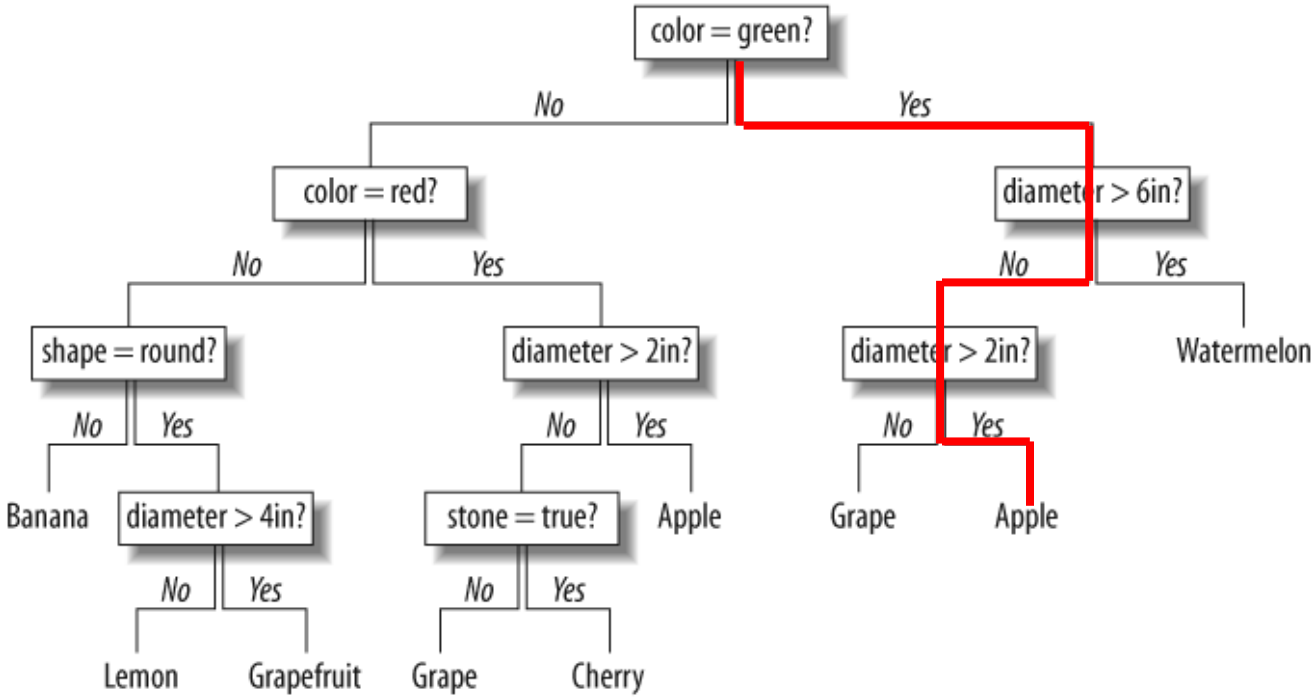
Source: r-bloggers.com

First checkpoint summary

- Supervised learning is all about controlling the data fitting process
- Core components: model + loss function + learning algorithm
  - MSE for regression, Cross-Entropy for classification
- L1 (LASSO) and L2 (ridge) regularization
  - Tuning of regularization strength
- The concept of margin in SVM → Hinge Loss
- Kernel tricks = extension of SVM to non-linear feature space

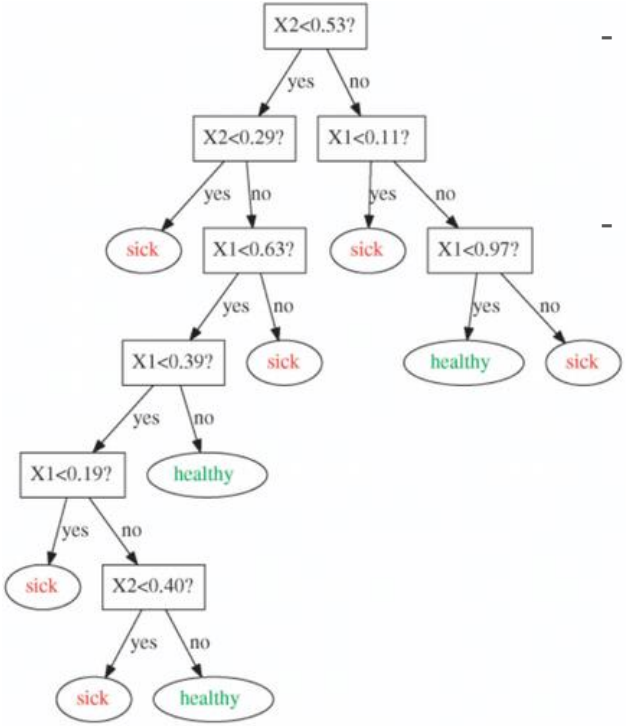
Decision tree

Decision tree

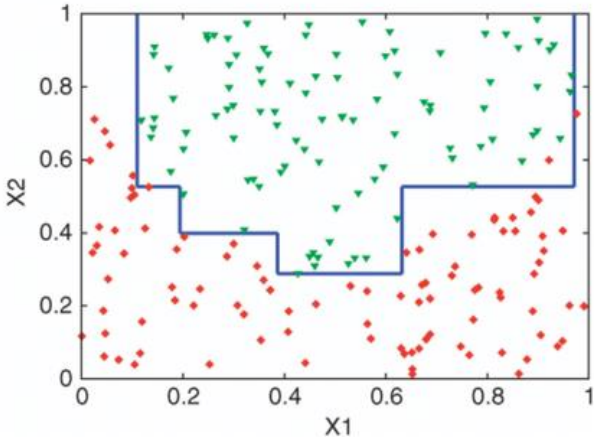


Source: Programming collective intelligence by Toby Segaran

Decision tree behaviors

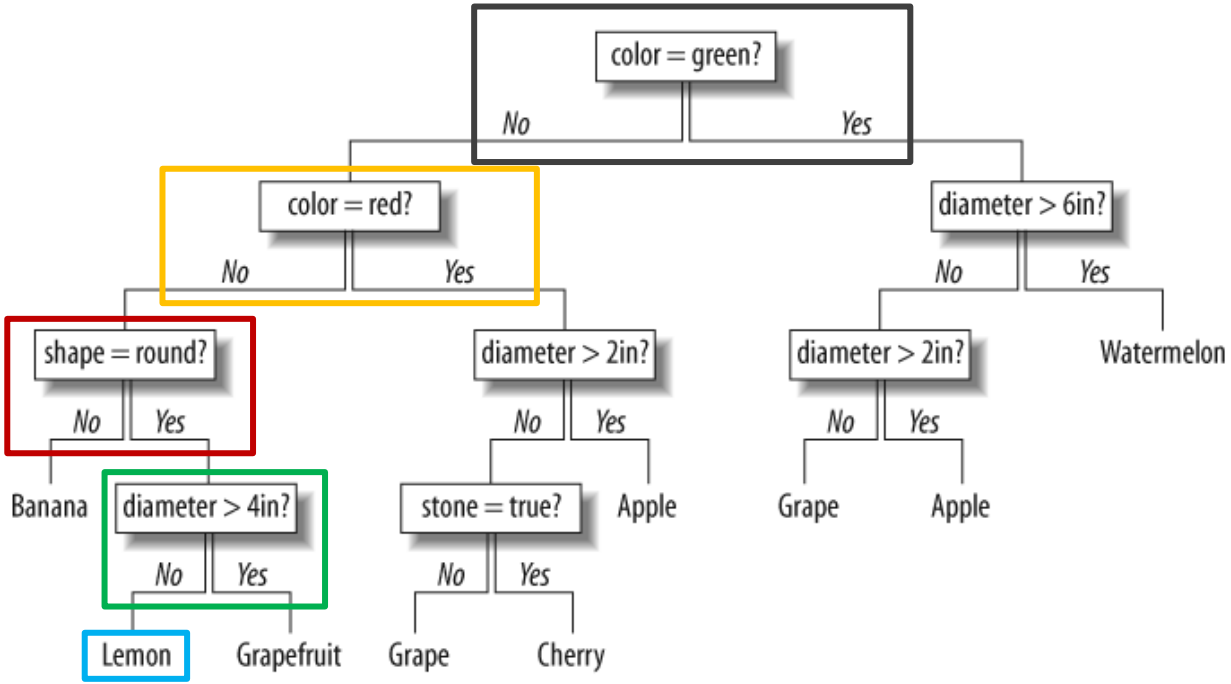


- Each decision is a threshold on each feature
  - Piecewise linear
  - Parallel to an axis
- Good for criteria-based classification



Miller, C. “Screening meter data: Characterization of temporal energy data from large groups of non-residential buildings”

Tree building



- Pick a feature & a criterion, but how?

Splitting quality

100 healthy, 100 sick

85 healthy, 5 sick

15 healthy, 95 sick

45 healthy, 51 sick

55 healthy, 49 sick

- Gini impurity: $\sum p(1-p)$
- Entropy: $-\sum p \ln(p)$
  - Minimal at p = 0 or 1 → Perfect split
  - Maximal at p = 0.5 → 50-50 split
- Search for feature and cutoff that yield lowest impurity or entropy

Control mechanisms for tree building

1. Too few samples to make a split

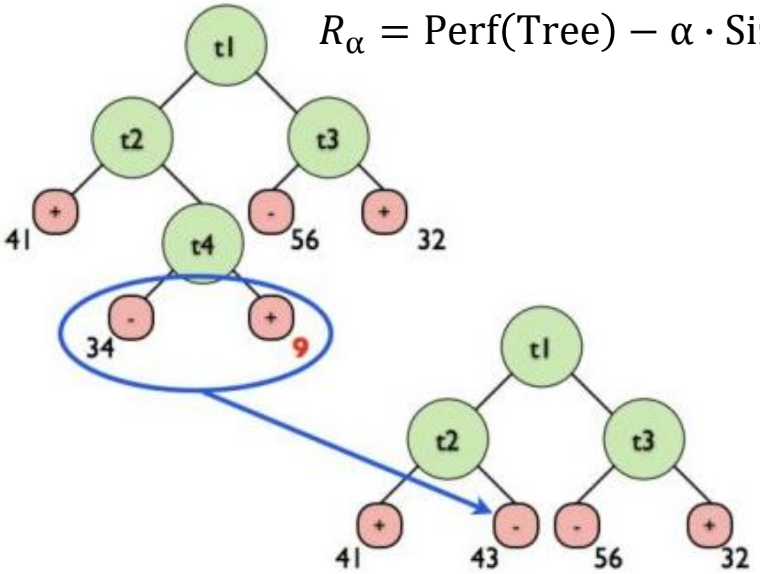
3. Impurity or entropy does not change much after the split

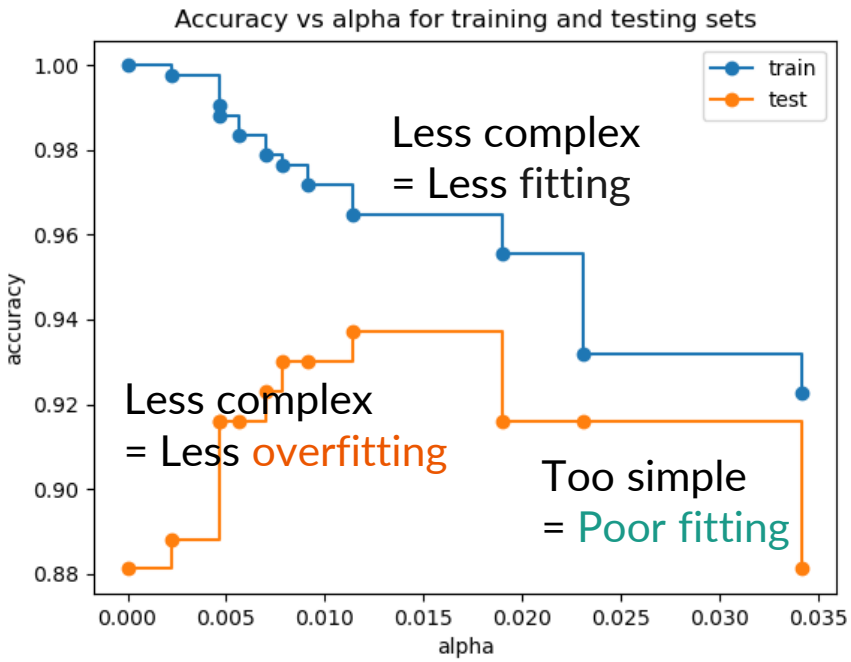
2. Too few samples on either branch

- Limit the tree size
- Limit the improvement in quality
- Limit the number of samples that support a split

Tree pruning (post-processing)

$$R_\alpha = \text{Perf(Tree)} - \alpha \cdot \text{Size}$$





Patel, N. and Upadhyay, S. “Study of Various Decision Tree Pruning Methods with their Empirical Comparison in WEKA” IJCA 2012

Image from scikit-learn,org

Regression tree

Predictors | Target

Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	52
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

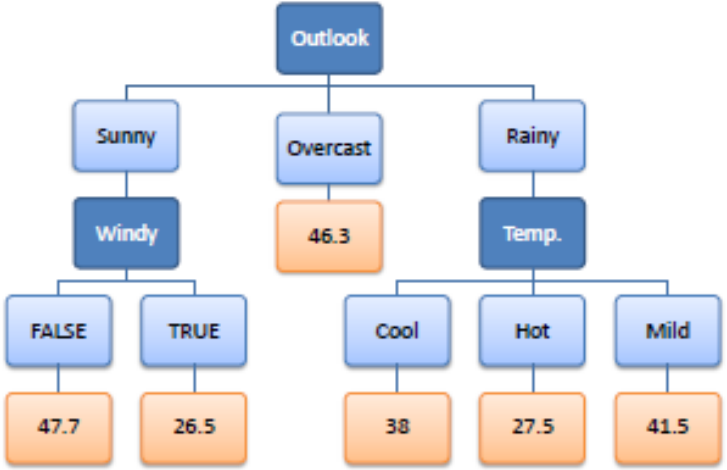


Image from saedsayad.com

- Use decision tree to group data points and predict the average

Regression tree in action

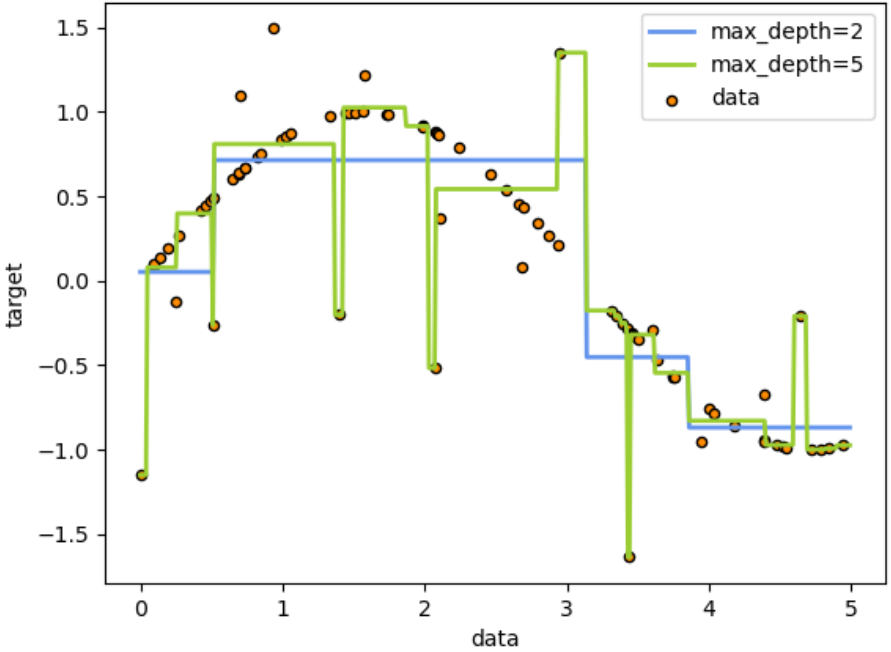
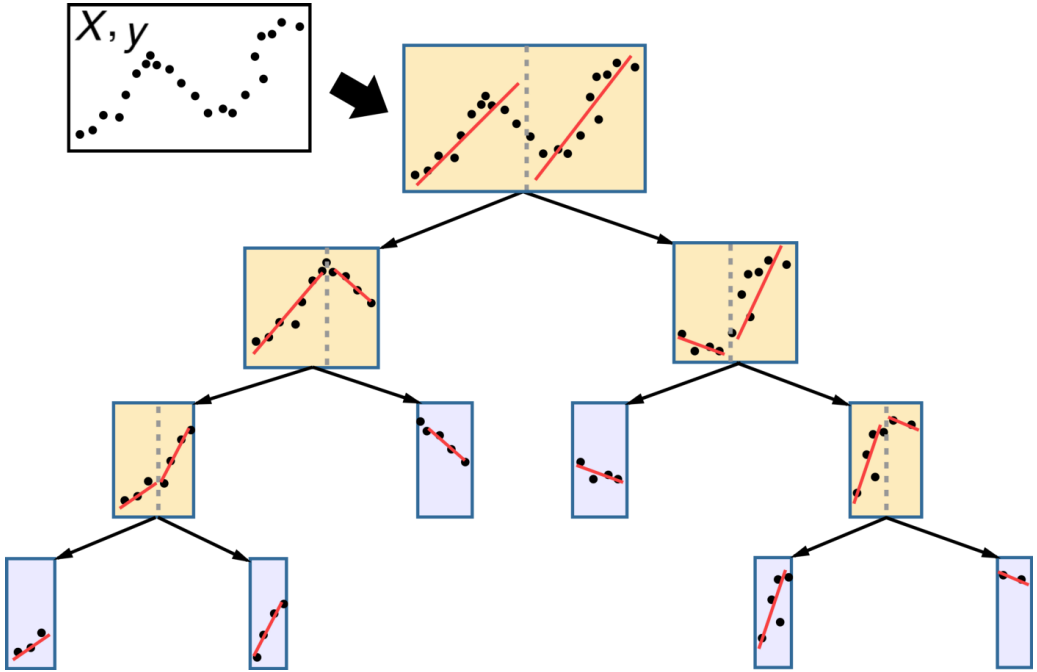


Image from scikit-learn,org

- Low depth = less complex = smoother prediction values
- High depth can lead to overfitting

Decision tree with regression model



https://towardsdatascience.com/introduction-to-model-trees-6e396259379a

- For each group of samples, use the data to fit a regression model

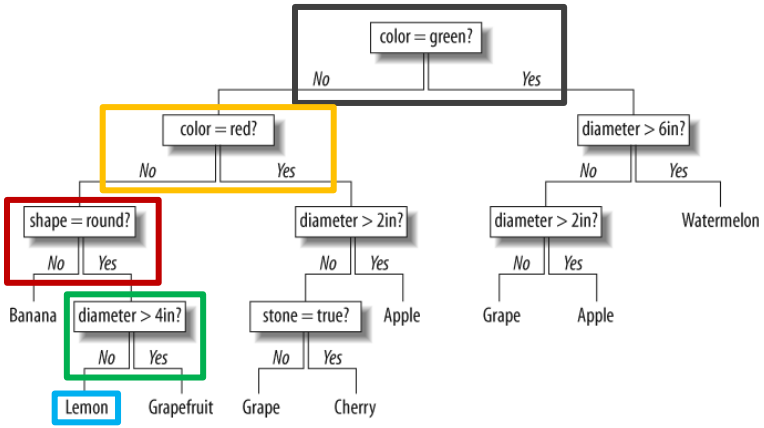
Feature selection

Using all features can be detrimental

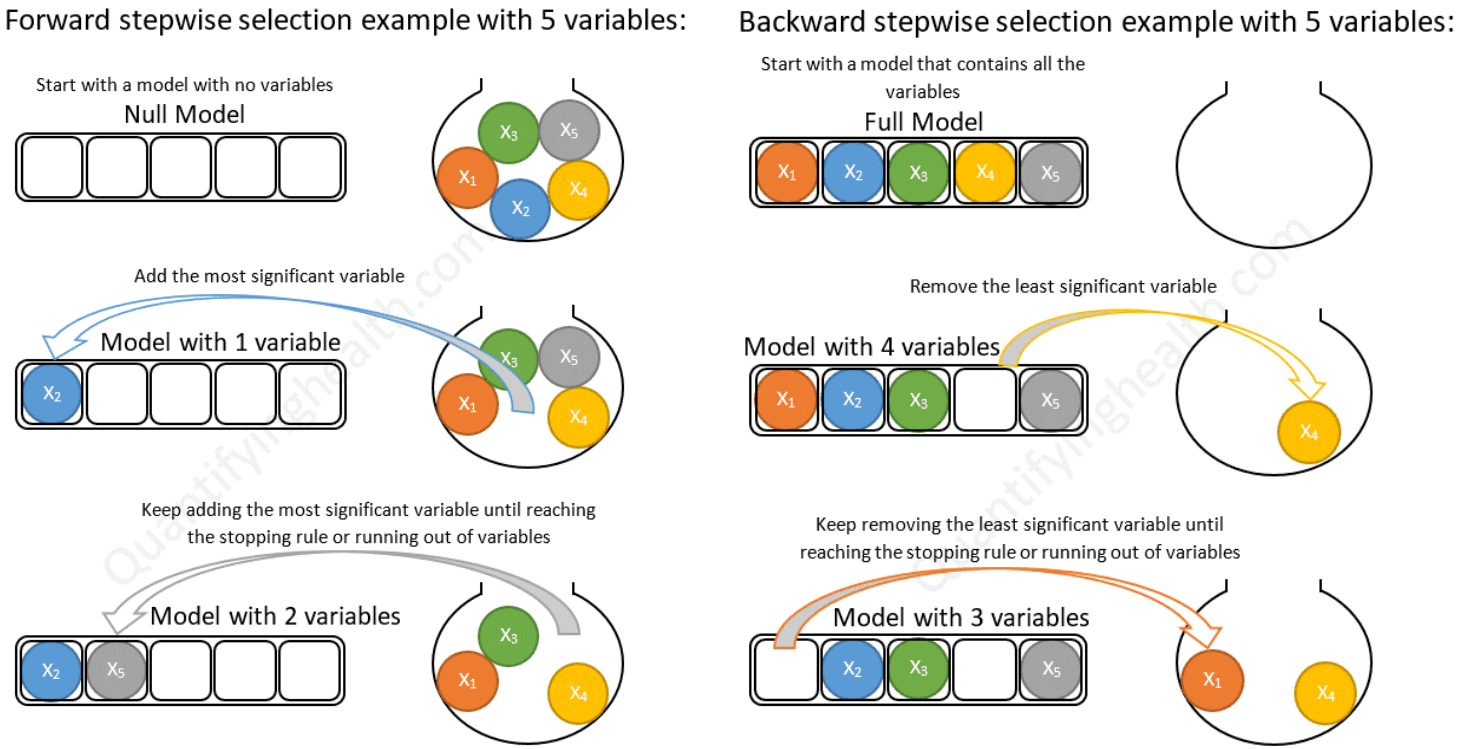
- **Linear model:** $\hat{y}_i = b_0 + b_1 x_{i,1} + \cdots + b_n x_{i,n}$
  - LASSO

- **Tree model:**
  - Repeatedly using the same feature
  - Early decision affects the rest

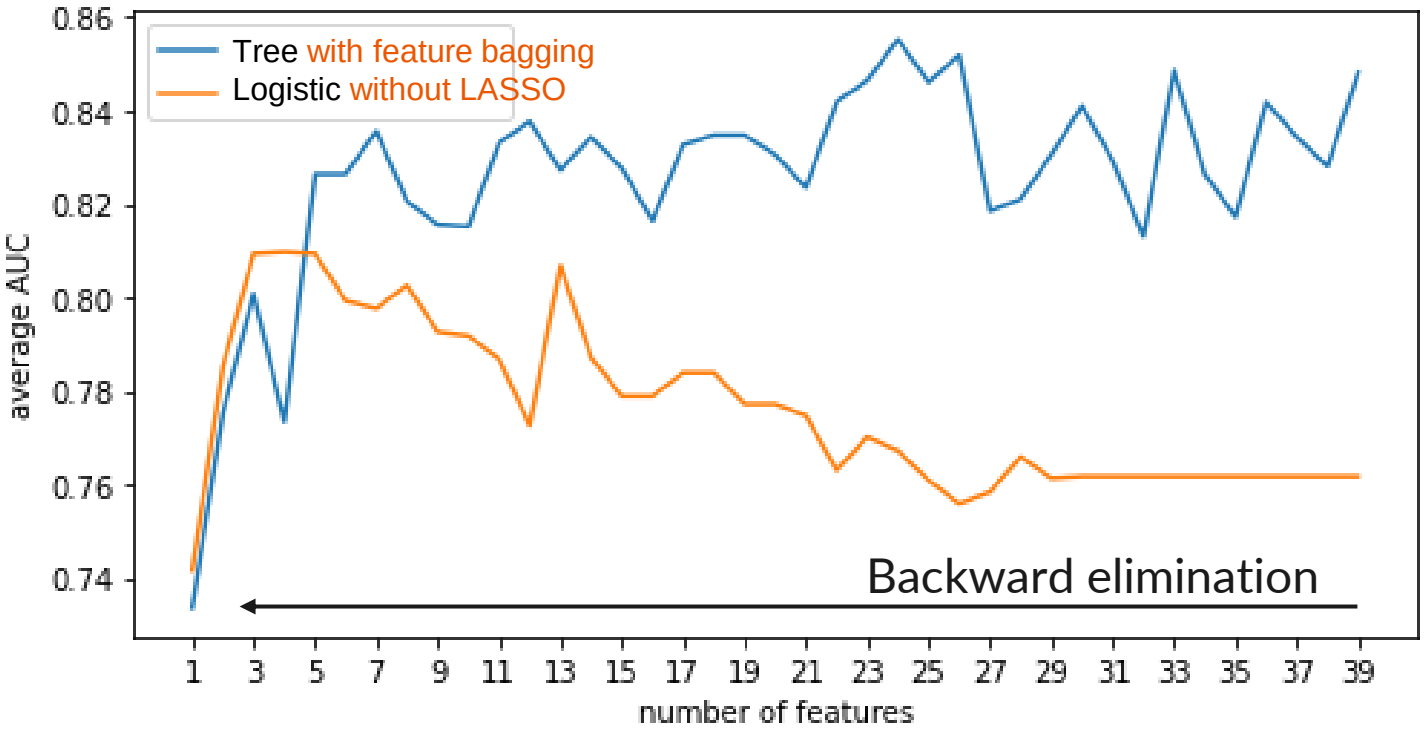
- Feature bagging
  - Look at only *N* features at each step
  - Force model to use diverse features



Iterative feature selection

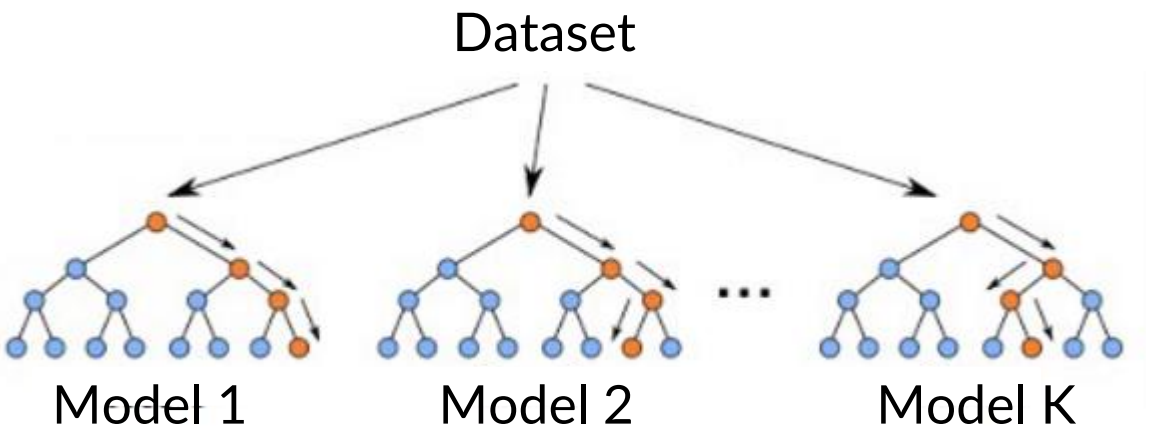


Impact of feature selection



Ensemble approaches

Training and aggregating multiple models

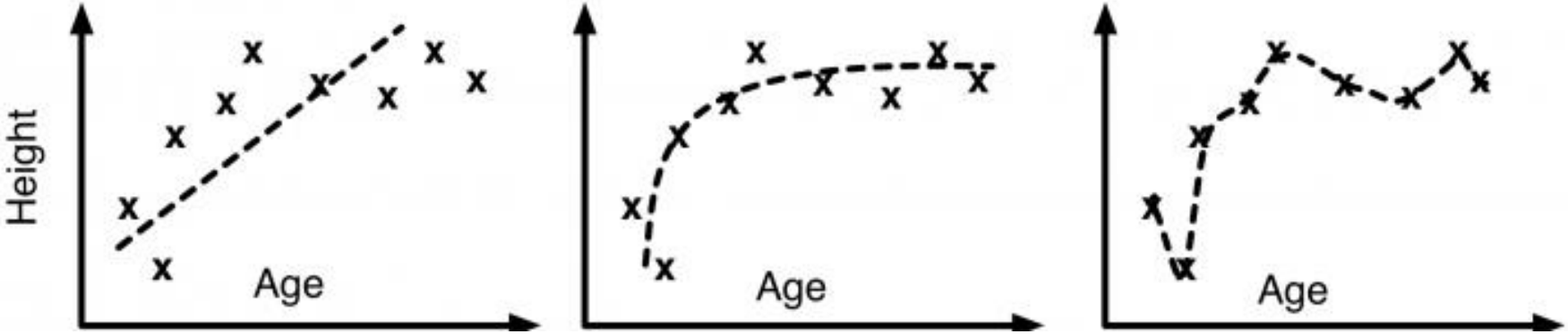


- Bagging: Generate random sets of samples to train multiple models
- Boosting: The *k*-th model address the errors made by earlier models

Impact of ensemble

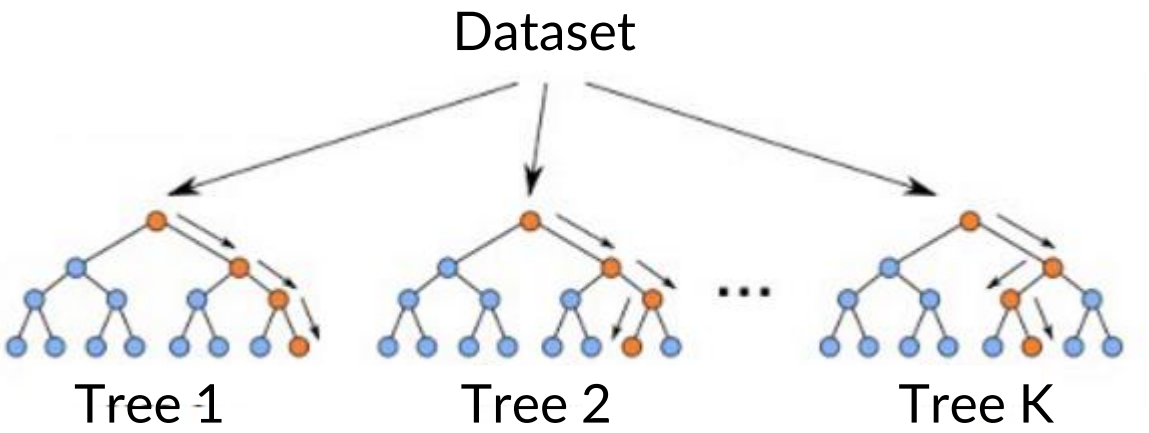
Boosting solves underfitting

Bagging prevent overfitting



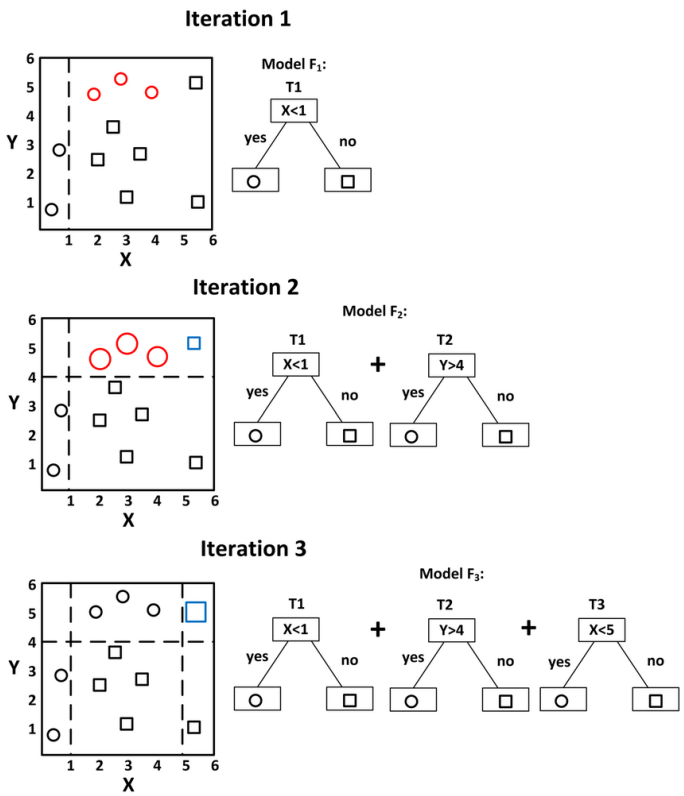
https://realkm.com/2018/04/23/optimization-and-complexity-the-cost-of-complexity-systems-thinking-modelling-series/

Random forest



- Sample 80% of the dataset to train each decision tree
- Each tree may overfit to different part of the dataset
- But the consensus should be correct

Boosting for classification = weight the error



- Ensemble predictor = weighted average
  - $C_n(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \cdots + \alpha_n f_n(x)$

- **Adaptive Boosting** (AdaBoost)
- **Exponential loss:** $\sum_{y_i \neq f_n(x_i)} e^{-y_i C_{n-1}(x_i)}$
  - Weight error made by *n*-th model using the error made by the first *n*−1 models

- α_n is based on the performance of $f_n(x)$

Boosting for regression = fit the residual directly

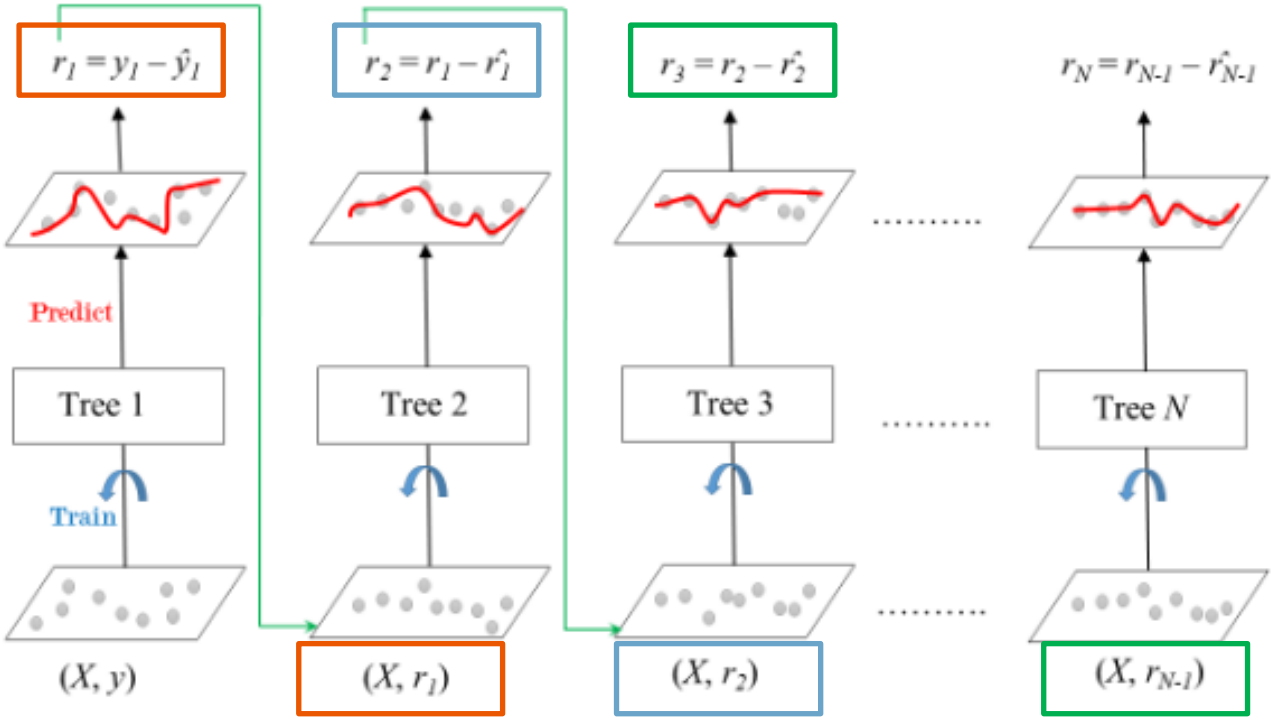
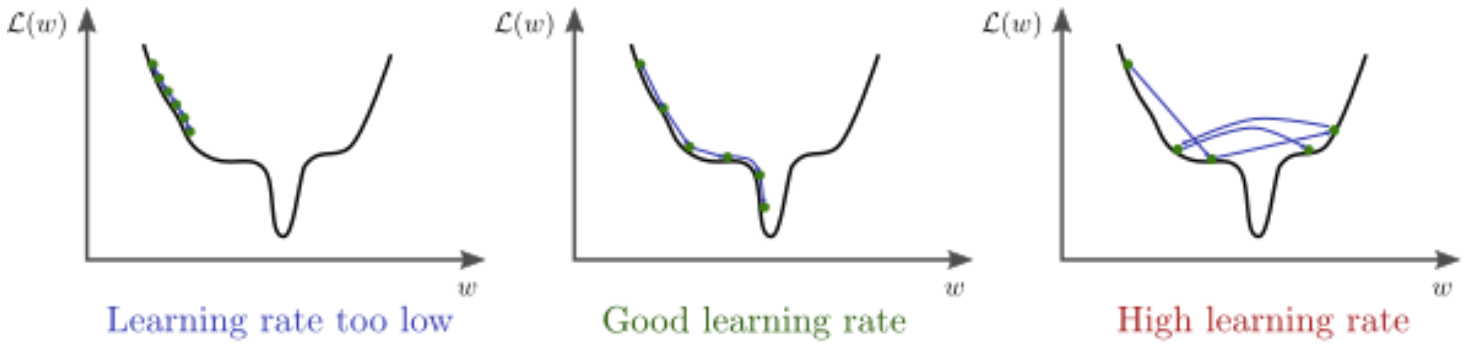


Image from geeksforgeeks.org

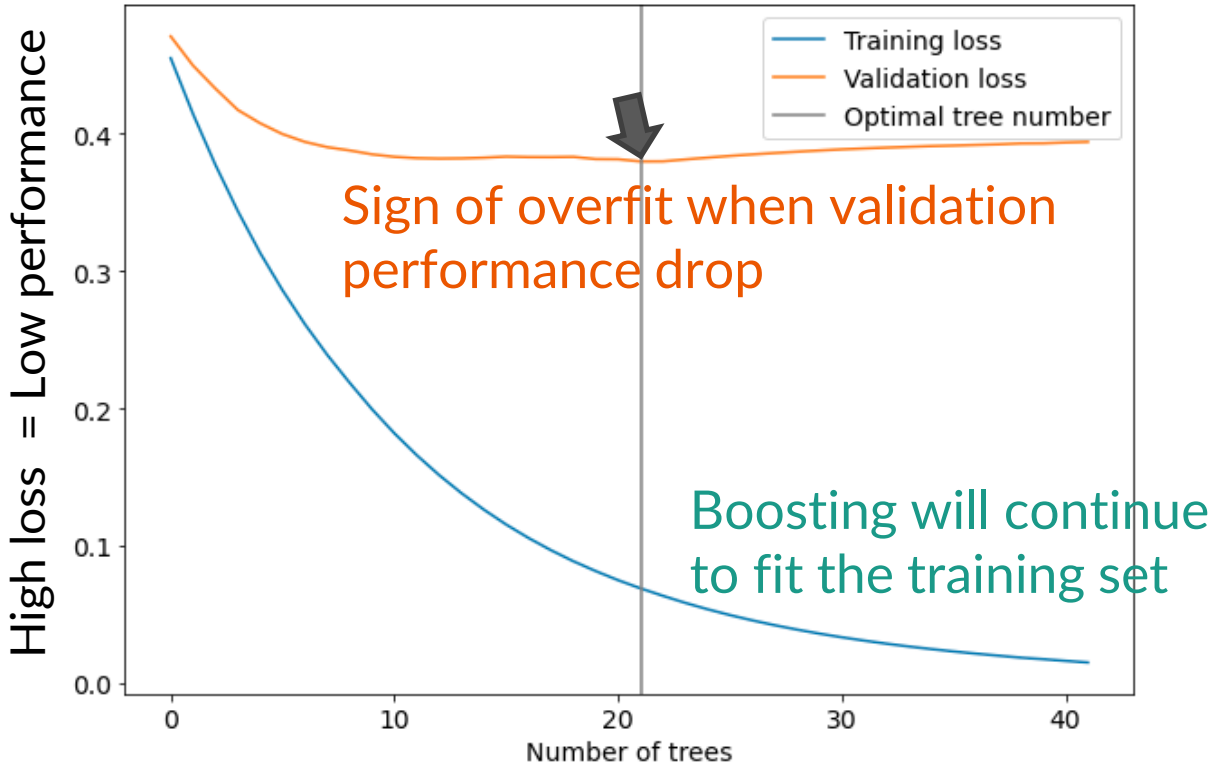
Controlling the boosting process



http://www.bdhammel.com/learning-rates/

- Learning rate: how much to trust the next update
  - $C_n(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \cdots + \alpha_n f_n(x)$
- Too low → slow training process → many models → computational cost
- Too high → overshoot the optimal point

Early stopping with validation set

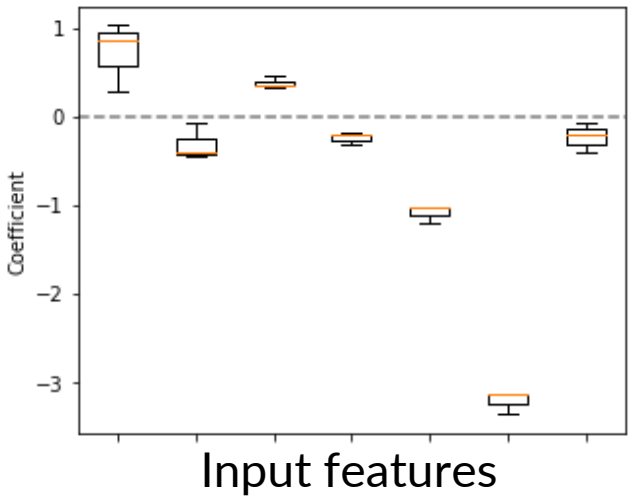


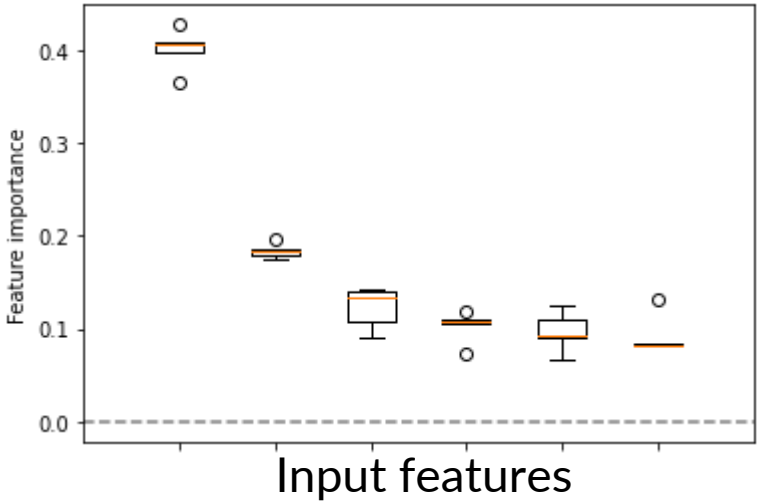
Second checkpoint summary

- Decision tree model is suitable for criteria-based task
- Control mechanisms for tree model
  - Tree size and splitting criteria
- Benefits of feature selection
- Ensemble approaches
  - Bagging prevents overfitting, boosting solves underfitting
- Learning rate and early stopping

Explainability

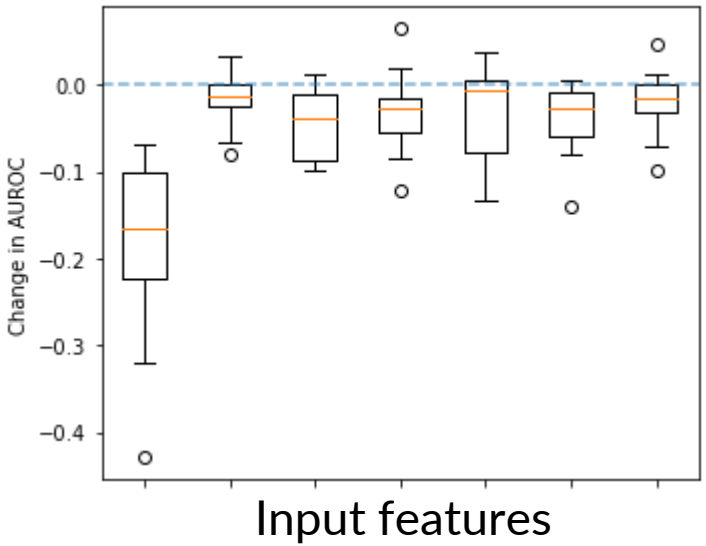
Feature importance





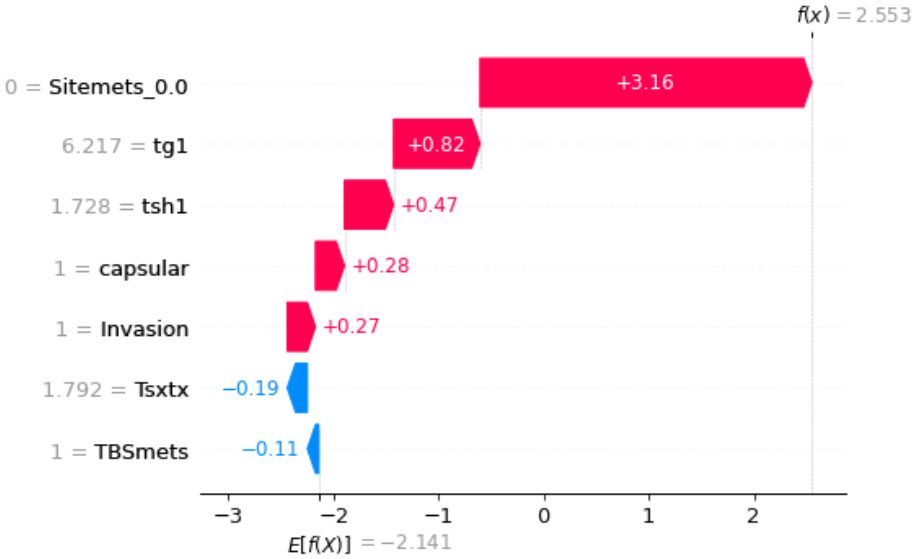
- Coefficients of linear, logistic, and SVM models
- Average improvement in impurity or entropy in tree models
- Model-level explanation

Change in performance after dropping a feature



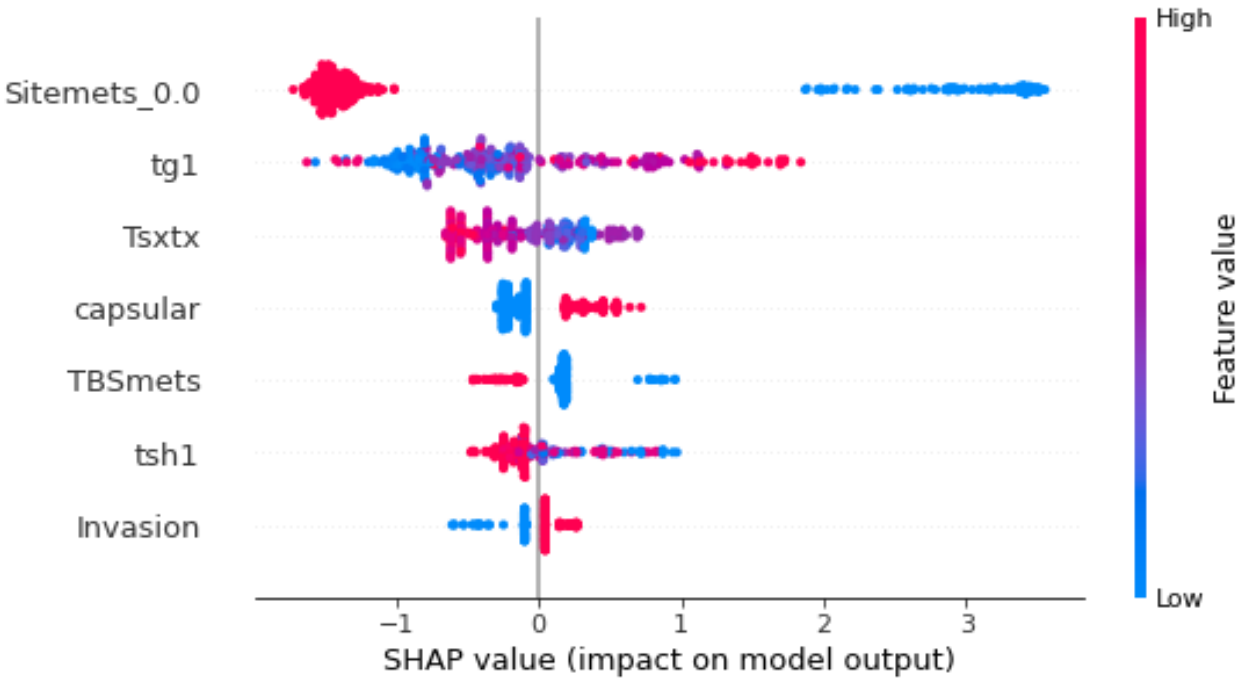
- Compare performance with and without each input feature
- Big drop = important

Shapley value



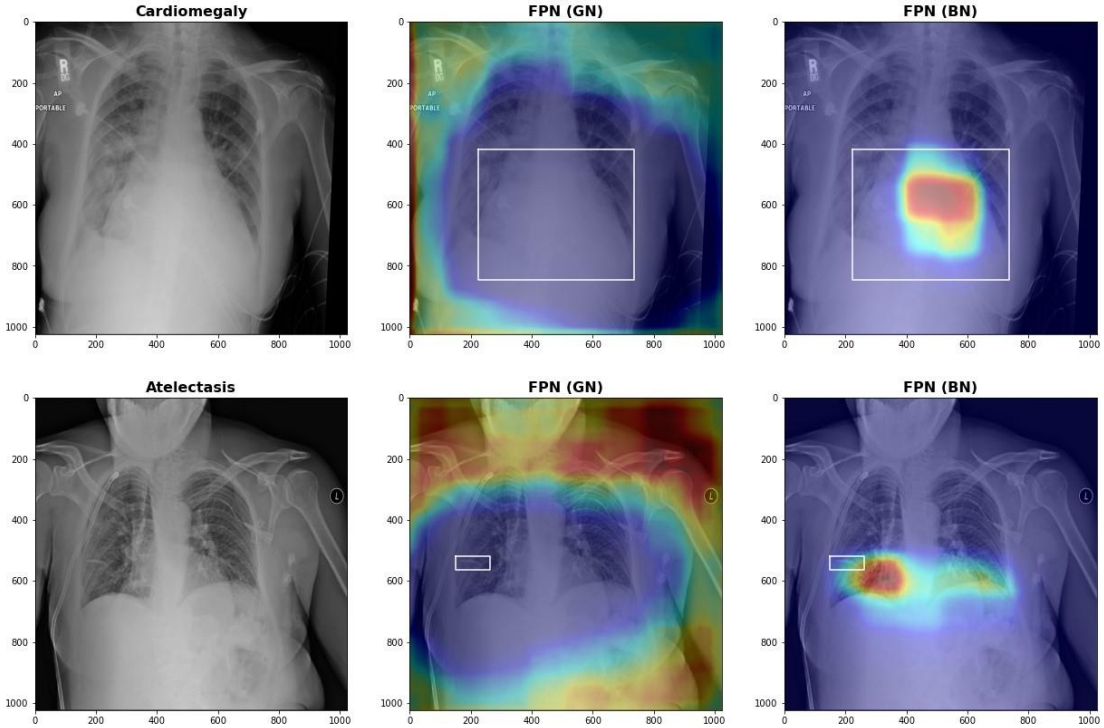
- Gain in performance from adding a feature *i*
  - $\varphi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n-|S|-1)!}{n!} [v(S \cup \{i\}) - v(S)]$
- Sample-level explanation

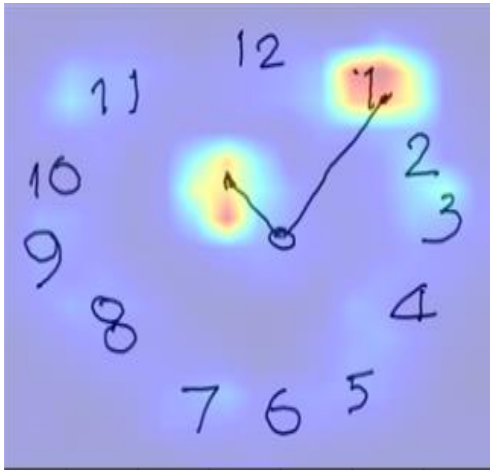
Shapley value distribution on the whole dataset



- Direction and magnitude of effect on the predicted values

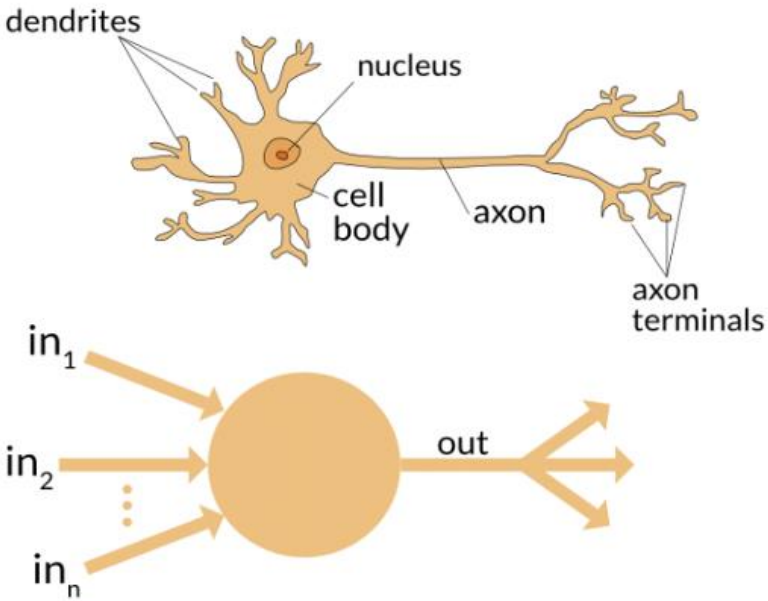
Explainability is needed for complex model

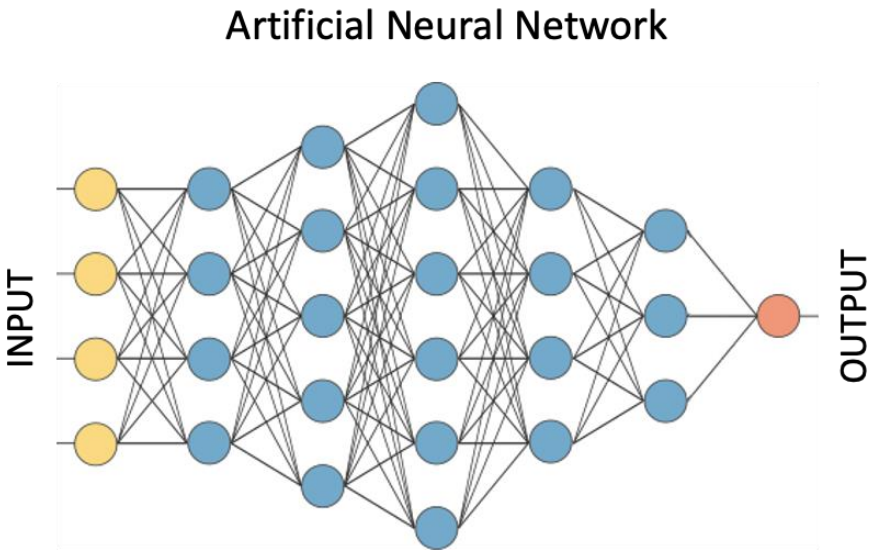




A touch of deep learning

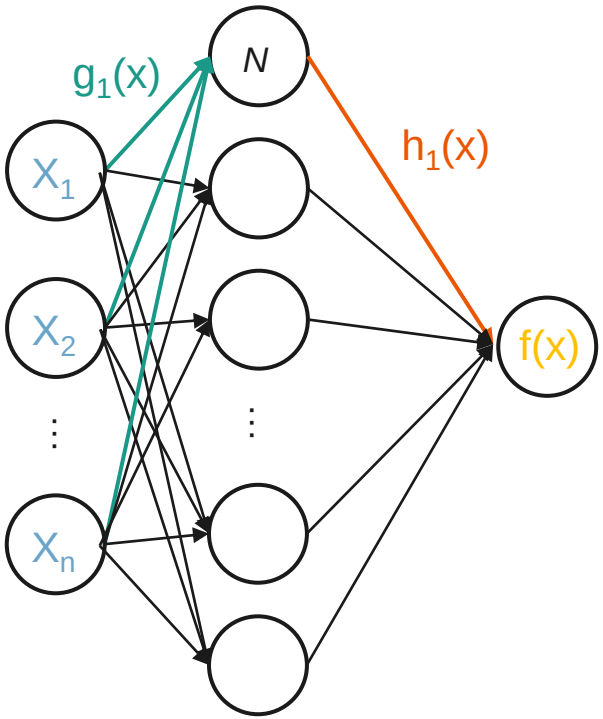
Artificial neural network





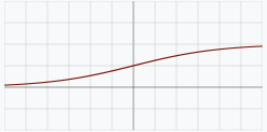
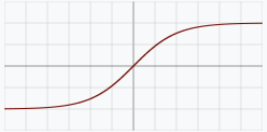
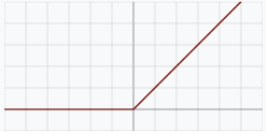
- Network of individually-simple computation node: out = $f(w_1 in_1 + w_2 in_2 + \ldots + w_n in_n)$

Signal passing in artificial neural network



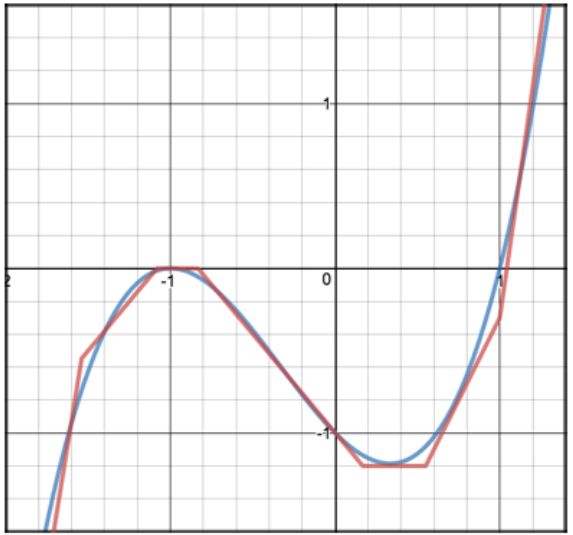
- Input of neuron *N* is a linear combination of input features X_i's
  - $g_1(x) = w_{1,1}x_1 + \cdots + w_{1,n}x_n$
- Output of neuron *N* is a non-linear activation
  - Sigmoid: $h_1(x) = \frac{1}{1+e^{-g_1(x)}}$
- Input of the next neuron is also a linear combination
  - $f(x) = u_1 h_1(x) + \cdots + u_m h_m(x)$

Activation functions

Name	Plot	Function, $f(x)$	Derivative of f, $f'(x)$	Range
Logistic, sigmoid, or soft step		$\sigma(x) = \frac{1}{1+e^{-x}}$ [1]	$f(x)(1-f(x))$	$(0,1)$
tanh		$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - f(x)^2$	$(-1,1)$
Rectified linear unit (ReLU)[11]		$\begin{cases} 0 & \text{if } x \le 0 \\ x & \text{if } x > 0 \end{cases} = \max\{0, x\} = x\mathbf{1}_{x>0}$	$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$[0,\infty)$

wikipedia.com

Universal approximation theorem



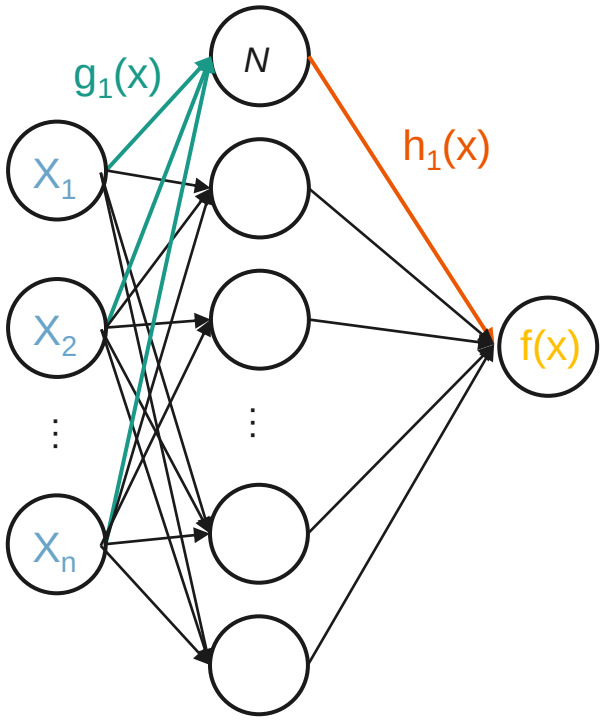
- Cybenko, 1989

- Any continuous function $y = f(x)$
- Any non-polynomial activation function
- Any small number ε

- There is an artificial neural network with only one hidden layer that can mimic the function $y = f(x)$ with maximum absolute error less than ε

https://towardsdatascience.com/can-neural-networks-really-learn-any-function-65e106617fc6

Artificial neural network is differentiable



- $g_1(x) = w_{1,1}x_1 + \cdots + w_{1,n}x_n$
- $h_1(x) = \frac{1}{1+e^{-g_1(x)}}$
- $f(x) = u_1 h_1(x) + \cdots + u_m h_m(x)$

- MSE: $\frac{1}{n}\sum_i \left(y_i - f(x)\right)^2$

- $\frac{\delta \mathrm{MSE}}{\delta w_{1,1}} = \frac{\delta \mathrm{MSE}}{\delta f}\frac{\delta f}{\delta h_1}\frac{\delta h_1}{\delta g_1}\frac{\delta g_1}{\delta w_{1,1}} + \cdots$
- Update with gradient descent

Beyond overfitting

Deep learning

10,000 data points
vs
10M parameters

5x + y = 13
-3x + 7y = 9

Unique solution!

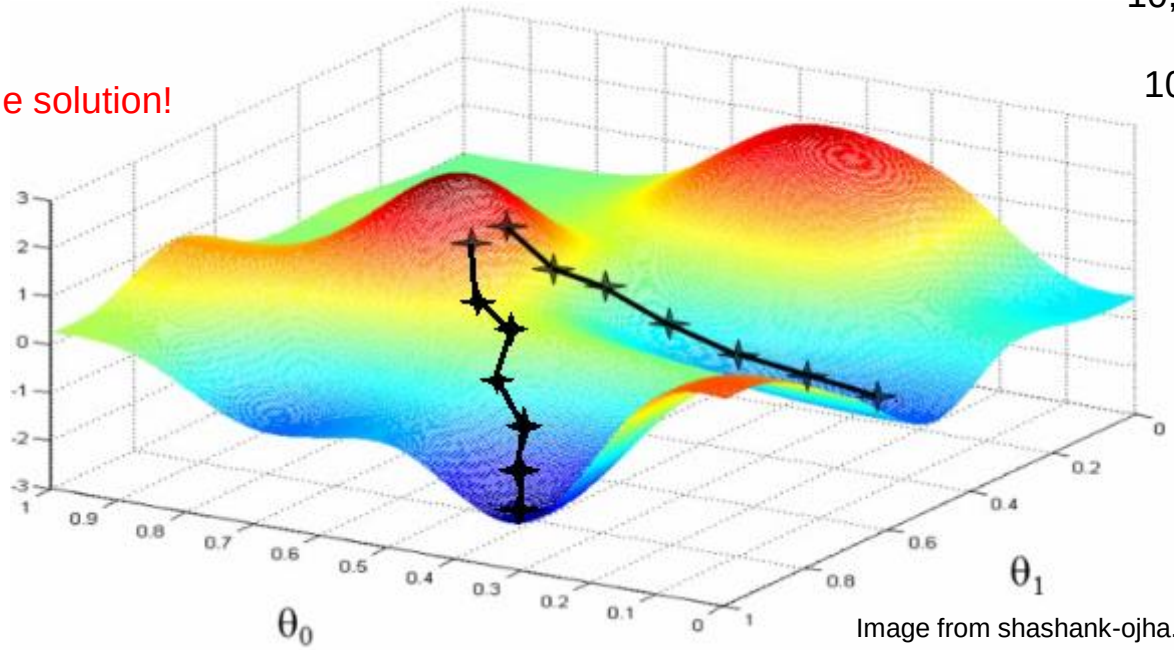


Image from shashank-ojha.github.io

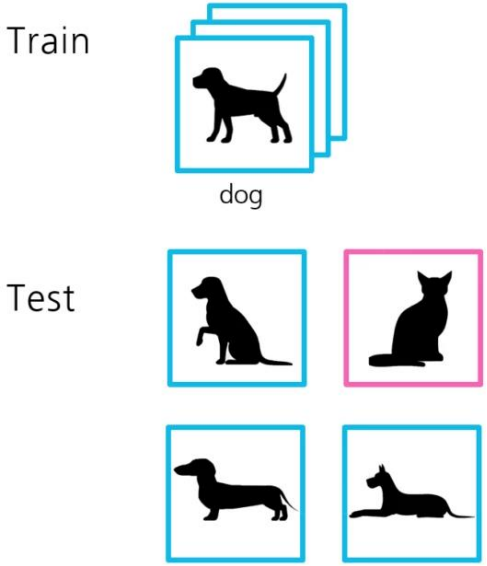
5x + y – 3z = 13
-3x + 7y + z = 9

Many solutions!

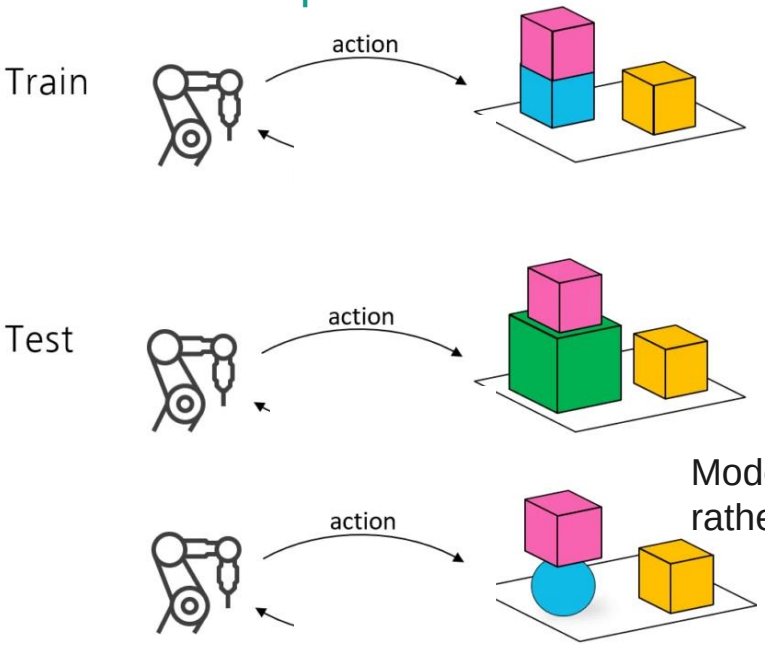
- More parameters than sample size = multiple equally-good solutions
- Some might generalize well, some might not

What the model learned (and not learned)

Out of distribution



Unintended pattern



Model uses color rather than shape

Image from https://safe-intelligence.fraunhofer.de/en/articles/out-of-distribution-detection-for-reinforcement-learning

Any question?

- See you next week on September 26th 9-10:30am