



CSCI E-82


Advanced Machine Learning,
Data Mining & Artificial Intelligence
Lecture 8

Classification, Part 2

Peter V. Henstock

Fall 2018

© 2018 Peter V. Henstock



What is Machine Learning?

© 2018 Peter V. Henstock

Machine Learning

- Set of methods that enables solving problems that traditional programming cannot solve
- Can you write a program to recognize Chinese characters in images?
- Can you write a program to assist oncologists recognize tumors?
- Can you recommend songs to strangers?
- Can you assess credit card transactions?
- What enables you to recognize Picasso's or Mozart's style?

© 2018 Peter V. Henstock

Machine Learning Alternative

- Don't code a set of instructions
- Allow machine learning methods to train for a given task on a large set of data
 - Very little coding
 - Lots of information
- If problem changes, give it new data
- Much cheaper approach

© 2018 Peter V. Henstock

Traditional Computing

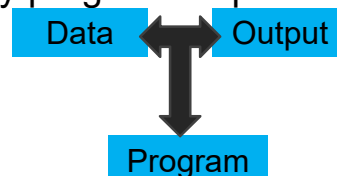
- Computers have traditionally been used as fast calculators
- Fast calculations have been used to execute series of instructions provided by developers



© 2018 Peter V. Henstock

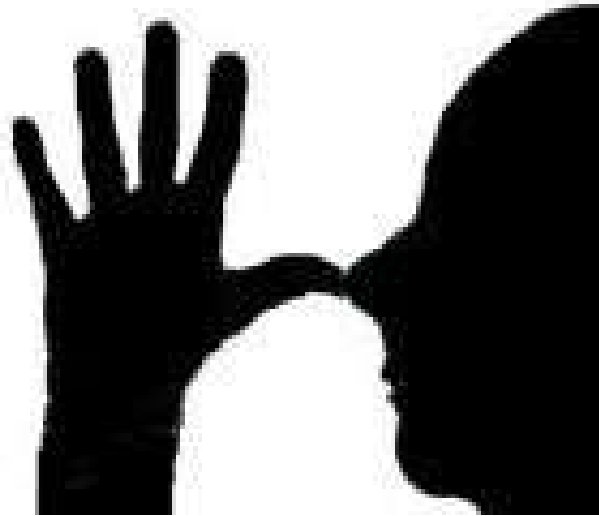
Machine Learning Paradigm Shift

- Traditional
- Machine Learning
 - Instructions are not provided
 - Data provides the “instructions”
 - Infer instructions from the data
 - Can apply program for predictions, etc.



© 2018 Peter V. Henstock

My classifier is better than yours...



© 2018 Peter V. Henstock

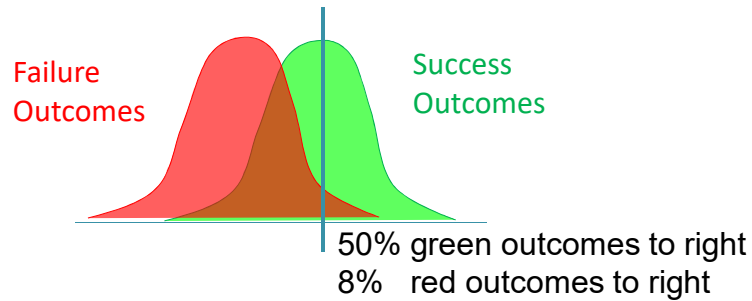
Limitations of Confusion Matrix

Confusion Matrix	Predicted True	Predicted False	
Actually True	TP	FN	$P = TP + FN$
Actually False	FP	TN	$N = FP + TN$
	$P' = TP + FP$	$N' = FN + TN$	

- One fundamental issue that we have with a confusion matrix by its nature
- What does it assume?

© 2018 Peter V. Henstock

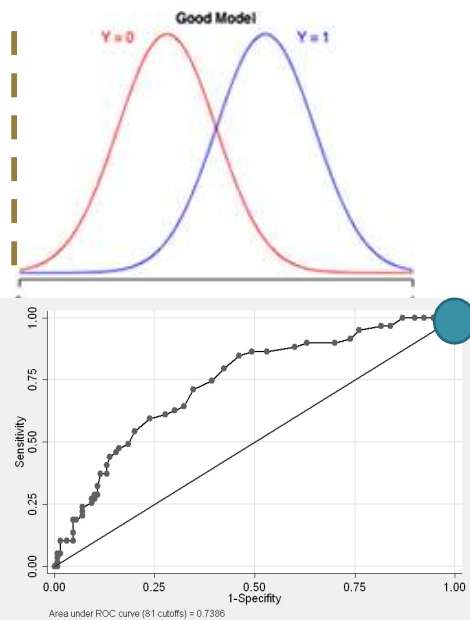
How to assess classification scores?



- Want to summarize & compare
- If threshold at blue line:
 - Correctly predicted 50% of green data to right
 - Incorrectly predicted 8% of red data to right
- Where to draw the line?

© 2010 PETER V. HENSTOCK

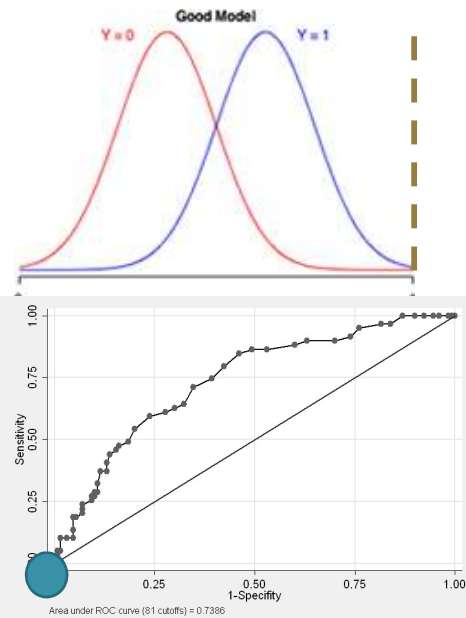
From Histogram to ROC



- Each point on ROC is a threshold
- Y= fraction of blue to the right of threshold
- X = fraction of red to right of threshold

© 2018 Peter V. Henstock

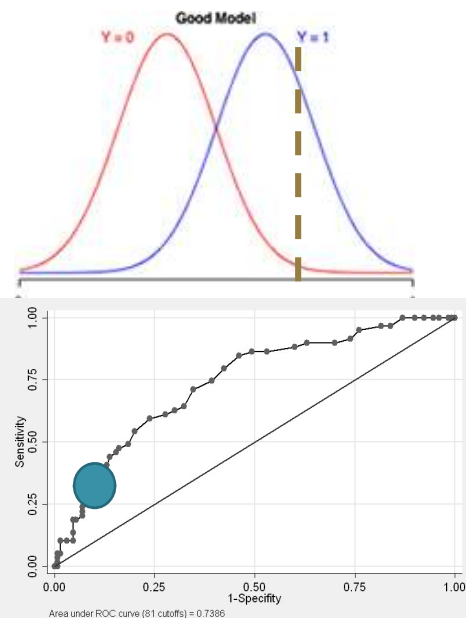
From Histogram to ROC



- Each point on ROC is a threshold
- Y= fraction of blue to the right of threshold
- X = fraction of red to right of threshold

© 2018 Peter V. Henstock

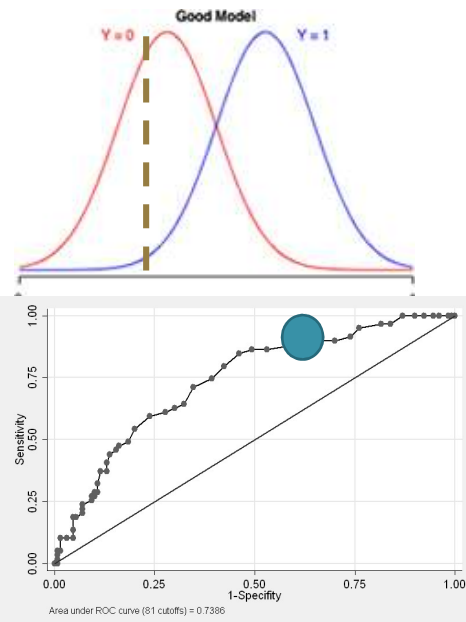
From Histogram to ROC



- Each point on ROC is a threshold
- Y= fraction of blue to the right of threshold
- X = fraction of red to right of threshold

© 2018 Peter V. Henstock

From Histogram to ROC



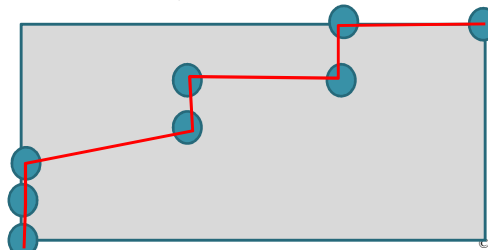
- Each point on ROC is a threshold
- Y = fraction of blue to the right of threshold
- X = fraction of red to right of threshold

© 2018 Peter V. Henstock

How I create ROC

True	False
98	
97	
85	
	85
	85
75	
	51
	51
25	
	22

- Sort based on score
- Assume True are higher
- Start bottom left (0,0)
- Loop in decreasing order
 - If True, 1/T upward
 - If False, 1/F to right
 - If ties, include cumulative



© 2018 Peter V. Henstock

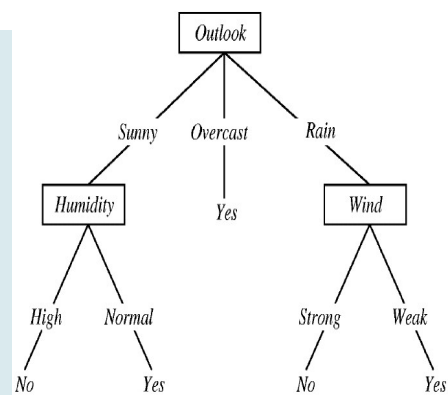
Decision Trees

© 2018 Peter V. Henstock

Decision Tree Classifier Example

Data Matrix **Training Examples** Label

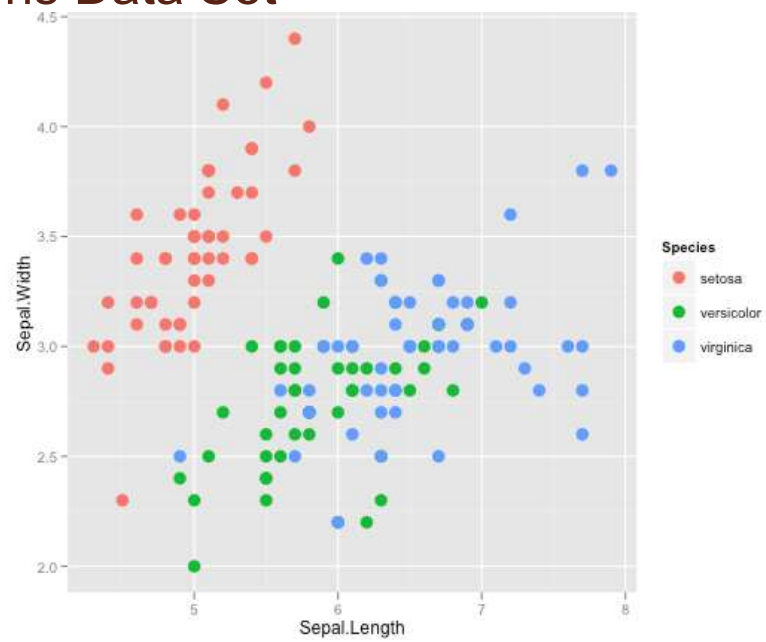
Day	Outlook	Temp	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



- Which is the most important feature?
- Which is the least important feature?

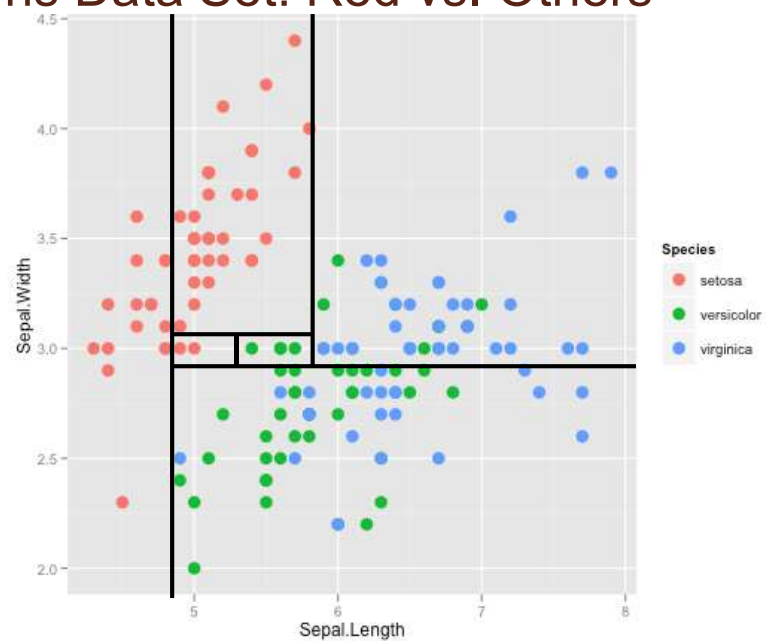
©2017 Peter V. Henstock

Iris Data Set



© 2018 Peter V. Henstock

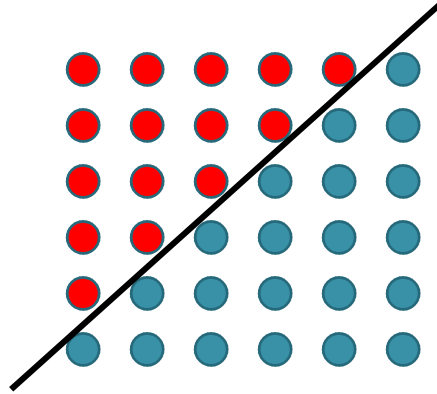
Iris Data Set: Red vs. Others



© 2018 Peter V. Henstock

What can decision trees do?

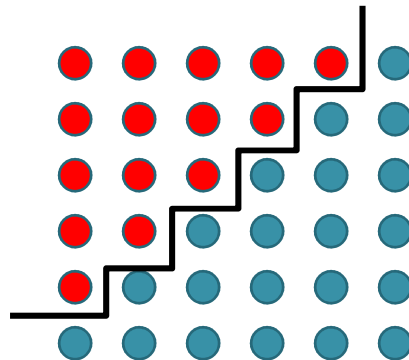
- Can we learn this line from data?



© 2018 Peter V. Henstock

What can decision trees do?

- Can we learn this line from data?



© 2018 Peter V. Henstock

Impurity Measures

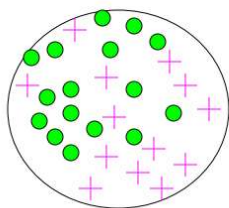
- Error rate
- Entropy of distribution
 - $H = \sum_{pts} P_{class}(pts) \log_2[P_{class}(pts)]$
 - Entropy is 0 when classes are pure
 - Entropy is 1 when equally split
- Gini Index
 - $Gini = \sum_{pts} P_{class}(pts) (1 - P_{class}(pts))$

© 2018 Peter V. Henstock

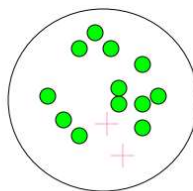
Impurity Measure

- (No, we are not talking about your college fraternity purity test)

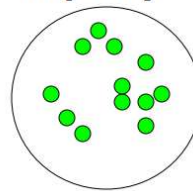
Very impure group



Less impure



Minimum impurity

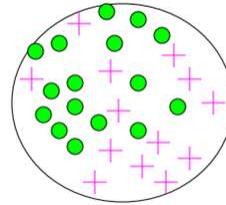


- <http://homes.cs.washington.edu/~shapiro/EE596/notes/InfoGain.pdf>

© 2018 Peter V. Henstock

Entropy Measure

- Entropy = $-\sum_i p_i \log_2(p_i)$
- Entropy = measure of chaos

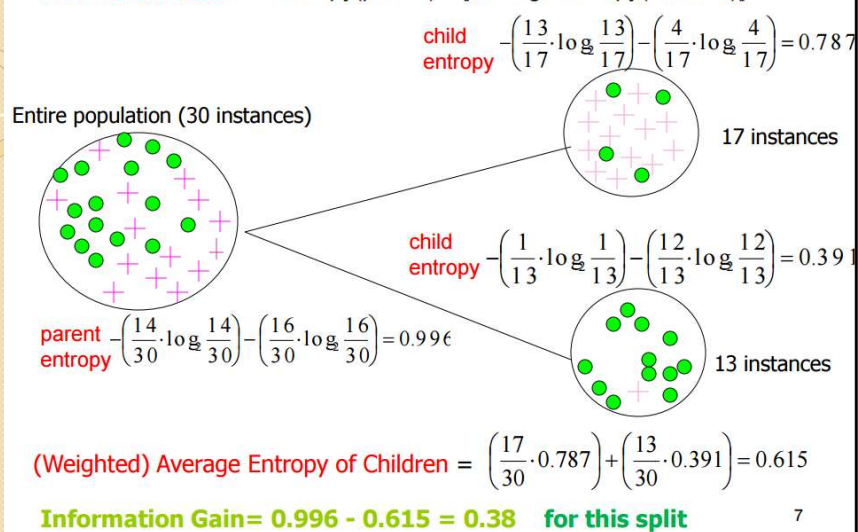


- 16/30 green circles; 14/30 purple “+”
- Entropy = $-\frac{16}{30} \log_2(\frac{16}{30}) - \frac{14}{30} \log_2(\frac{14}{30})$
- Entropy = 0.99
- Superior way of measuring impurity

© 2018 Peter V. Henstock

Calculating Information Gain

Information Gain = entropy(parent) – [average entropy(children)]



- <http://homes.cs.washington.edu/~shapiro/EE596/notes/InfoGain.pdf>

© 2018 Peter V. Henstock

Pruning

Why prune decision trees?



• <http://www.hydrangeashydrangeas.com/pruning.html>

© 2018 Peter V. Henstock

Bias Variance Tradeoff

- In decision trees, where will overfitting manifest itself?

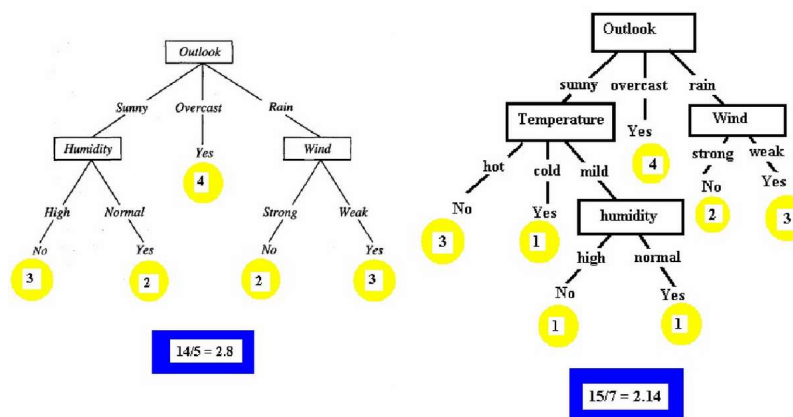
© 2018 Peter V. Henstock

Where will overfitting happen?

- In a decision tree that is trained on your training data, where will overfitting manifest itself?
 - 1) Errors from the training examples
 - Unavoidable error in labels or features
 - 2) Sparsely populated nodes and leaves
 - Insufficient training data
 - More complex tree structures

© 2018 Peter V. Henstock

Occam's Razor: Population / Leaf



- <http://www.slideshare.net/cnu/machine-learning-lecture-3>

© 2018 Peter V. Henstock

Pruning Decision Trees

- Replace a subtree near leaf with a leaf
- Goal is to lower the predicted error rate
- Remove parts of tree that don't contribute to classification of test data
- Can use new samples if available

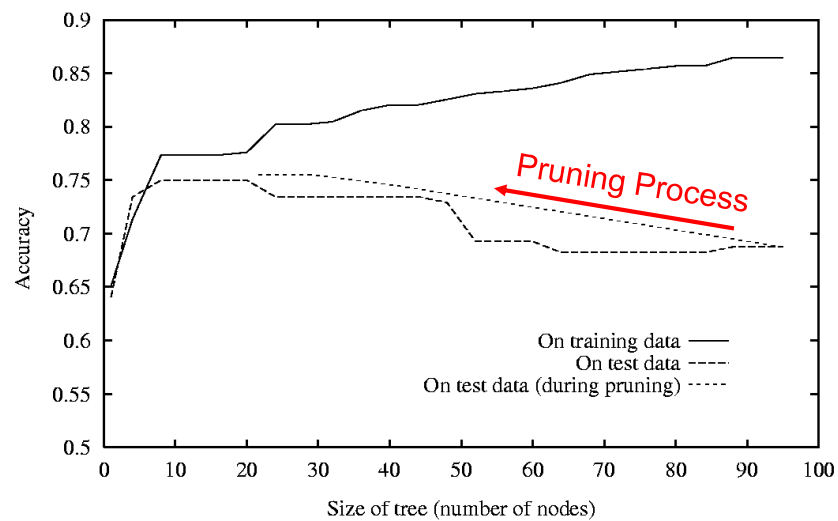
© 2018 Peter V. Henstock

Pruning

- Prepruning stops the tree from growing
 - Reduces computation
 - Stop using a chi-squared test
 - Asking whether distribution before and after adding the node is the same
- Postpruning prunes a fully grown tree
 - More common approach
 - Requires more computation
 - Generally more reliable than prepruning
 - Approach used by C4.5

© 2018 Peter V. Henstock

Effect of Pruning



- <http://jmvidal.cse.sc.edu/talks/decisiontrees/reducederrorprun.html>

© 2018 Peter V. Henstock

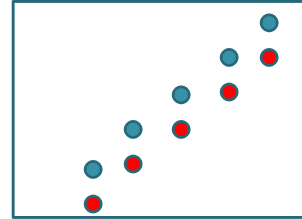
Good and Bad of Decision Trees

- Good
- Bad
- Ugly

© 2018 Peter V. Henstock

Good and Bad of Decision Trees

- Good:
 - No assumptions about distributions of data
 - Fairly robust compared to statistical approaches
- Bad:
 - How many rules needed?
 - Hard limiter at each node
- Ugly:
 - 9 out of 10 dentists recommend flossing
 - Lupus is a collection of 17 or 24 symptoms
 - No good way to represent this in decision tree



© 2018 Peter V. Henstock

Naïve Bayes

© 2018 Peter V. Henstock

Bayes Rule Revisited

- $P(\text{Class})$ = prior
- $P(\text{Evidence} \mid \text{Class})$
- $P(\text{Class} \mid \text{Evidence})$ = posterior
- $P(\text{Class} \mid \text{Evidence}) = \frac{P(\text{Class}, \text{Evidence})}{P(\text{Evidence})}$
- $P(\text{Class} \mid \text{Evidence}) = \frac{P(\text{Evidence} \mid \text{Class}) P(\text{Class})}{P(\text{Evidence})}$

© 2018 Peter V. Henstock

Naïve Bayes Classifier

- Features: $f_1 \dots f_N$
- Train up a system
- Chose the most probable class given the set of features
- MAP = maximum a posterior probability
- $\text{Classi}_{\text{MAP}} = \text{argmax}_{\text{classi}} P(\text{classi} \mid f_1 \dots f_N)$
- $= \text{argmax}_{\text{classi}} \frac{P(f_1 \dots f_N \mid \text{classi}) P(\text{classi})}{P(f_1 \dots f_N)}$
- Can ignore denominator since same for each of the classes

© 2018 Peter V. Henstock

Naïve Bayes Classifier

- $\text{Classi}_{\text{MAP}} = \text{argmax}_{\text{classi}} P(\text{classi} | f_1 \dots f_N)$
 - Class = author (Shakespeare)
 - $f_1 \dots f_N$ = set of word frequencies
 - Impossible to compute as would have to identify all possible combinations of $f_1 \dots f_N$ in order to determine class
- $P(\text{classi})$ is easy
- $P(f_1 \dots f_N | \text{classi})$ is impossible
 - Every combination of features for a class
 - All combinations of words to determine if it's written by Shakespeare
 - Can't generalize it—have to see all combos

Naïve Bayes Assumption

- Features are conditionally independent given the class
- $P(f_1 \dots f_N | \text{classi}) = \prod_j p(f_j | \text{classi})$
- Is this a good assumption?

Naïve Bayes Assumption

- Features are conditionally independent given the class
- $P(f_1 \dots f_N \mid \text{class}_i) = \prod_j p(f_j \mid \text{class}_i)$
- Naïve Bayes classifier:
 - $\text{Class}_i = \underset{\text{class}_i}{\text{argmax}} P(\text{class}_i) \prod_j p(\text{feat}_j \mid \text{class}_i)$
 - $P(\text{feat}_j \mid \text{class}_i)$ would be probability of “thou” occurring in Shakespeare author class

© 2018 Peter V. Henstock

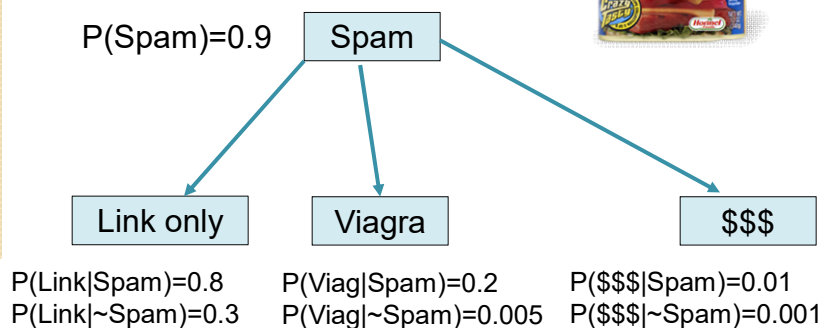
Naïve Bayes Assumption

- Features are conditionally independent given the class
- $P(f_1 \dots f_N \mid \text{class}_i) = \prod_j p(f_j \mid \text{class}_i)$
- Naïve Bayes classifier:
 - $\text{Class}_i = \underset{\text{class}_i}{\text{argmax}} P(\text{class}_i) \prod_j p(\text{feat}_j \mid \text{class}_i)$
 - How should you actually program this?

© 2018 Peter V. Henstock

Generative Model

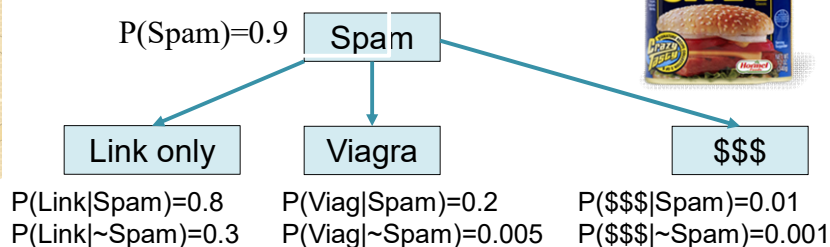
- Classic example: spam filtering
- Generative model:



Should $P(\text{Link}|\text{Spam})$ and $P(\text{Link}|\sim\text{Spam})$ add up to 1.0?

© 2018 Peter V. Henstock

Generative Model



$$P(\text{Spam} \mid \text{"Save $$$ on Viagra"}) = P(\text{Spam} \mid \sim\text{Link}, \text{Viag}, \text{$$$})$$

$$\rightarrow P(\sim\text{Link}, \text{Viagra}, \text{$$$} \mid \text{Spam}) P(\text{Spam}) / k$$

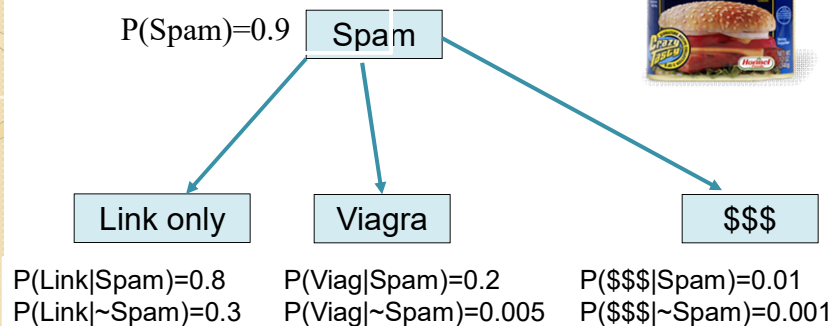
In Naïve Bayes \rightarrow independent conditional probabilities

$$P(\sim\text{Link}|\text{Spam})P(\text{Viagra}|\text{Spam})P(\text{$$$}|\text{Spam})P(\text{Spam})$$

- $(1-0.8)*(0.2)*(0.01)*(0.9) = 0.00036 / k$
- Does this seem low?

© 2018 Peter V. Henstock

Generative Model



- $P(\text{Spam} \mid \text{"Save \$\$ \$ on Viagra"}) = P(\text{Spam} \mid \sim\text{Link}, \text{Viag}, \text{\$}\$ \$)$
- $P(\sim\text{Link}, \text{Viagra}, \text{\$}\$ \$ \mid \text{Spam}) P(\text{Spam})/k$
- In Naïve Bayes \rightarrow independent conditional probabilities
- $P(\sim\text{Link}|\text{Spam})P(\text{Viagra}|\text{Spam})P(\text{\$}\$ \$|\text{Spam})P(\text{Spam})$
 - $(1-0.8)*(0.2)*(0.01)*(0.9) = 0.00036 / k$
- $P(\sim\text{Link}|\sim\text{Spam})P(\text{Viagra}|\sim\text{Spam})P(\text{\$}\$ \$|\sim\text{Spam})P(\sim\text{Spam})$
 - $0.7*0.005*0.001*0.1 = 3.5\text{E-}7 / k$

© 2018 Peter V. Henstock

Naïve Bayes

- $P(\text{Class} \mid \text{feature}_1, \dots, \text{feature}_N)$
- $= \prod_i P(\text{feature}_i \mid \text{Class}) * P(\text{Class})/k$
- Simple solution for NP-hard problem but easy with Naïve Bayes
- MAP = maximum a posterior
- $\text{MAP}(\text{spam}) = \text{argmax}_{\text{Class}} \prod_i P(\text{feature}_i \mid \text{Class}) P(\text{Class})$
 - Essentially removed our norm constant k
 - Gives us a classifier

© 2018 Peter V. Henstock

Naïve Bayes

- $P(\text{feature}_i | \text{Class}) = \frac{P(\text{feature}_i, \text{Class})}{P(\text{Class})}$
- What if you don't encounter $P(\text{feature}_i | \text{Class})$ in your data?

© 2018 Peter V. Henstock

Naïve Bayes

- $P(\text{feature}_i | \text{Class}) = \frac{P(\text{feature}_i, \text{Class})}{P(\text{Class})}$
- What if you don't encounter $P(\text{feature}_i | \text{Class})$ in your data?
- $P(f_j | \text{class}_i) = \frac{|f_j \cap \text{class}_i|}{|\text{class}_i|} \rightarrow \frac{|f_j \cap \text{class}_i| + mp}{|\text{class}_i| + m}$
- m = general weight given to prior
- p = prior estimate for $P(f_j | \text{class}_i)$
- Similar idea to Yates correction in chi-sq

© 2018 Peter V. Henstock

Range of values for Naïve Bayes

- Recall that Naïve Bayes is a product of many independent conditional probabilities
- What will the typical range of values be for the output?

© 2018 Peter V. Henstock

Range of values for Naïve Bayes

- Recall that Naïve Bayes is a product of many independent conditional probabilities
- What will the typical range of values be for the output?
- Removed all the correlations between values so getting repeated entries
- Result tends to be near 0.0 or 1.0
 - Overconfident in its prediction

© 2018 Peter V. Henstock

Example for Tweet Analysis

- Examined tweets from two girls over a one month period

	Mary	Alice
OMG	0.6	0.3
LOL	0.5	0.6
BFF	0.2	0.3

- Cell = Fraction of tweets containing this term
- $P(\text{Mary}) = P(\text{Alice}) = 0.5$
- Who sent a tweet “OMG what is he thinking? LOL”

©2017 Peter V. Henstock

Example for Tweet Analysis

- $P(\text{Mary}) = P(\text{Alice}) = 0.5$

- Who sent a tweet with LOL and OMG?

	Mary	Alice
OMG	0.6	0.3
LOL	0.5	0.6
BFF	0.2	0.3

- Naïve Bayes treats each term independently
- Essentially looking for posterior:
- $$P(\text{Class}|\text{Evidence}) = \frac{P(\text{Evidence}|\text{Class})P(\text{Class})}{P(\text{Evidence})}$$
- $P(\text{Evidence})$ is usually constant $\rightarrow 1$
- $P(\text{Evidence}|\text{Mary}) = 0.5 * 0.6 * 0.8 \rightarrow 0.15$
- $P(\text{Evidence}|\text{Alice}) = 0.6 * 0.3 * 0.7 \rightarrow 0.09$
- MAP Mary|Evidence = $0.15 * P(\text{Mary}) = 0.075$
- MAP Alice|Evidence = $0.09 * P(\text{Alice}) = 0.045$

©2017 Peter V. Henstock

What about probabilities?

- MAP Mary|Evidence = $0.15 \cdot P(\text{Mary}) = 0.075$
- MAP Alice|Evidence = $0.09 \cdot P(\text{Alice}) = 0.045$
- To compute the posterior probabilities:
 - Posteriors are normalized to 1.0 with space of two authors
- $P(\text{Mary} \mid \text{LOL,OMG,}\sim\text{BFF}) = 0.075 / (0.075 + 0.045) = 0.625$
- $P(\text{Alice} \mid \text{LOL,OMG,}\sim\text{BFF}) = 0.045 / (0.075 + 0.045) = 0.375$

© 2018 Peter V. Henstock

Banko & Brill 2001 ACL

- Everyone cites this paper

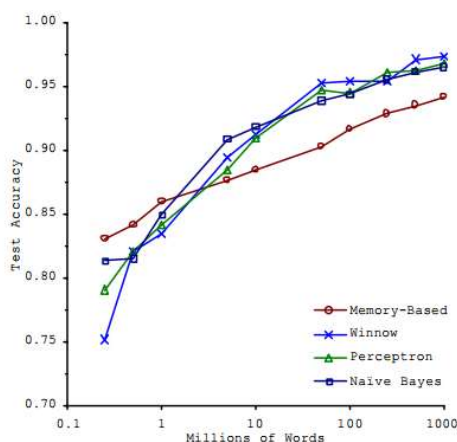


Figure 1. Learning Curves for Confusion Set Disambiguation

© 2018 Peter V. Henstock

Ensemble Methods

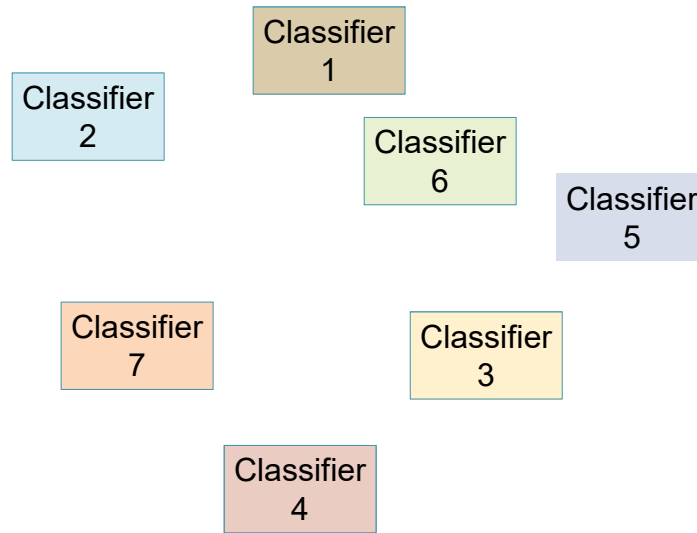
© 2018 Peter V. Henstock

Decision Tree

- So far, feed all the data into one classifier in the form of a single tree
- What if you had multiple classifiers?
 - How could one come up with classifiers?
 - Or multiple decision trees?

© 2018 Peter V. Henstock

Handling Multiple Classifiers



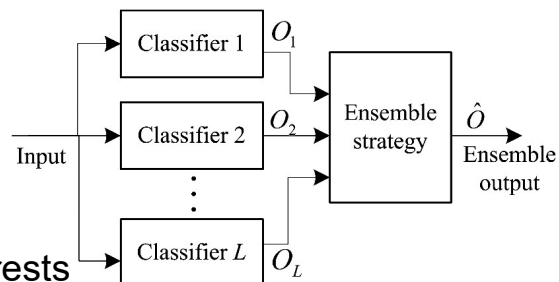
© 2018 Peter V. Henstock

Ensemble Methods

- Combination of multiple learned classifiers together to create a (better) composite classifier

- Examples:

- Bagging
- Random forests
- Boosting
- Error correcting
- Stacking



<http://opticalengineering.spiedigitallibrary.org/article.aspx?articleid=1352706>

© 2018 Peter V. Henstock

Bagging = Bootstrap Aggregation

- Designed to increase accuracy
- You ask your friends what's their favorite stock picks and several recommend AAPL
- Bagging of classifiers is equivalent
- Sample the tuples k times with replacement
- Train on samples and test on the rest
- Vote on the test tuples

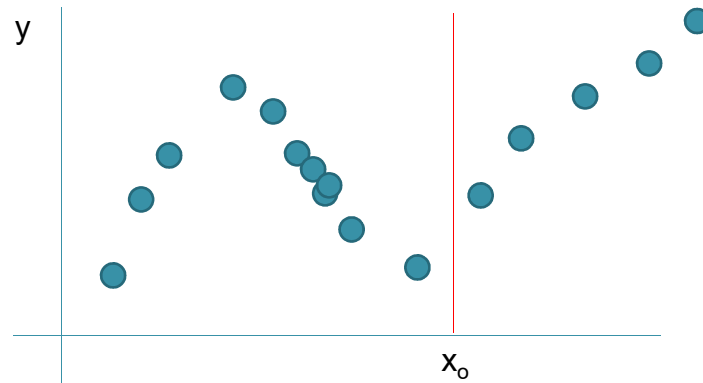
© 2018 Peter V. Henstock

Bootstrap Aggregation = Bagging

- Improves any classifier by aggregating multiple versions of a given classifier
- Resample by drawing k points at random from the original data set for training and make one prediction
- Draw samples randomly (uniform)
- Each sampling creates one prediction

© 2018 Peter V. Henstock

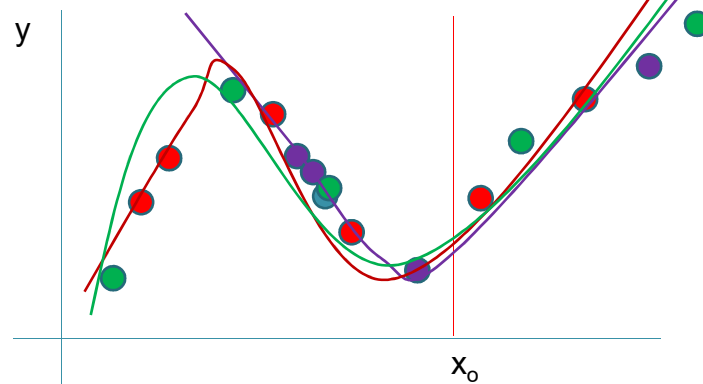
Bagging for Regression



- Here is the full data set
- What is the predicted value of y at x_0 ?
- What is the confidence?

© 2018 Peter V. Henstock

Bagging for Regression



- Resample points with replacement
- Fit multiple curves
- Average results

© 2018 Peter V. Henstock

Bagging for Regression

- If you were trying to estimate y , what two properties would you really like to have for that estimate?
- 1)
- 2)

© 2018 Peter V. Henstock

Bagging for Regression

- If you were trying to estimate y , what two properties would you really like to have for that estimate?
- 1) Unbiased:
 - Mean of estimate = mean of population
- 2) Low variance

© 2018 Peter V. Henstock

Bagging Math

- Sample rows from pairs of X,Y randomly
X=features Y=labels
- Each sample, estimate y_{est} at x_o point
- Let's assume each y_{est} is unbiased
- Compute the mean
 - $E(y_{est}) = f(x_o)$ by definition of unbiased
- Compute the variance
 - $E[(y_{est} - y_o)^2] = E[y_{est}^2 - 2y_{est}y_o + y_o^2] = Var(y_{est})$

© 2018 Peter V. Henstock

Bagging Math

- $y_{est}^{(i)}$ is estimate for “bag” i of k “bags”
- $Z = \frac{1}{k} \sum_{i=1}^k y_{est}^{(i)}$
- Compute the mean
 - $E[Z] = \frac{1}{k} \sum_{i=1}^k E[y_{est}^{(i)}] = f(x_o)$ unbiased
- Compute the variance
 - $E[(Z - y_{est})^2] = Var(Z) = Var[\frac{1}{k} \sum_{i=1}^k y_{est}^{(i)}]$
 - $= \frac{1}{k^2} Var[\sum_{i=1}^k y_{est}^{(i)}] = \frac{1}{k^2} \sum_{i=1}^k Var[y_{est}^{(i)}] = \frac{1}{k} Var(y_{est})$
- This means
 - Bagging estimate is an unbiased estimate
 - Variance of estimate is reduced by taking more samples k

© 2018 Peter V. Henstock

Bagging for Regression

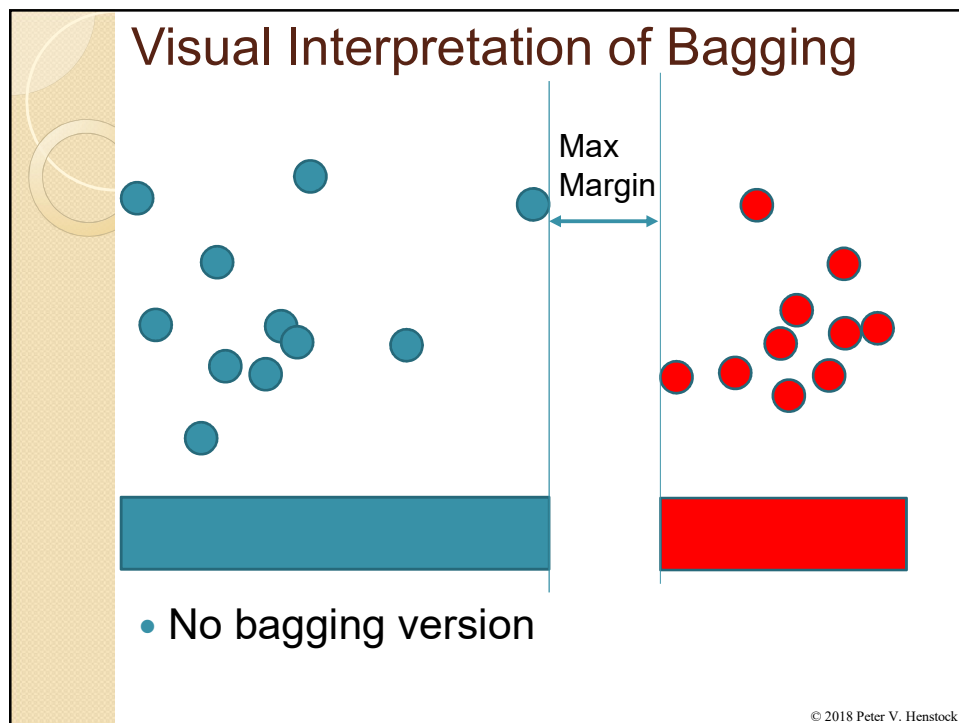
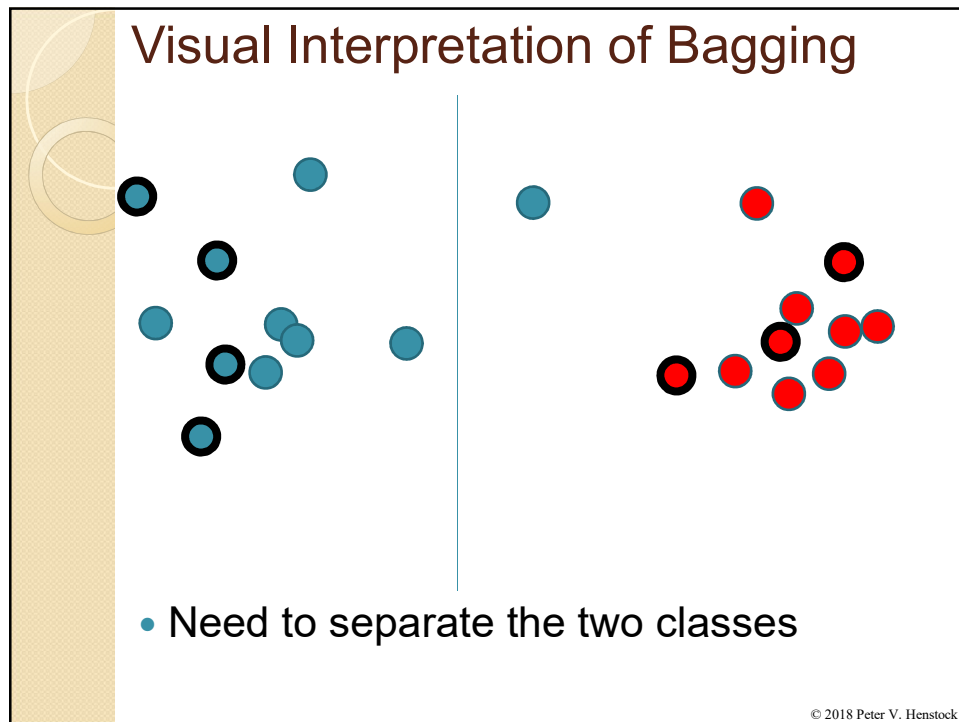
- Results of the math (not shown) give two interesting facts:
 - 1) Mean value of estimate is unbiased
 - 2) Variance is σ^2/k where $k=\text{\#resamples}$
- This means we can come up with a good estimate for $f(x)$ and the variance of that estimate decreases with the number of resampling steps \rightarrow small

© 2018 Peter V. Henstock

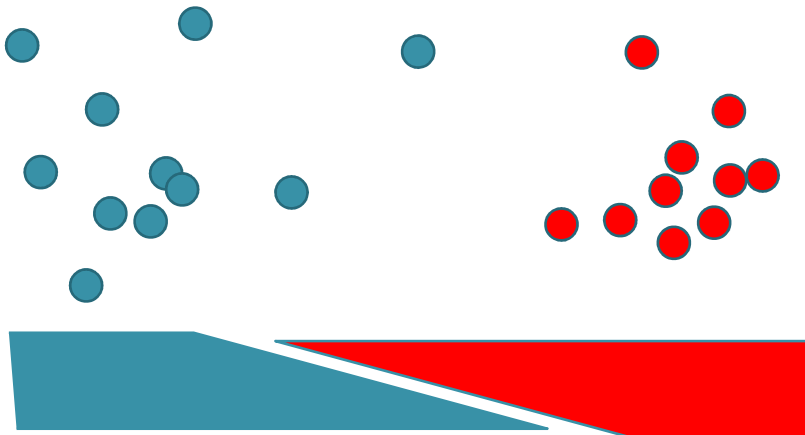
Bagging Compared

- Takes longer to produce the results
- Results are more accurate
- Results are more robust to outliers
- Can also be used to predict continuous values by taking averages of predictions

© 2018 Peter V. Henstock



Visual Interpretation of Bagging



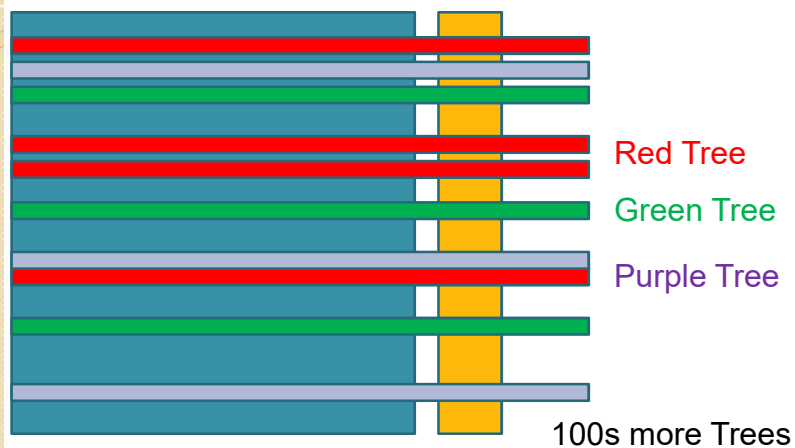
- Bagging version samples points and gives an extended confidence area

© 2018 Peter V. Henstock

Bagged Tree

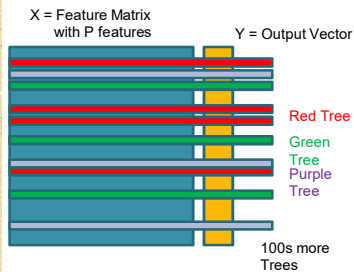
X = Feature Matrix
with P features

Y = Output Vector



© 2018 Peter V. Henstock

Random Forest



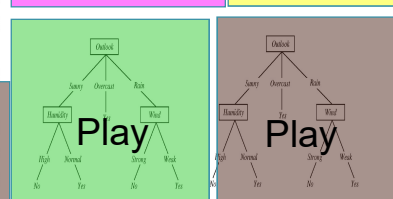
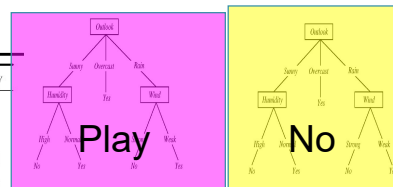
- For each tree
 - Sample rows from the X and Y
 - **For each node in tree: sample features**
 - Choose to split on best feature from sample
 - Construct decision tree without pruning (C4.5)
- Testing: vote using predictions of all trees

© 2018 Peter V. Henstock

Random Forest

Training Examples

Day	Outlook	Temp	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



Voted 75% chance of play

© 2017 Peter V. Henstock

Random Forest

Training Examples

Day	Outlook	Temp	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Picture is wrong

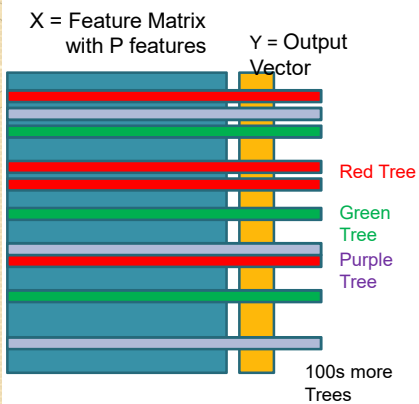
Samples subsets of rows for each tree

Rows need not be contiguous within data

Samples of columns are per-node not tree

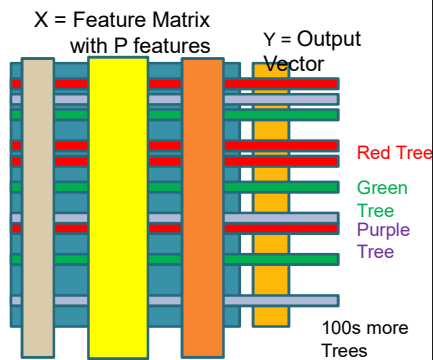
©2017 Peter V. Henstock

Bagged Tree



Number of trees
When to stop tree
Fraction sampled

Random Forest

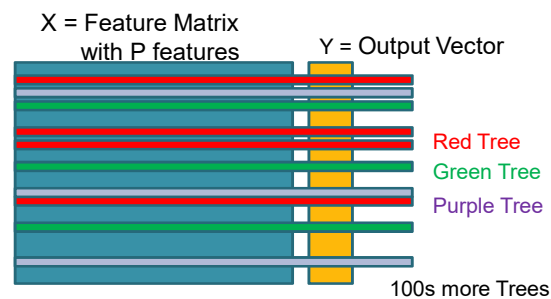


Number of trees
When to stop tree
Fraction sampled
#features per node

© 2018 Peter V. Henstock

OOB = Out of Bag Scoring

- Cross-validation divides data into k sets
 - $K-1$ sets used for training, 1 set for testing
- OOB: test results using samples not selected in the bagging for training
 - Very accurate and “free”



© 2018 Peter V. Henstock

Proximities

- Throw all data down each trained tree
- Compute the proximity of paired entries
 - Count how often the same rows end up in the same leaf
- Useful for
 - Replacing missing values
 - Inferring space of variables
 - Understanding class prototypes
 - Measure of sensitivity analysis of features
 - Finding outliers

© 2018 Peter V. Henstock

Random Forests

- Good
- Bad
- Ugly

© 2018 Peter V. Henstock

Random Forests

- Good
 - Among the top classifier performance
 - Produces a variable importance
 - Works well for imbalanced data sets
 - Don't have to decide on features in advance
 - Parallelizable
- Bad
 - Difficult to improve upon them
- Ugly
 - Random in nature so hard to understand
 - Voting is somewhat simple

© 2018 Peter V. Henstock

Random Forest in the News

© 2018 Peter V. Henstock

“Self-Learning” → curing cancer

Self-learning Computer Algorithms Enable Two-day Cancer Diagnosis

🕒 Wed, 10/07/2015 - 3:29pm

👤 by Mette Haagen Marcussen, Technical University of Denmark

✉️ Get the latest news in High Performance Computing, Informatics, Data Analysis Software and more - Sign up now!



Researchers have combined genetics with computer science and created a new diagnostic technology based on advanced self-learning computer algorithms that can identify the source of the disease with 85 percent certainty and, thus, target treatment.

In by far the majority of cancer cases, the doctor can quickly identify the source of the disease, for example cancer of the liver, lungs, etcetera. However, in about one in 20 cases, the doctor can confirm that the patient has cancer — but cannot find the source. These patients then face the prospect of a long wait with numerous diagnostic tests and attempts to locate the origin of the cancer before starting any treatment.

© 2018 Peter V. Henstock

Machine Learning News & Reality

- “Self-learning Computer Algorithms Enable Two-day Cancer Diagnosis”
- http://www.scientificcomputing.com/news/2015/10/self-learning-computer-algorithms-enable-two-day-cancer-diagnosis?et_cid=4878817&et_rid=328907261&location=top
- Now, researchers at * have combined genetics with computer science and **created a new diagnostic technology based on advanced self-learning computer algorithms** which — on the basis of a biopsy from a metastasis — can with 85 percent certainty **identify the source of the disease and, thus, target treatment and, ultimately, improve the prognosis for the patient.**
- TumorTracer by Marquardt et al. 2015
- <http://www.biomedcentral.com/1755-8794/8/58>

© 2018 Peter V. Henstock

Actual paper: Methods

We used **publicly available somatic mutation data** from the COSMIC database to **train random forest classifiers** to distinguish among those tissues of origin for which sufficient data was available. We selected feature sets using cross-validation and then derived two final classifiers (with or without copy number profiles) **using 80 % of the available tumors. We evaluated the accuracy using the remaining 20 %.**

© 2018 Peter V. Henstock

Results of Article

“The **cross-validation accuracy** was highest when all three types of information were used. On the left-out COSMIC data not used for training, we achieved a classification accuracy of **85 %** across 6 primary sites (with copy numbers), and **69 %** across 10 primary sites (...). Importantly, a derived confidence score could distinguish tumors that could be identified with **95 %** accuracy (32 %/75 % of tumors w w/o copy numbers) from those that were less certain. Accuracy in the **independent data sets** was **46 %**, **53 %** and **89 %** respectively, similar to the accuracy expected from the training data.”

© 2018 Peter V. Henstock

Boosting

© 2018 Peter V. Henstock

Error Correcting Codes

© 2018 Peter V. Henstock

How do you do multiple classes?

© 2018 Peter V. Henstock

Handling Multiple Labels

- Multiple approaches to this problem
- Decision trees with multiple labels are available in python and R
- A common alternative is to use multiple decision trees:
 - A vs. (B and C)
 - B vs. (A and C)
 - C vs. (A and B)

© 2018 Peter V. Henstock

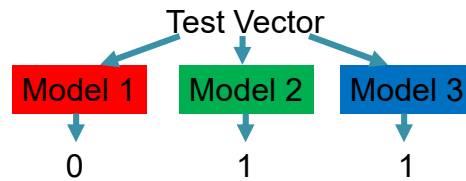
Error-Correcting Output Coding

- Apply binary classifiers to multiclass using an ensemble instead
- Sample from the output classes A,B,C,D
 - Sample AC → class0 BD → class1
 - Sample AD → class0 BC → class1
 - Sample ABD → class0 C → class1
- Train up these classification models for the assigned 2-class problem

© 2018 Peter V. Henstock

Error-Correcting Output Coding

- Sample AC → class0 BD → class1
- Sample AD → class0 BC → class1
- Sample ABD → class0 C → class1

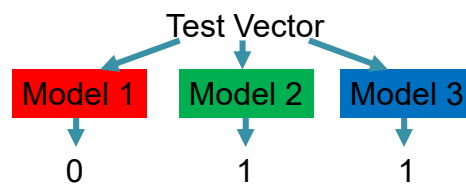


- So, which class output should it be?

© 2018 Peter V. Henstock

Error-Correcting Output Coding

- Sample AC → class0 BD → class1
- Sample AD → class0 BC → class1
- Sample ABD → class0 C → class1



- Each class is essentially a binary vector from the class assigned in training
 - A: 000 B=110 C=011 D=100
 - Predict class closest to the test vector output

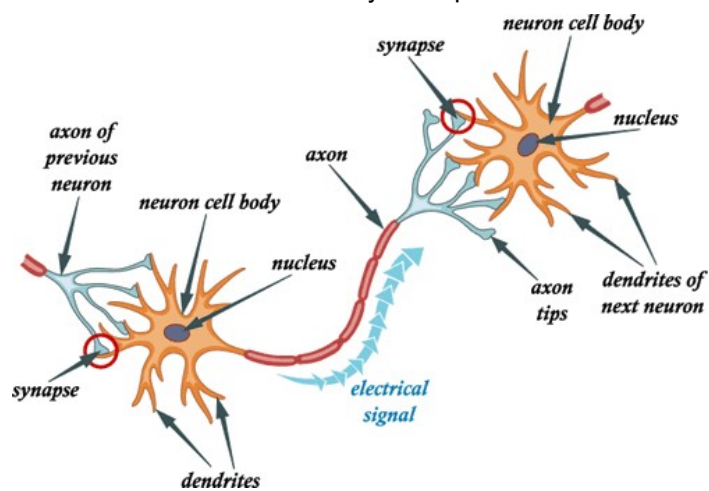
© 2018 Peter V. Henstock

Neural Networks

© 2018 Peter V. Henstock

Neurons

- <http://biology.stackexchange.com/questions/25967/nerves-neurons-axons-and-dendrites-by-example>



© 2018 Peter V. Henstock

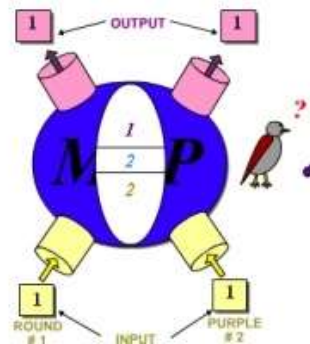
Synapse

- Charges travel along axon to synapse
- Charge causes vesicles of specific transmitter charges to cross synapse
- Receptors located on membrane
 - Bind to transmitted molecules
 - Conformation change of membrane
 - Specific ions are exchanged
- Synapses adapt
 - #vesicles and #receptors change

© 2018 Peter V. Henstock

McCulloch & Pitts 1943

- Concept of neuron being formal units within the brain
- Developed a mathematical model of neurons
- Built on Turing model



© 2018 Peter V. Henstock

Rosenblatt 1957

- Developed perceptron in 1957
- “*Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*” 1962
- First finite state machine
- First device that could learn



NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo
of Computer Designed to
Read and Grow Wiser

WASHINGTON, July 7 (UPI)—The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's \$2,000,000 “704” computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of \$100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. At 40, human be-

ings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

The Navy said the perceptron would be the first non-living mechanism “capable of receiving, recognizing and identifying its surroundings without any human training or control.”

The “brain” is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

1958 New York
Times...



In today's demonstration, the “704” was fed two cards, one with squares marked on the left side and the other with squares on the right side.

Learns by Doing

In the first fifty trials, the machine made no distinction between them. It then started registering a “C” for the left squares and “O” for the right squares.

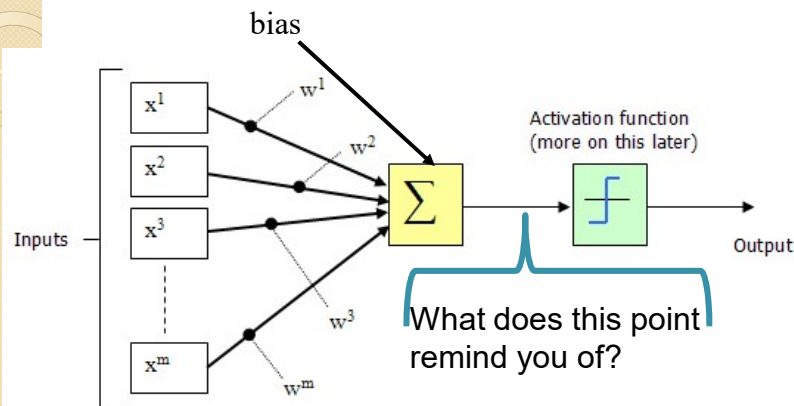
Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a “self-induced change in the wiring diagram.”

The first Perceptron will have about 1,000 electronic “association cells” receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.



© 2018 Peter V. Henstock

General model

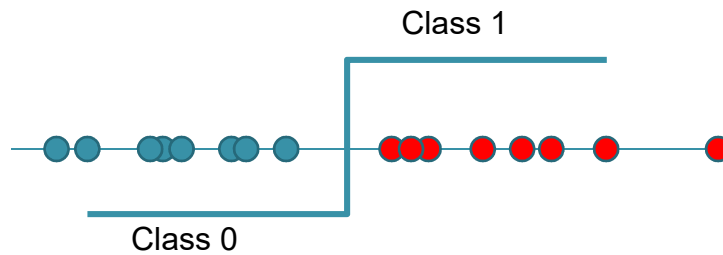


<http://www.codeproject.com/Articles/16419/AI-Neural-Network-for-beginners-Part-of>

© 2018 Peter V. Henstock

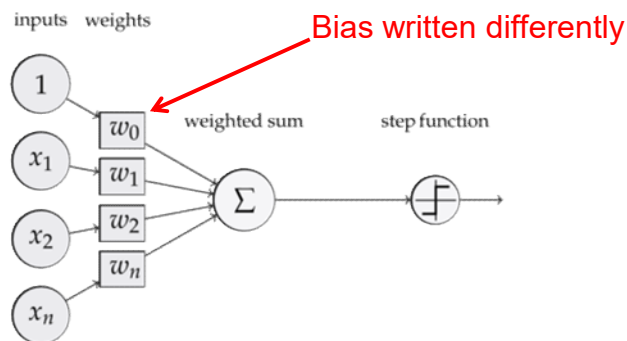
Notion of bias

- Bias shifts the threshold to appropriate place given the other weights
- Each dot
 - = sum of weighted inputs
 - = input from one data row



© 2018 Peter V. Henstock

Perceptron

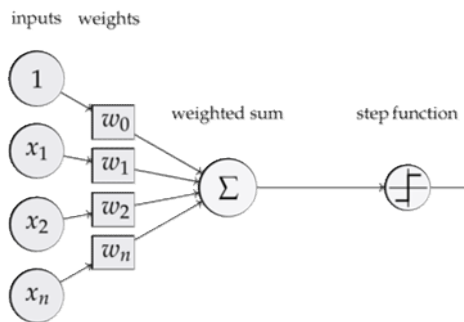


<https://blog.dbrgn.ch/2013/3/26/perceptrons-in-python/>

© 2018 Peter V. Henstock

How to Use Perceptron?

- Present a set of features to inputs
- Get a weighted sum \rightarrow step \rightarrow output
- Binary classifier as shown below
- Need to learn the weights to make the right decision



© 2018 Peter V. Henstock

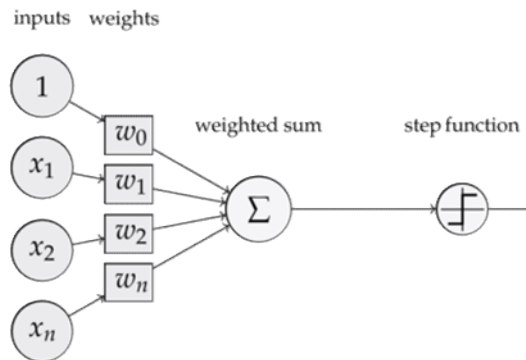
Perceptron Learning

- $w_i \leftarrow w_i + \Delta w_i$
- $\Delta w_i = \alpha(t - y)x_i$
 - t = target output (from training)
 - y = actual output
 - If target == output \rightarrow do nothing
 - If target != output \rightarrow adjust weight
 - Adjustment weight based on x_i
 - Different formulations has x_i in $\{-1, 1\}$

© 2018 Peter V. Henstock

How to Use Perceptron?

- Loop over “epochs”
 - Loop over data inputs (training rows):
 - Present features & output to perceptron
 - If weights are incorrect, modify them
- Learns a decision boundary



© 2018 Peter V. Henstock

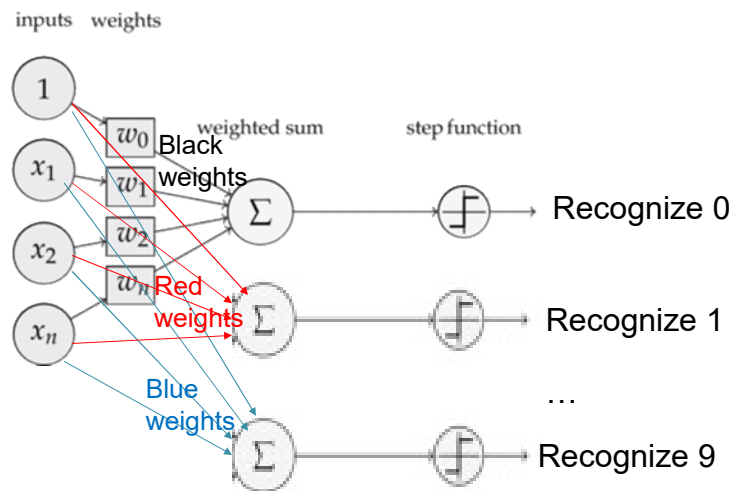
MNIST Digits



© 2018 Peter V. Henstock

What if we're recognizing multiple numbers?

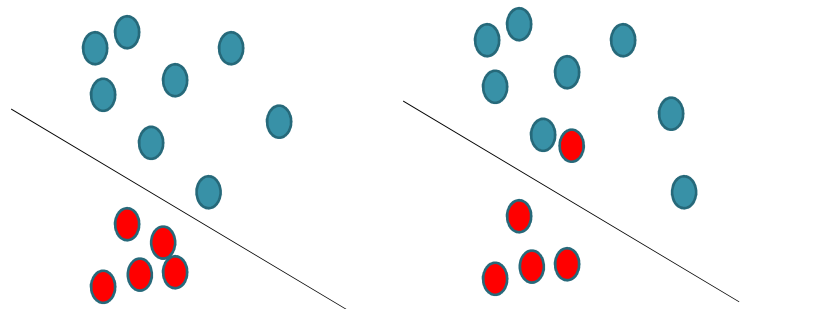
- Multiple sets of weights and outputs



© 2018 Peter V. Henstock

Perceptron Training Rule

- If the learning rate is sufficiently small
- It will converge if the training data are linearly separable
- What if there is noise?



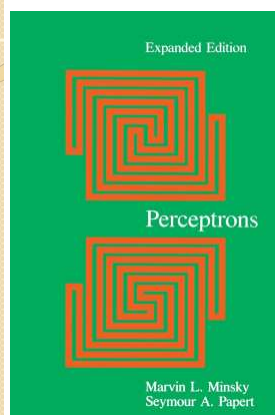
© 2018 Peter V. Henstock

Perceptron Learning

- Present all training cases to the input
- If output unit is correct, do nothing
- If output is wrong:
 - If output = 0, weights = weights + input
 - If output = 1, weights = weights – input
- Guaranteed to find set of weights that give the right answer...*if such a set exists*

© 2018 Peter V. Henstock

Perceptrons 1969



- Minsky & Papert
- Founded AI Lab @ MIT
- Developed logo
- Head-mounted display
- Confocal microscopy
- Randomly wired NN
- Book started the “AI Winter”
 - Lasted 12-15 years

© 2018 Peter V. Henstock

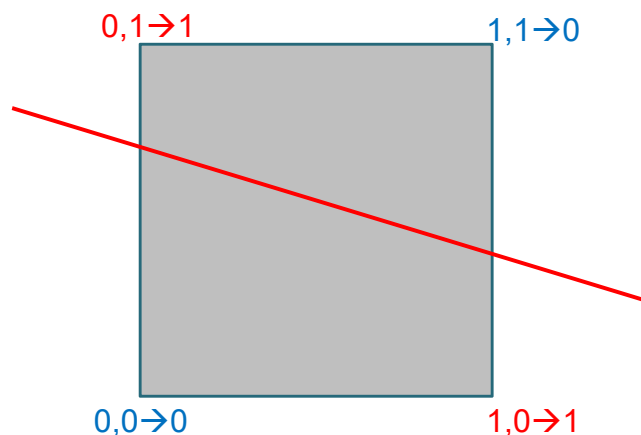
Minsky & Papert

- Proved that the translations with wrap-around could not work with perceptrons
 - “Group Invariance Theorem”
- Showed that perceptrons could not solve the XOR case
- Conclusion is that you have to hand-code feature detectors rather than have them magically learn

© 2018 Peter V. Henstock

Geometric Case: XOR

- Not linearly separable → perceptron fails



© 2018 Peter V. Henstock

AI Winter

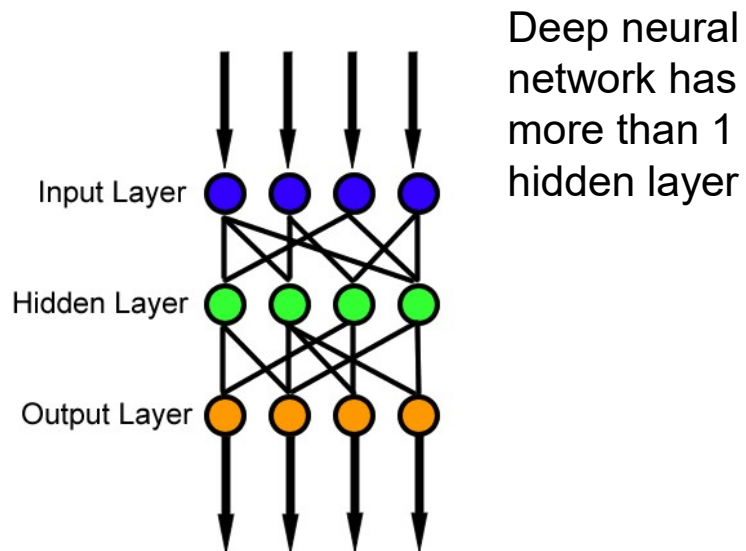
- Limited research
- No funding for neural networks
- Solutions had actually been published
- 1980s
 - Greater computational power
 - Overcame limitations of perceptron

© 2018 Peter V. Henstock

Alternative to Perceptron

© 2018 Peter V. Henstock

Feedforward neural network



© 2018 Peter V. Henstock

Steps to more advanced NN

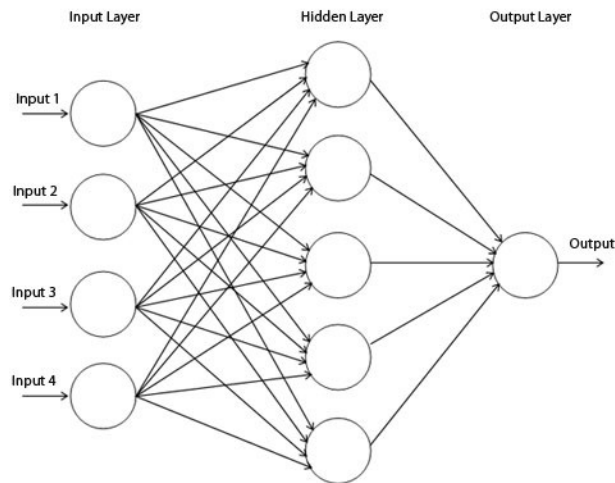
- Need to have a nonlinear step function
- Need to have an extra layer(s) as single layer cannot recognize the variety of patterns

7 7 7 7 7 7 7 7

- Need adaptive learning of weights at each level
 - Output weights are easy
 - Hidden layers are unguided but are essentially doing feature selection

© 2018 Peter V. Henstock

Multilayer feed-forward (2-layer)



- Which layer doesn't count?
- Is it fully connected?

© 2018 Peter V. Henstock

Learning Linear Neuron

- Perceptron learning does not work for multiple layers
- Perceptrons have a property they learn and can be optimized in a convex space
 - Average of any two good sets of weights is also a good weight
 - Property fails for multiple layers
- Linear Neuron optimizes a cost function
 - Output values converge to the targets
 - Example is least-squares error

© 2018 Peter V. Henstock

Linear Neuron

- $Y = \sum_i w_i x_i = w^T x$
- Goal is to generalize this to the multi-layer neuron
- Could just solve since this is essentially a linear regression where optimize w in a least-squares case to approximate Y
- Iterative approach will generalize by shifting weights gradually

© 2018 Peter V. Henstock

Learn by Modifying Weights

- Goal is to automate feature selection
- One approach: randomly tweak weights
 - Not efficient
 - Really inefficient near optimum
- Perturb all weights but hard to deconvolve the results
- Randomly modify activities of hidden units

© 2018 Peter V. Henstock

Gradient Descent

- $w_i \leftarrow w_i + \Delta w_i$
- $\Delta w = -\alpha \nabla E(w)$ for vector w
- $\nabla E(w) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \frac{\partial E}{\partial w_3}, \dots, \frac{\partial E}{\partial w_n} \right]$

© 2018 Peter V. Henstock

Delta Rule or Least-Squares

- Error = $\frac{1}{2} \sum_{\text{training } k} (t^k - y^k)^2$
 - y^k is output; t^k is target
- How can we optimize the error?
- $\frac{\partial \text{Error}}{\partial w_j} = \frac{1}{2} \sum_i \frac{\partial y^i}{\partial w_j} \frac{\partial \text{Error}^i}{\partial y^i} = -\sum_k x_j^k (t^k - y^k)$
- $\Delta w_j = -\alpha \frac{\partial \text{Error}}{\partial w_j} = \alpha \sum_k x_j^k (t^k - y^k)$
- How did we do that?

© 2018 Peter V. Henstock

Compute Gradient Descent

- Optimize: $\frac{1}{2} \sum_{k=inputs} (t_k - y_k)^2$
- $\frac{\partial E}{\partial w_i} \Rightarrow \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{k=inputs} (t_k - y_k)^2$
- $= \frac{1}{2} \sum_{k=inputs} \frac{\partial}{\partial w_i} (t_k - y_k)^2$
- $= \frac{1}{2} \sum_{k=inputs} 2(t_k - y_k) \frac{\partial}{\partial w_i} (t_k - y_k)$
- But $y_k = weights * x_k$
- $= \sum_{k=inputs} (t_k - y_k) \frac{\partial}{\partial w_i} (t_k - \sum_{j=1}^P w_j x_{kj})$
- $= \sum_{k=inputs} (t_k - y_k) (-x_{ki})$

© 2018 Peter V. Henstock

Gradient Descent Algorithm

- Initialize all the weights w
- Loop
 - Present each training instance to NN
 - Input x vector
 - Compute the output y
 - Compare y to the t from the training data
 - Compute all the Δw
 - Update the new w weights by moving a distance $\alpha \Delta w$ downhill

© 2018 Peter V. Henstock

α and the 3 bears



<http://jessicabarrah.blogspot.com/2012/02/fairy-tales.html>

- What happens if α is too small?
- What happens if α is too large?

© 2018 Peter V. Henstock

Perceptron vs. Gradient Descent

- Perceptron will work if:
 - Training examples are linearly separable
 - Learning rate is sufficiently small
 - $w_i \leftarrow w_i + \alpha(t - y)x_i$
- Gradient Descent converges minimum squared error if
 - Learning rate is sufficiently small
 - Works even if training data not separable
 - $w_i \leftarrow w_i + \alpha(t - y)x_i$
- So where is the difference?

© 2018 Peter V. Henstock

What's Missing?

- Perceptron
 - $w_i \leftarrow w_i + \alpha(t - y)x_i$
- Gradient Descent
 - $w_i \leftarrow w_i + \alpha(t - y)x_i$

© 2018 Peter V. Henstock

What's Missing?

- Nonlinearity
- Why?
- The step function is not differentiable
- Therefore you cannot perform a gradient search

© 2018 Peter V. Henstock

What's Missing?

- Nonlinearity
- Why?
- The step function is not differentiable
- Therefore you cannot perform a gradient search
- So which nonlinearity?

© 2018 Peter V. Henstock

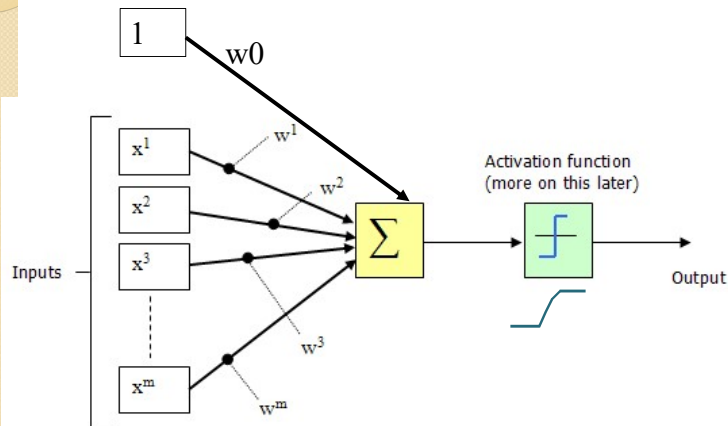
So which nonlinearity?

- Typically use a sigmoid
- $\sigma(x) = \frac{1}{1 + e^{-x}}$
- $\sigma'(x) = \sigma(x)(1 - \sigma(x))$
 - Let you prove it if you are interested in the math
- Sigmoid is standard biological dose-response or binding relationship
 - Believed neurons follow these

© 2018 Peter V. Henstock

Sigmoid and bias

- Bias term: w_0 changed threshold
- In sigmoid what does w_0 bias do?

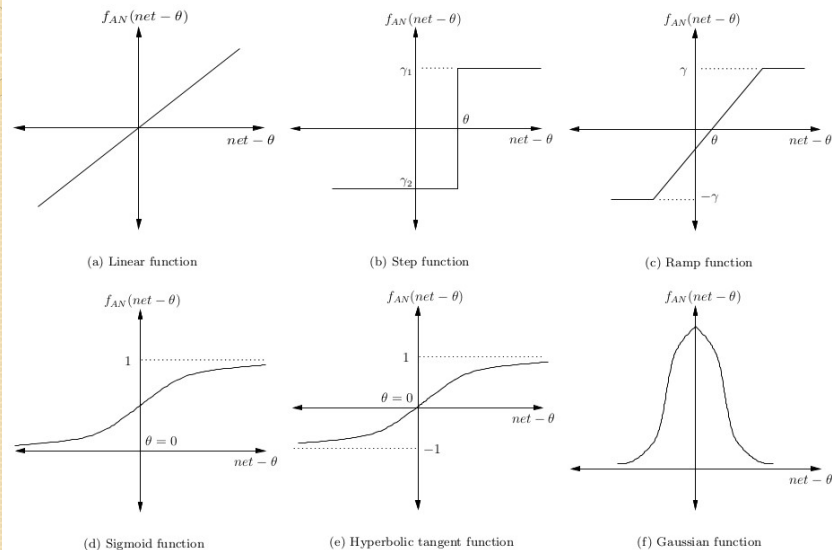


<http://www.codeproject.com/Articles/16419/AI-Neural-Network-for-beginners-Part-of>

© 2018 Peter V. Henstock

Activation Functions

- <http://www.turinfofinance.com/misconceptions-about-neural-networks/>



© 2018 Peter V. Henstock

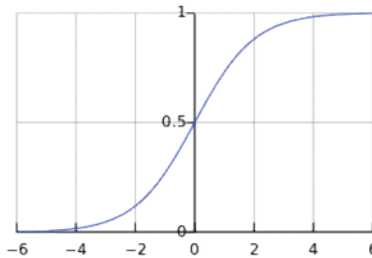
Activation Function

- Sum of inputs would normally only be capable of producing linear regression
- Activation function provides nonlinear regression since output = $f(\sum \text{inputs})$
- Multilayer feed-forward network can approximate pretty much any function given enough hidden units and corresponding training

© 2018 Peter V. Henstock

Sigmoidal Neuron

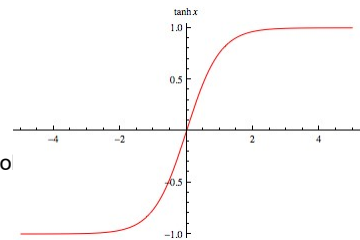
- $Z = b + \sum x_i w_i$
- $Y = 1 / [1 + \exp(-z)]$
- Smooth derivatives
- Also called “squashing function” since it maps $[-\infty, \infty] \rightarrow [0, 1]$
- Wikipedia



© 2018 Peter V. Henstock

Sigmoidal Neuron using tanh

- $Z = b + \sum x_i w_i$
- $Y = \tanh(Z)$
- Smooth derivatives
- Negative values



- <http://mathworld.wolfram.com/HyperbolicTangentFunction.html>

© 2018 Peter V. Henstock

Logistic Neuron

- $y = 1/[1 + \exp(-z)] \quad z = b + \sum x_i w_i$
- $\frac{dy}{dz} = \frac{-(-e^{-z})}{(1+e^{-z})^2} = \frac{1}{(1+e^{-z})} \frac{e^{-z}}{(1+e^{-z})} = y(1-y)$

- From before:

$$\frac{\partial \text{Error}}{\partial w_j} = \frac{1}{2} \sum_i \frac{\partial y^i}{\partial w_j} \frac{\partial \text{Error}^i}{\partial y^i} = -\sum_i x_j^i (t^i - y^i)$$

- Parts with ∂y^i change

$$\frac{\partial y}{\partial w_j} = \frac{\partial z}{\partial w_j} \frac{dy}{dz} = x_j y(1-y)$$

$$\frac{\partial \text{Error}}{\partial w_j} = \sum_i \frac{\partial y^i}{\partial w_j} \frac{\partial \text{Error}^i}{\partial y^i} = -\sum_i x_j y^i (1 - y^i) (t^i - y^i)$$

- Now have dError/dw without hidden layer

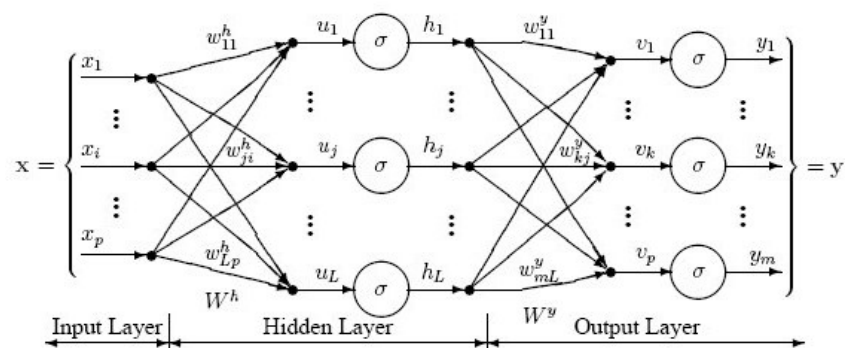
© 2018 Peter V. Henstock

Backpropagation

- Minsky & Papert stalled the field
- Knew that if you put perceptrons or neural networks together, then you could overcome the limitations
- Couldn't figure out how to train the merged networks
- ...until backpropagation emerged
 - Rumelhart, Hinton & Williams 1986 Nature 323, 533-536

© 2018 Peter V. Henstock

Multiple layer Perceptron



© 2018 Peter V. Henstock

Feedforward with Backpropagation

- Estimated 90% of non-academic models are based on this approach
- Multi-layer networks required in many applications

© 2018 Peter V. Henstock

Derivative Chain rule

- Reminder

- $$\frac{dh}{dx} = \frac{dh}{dg} \frac{dg}{dx}$$

© 2018 Peter V. Henstock

3 Versions of Backprop

- All versions get us to the same place
- It's a somewhat tricky algorithm and I found that different versions were helpful
- My favorite is the single slide version
- **Version 1**
- Version 2 from Deep Learning
- Version 3 from Hinton's slide

© 2018 Peter V. Henstock

Backpropagation

- “Backward propagation of errors”
- Optimize: $\frac{1}{2} \sum_{k=inputs} (t_k - y_k)^2$
- $\frac{\partial E}{\partial w_i} \Rightarrow \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{k=inputs} (t_k - y_k)^2$
- $= \frac{1}{2} \sum_{k=inputs} \frac{\partial}{\partial w_i} (t_k - y_k)^2$
- $= \frac{1}{2} \sum_{k=inputs} 2(t_k - y_k) \frac{\partial}{\partial w_i} (t_k - y_k)$
- $= \sum_{k=inputs} (t_k - y_k) (-1) \frac{\partial y_k}{\partial w_i}$
- $= \sum_{k=inputs} (t_k - y_k) (-1) \frac{\partial y_k}{\partial sum_k} \frac{\partial sum_k}{\partial w_i}$

© 2018 Peter V. Henstock

Backpropagation

- $\sum_{k=inputs} (t_k - y_k)(-1) \frac{\partial y_k}{\partial sum_k} \frac{\partial sum_k}{\partial w_i}$
- But $\frac{\partial y_k}{\partial sum_k} = \frac{\partial f(sum_k)}{\partial sum_k} = y_k(1-y_k)$
- And $\frac{\partial sum_k}{\partial w_i} = \frac{\partial \sum_j w_j x_{k,j}}{\partial w_i} = x_{k,i}$
- $-\sum_{k=inputs} (t_k - y_k) y_k(1-y_k) x_{k,i}$
- This allows us to compute outputs from previous layer (i.e. top hidden layer)

© 2018 Peter V. Henstock

Computing input → hidden weights

- $\frac{\partial E}{\partial sum_j} \Rightarrow \sum_{k=out(j)} \frac{\partial E_k}{\partial sum_k} \frac{\partial sum_k}{\partial sum_j}$
- Let $dk = \frac{-\partial E_k}{\partial sum_k}$
- $\frac{\partial E}{\partial sum_j} \Rightarrow \sum_{k=out(j)} -dk \frac{\partial sum_k}{\partial sum_j}$
- $= \sum_{k=out(j)} -dk \frac{\partial sum_k}{\partial y_j} \frac{\partial y_j}{\partial sum_j}$
- $= \sum_{k=out(j)} -dk w_{kj} \frac{\partial y_j}{\partial sum_j}$
- w_{kj} is weight from j to k
- $= \sum_{k=out(j)} -dk w_{kj} y_j(1 - y_j)$

© 2018 Peter V. Henstock

- $\frac{\partial E}{\partial \text{sum}_j} = \sum_{k=\text{out}(j)} -dk \, w_{kj} y_j (1 - y_j)$
- $dj = -\frac{\partial E}{\partial \text{sum}_j} = y_j (1 - y_j) \sum_{k=\text{out}(j)} dk \, w_{kj}$

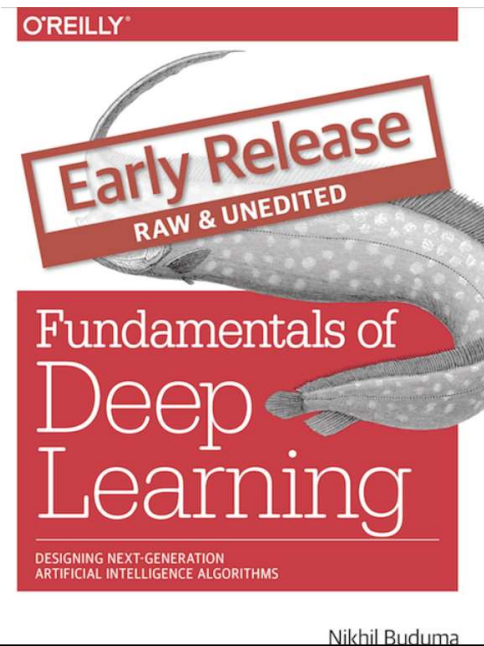
© 2018 Peter V. Henstock

Backpropagation Algorithm

- We know what the inputs are doing
- We know what the outputs are doing
- We can do a gradient descent between the input and output easily for 1 layer
- We don't know what the hidden layers are doing
- Hidden layers will affect the errors
- If solve for one hidden layer, we can repeat for the other layers

© 2018 Peter V. Henstock

Backpropagation

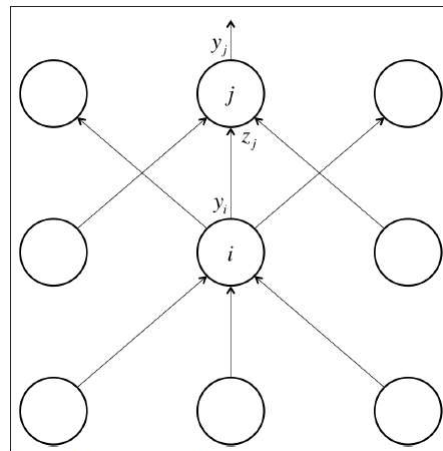


- \$25 preorder
- Jan 2016 release
- Covers theano

Nikhil Buduma

© 2018 Peter V. Henstock

Notation Change



- Here $z_j = \sum_i w_{ij} y_i$
- Weights $w_{2,3}^{(1)}$ goes node 2 to node 3
 - (1) indicates it's a weight going to 1st hidden

© 2018 Peter V. Henstock

Backpropagation Version 2

- Minimize Error $E = \frac{1}{2} \sum_{j=\text{inputs}} (t_j - y_j)^2$
- $\frac{dE}{dy_j} \rightarrow -(t_j - y_j)$ at output layer
- Going one step deeper
- Want to calculate error at layer below output at node i
- Need to relate outputs of layer i to sigmoids of all neurons in layer j

© 2018 Peter V. Henstock

Next layer

- $\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial z_j} \frac{dz_j}{dy_i} = \sum_j w_{ij} \frac{\partial E}{\partial z_j}$
- $\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j} \frac{dy_j}{dz_i} = y_j(1 - y_j) \frac{\partial E}{\partial y_j}$
 - $\frac{dy}{dz} = \frac{-(-e^{-z})}{(1+e^{-z})^2} = \frac{1}{(1+e^{-z})} \frac{e^{-z}}{(1+e^{-z})} = y(1 - y)$
 - Property of derivative of sigmoid (logit)
- Combine 1st and 2nd equations
- $\frac{\partial E}{\partial y_i} = \sum_j w_{ij} y_j(1 - y_j) \frac{\partial E}{\partial y_j}$
- Now expressed $\frac{\partial E}{\partial y_i}$ in terms of y_j

© 2018 Peter V. Henstock

How do we learn the weights?

- $\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial z_j} \frac{\partial z_j}{\partial w_{ij}} = [y_j(1 - y_j) \frac{\partial E}{\partial y_j}] y_i$
 - Red is from previous slide
 - Black is just partial-derivative of the connection $y_i * w_{ij} \rightarrow z_j$
- Need to update all weights in data set
- $\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}}$
- $\Delta w_{ij} =$
- $= -\alpha \sum_{k=\text{inputs}} y_i^{(k)} y_j^{(k)} (1 - y_j^{(k)}) \frac{\partial E^{(k)}}{\partial y_j^{(k)}}$

© 2018 Peter V. Henstock

Summarize Backpropagation

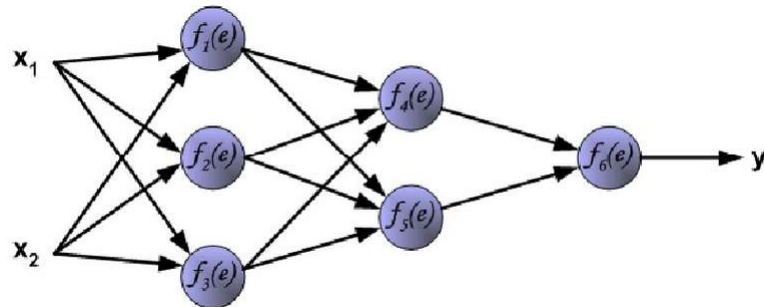
- Optimizing LS Error of target & y's
- Forward step
 - Input training with fixed weights
 - Propagate through sigmoids of each layer to the target and compare to output y
- Backward step
 - Compute $\frac{\partial E}{\partial w_{ij}}$ using backward propagation of the errors in terms of $\frac{\partial E}{\partial y_i}$ as $f(y_j)$
 - Enables computation of new weights

© 2018 Peter V. Henstock

Example:

- <http://www.slideshare.net/aorriols/lecture11-neural-networks>

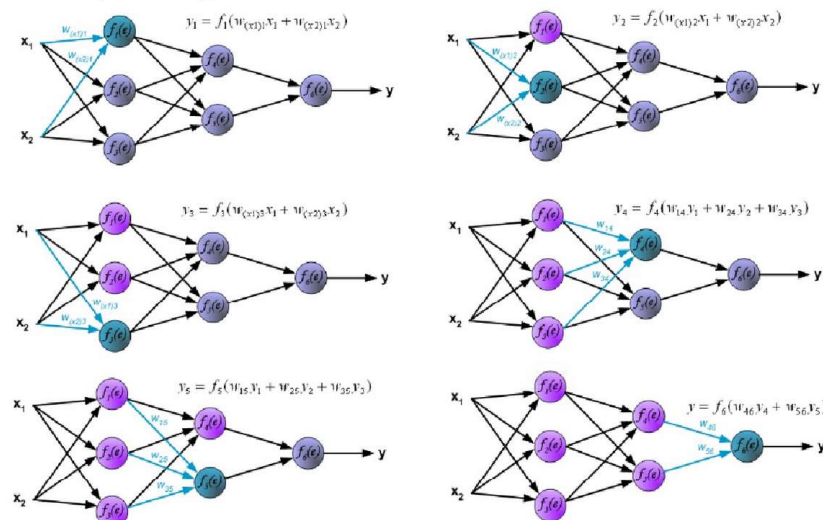
Example borrowed from: http://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html



© 2018 Peter V. Henstock

1) Forward propagation

1. Sweep the weights forward

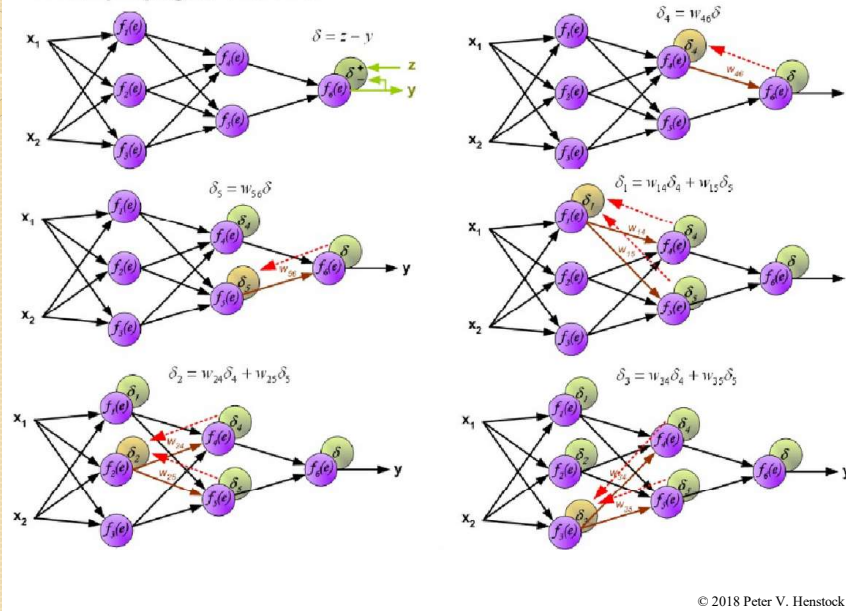


- Blue = not yet; Teal=Current; Purple = done

© 2018 Peter V. Henstock

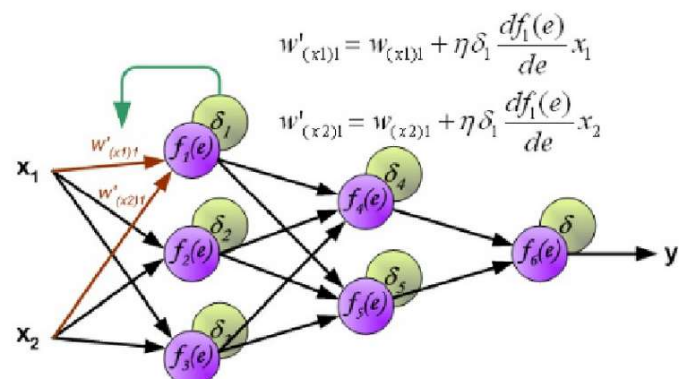
2) Backpropagate

2. Backpropagate the error



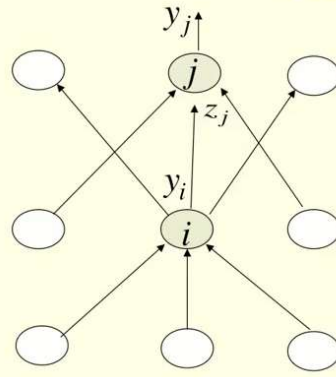
3) Update weights

3. Modify the weights of each neuron



Backpropagation on one slide

Backpropagating dE/dy



$$\frac{\partial E}{\partial z_j} = \frac{dy_j}{dz_j} \frac{\partial E}{\partial y_j} = y_j (1 - y_j) \frac{\partial E}{\partial y_j}$$

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{dz_j}{dy_i} \frac{\partial E}{\partial z_j} = \sum_j w_{ij} \frac{\partial E}{\partial z_j}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial z_j}{\partial w_{ij}} \frac{\partial E}{\partial z_j} = y_i \frac{\partial E}{\partial z_j}$$

- Just the chain rule
- Hinton, Srivastava and Swersky slides

© 2018 Peter V. Hestock

Practical Advice for Classification

© 2018 Peter V. Hestock

No Free Lunch Theorem

- Wolpert 1996
- No learning algorithm can be guaranteed to succeed on all tasks
- Main goal is to predict unseen examples correctly
- NFLT states that for all machine learning algorithms, average of all predictions on unseen 2-class examples = $\frac{1}{2}$
- How can this be?

© 2018 Peter V. Henstock

No Free Lunch Theorem

- Each time you train your data on a particular space, it is less trained on other feature spaces

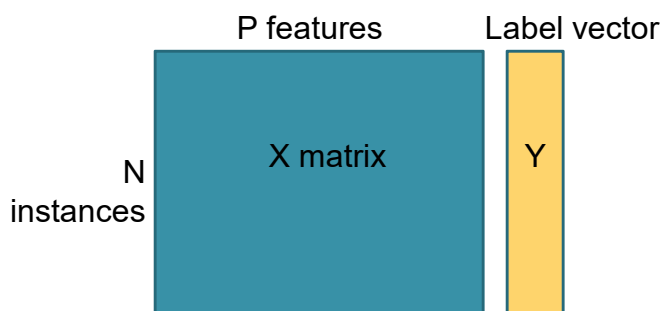
© 2018 Peter V. Henstock

No Free Lunch Theorem

- Consequence
- Can't run your favorite algorithm and expect it to work on all data sets
- Need to run multiple algorithms and compare using cross-validation

© 2018 Peter V. Henstock

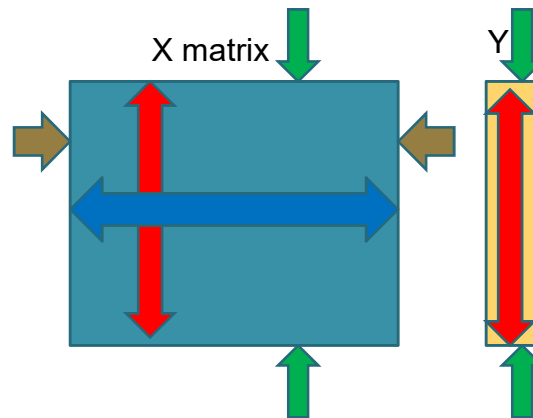
Are bias & variance useful?



- We can send this through any or all of our favorite algorithms
- What else can we do with this?

© 2018 Peter V. Henstock

Are bias & variance useful?

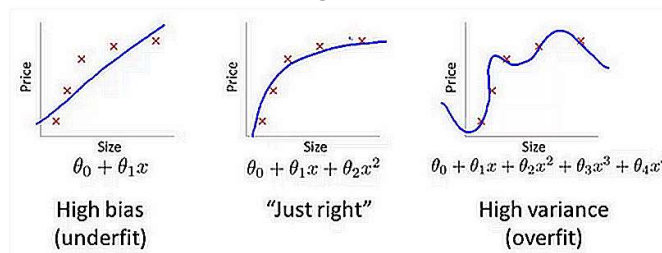


- Increase our training set size
- Reduce our training set size
- Increase #features
- Reduce #features

© 2018 Peter V. Henstock

Over or Underfitting?

- What should we look at?
- Plot curves for regression



- What about high dimensional spaces for classification or clustering?
- How can we tell if we're over/underfitting?

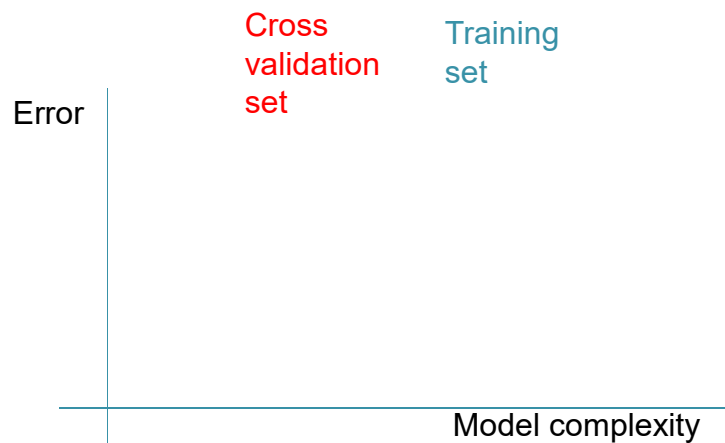
© 2018 Peter V. Henstock

Cross-validation

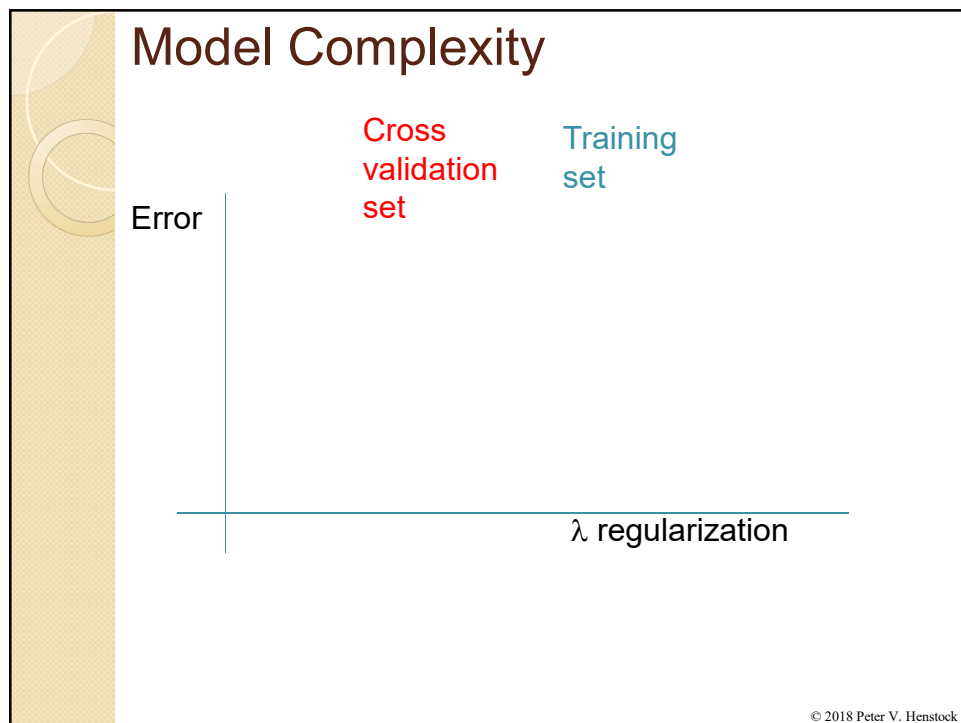
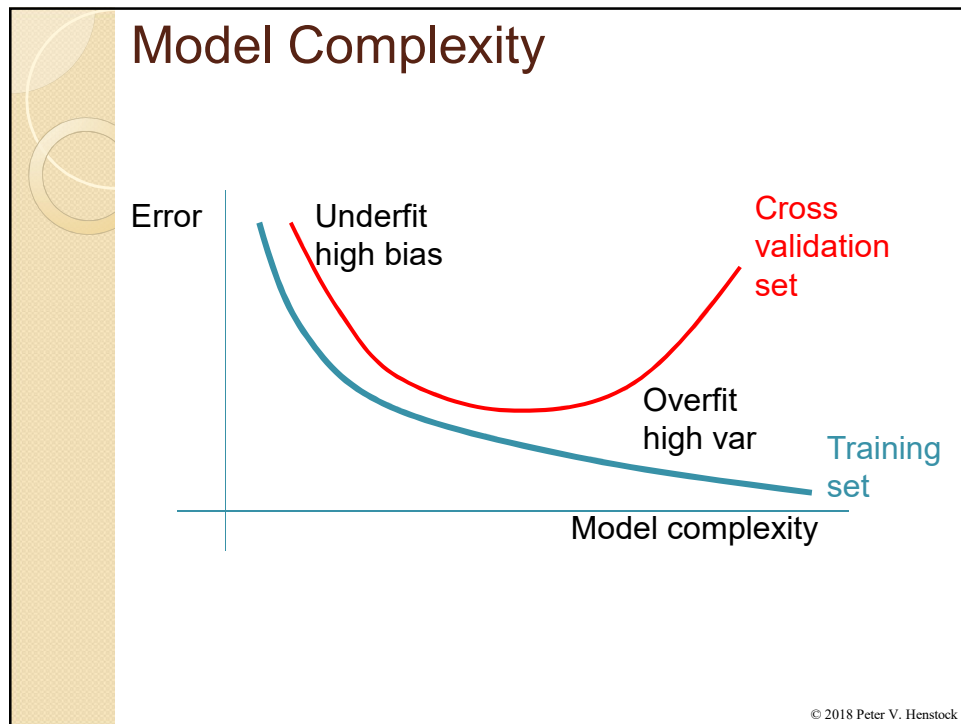
- Typically have 3 data sets
 - Training set 60%
 - Cross-validation/test set 20%
 - Blind set 20%
- Work your machine learning magic and optimize on your training set
- Evaluate your decisions on your cross-validation/test set
- Final performance evaluation: blind set

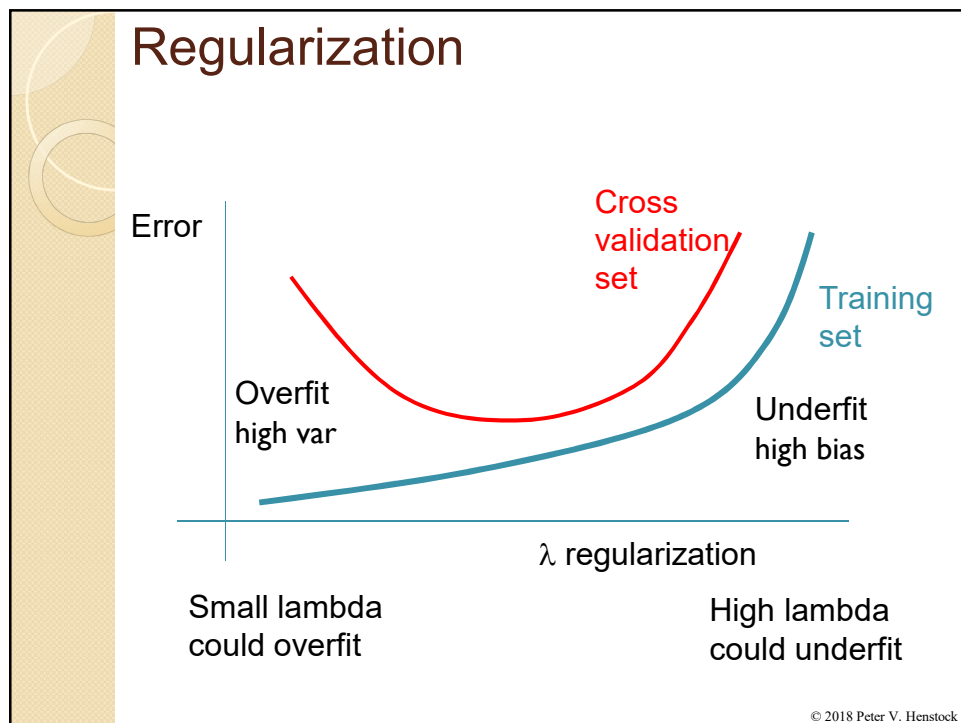
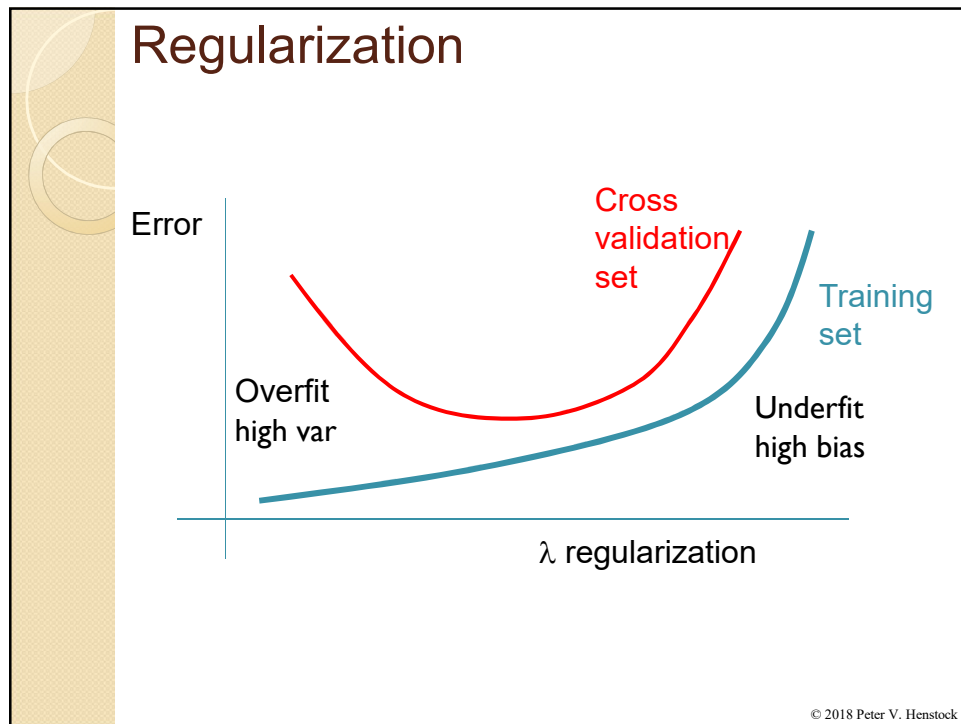
© 2018 Peter V. Henstock

Model Complexity

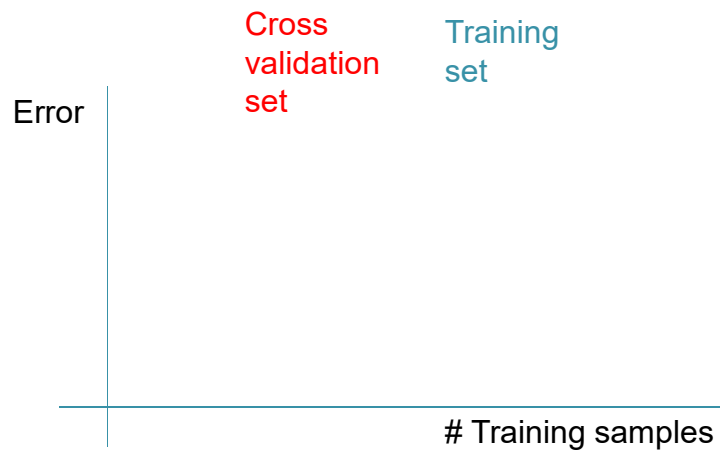


© 2018 Peter V. Henstock



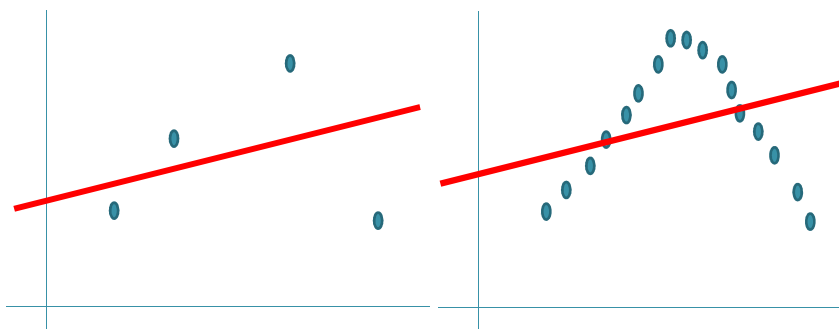


Learning Curve/Sample subsets



© 2018 Peter V. Henstock

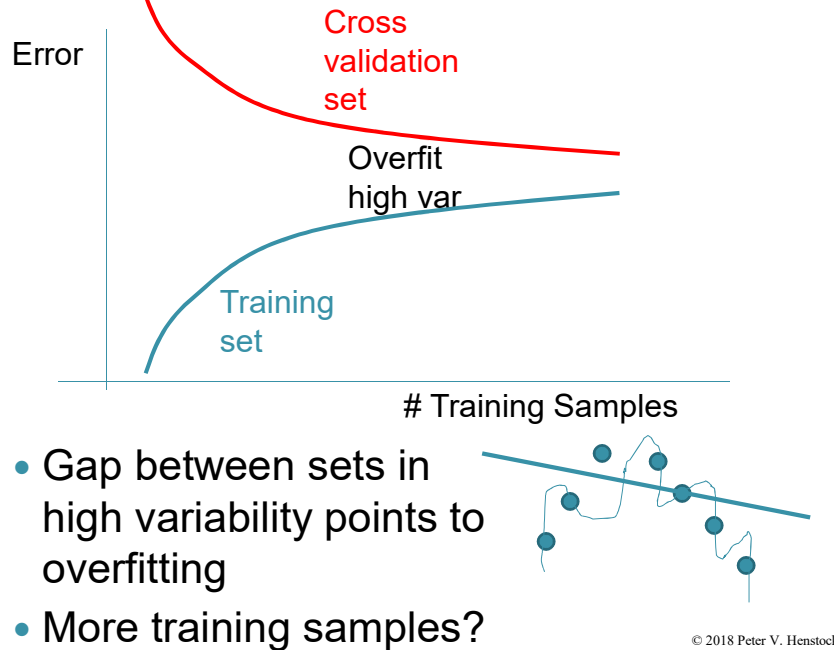
Training samples



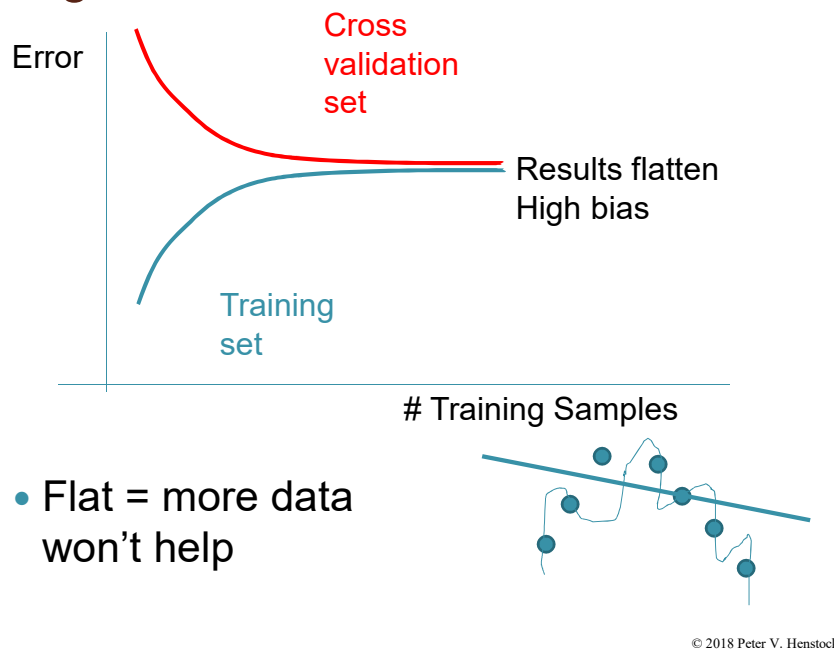
- More points for a high bias

© 2018 Peter V. Henstock

High Variance Case



High Bias Case



What choices to make

- High bias
 - Add complexity to model
 - Procure more features
 - Add polynomial terms or other variations
 - Decrease the λ regularization
 - Add hidden layers
- High variability
 - Add more data instances
 - Reduce feature space
 - Increase the λ regularization
 - Reduce hidden layers

© 2018 Peter V. Henstock

Summary of Strategies

- Obtain more training examples
 - Helps for high variance
- Generate more features
 - Helps for high bias
- Select feature sets (reduce)
 - Helps for high variance
- Modify parameters (regularization)
 - Higher lambda → helps high variance
 - Lower lambda → helps high bias
- Use cross-validation and test-on training to figure out where you are in bias/variance

© 2018 Peter V. Henstock