PCA

The central idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set. This is achieved by transforming to a new set of variables, the principal components (PCs), which are uncorrelated, and which are ordered so that the first few *retain most of the variation present in* all *of the original variables.*
[Jolliffe, Pricipal Component Analysis, 2^{nd} edition]

http://www.stat.columbia.edu/~fwood/Teaching/w4315/Fall2009/pca.pdf

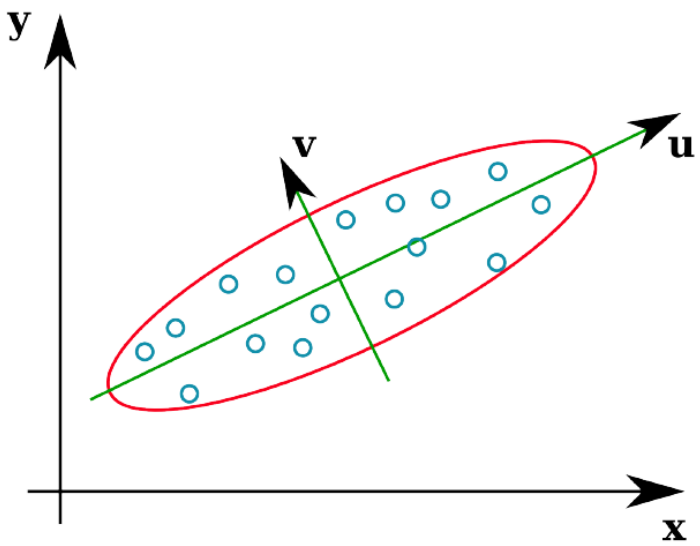


Figure 1: PCA for Data Representation

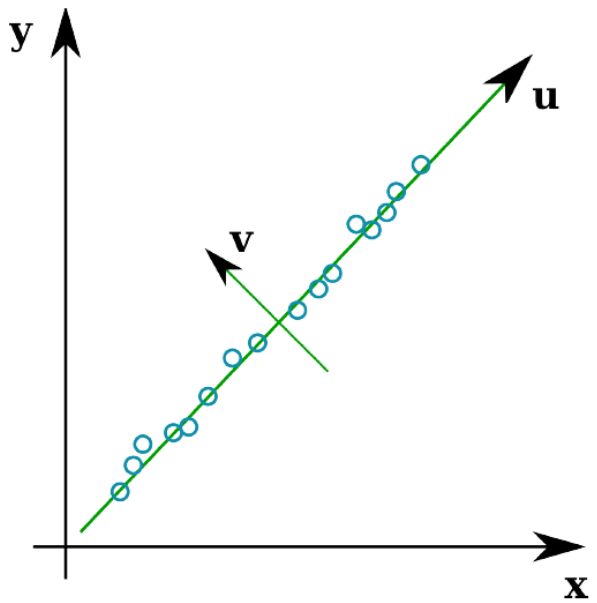
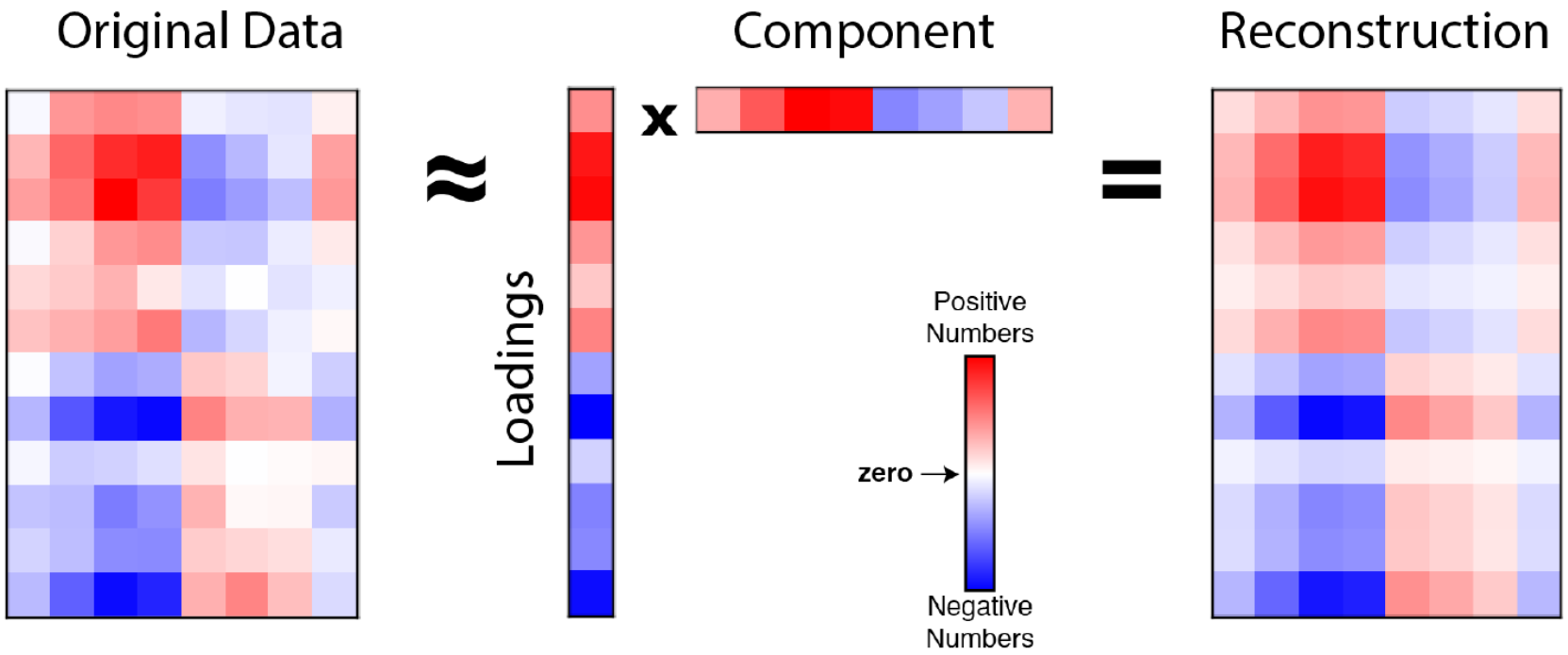


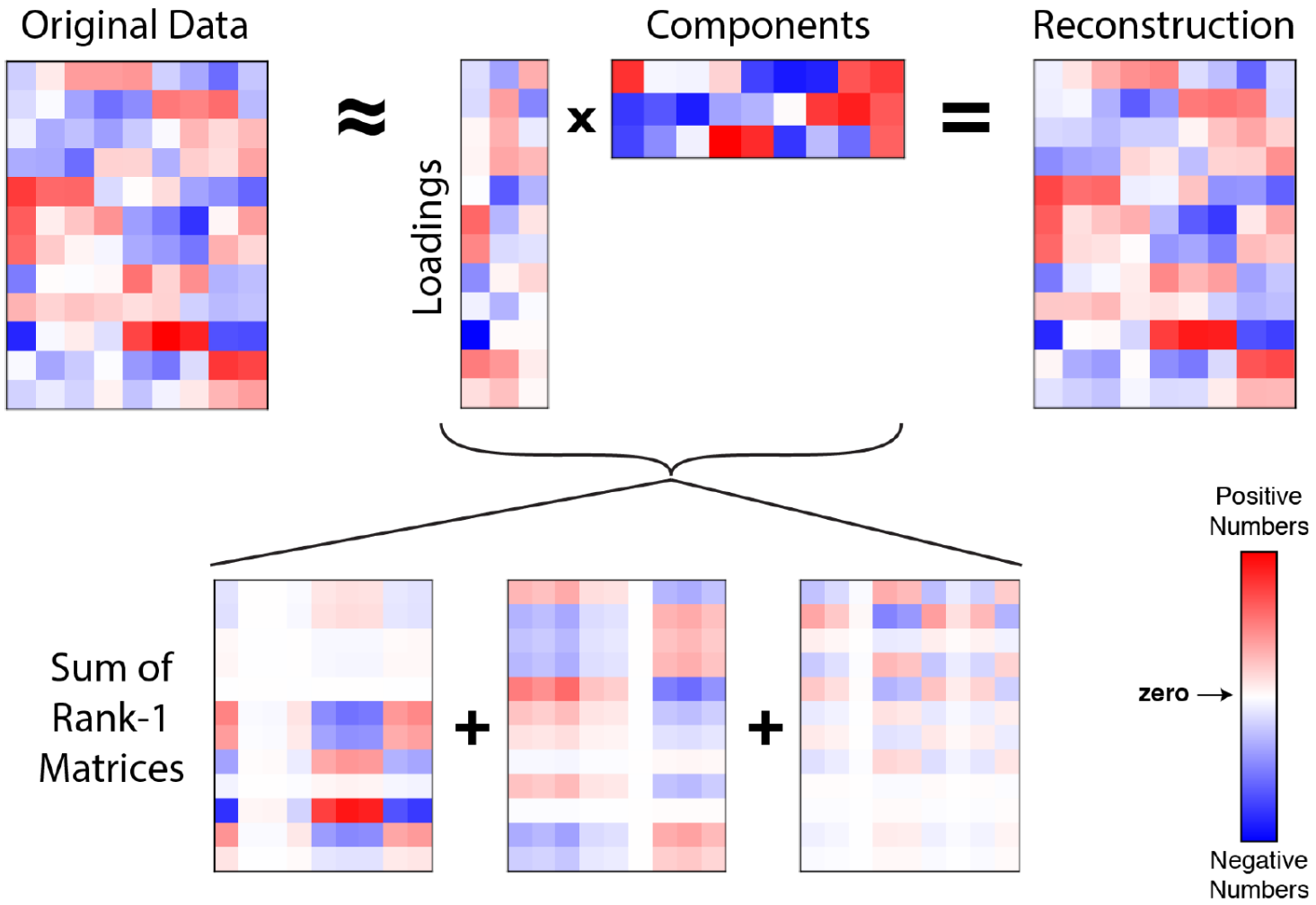
Figure 2: PCA for Dimension Reduction

http://www.stat.columbia.edu/~fwood/Teaching/w4315/Fall2009/pca.pdf

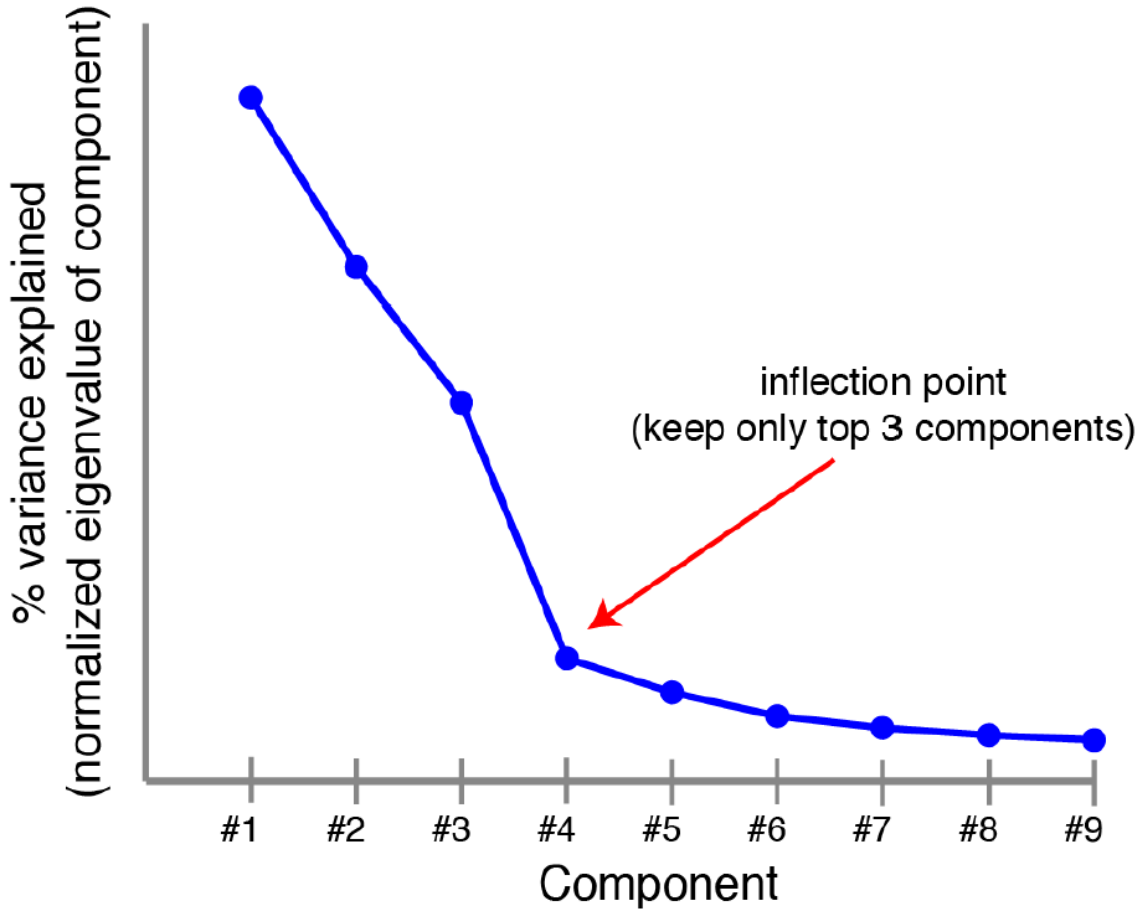


Example reconstruction of data with 1 principal component. An example data matrix (*left*) with $n = 12$ observations and $p = 8$ features is approximated by the outer product $\mathbf{wc}^T$ (*middle*) which produces a rank-one matrix (*right*). Note $\mathbf{w}$ is labeled as *loadings* and $\mathbf{c}^T$ is labeled as *component*

http://alexhwilliams.info/itsneuronalblog/2016/03/27/pca/#an-alternative-optimization-problem



http://alexhwilliams.info/itsneuronalblog/2016/03/27/pca/#an-alternative-optimization-problem



Scree plot. Principal components are ranked by the amount of variance

http://alexhwilliams.info/itsneuronalblog/2016/03/27/pca/#an-alternative-optimization-problem

The Optimal Hard Threshold for Singular Values is $4/\sqrt{3}$

Matan Gavish, *Student Member, IEEE*, and David L. Donoho, *Member, IEEE*

$$\lambda = \frac{4\sigma\sqrt{n}}{\sqrt{3}}$$

σ is the noise in the dataset, keep components which are above this threshold

Limitations of PCA

- Assumes Normality of the data
- Cannot Model non-Linear relationship
- Becomes inconsistent when p>n

Extensions of PCA

- **Sparse PCA** – replace L2 norm with L1 norm (like ridge regression vs LASSO), force to 0 some of the variables in the components.
- **Logistic PCA** (for 0/1 variables)
- **Robust PCA.** If you have outliers in your dataset, use the sum of the absolute value of the residuals (L1 loss) or a Huber loss function (Kwak, 2008). There are some alternative formulations of robust PCA, see e.g. Candes et al. (2009) and Netrapalli et al. (2014).
- **Poisson PCA and PCA on ordinal data.** See Rennie & Srebro (2005) for some discussion of appropriate loss functions.
- **Zero-Inflated dimensionality reduction.** Some datasets, such those from single-cell RNAseq, have more zero entries than would be expected under a Poisson noise model. This can arise from technical variability — mRNA is fragile, and lowly expressed genes have less starting material, leading to "dropout" of lowly expressed genes to zero. Pierson & Yau (2015) develop a model to account for this flavor of noise, and their work can be mapped onto the optimization framework described in this post.

T-SNE

Journal of Machine Learning Research 9 (2008) 2579-2605 Submitted 5/08; Revised 9/08; Published 11/08

Visualizing Data using t-SNE

Laurens van der Maaten LVDMAATEN@GMAIL.COM
TiCC
Tilburg University
P.O. Box 90153, 5000 LE Tilburg, The Netherlands

Geoffrey Hinton HINTON@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
6 King's College Road, M5S 3G4 Toronto, ON, Canada

Editor: Yoshua Bengio

Goals:

- Discover Natural Clusters
- Linear Relationship
- Visualize
- Each high-dimensional object is represented as a low-dimensional object
- Preserve the neighborhood
- Distant points correspond to dissimilar objects
- Scalability: should work for large, high-dimensional data sets

Main Idea:

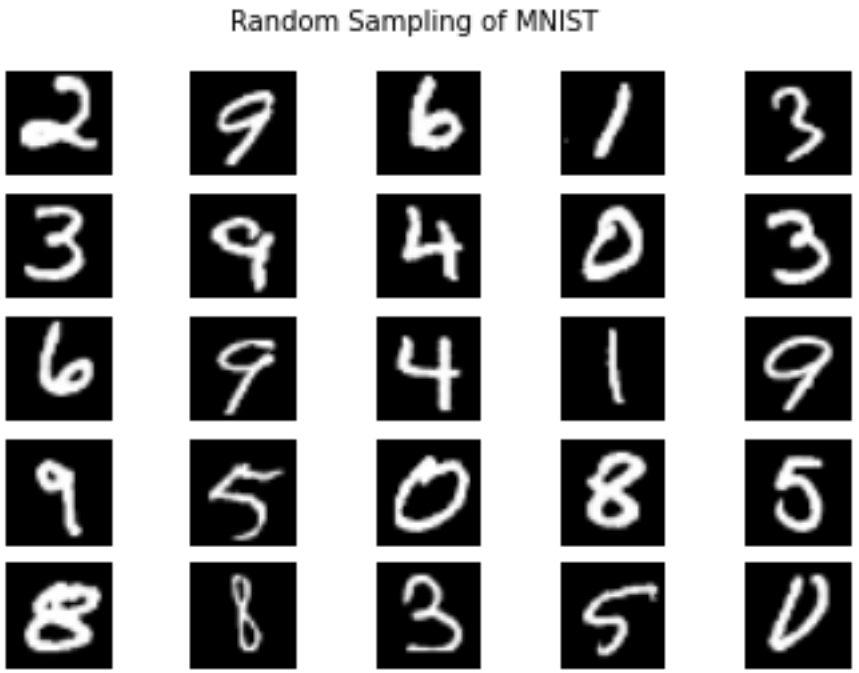
- Compute NxN similarity matrix in the original high-dimensional space
- Compute NxN similarity matrix in the low-dimensional space – a learning objective
- Compare the two similarity matrices – we would like to make them as similar as possible. Use Kullback-Leibler divergence
- Iteratively learn low-dimensional embedding

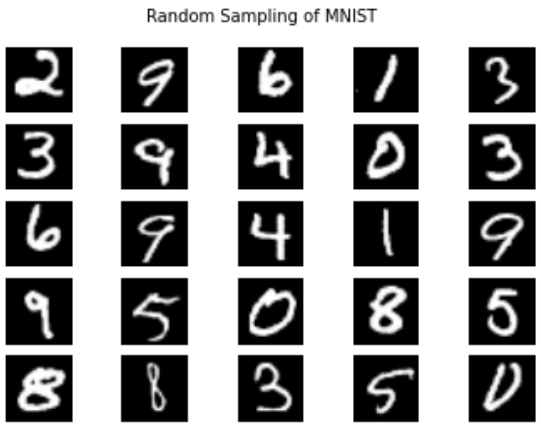
- This presentation is based on the following resources:

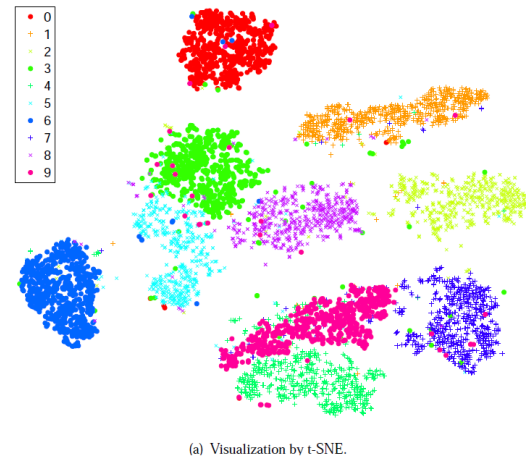
https://docs.google.com/presentation/d/1uOWkXVdEL_P9kO-kCmjkeLdVGR-oYb-3_mLlXp0Ezs8/edit#slide=id.p

http://kawahara.ca/visualizing-data-using-t-sne-slides/

http://yann.lecun.com/exdb/mnist/



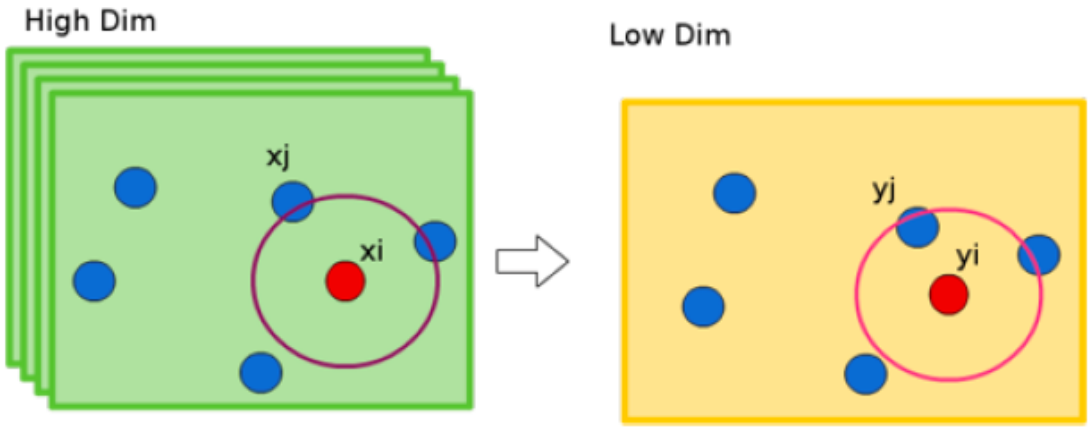




(a) Visualization by t-SNE.

X → Y

Step 1. Conditional similarity between two data points



Similarity of datapoints ($\mathbf{x}_i$) in data space R^D

$$p_{j|i} = \frac{exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2})}{\sum_{k \neq m} exp(-\frac{\|\mathbf{x}_k - \mathbf{x}_m\|^2}{2\sigma_i^2})}$$

$p_{j|i}$ measures how close $\mathbf{x}_j$ is from $\mathbf{x}_i$, considering Gaussian distribution around $\mathbf{x}_i$ with a given variance σ_i^2.

Step 1. Symmetric similarity

Similarity of datapoints ($\mathbf{x}_i$) in data space R^D

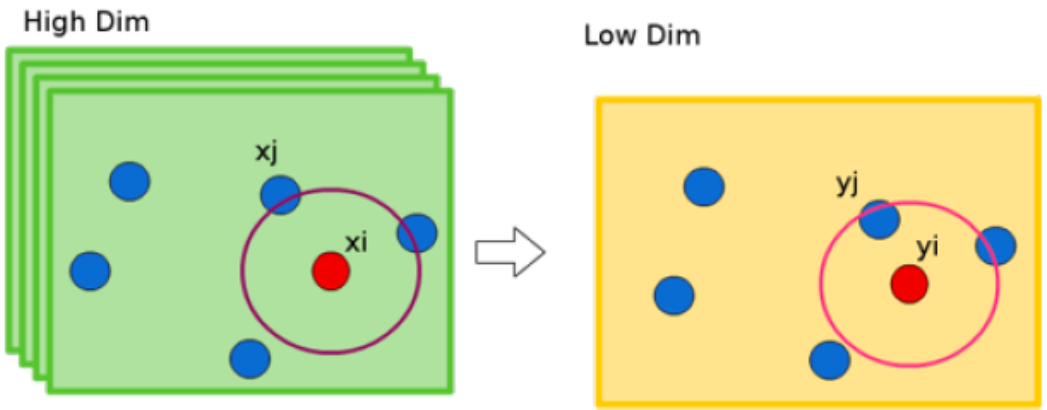
$$p_{j|i} = \frac{exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2})}{\sum_{k \neq m} exp(-\frac{\|\mathbf{x}_k - \mathbf{x}_m\|^2}{2\sigma_i^2})} \tag{1}$$

Make the similarity metric p_{ij} symmetric. The main advantage of symmetry is simplifying the gradient (learning stage):

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N} \tag{2}$$

- we set $p_{ii} = 0$, as we interested in pairwise similarities
- σ_i is chosen such that the data point has a fixed **perplexity** (effective number of neighbors).

Step 2. Similarity of map points in Low Dimension



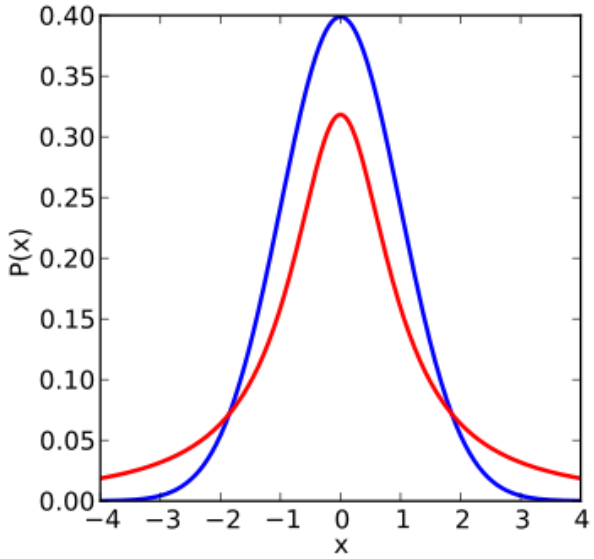
Student t-distribution with 1DoF

$$q_{ij} = \frac{\frac{1}{1+\|\mathbf{y}_i - \mathbf{y}_j\|^2}}{\sum_{k \neq m} \frac{1}{1+\|\mathbf{y}_k - \mathbf{y}_m\|^2}} \tag{3}$$

- we set $q_{ii} = 0$, as we interested in pairwise similarities
- heavy-tail (will be discussed later)
- still closely related to the Gaussian
- computationally convenient (no exponent)

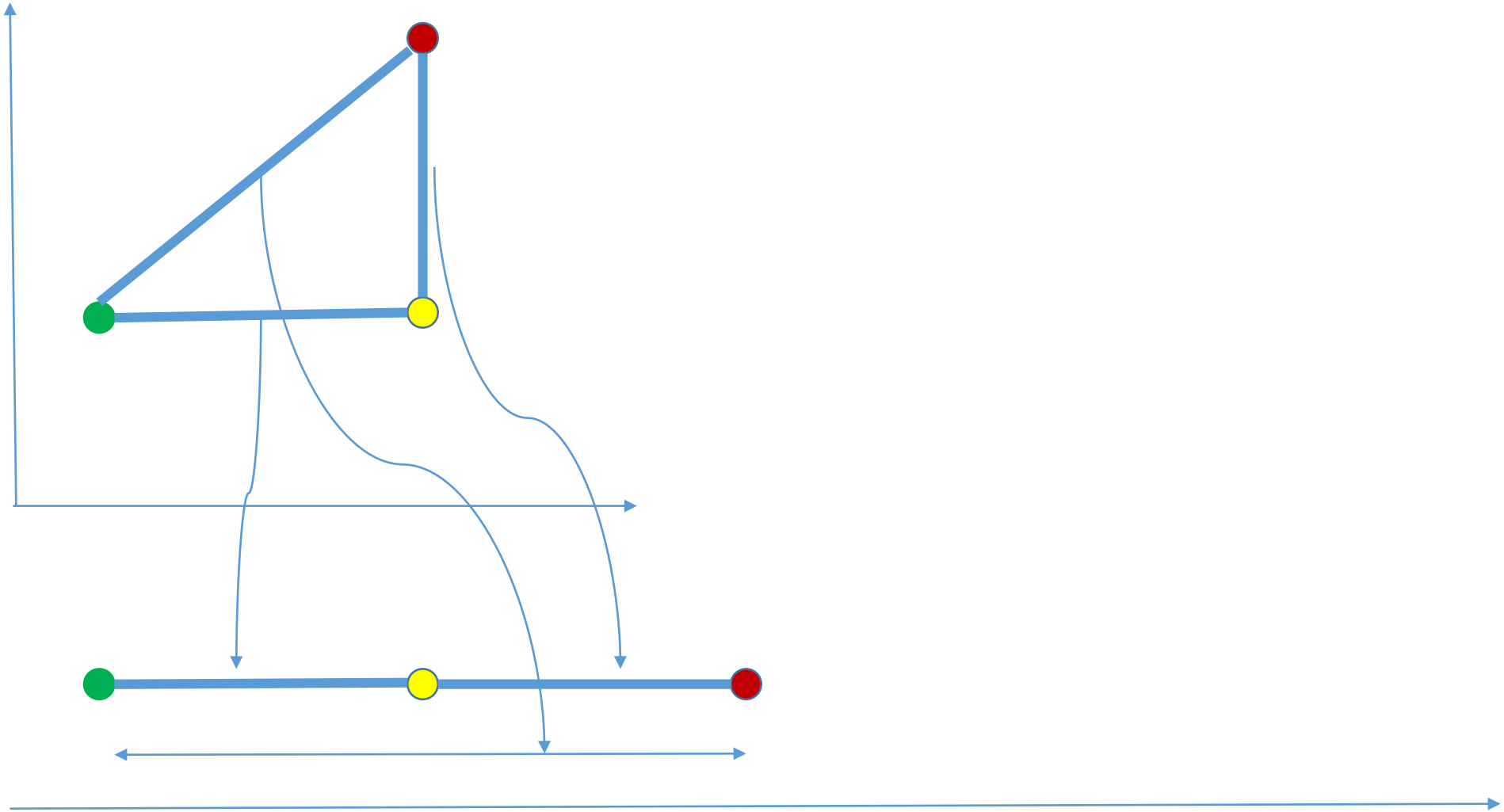
Why switch Distribution?

Blue = Gaussian
Red = Student's T



Student's has longer tails…
gives higher probabilities to
points that are further away

Why switch Distribution?



cost function: Kullback-Leibler divergence

iteration

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$$

momentum parameter

learning rate

$$p_{j|i} = \frac{exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq m} exp\left(-\frac{\|\mathbf{x}_k - \mathbf{x}_m\|^2}{2\sigma_i^2}\right)}$$

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N}$$

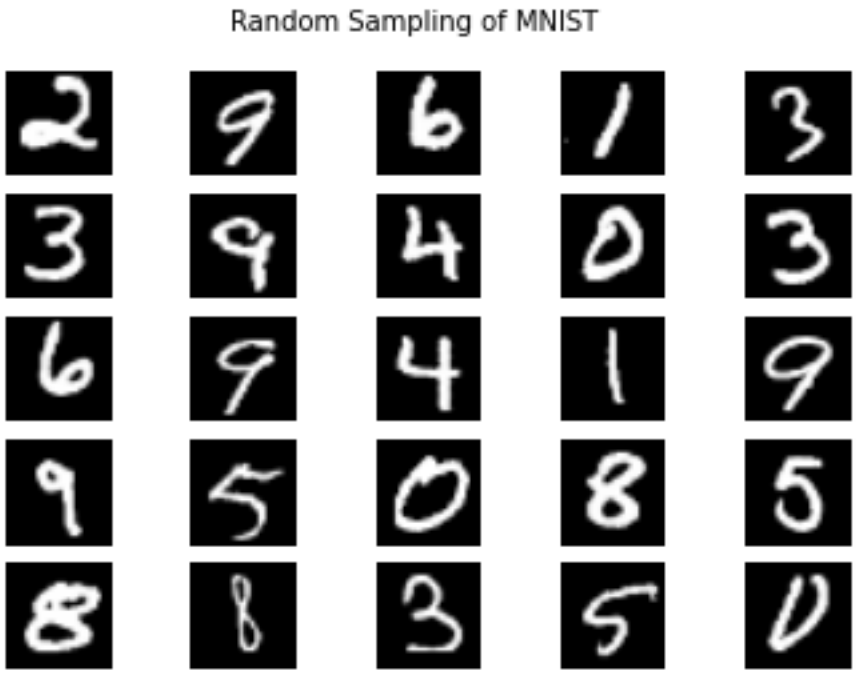
$$q_{ij} = \frac{\frac{1}{1+\|\mathbf{y}_i - \mathbf{y}_j\|^2}}{\sum_{k \neq m} \frac{1}{1+\|\mathbf{y}_k - \mathbf{y}_m\|^2}}$$

Iteratively adjust Y to make $[p_{ij}]$ similar to $[q_{ij}]$
(minimize distance between two matrices)

Kullback-Leibler Divergence

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

http://yann.lecun.com/exdb/mnist/



http://yann.lecun.com/exdb/mnist/

THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

he MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

our files are available on this site:

```
ain-images-idx3-ubyte.gz:  training set images (9912422 bytes)
ain-labels-idx1-ubyte.gz:  training set labels (28881 bytes)
10k-images-idx3-ubyte.gz:  test set images (1648877 bytes)
10k-labels-idx1-ubyte.gz:  test set labels (4542 bytes)
```

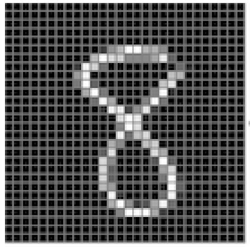
lease note that your browser may uncompress these files without telling you. If the files you downloaded have a larger size than the above, they have been uncompressed by your browser. Simply rename them to remove the .gz extension. Some people have asked me "my ave to write your own (very simple) program to read them. The file format is described at the bottom of this page.

he original black and white (bilevel) images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. the images were nage so as to position this point at the center of the 28x28 field.

/ith some classification methods (particuarly template-based methods, such as SVM and K-nearest neighbors), the error rate improves when the digits are centered by bounding box rather than center of mass. If you do this kind of pre-processing, you should report it in your p

he MNIST database was constructed from NIST's Special Database 3 and Special Database 1 which contain binary images of handwritten digits. NIST originally designated SD-3 as their training set and SD-1 as their test set. However, SD-3 is much cleaner and easier to rec ensus Bureau employees, while SD-1 was collected among high-school students. Drawing sensible conclusions from learning experiments requires that the result be independent of the choice of training set and test among the complete set of samples. Therefore it was necess

Each Image is Digitized



28 x 28
784 pixels

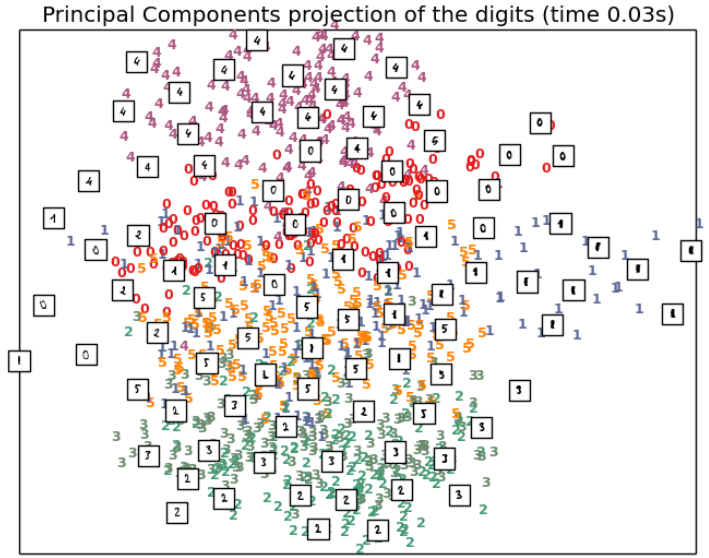
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

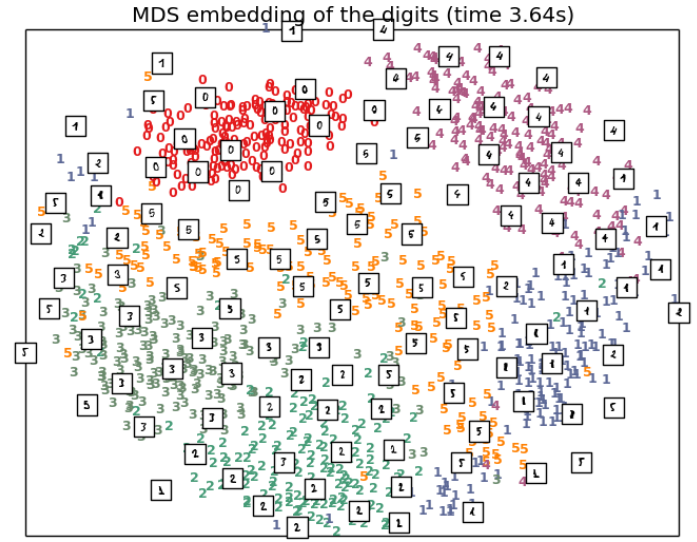
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 4 62 146 182 254 254 181 176 139 15 0 0 0 0 0 0 0 ...]

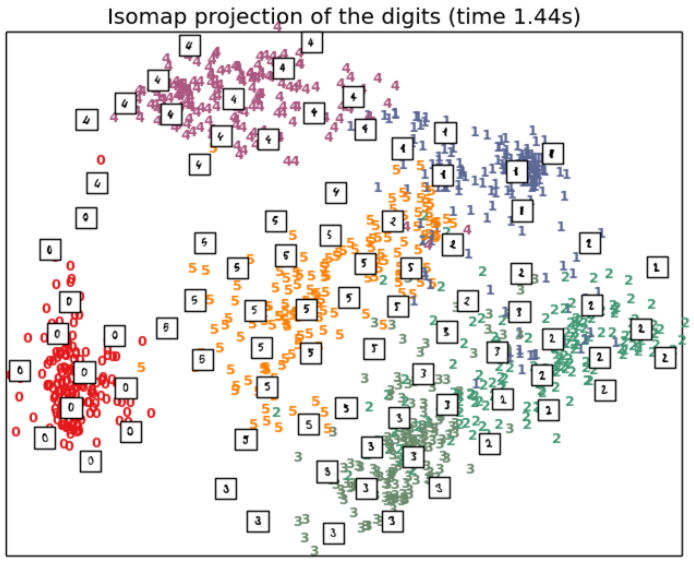
https://projector.tensorflow.org/

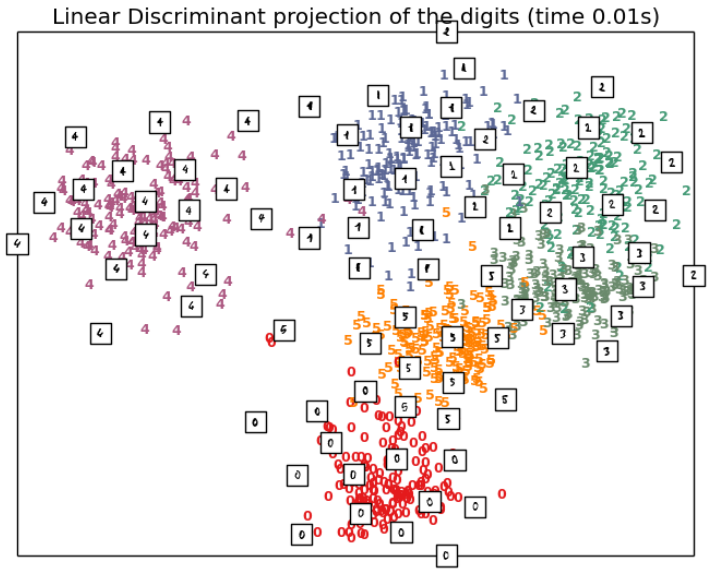
Cons:

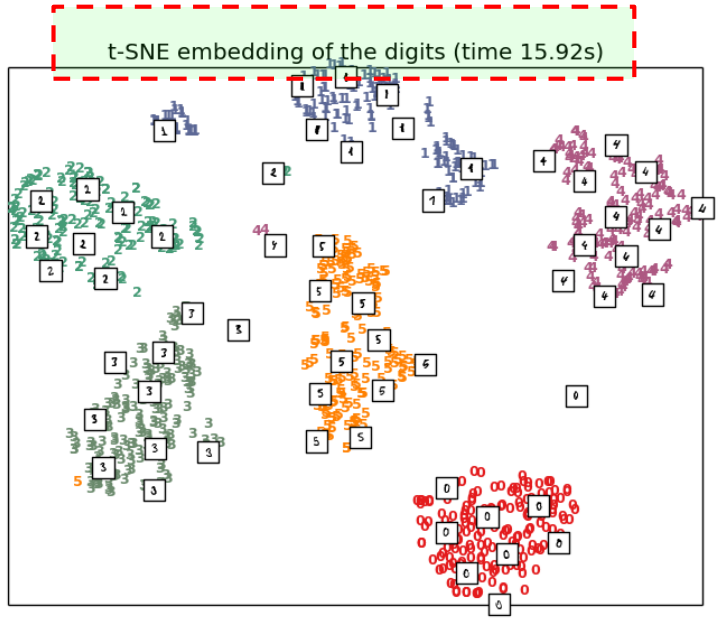
- Too sensitive – may discover clusters in multivariate Normal data
- Cannot be used prognostically: need to rerun t-SNE for each new observation









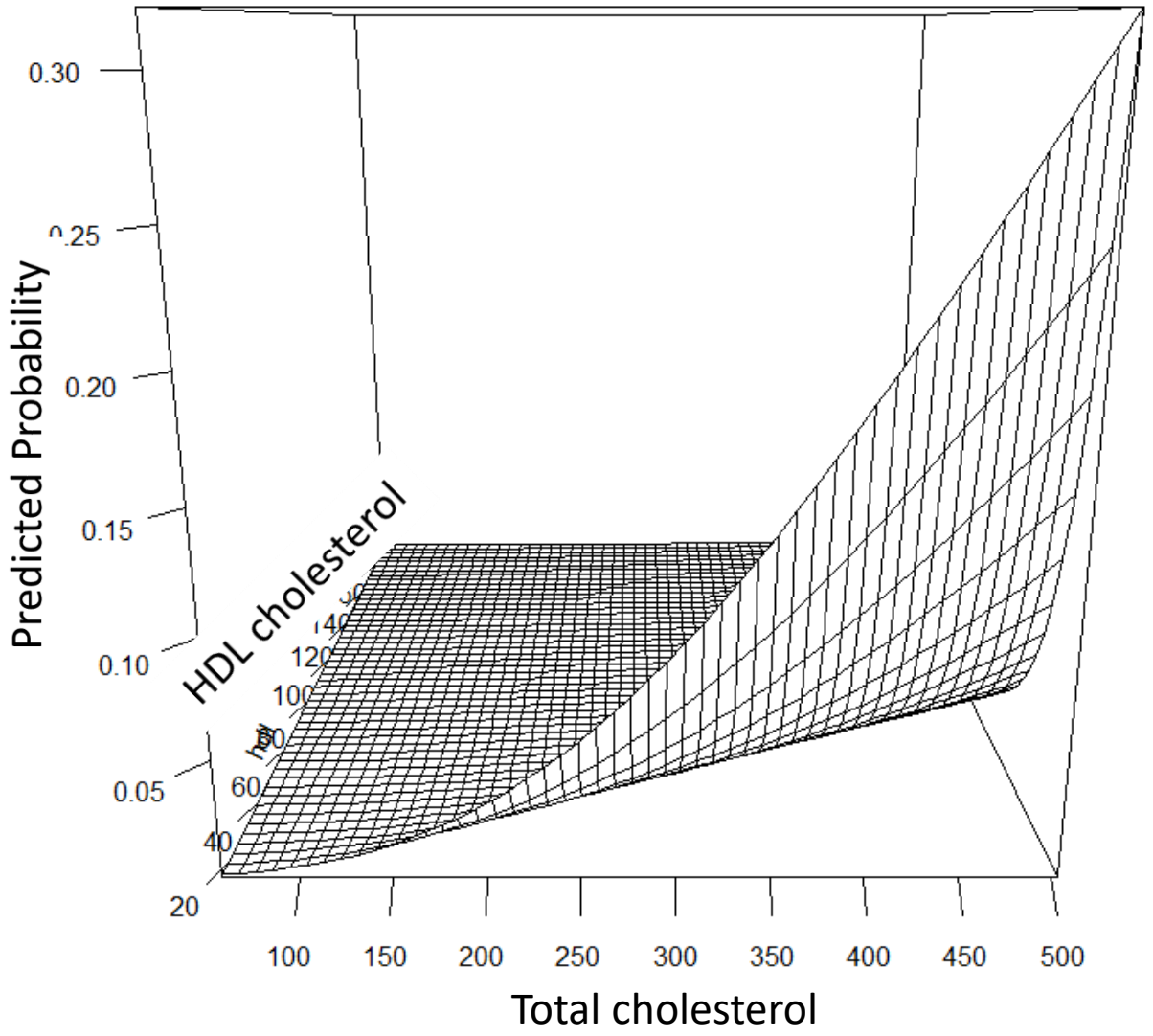


http://scikit-learn.org/stable/auto_examples/manifold/plot_lle_digits.html

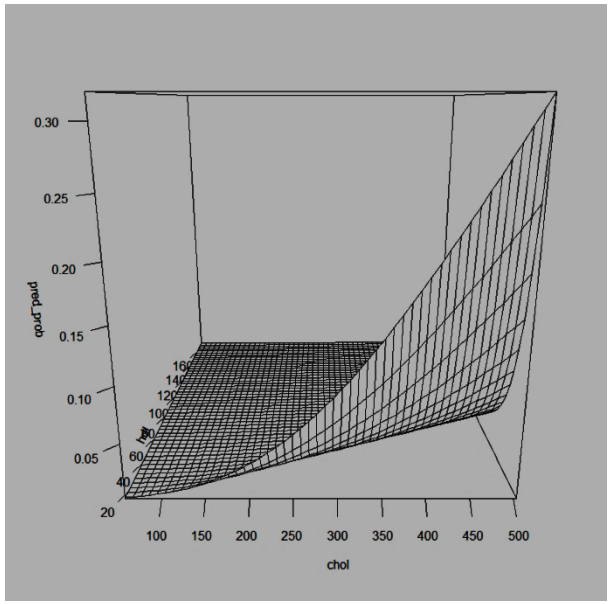
Classification And Regression Trees (CART)

From Logistic Regression to Supervised Learning Methods

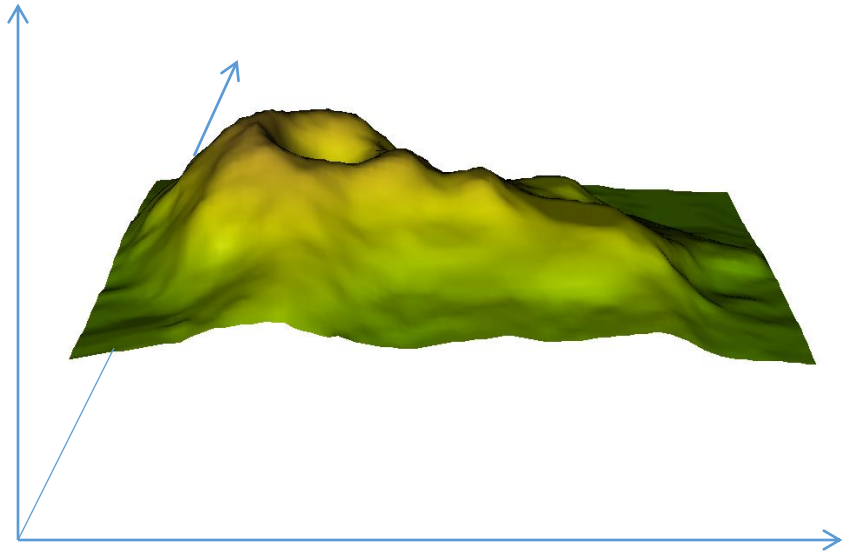
Predicted probability versus total cholesterol and hdl cholesterol



From Logistic Regression to Supervised Learning Methods



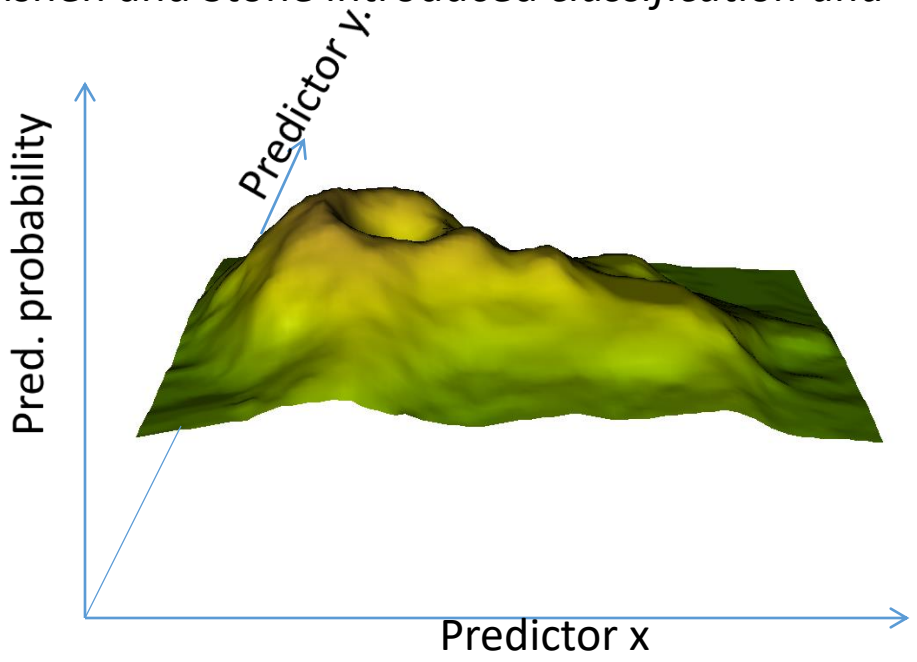
Pred. prob



Supervised Learning.
Classification and Regression Trees (CART)

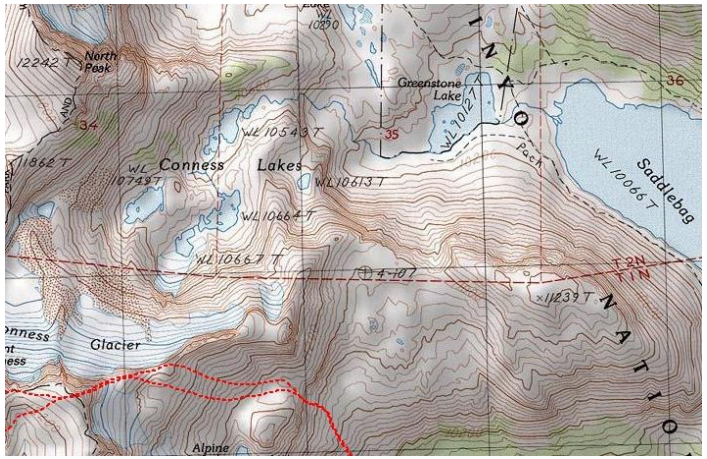
1980s Breiman, Friedman, Olshen and Stone introduced *classification and regression trees*

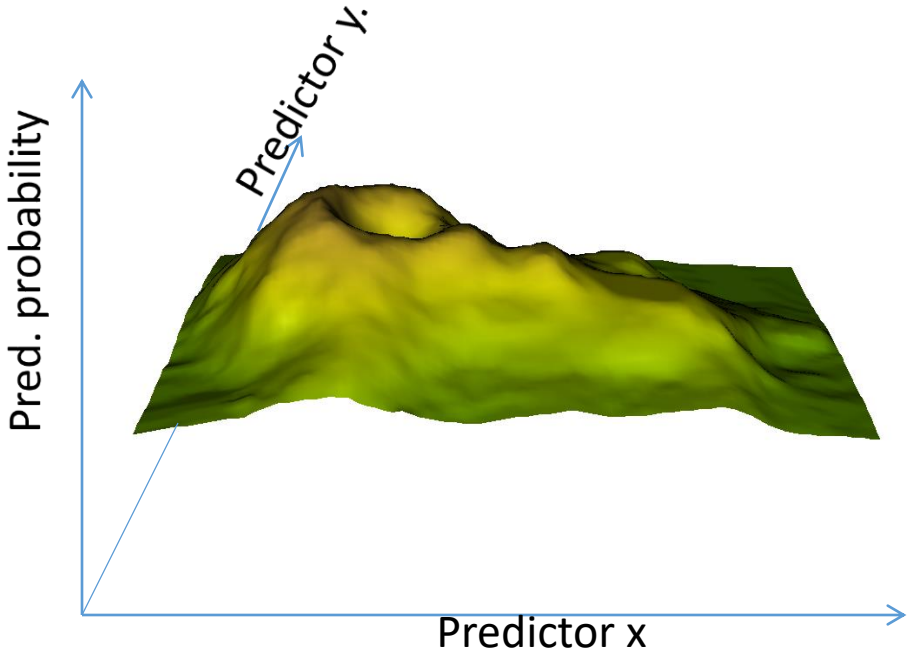
JH Friedman



Era of more powerful computers. The first example in this talk of a statistical learning method

Supervised Learning.
Classification and Regression Trees (CART)

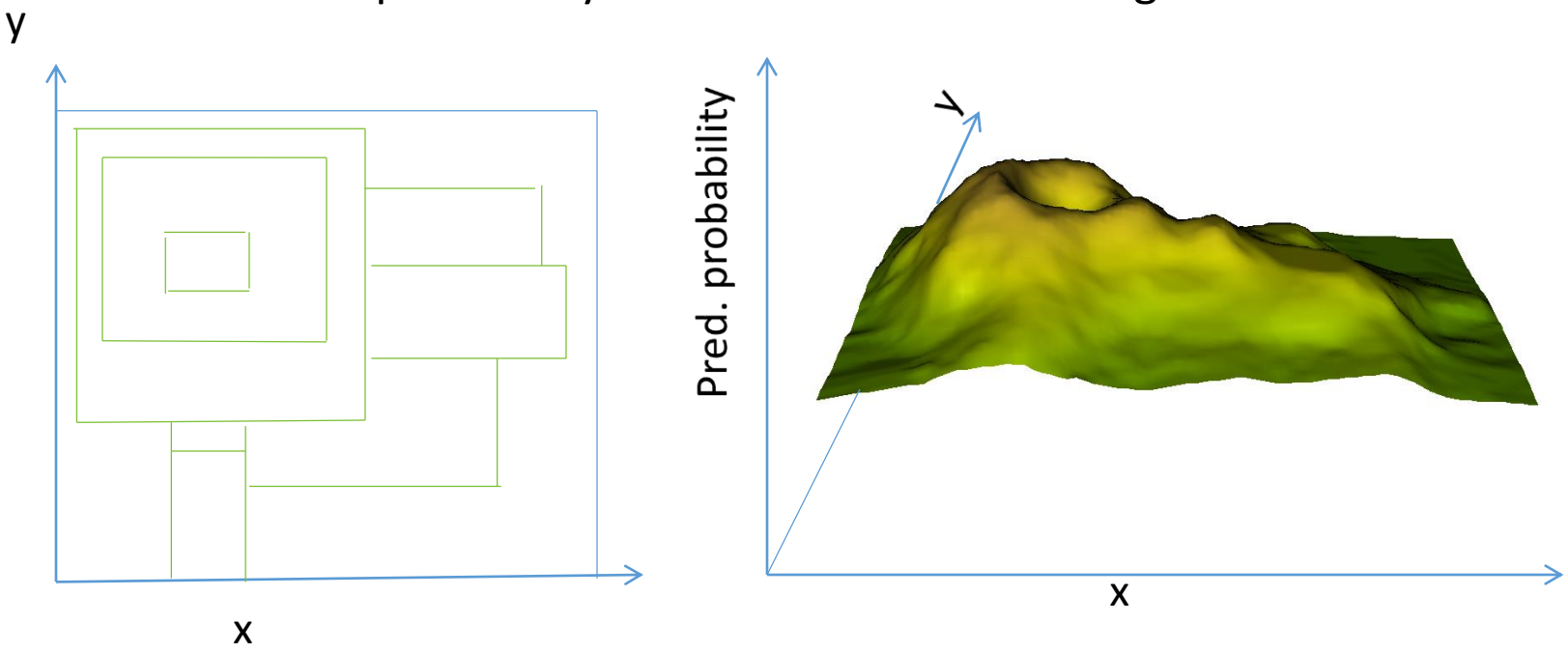




Main idea: Split the space into regions similarly to topographic maps and model constant probability of an outcome in each region

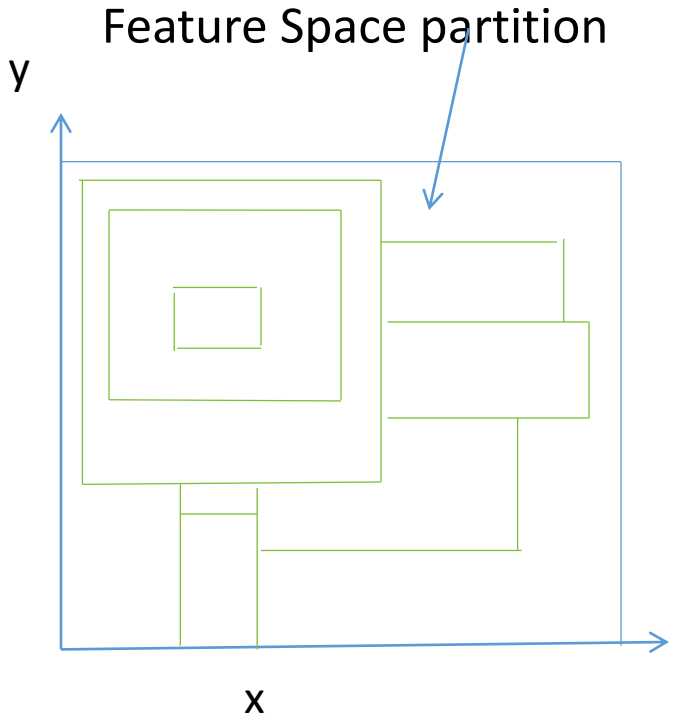
Supervised Learning.
Classification and Regression Trees (CART)

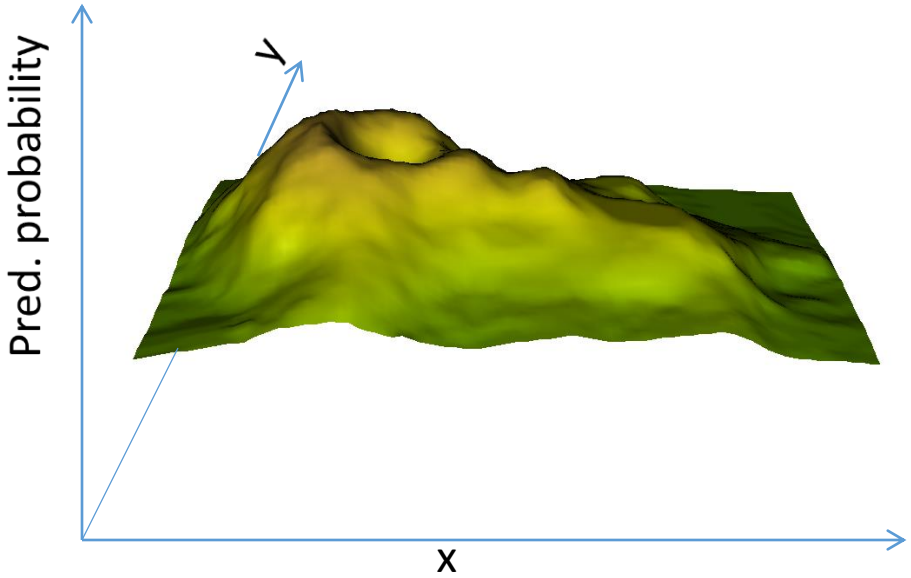
Main idea: Split the space into regions and model constant probability of an outcome in each region



Supervised Learning.
Classification and Regression Trees (CART)

Terminology

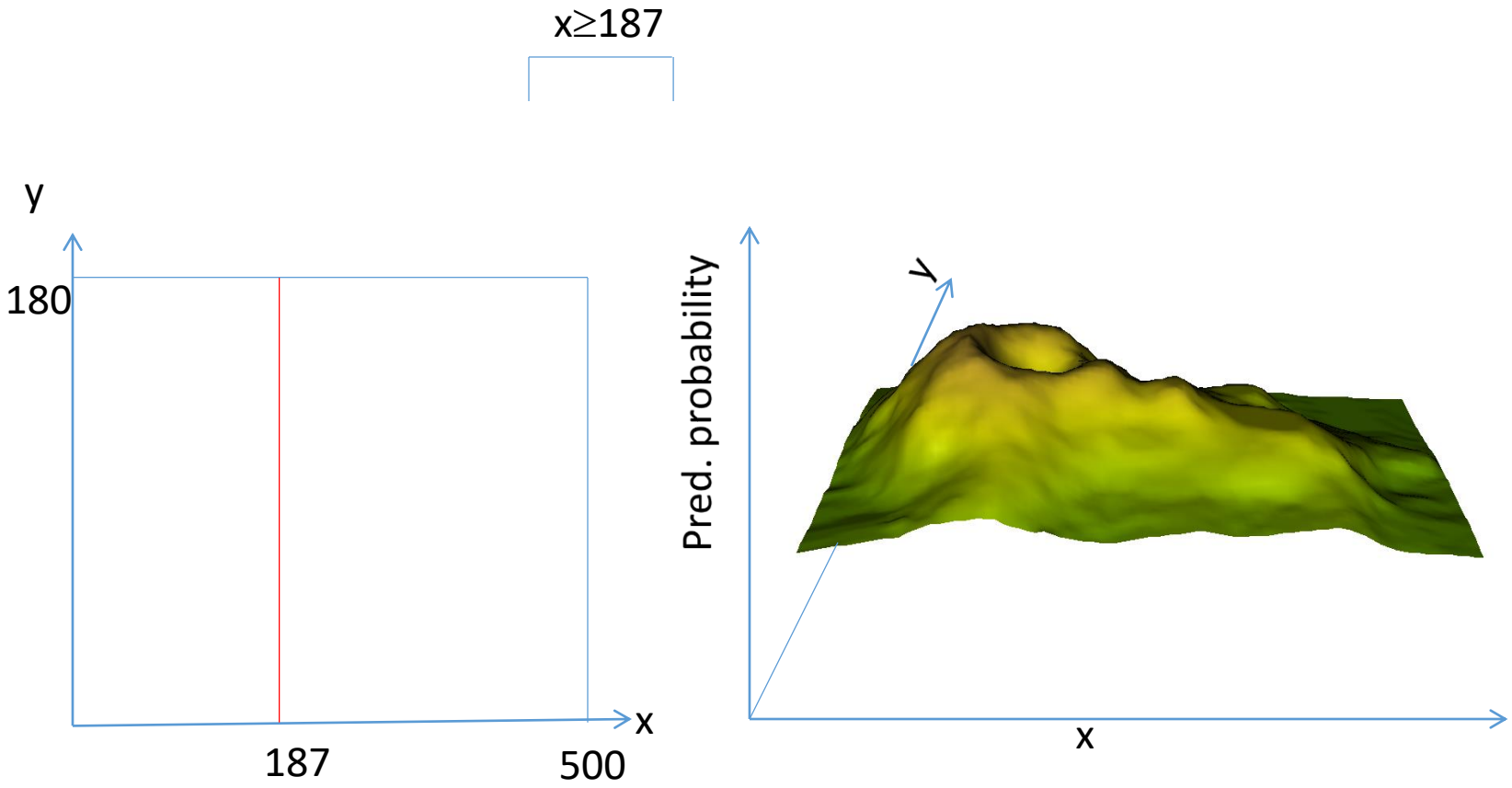




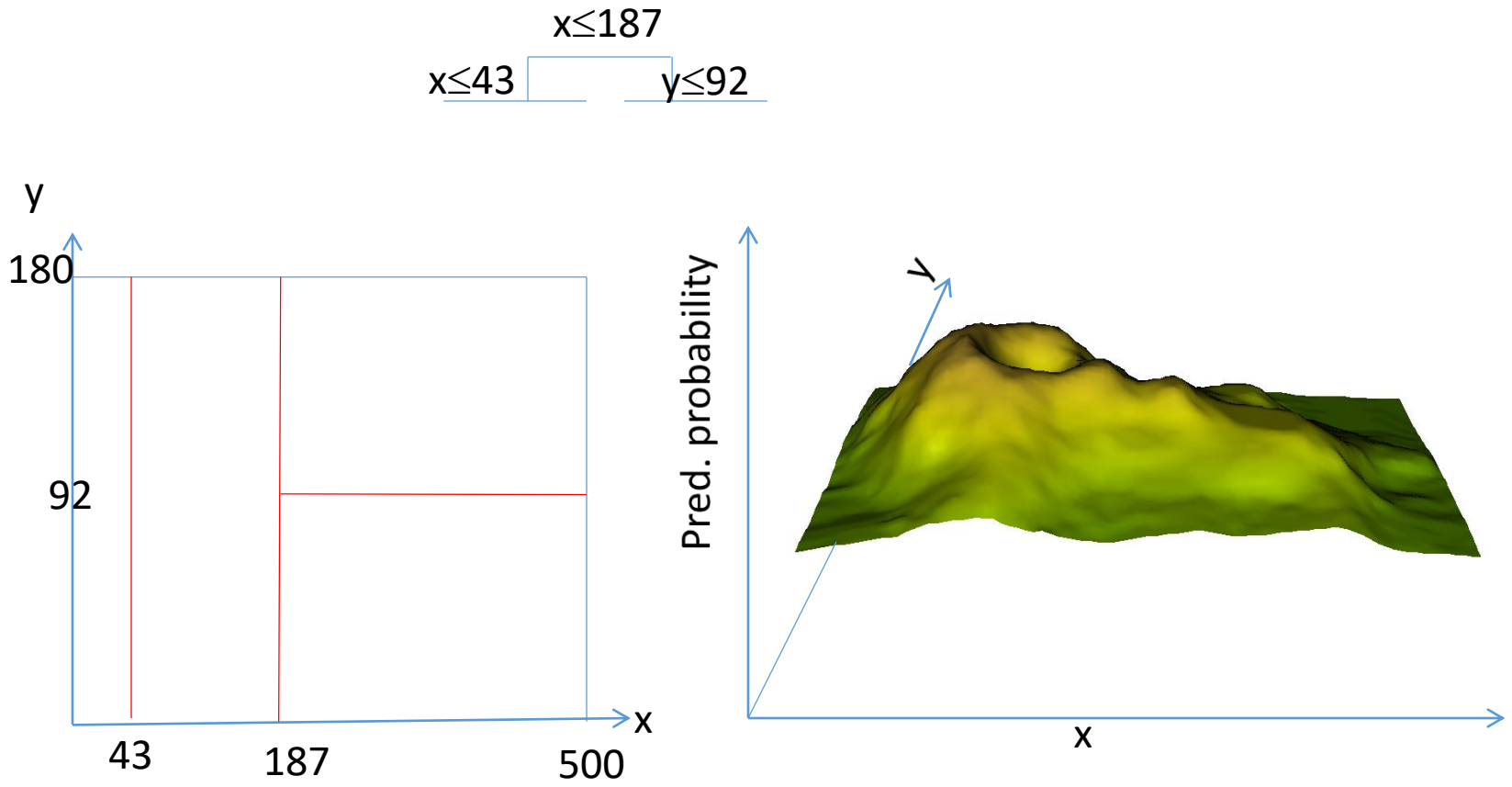
Features, Inputs, Attributes =predictor variables

Response=outcome variable

Supervised Learning.
Classification and Regression Trees (CART)

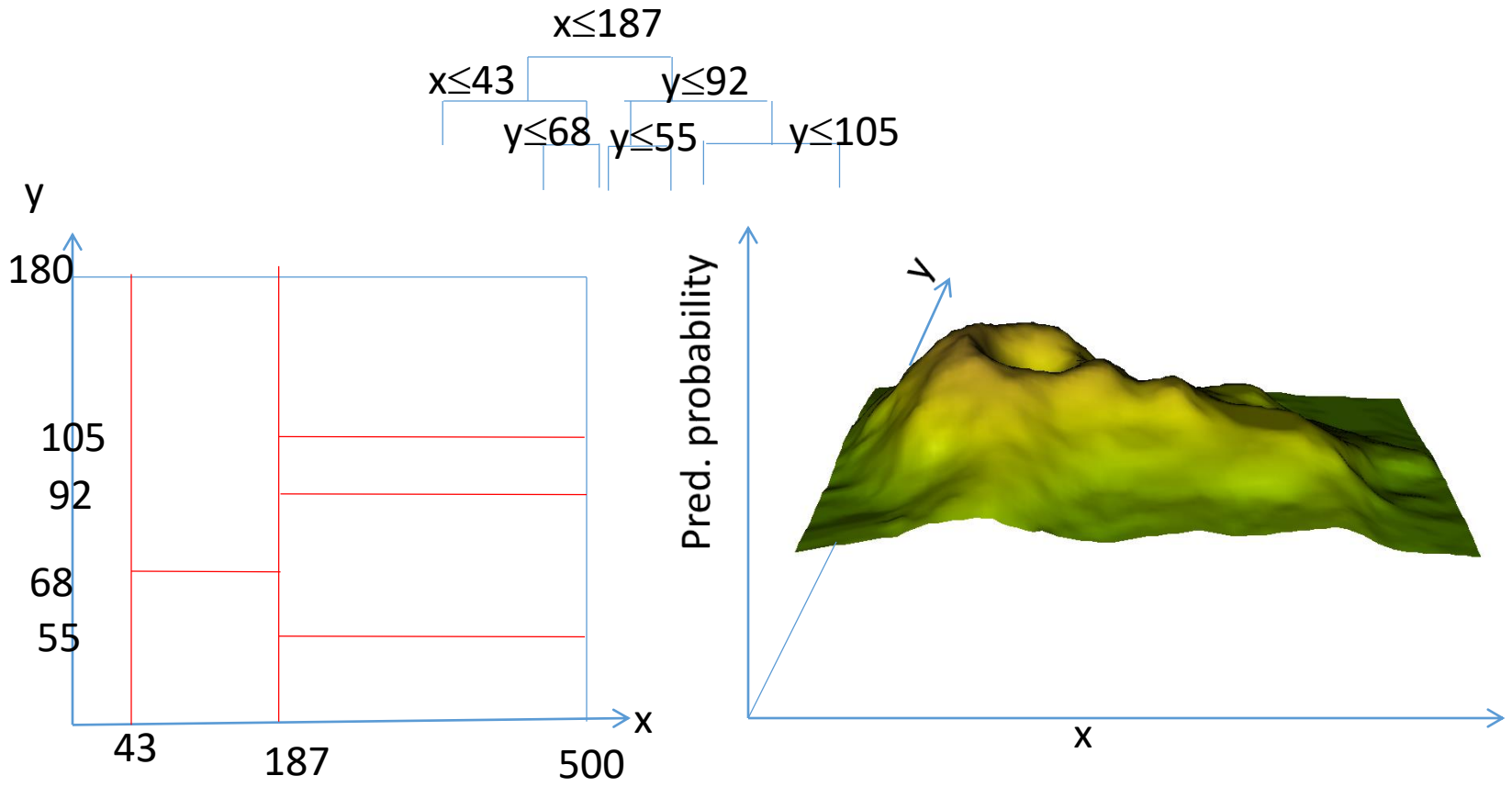


Supervised Learning.
Classification and Regression Trees (CART)



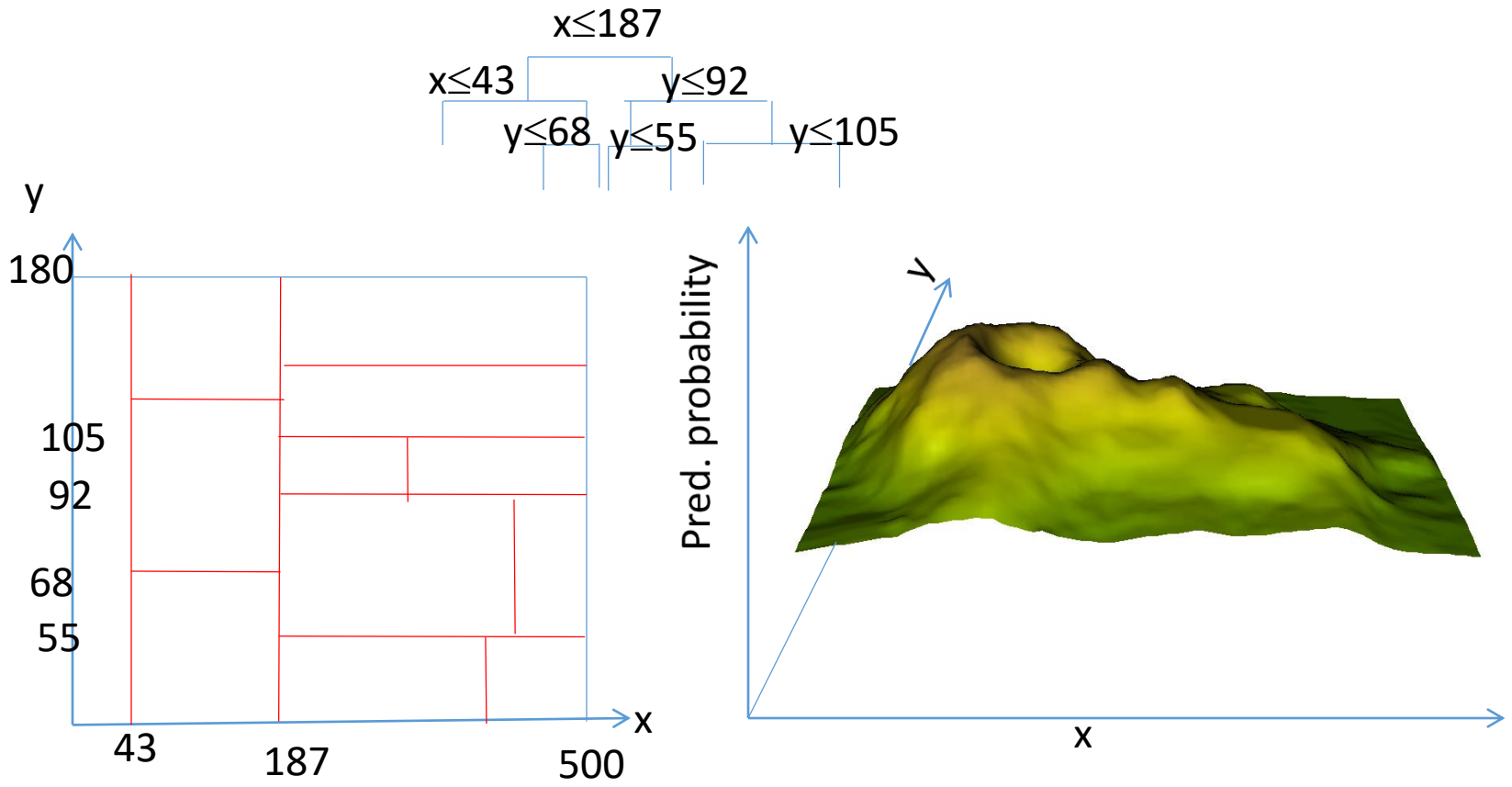
At each node select **the best predictor** for splitting **and the best cutoff value**

Supervised Learning.
Classification and Regression Trees (CART)



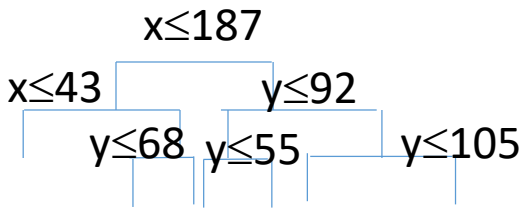
At each node select **the best predictor** for splitting and **the best cutoff value**

Supervised Learning.
Classification and Regression Trees (CART)



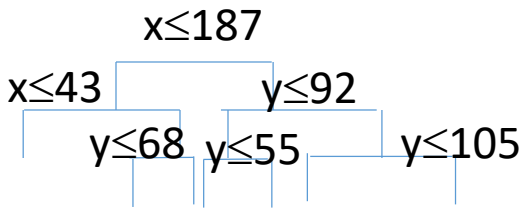
At each node select **the best predictor** for splitting and **the best cutoff value**

Supervised Learning.
Classification and Regression Trees (CART)



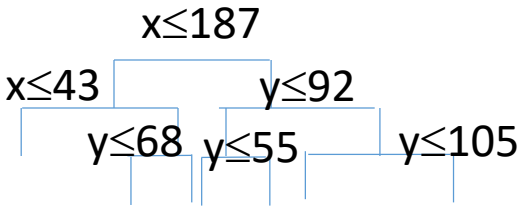
- At each node select **the best predictor** for splitting and **the best cutoff value**

Supervised Learning.
Classification and Regression Trees (CART)



- At each node select **the best predictor** for splitting and **the best cutoff value**
- The goal is to separate cases from non-cases as much as possible.

Supervised Learning.
Classification and Regression Trees (CART)



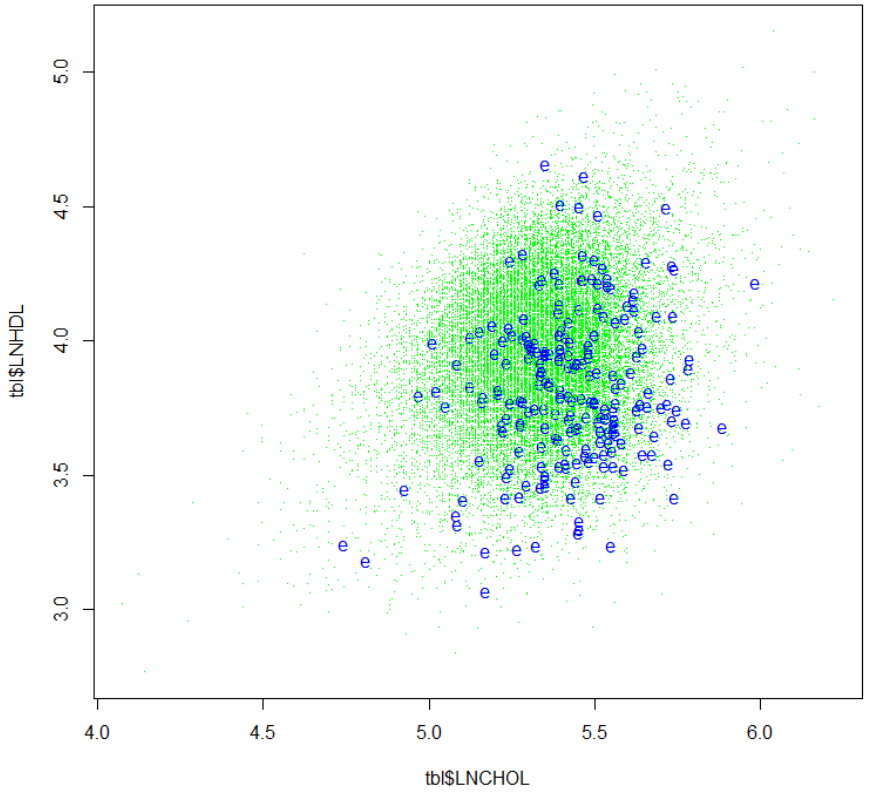
- At each node select **the best predictor** for splitting and **the best cutoff value** based on the node impurity measures: misclassification error or Gini index or cross-entropy or deviance
- The goal is to separate cases from non-cases as much as possible.

CART algorithm is an example of the **greedy algorithm.**

A **greedy algorithm** is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum. In many problems, a greedy strategy does not in general produce
an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a global optimal solution in a reasonable time.

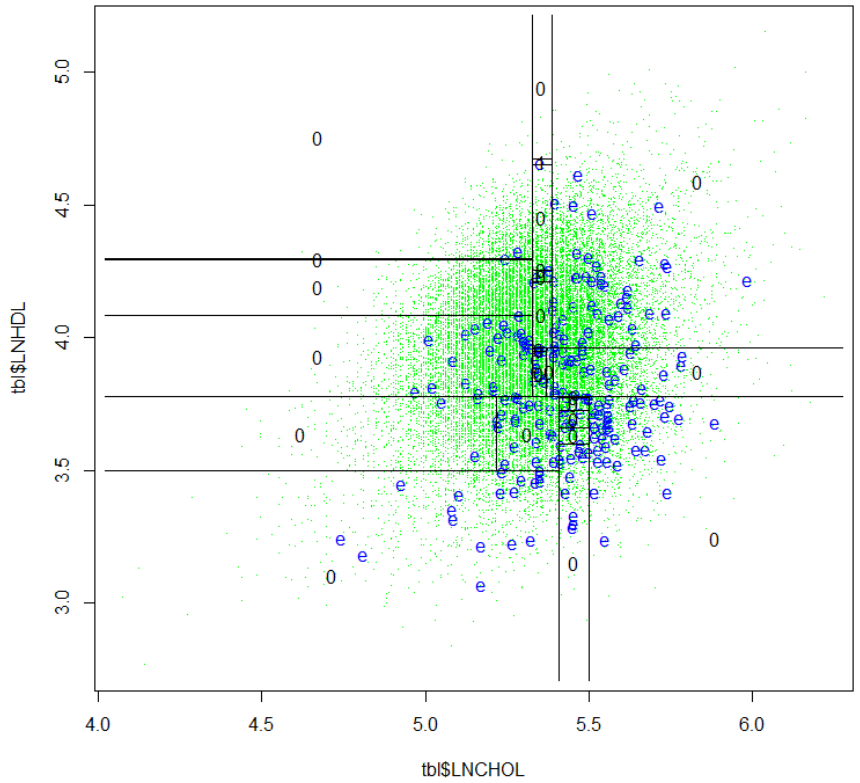
Supervised Learning.
Classification and Regression Trees (CART)

Example: predicting hard CHD using log(total cholesterol) and log(HDL cholesterol)



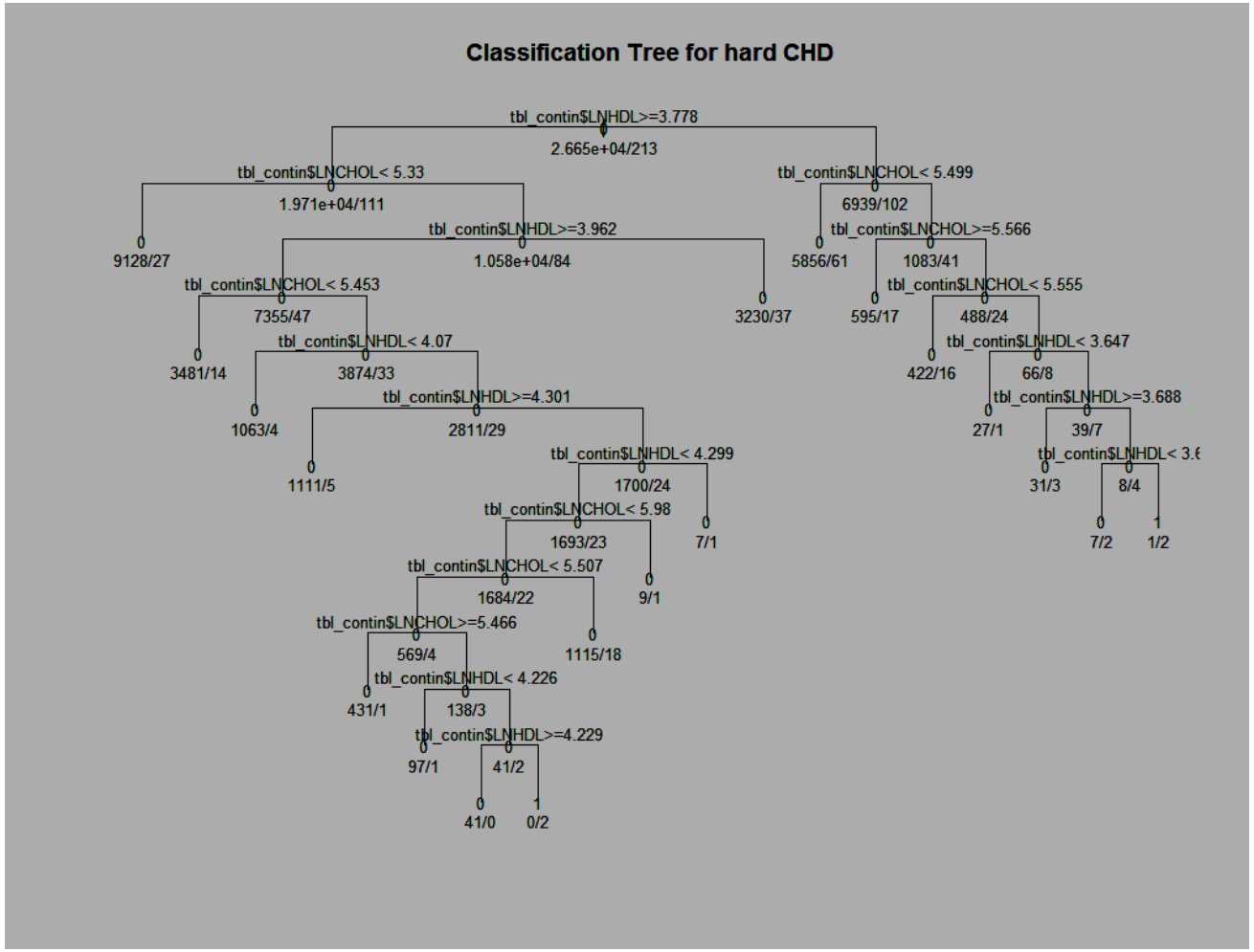
Supervised Learning.
Classification and Regression Trees (CART)

Example: predicting hard CHD using log(total cholesterol) and log(HDL cholesterol)

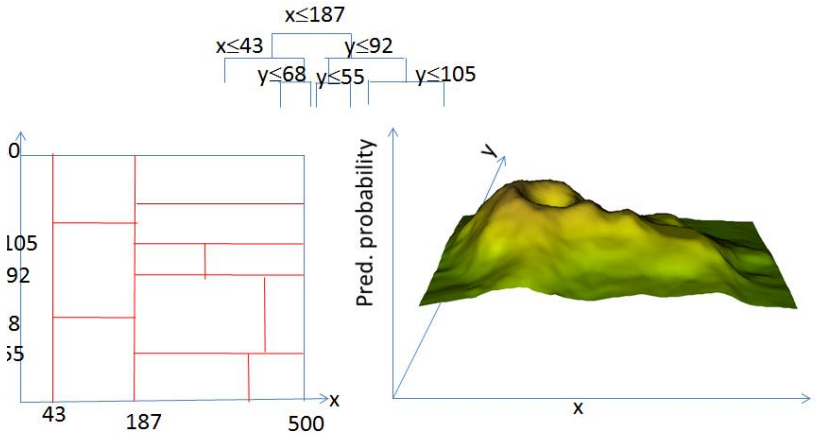


Supervised Learning.
Classification and Regression Trees (CART)

Example: predicting hard CHD using log(total cholesterol) and log(HDL cholesterol)



Supervised Learning.
Classification and Regression Trees (CART)



It is very easy to create a very large tree.
Large tree results in over-fitting the data.
So **prune** the tree!

Cost-complexity pruning:
By collapsing internal nodes, select the **least complex** but **most accurate** tree:
Tradeoff between the **size** and **goodness of fit of the tree**

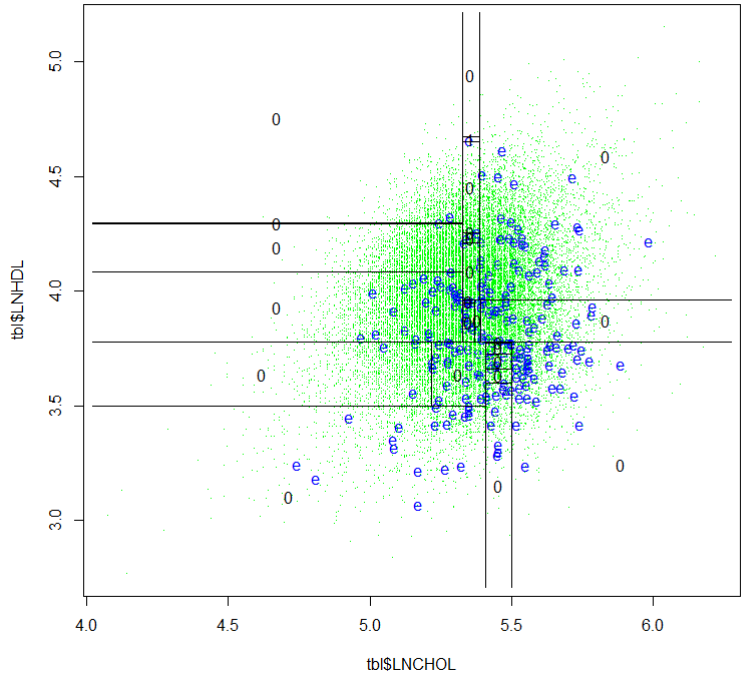
Misclassification error; Gini index; Cross-entropy or deviance

Supervised Learning.
Classification and Regression Trees (CART)

Example: predicting hard CHD using log(total cholesterol) and log(HDL cholesterol)

Advantages:

- these models go after interactions immediately
- Easy to explain and interpret



Supervised Learning.
Classification and Regression Trees (CART)

CARTS can be unstable.

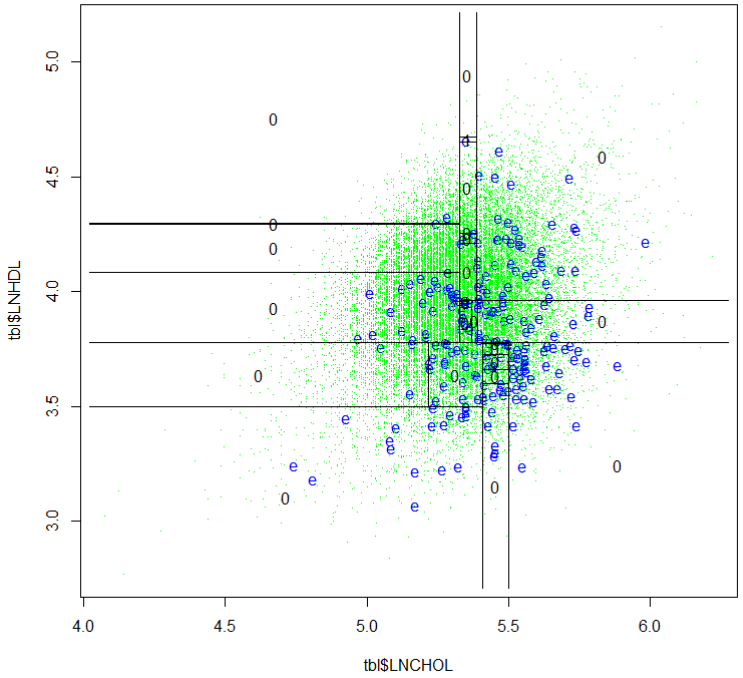
whole sample

removed 1% of obs

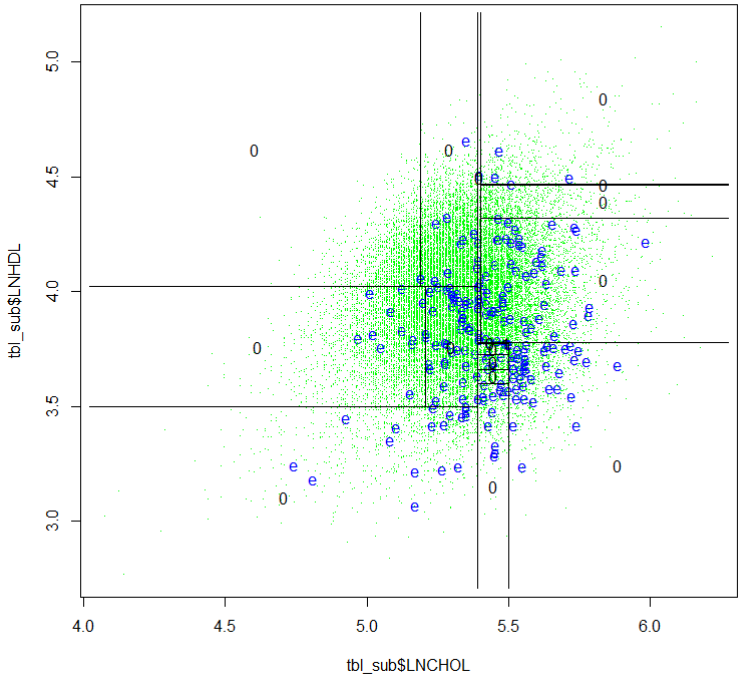
Supervised Learning.
Classification and Regression Trees (CART)

CARTS can be unstable.

whole sample



removed 1% of obs



Supervised Learning.
Classification and Regression Trees (CART)

Example: predicting hard CHD using log(total cholesterol) and log(HDL cholesterol)

Advantages:

- These models go after interactions immediately
- Easy to explain

Disadvantages:

- Results can be unstable

Supervised Learning.
Classification and Regression Trees (CART)

Example: predicting hard CHD using log(total cholesterol) and log(HDL cholesterol)

Advantages:

- These models go after interactions immediately
- Easy to explain

Disadvantages:

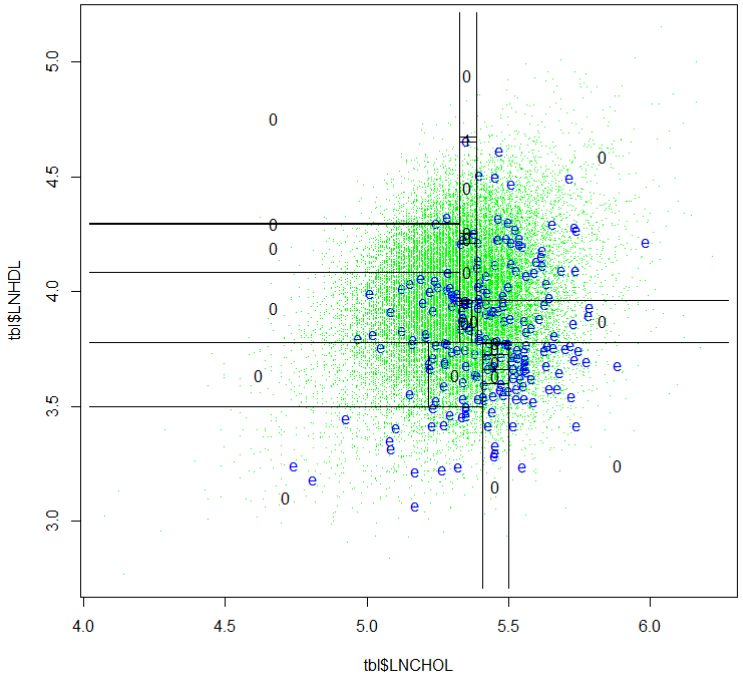
- Results can be unstable
- Hard to access uncertainty in inference about trees
- Hard to model linear relationships – allow linear splits

Random Forests

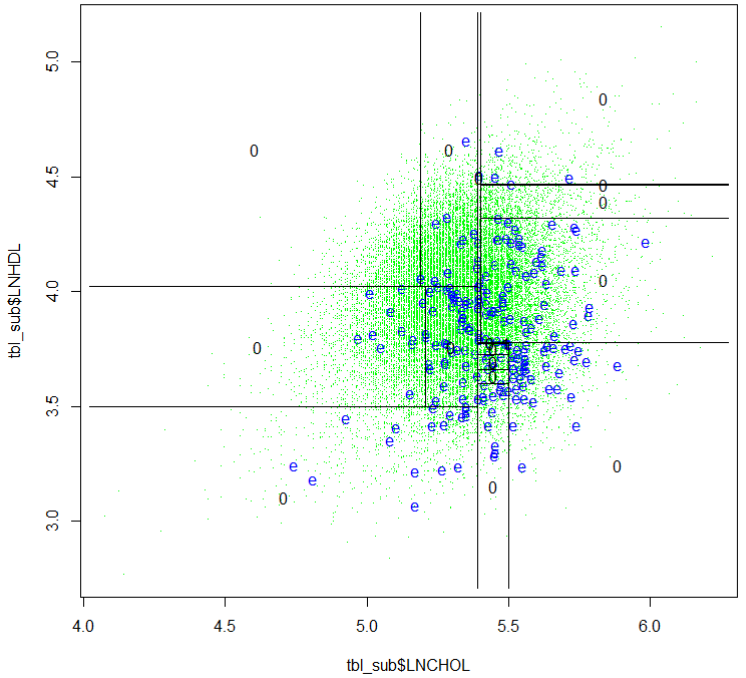
Extensions of CARTs: Random Forests. Main Idea.

1. CARTS can be unstable.

whole sample



removed 1% of obs



Extensions of CARTs: Random Forests. Main Idea.

1. CARTS can be unstable.
2. To stabilize them: main idea:

An average has lower variance than the random variable, therefore grow several trees and classify an observation by “majority vote” (called “bagging”)

Also called a committee of trees

Decision is made by consensus

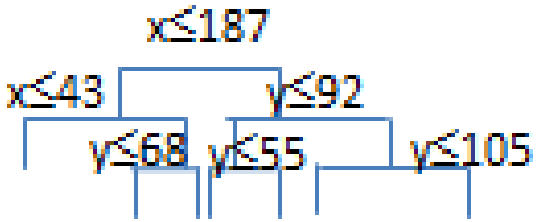
Extensions of CARTs: Random Forests. Main Idea

1. In 2001 Leo Breiman and Adele Cutler

Extensions of CARTs: Random Forests. Main Idea

Add two more steps:

1. At each node
    1. select a set of candidate predictors at random (hence Random)



Extensions of CARTs: Random Forests. Main Idea

Add two more steps:

1. At each node
    1. select a set of candidate predictors at random (hence Random)

Showed that final prediction works better (is much more stable) if trees are uncorrelated.

Ways to de-correlate the trees:

- Select random set of candidate predictors

Extensions of CARTs: Random Forests. Main Idea

Add two more steps:

1. At each node
    1. select a set of candidate predictors at random (hence Random)

Can be as many as sqrt(p) and as few as one

Showed that final prediction works better (is much more stable) if trees are uncorrelated.

Ways to de-correlate the trees:

- Select random set of candidate predictors

Extensions of CARTs: Random Forests. Main Idea

Add two more steps:

1. At each node
    1. select a set of candidate predictors at random (hence Random)

Showed that final prediction works better (is much more stable) if trees are uncorrelated.

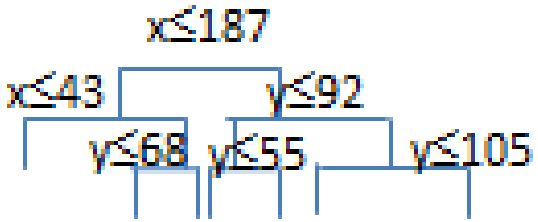
Ways to de-correlate the trees:

- Select random set of candidate predictors
- Use slightly different datasets (bootstrapping)

Extensions of CARTs: Random Forests. Main Idea

Add two more steps:

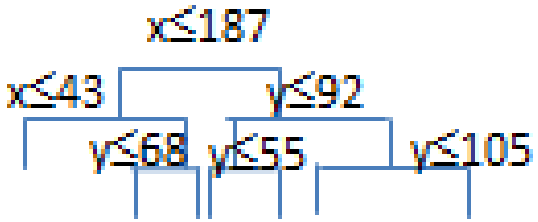
1. At each node
    1. select a set of candidate predictors at random (hence Random)
    2. Find the best predictor to use in the split at that node and optimal threshold



Extensions of CARTs: Random Forests. Main Idea

Add two more steps:

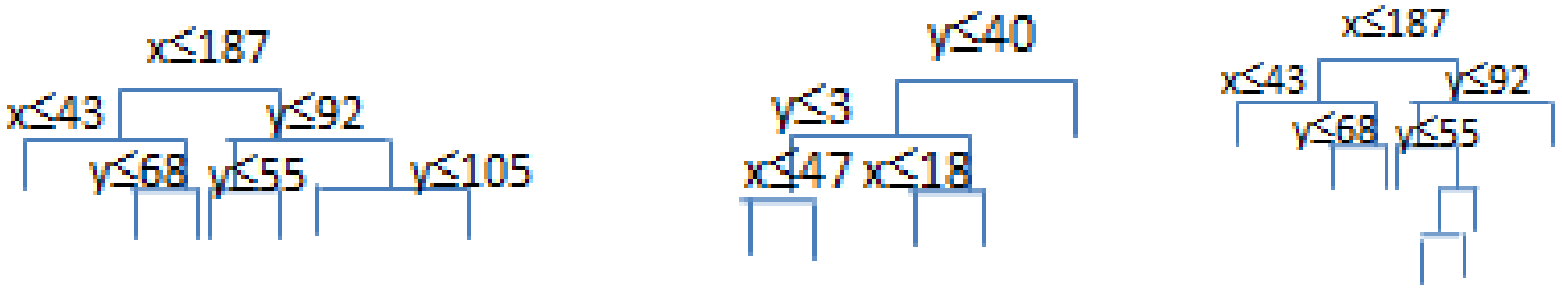
1. At each node
    1. select a set of candidate predictors at random (hence Random)
    2. Find the best predictor to use in the split at that node and optimal threshold
    3. Repeat



Extensions of CARTs: Random Forests. Main Idea

Add two more steps:

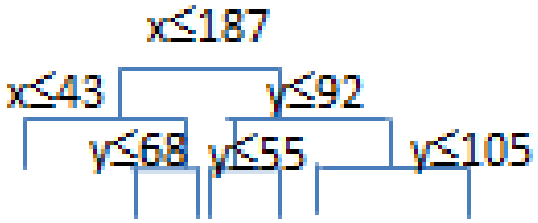
1. At each node
    1. select a set of candidate predictors at random (hence Random)
    2. Find the best predictor to use in the split at that node and optimal threshold
    3. Repeat
2. Create several trees (hence Forest)

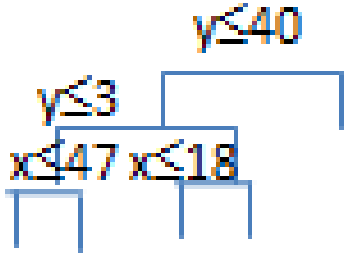


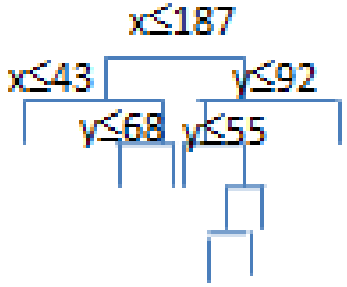
Extensions of CARTs: Random Forests. Main Idea

Add two more steps:

1. At each node
    1. select a set of candidate predictors at random (hence Random)
    2. Find the best predictor to use in the split at that node
    3. Repeat
2. Create several trees (hence Forest)
3. Classify an observation by taking the majority vote from the trees







Extensions of CARTs: Random Forests. Main Idea

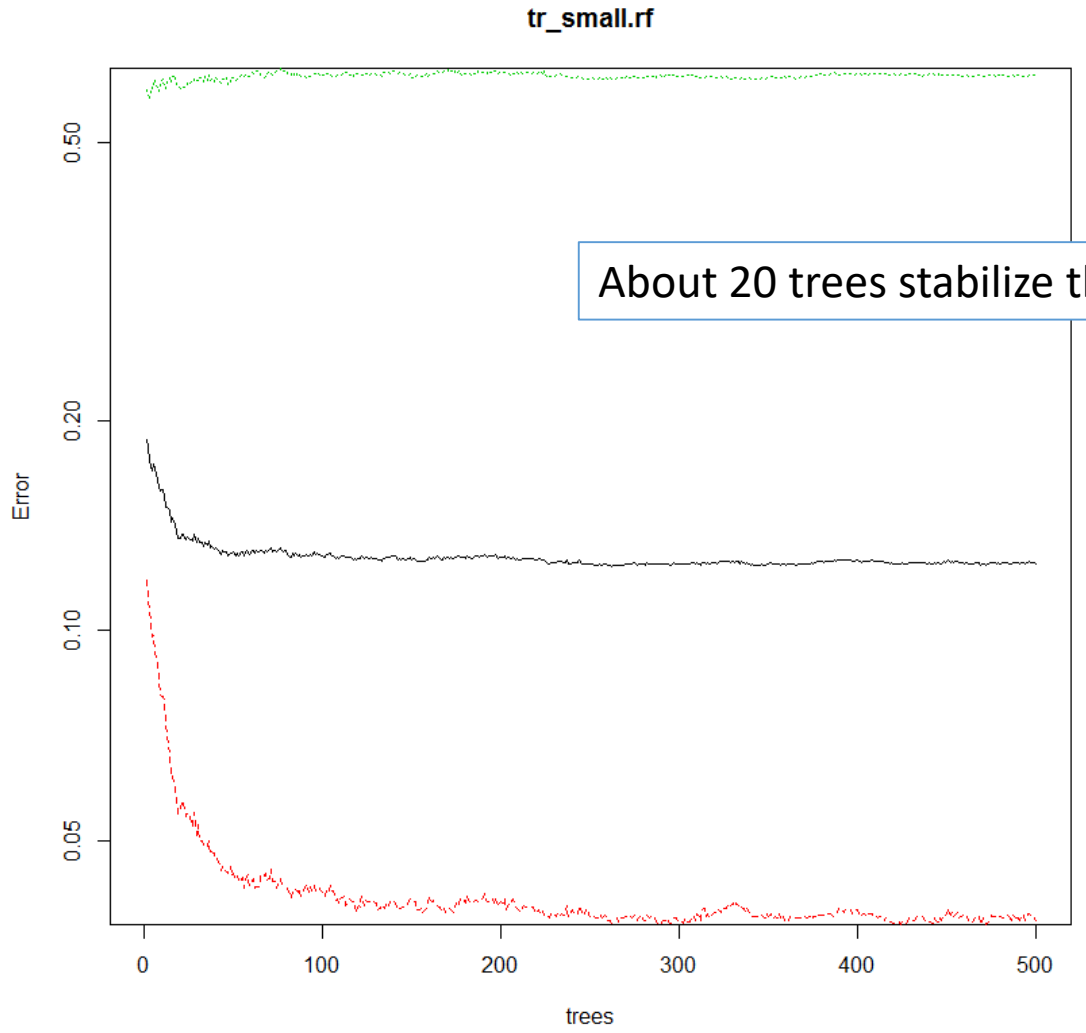
Add two more steps:

Can be as many as sqrt(p) and as few at one

1. At each node
    1. select candidate predictors at random (hence Random)
    2. Find the best predictor to use in the split at that node
    3. Repeat
2. Create several trees (hence Forest)
3. Classify an observation by taking the majority vote from the trees

Called Bagging

Extensions of CARTs: Random Forests. Main Idea



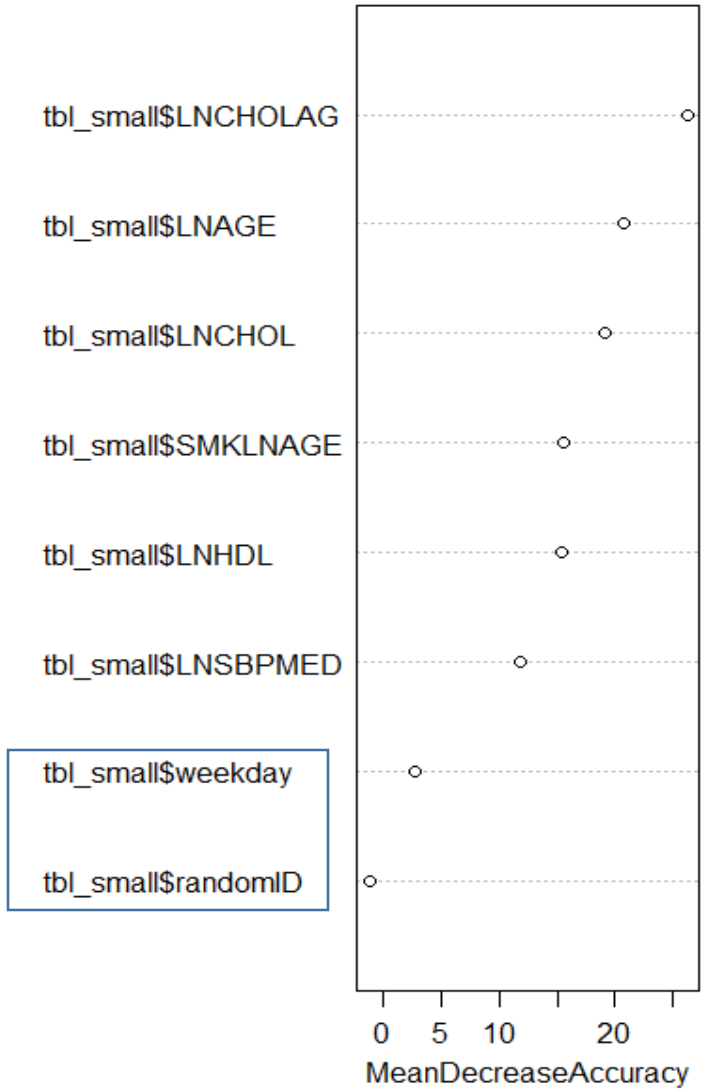
About 20 trees stabilize the prediction in our data

Extensions of CARTs: Random Forests.

Pros

1. One of the best-performing methods – stable
2. Accurate
3. Robust to outliers
4. Internal estimates of error, strength, variable importance
5. Simple to train and tune
6. Often used in variable selection

Extensions of CARTs: Random Forests.



Random Forests are

often used in variable selection

Short review of predictive models.
Random Forests

Pros:

1. One of the best-performing methods – stable
2. Accurate
3. Robust to outliers
4. Internal estimates of error, strength, variable importance
5. Simple to train and tune
6. Often used in variable selection
7. Popular
8. Implemented in several software packages

Extensions of CARTs: Random Forests.

Pros:

1. One of the best-performing methods – stable
2. Accurate
3. Robust to outliers
4. Internal estimates of error, strength, variable importance
5. Simple to train and tune
6. Often used in variable selection
7. Popular
8. Implemented in several software packages

Cons:

1. A “black-box” model – does not give the estimates of model or function
2. Limited interpretability

Extensions of CARTs: Random Forests. Boosting. Main Idea.

Add two more steps:

1. At each node
    1. select candidate predictors at random (hence Random)
    2. Find the best predictor to use in the split at that node
2. Create several trees (hence Forest)
3. Classify an observation by taking the majority vote from the trees. Average predicted probabilities.

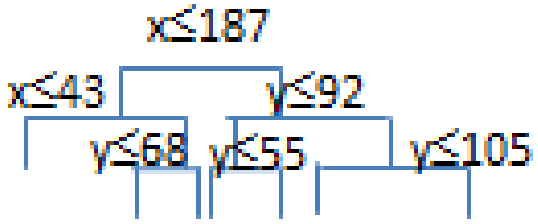
Called Bagging

Additional way to combine trees: do boosting.

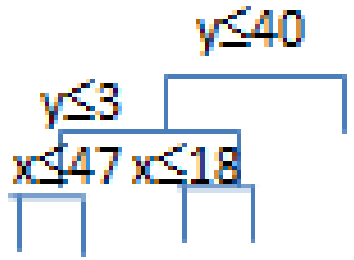
Hastie et al: "Boosting is one of the most powerful learning ideas introduced in the last twenty years"

Extensions of CARTs: Random Forests. Boosting. Main Idea.

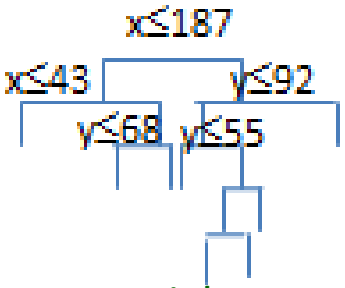
1. Calculate the first tree.
2. Use this tree to classify observation in your data.
3. Make misclassified observations “more important” by assigning them large weight.
4. Calculate new tree using weighted data.
5. Repeat M times.
6. For each tree calculate its weight – trees that perform better get more weight.
7. Calculate final prediction by taking weighted average of trees’s predictions



weight=.03



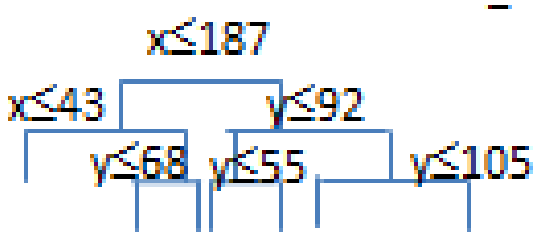
weight=.2



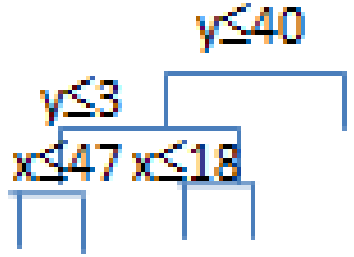
weight=.77

Extensions of CARTs: Random Forests. Boosting. Main Idea.

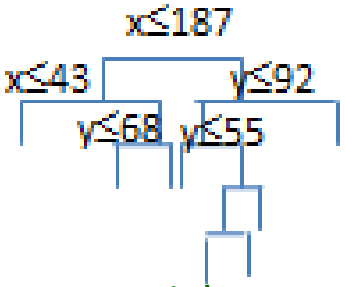
Boosting is not limited to CART but can be used with other prediction rules.



weight=.03



weight=.2

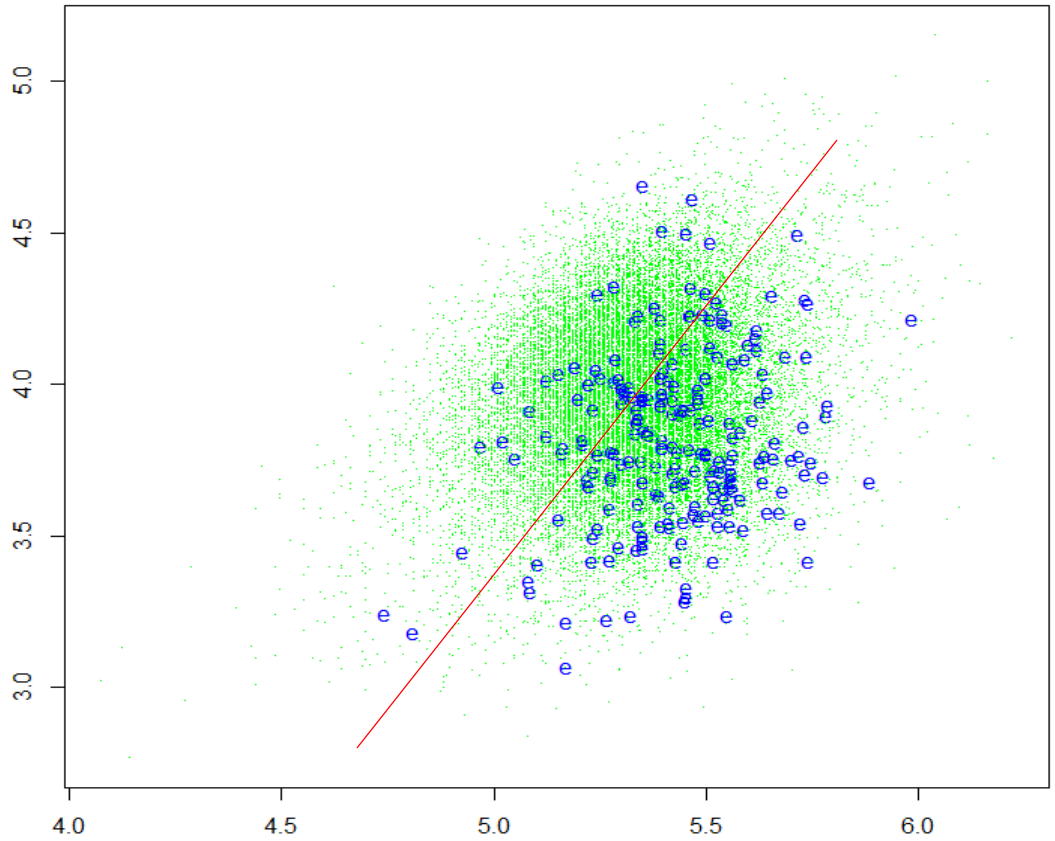


weight=.77

Support Vector Machines (SVM)

Support Vector Machines (SVM)

LDA, logistic regression results in linear boundary:



Support Vector Machines

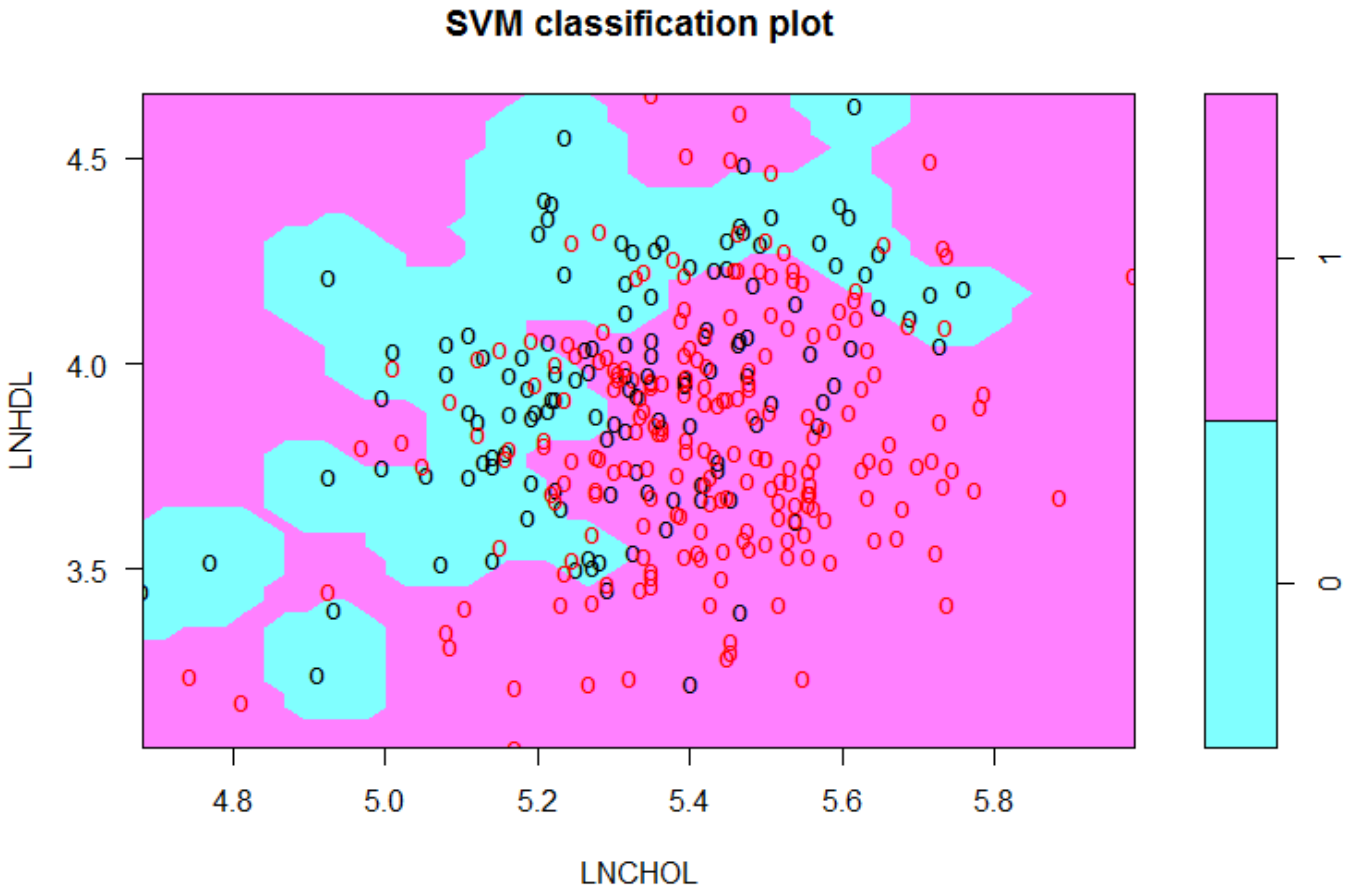
SVM Vapnik, Cortez (1996)

Support Vector Machines

SVM finds best, possibly non-linear boundaries between events and non-events:

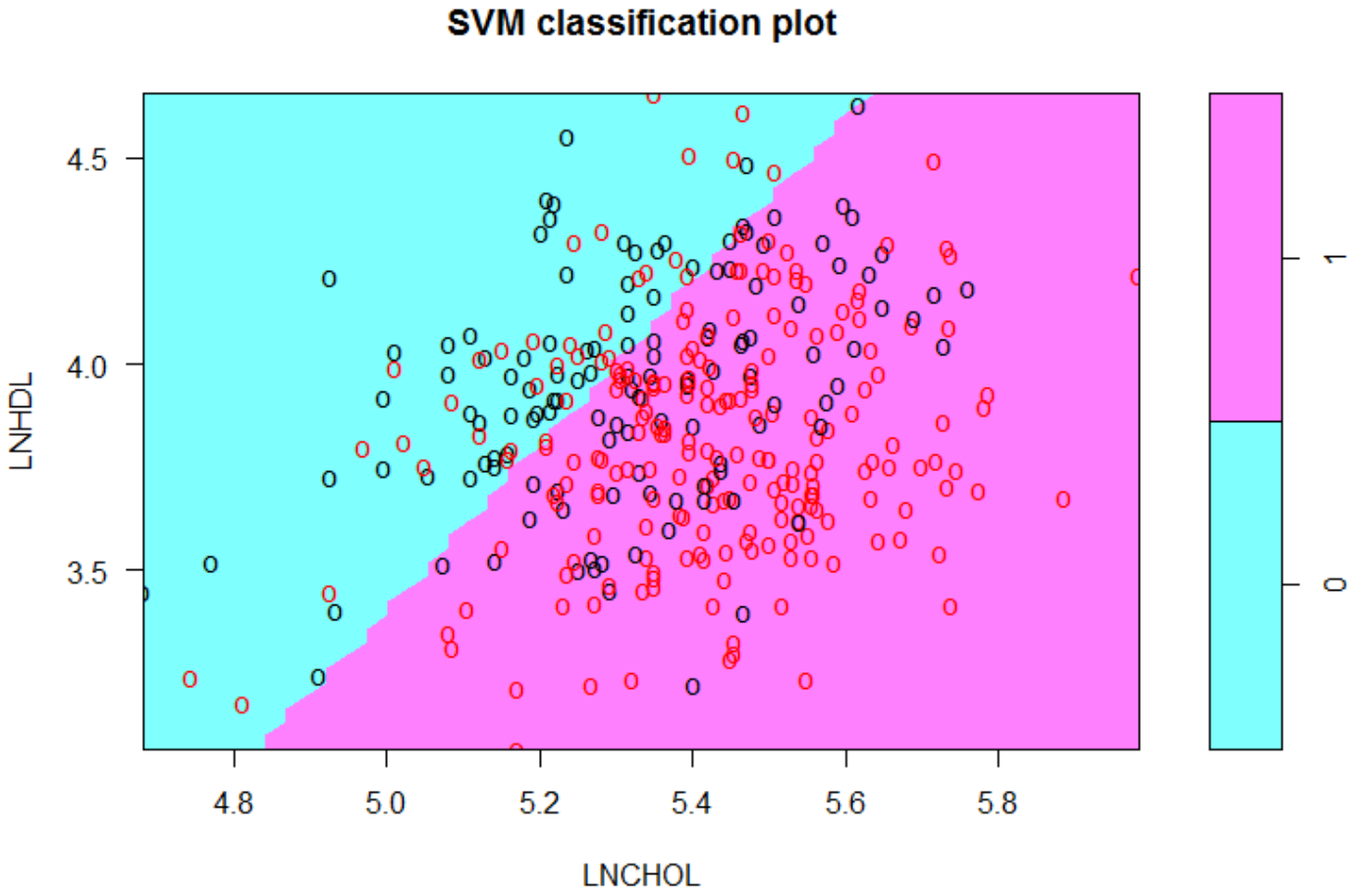
Support Vector Machines

SVM finds best, possibly non-linear boundaries between events and non-events:



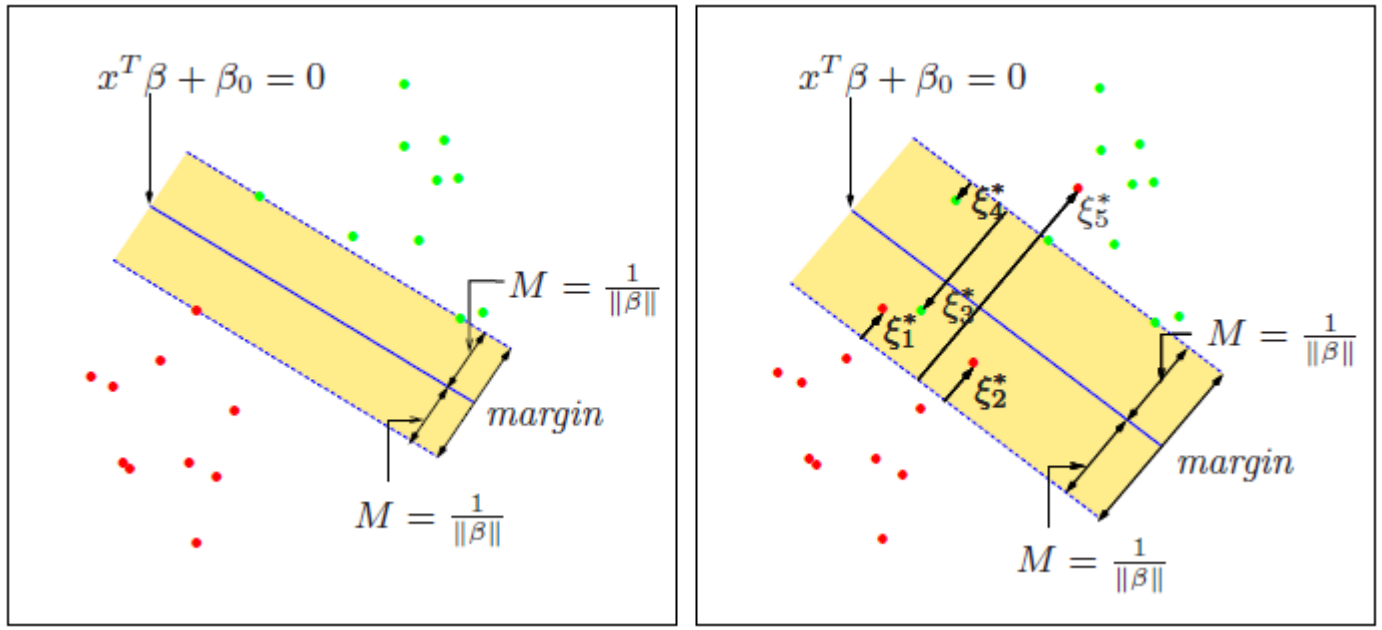
Support Vector Machines

Can control degree of non-linearity:



Support Vector Machines

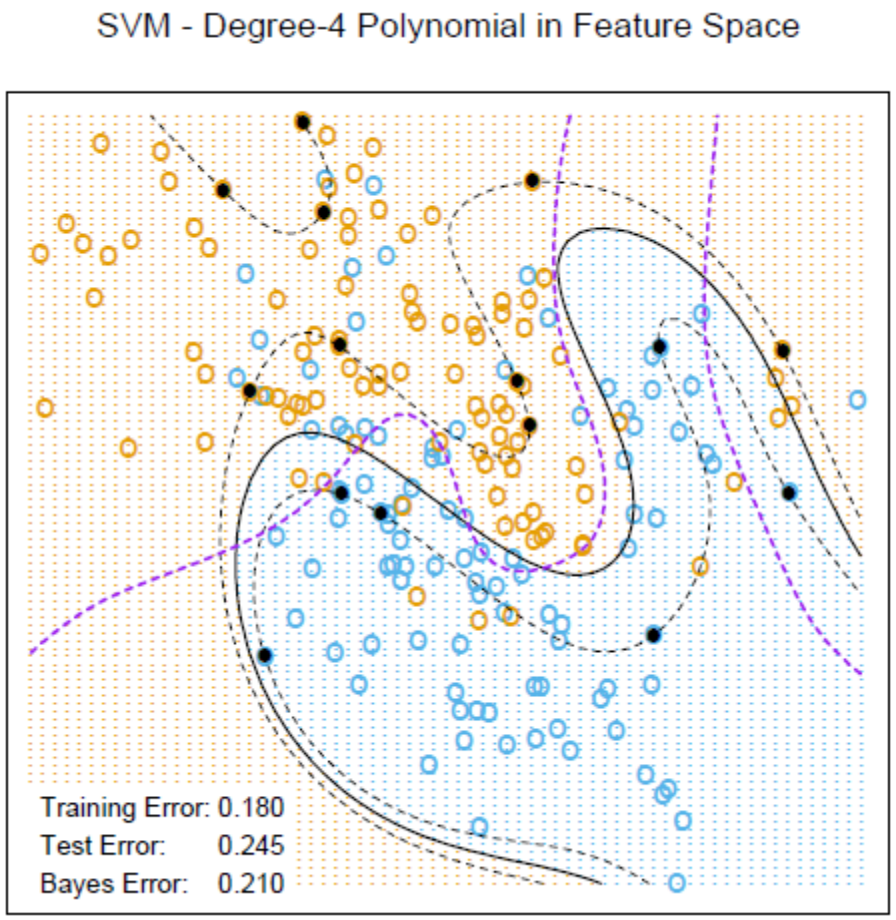
Main Idea:



Hastie et al

Left: separable case. Find the line that separates two groups by the widest margin.
Right: Groups overlap. Some observations are misclassified. Fix amount of allowed misclassification while find the boundary that has the largest margin

Support Vector Machines

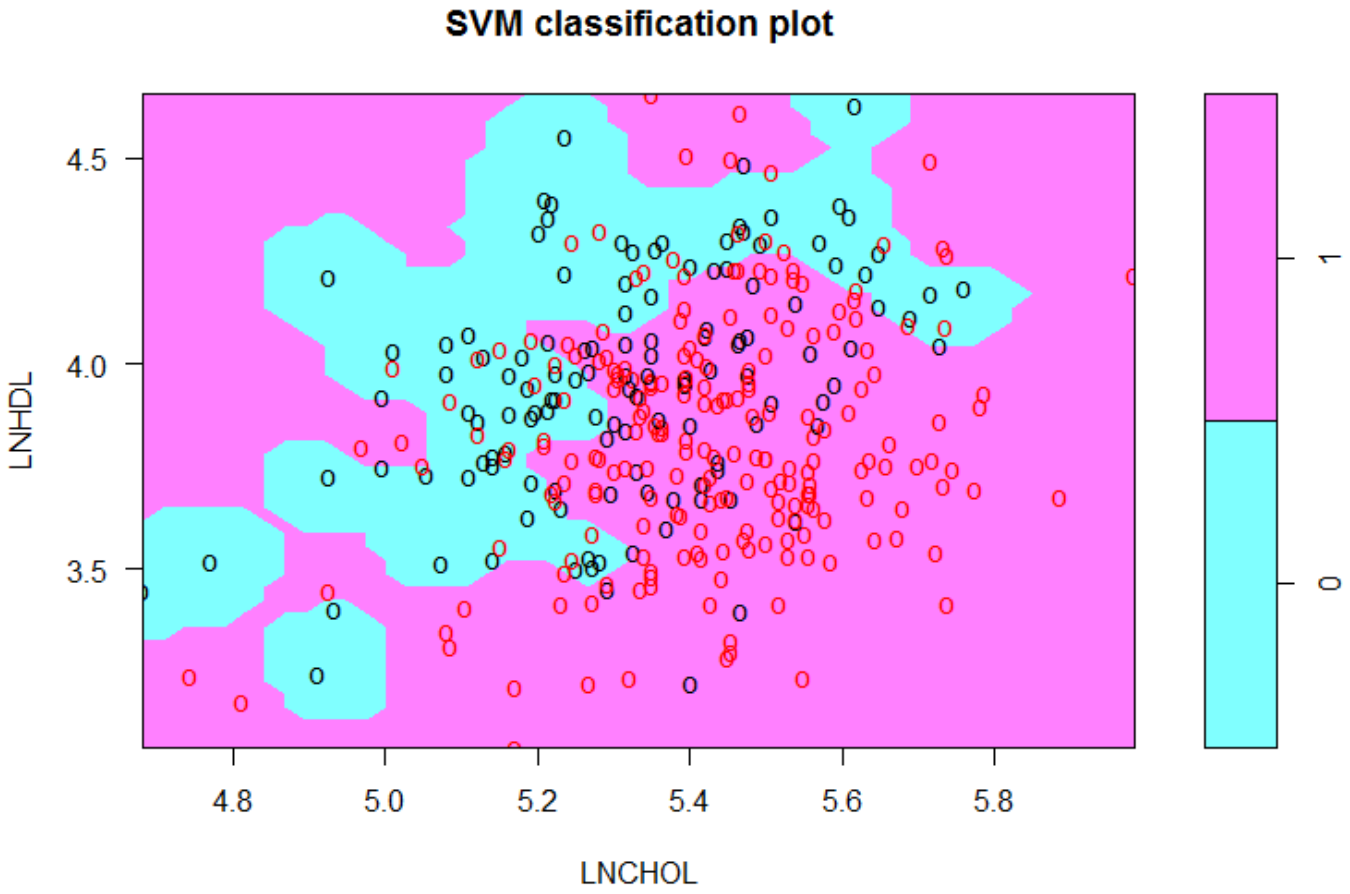


Hastie et al

1. Fix amount of allowed misclassification, called cost (C)
2. Split into two regions with non-linear boundary by maximizing the margin while staying within specified cost C

Support Vector Machines

SVM finds best, possibly non-linear boundaries between events and non-events:



Support Vector Machines

Pros:

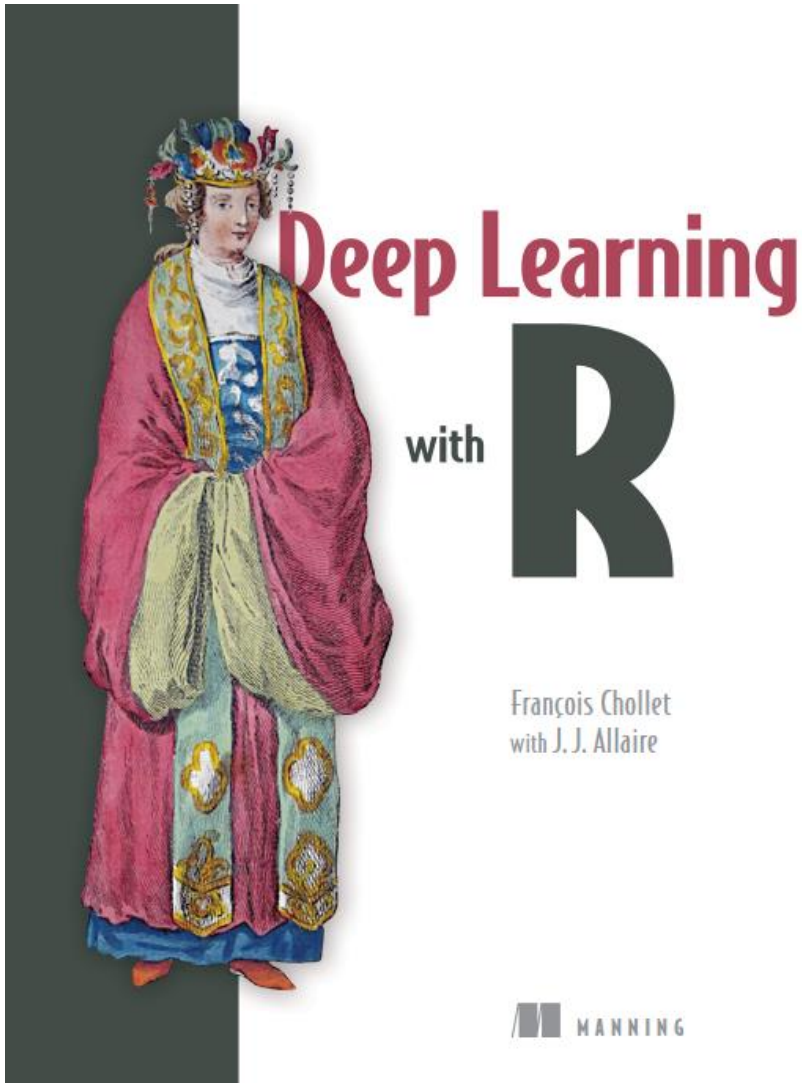
- Accommodates a wide range of boundaries and therefore can separate into two categories many different configuration of classes

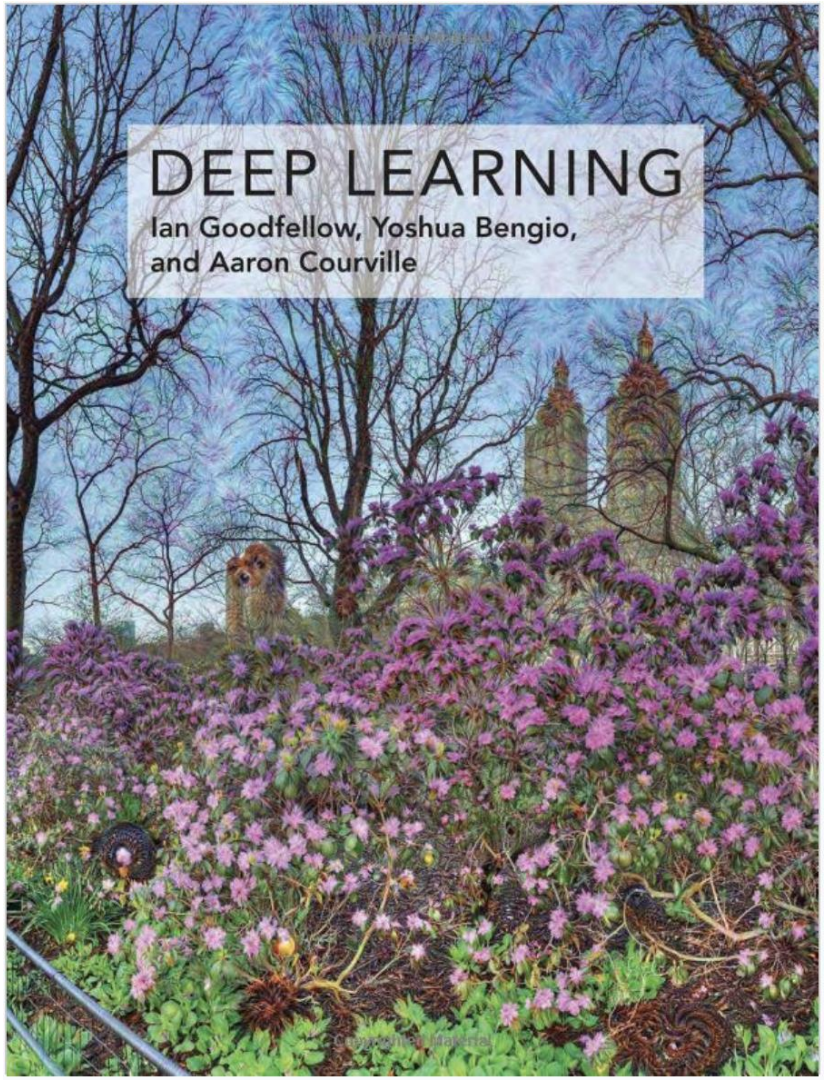
Cons:

- Cannot select variables (yet): if two predictors are important, have to build this knowledge into the model (curse of dimensionality)
- Can be computationally intensive

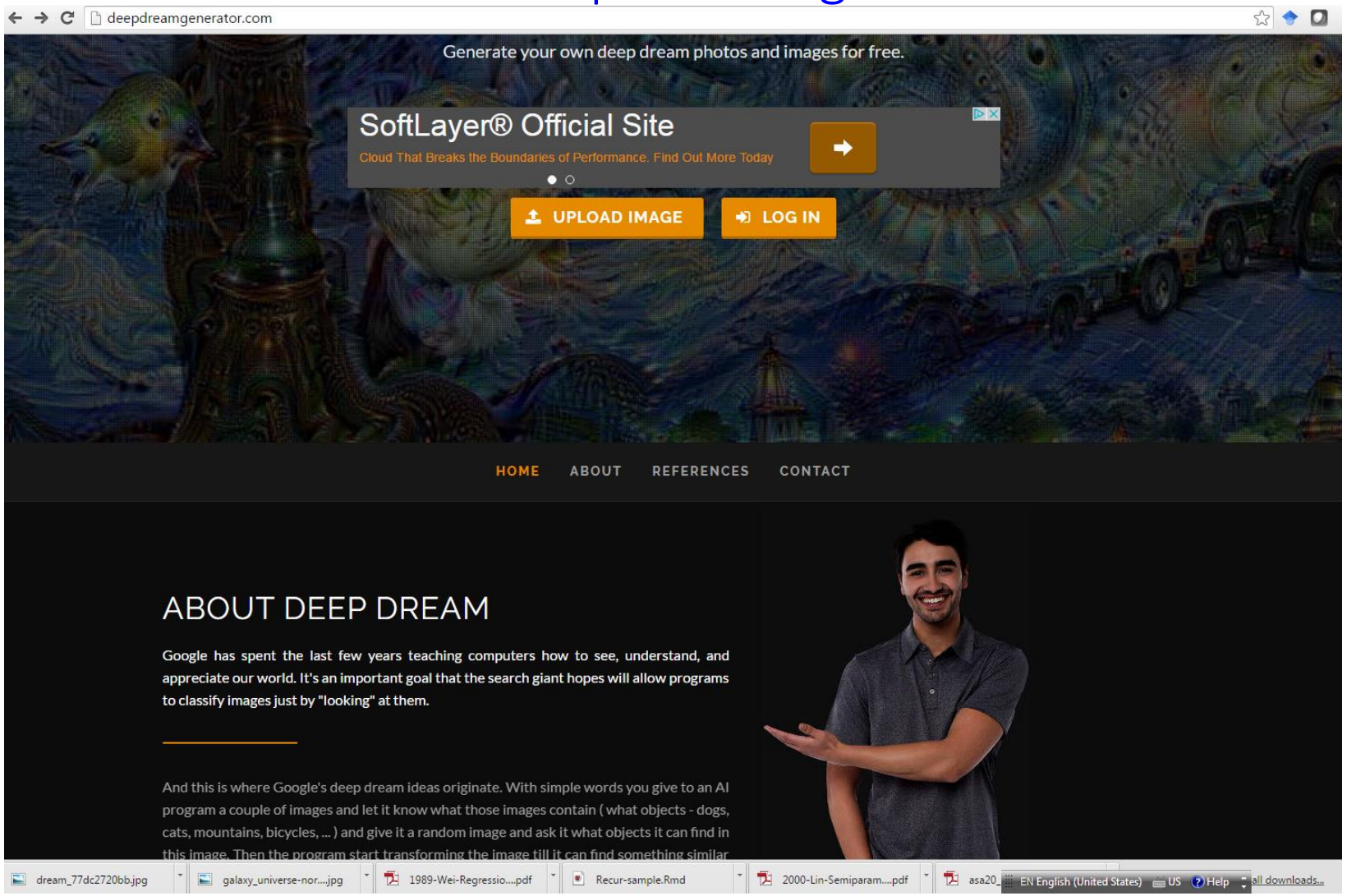
Deep Learning

Deep Learning



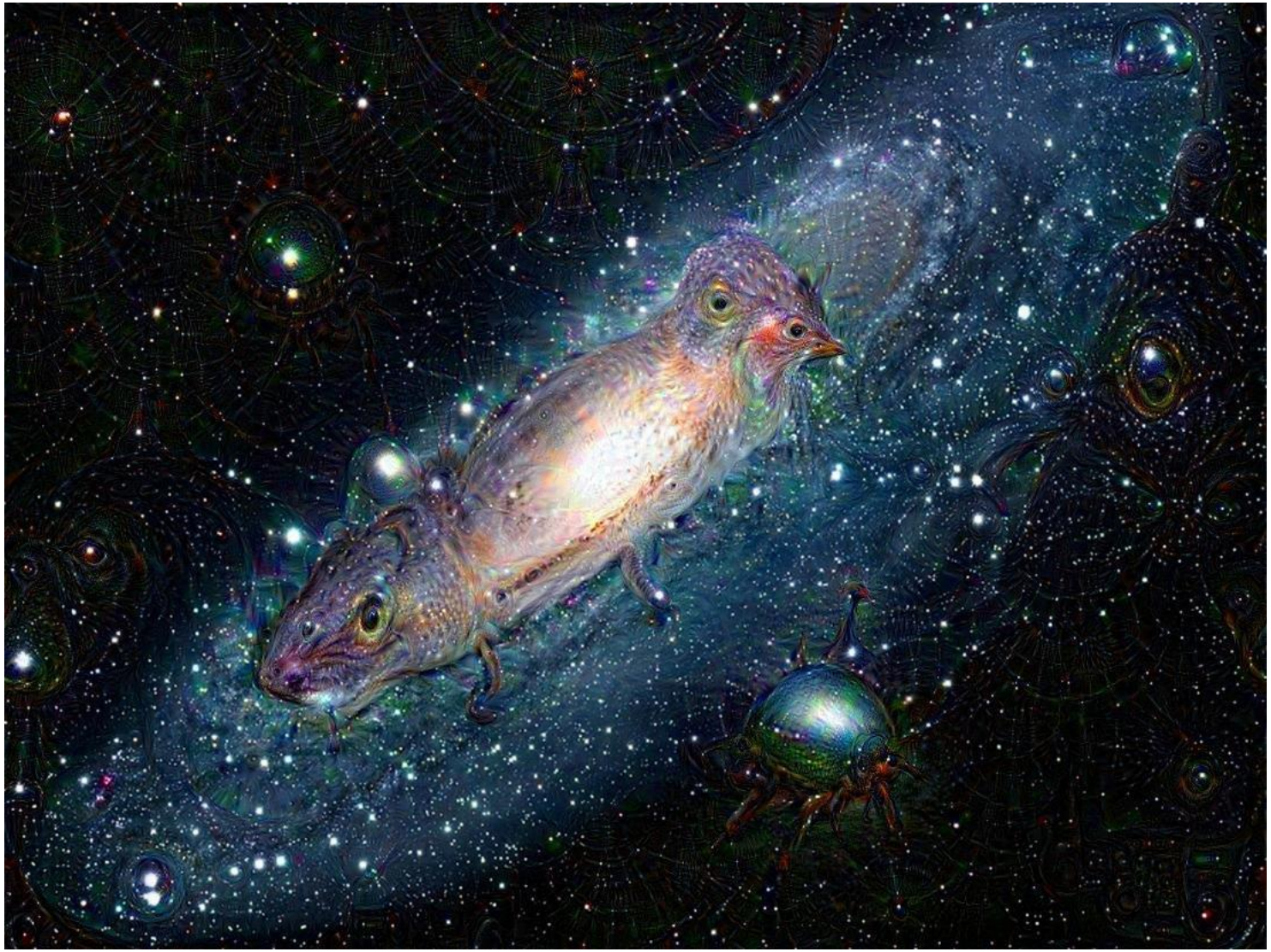


Neural Networks and Deep Learning



http://deepdreamgenerator.com/

http://deepdreamgenerator.com/



Gulshan, Varun, et al. "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs." *Jama* 316.22 (2016): 2402-2410.

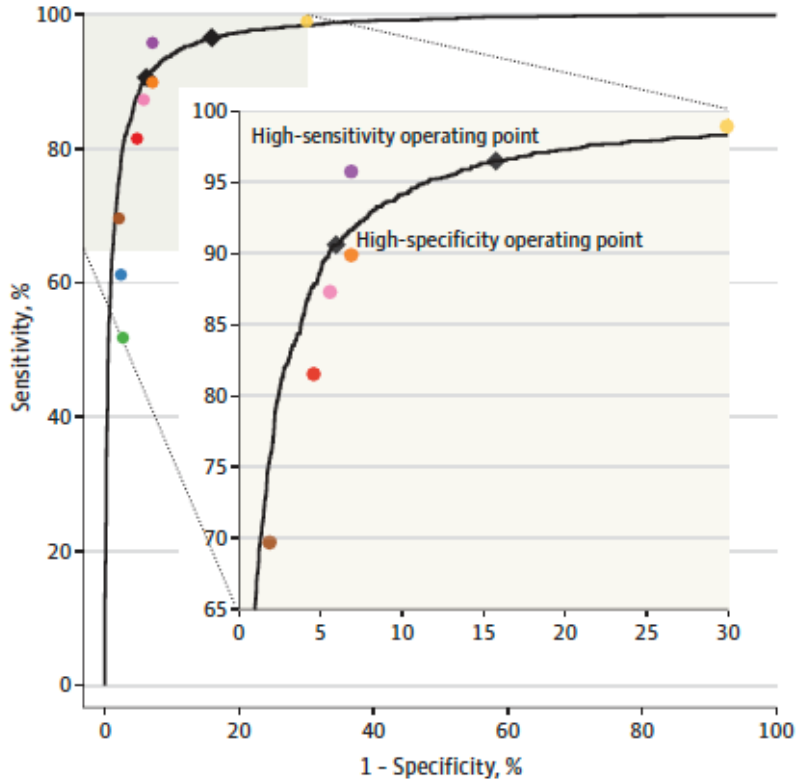
JAMA | Original Investigation | INNOVATIONS IN HEALTH CARE DELIVERY

Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs

Varun Gulshan, PhD; Lily Peng, MD, PhD; Marc Coram, PhD; Martin C. Stumpe, PhD; Derek Wu, BS; Arunachalam Narayanaswamy, PhD; Subhashini Venugopalan, MS; Kasumi Widner, MS; Tom Madams, MEng; Jorge Cuadros, OD, PhD; Ramasamy Kim, OD, DNB; Rajiv Raman, MS, DNB; Philip C. Nelson, BS; Jessica L. Mega, MD, MPH; Dale R. Webster, PhD

Performance of the Neural Network in the external dataset

Figure 3. Validation Set Performance for All-Cause Referable Diabetic Retinopathy in the EyePACS-1 Data Set (9946 Images)



Black curve – CNN Algorithm
Colored dots – ophtalmologists

AUC 97.4% (95% CI, 97.1%-97.8%).

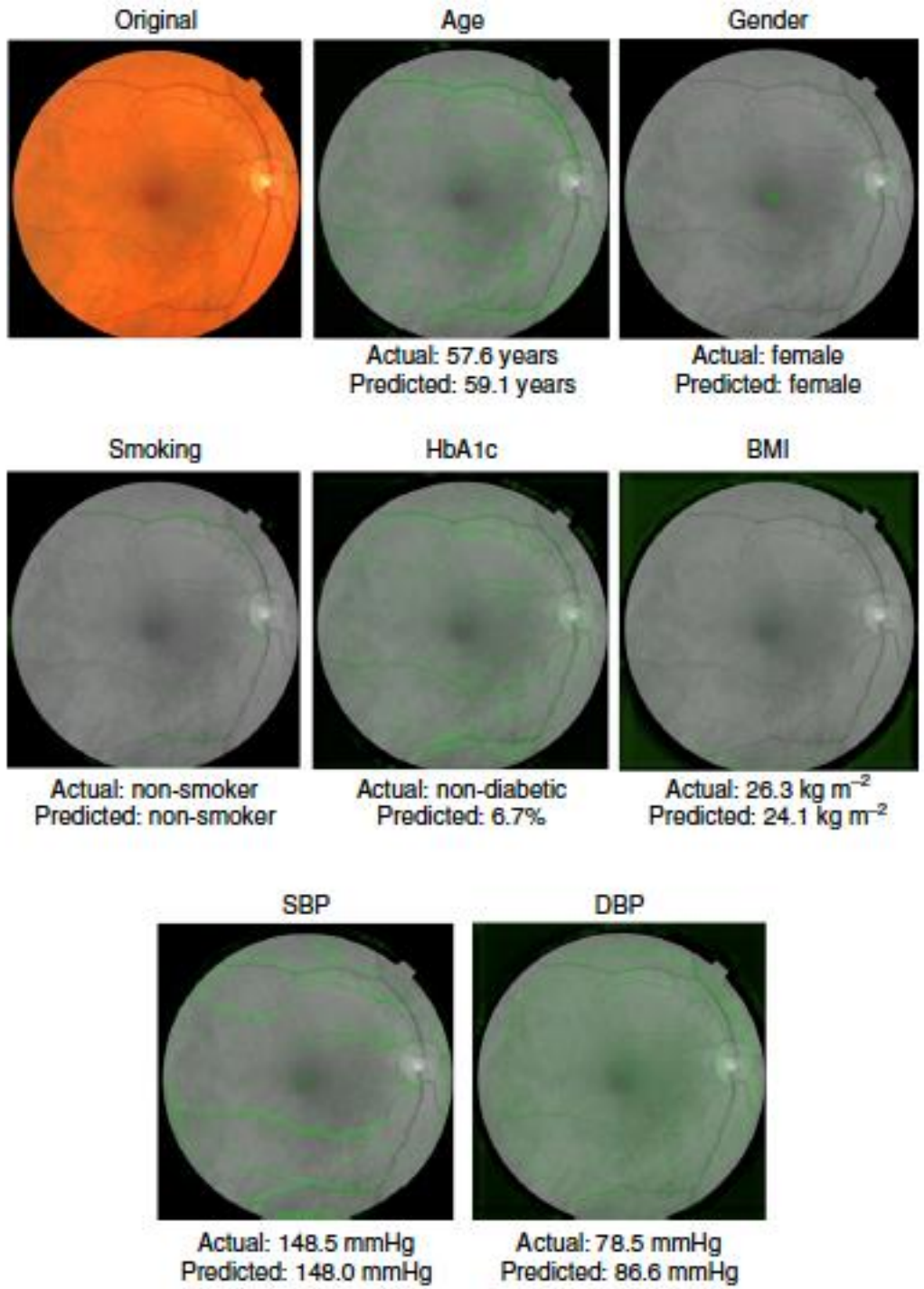
Poplin, Ryan, et al. "Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning." *Nature Biomedical Engineering* 2.3 (2018): 158.

ARTICLES
https://doi.org/10.1038/s41551-018-0195-0

nature biomedical engineering

Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning

Ryan Poplin[1,4], Avinash V. Varadarajan[1,4], Katy Blumer[1], Yun Liu[1], Michael V. McConnell[2,3], Greg S. Corrado[1], Lily Peng[1,4]* and Dale R. Webster[1,4]



Neural Network predicted:

Age – Mean abs error within 3.67 years

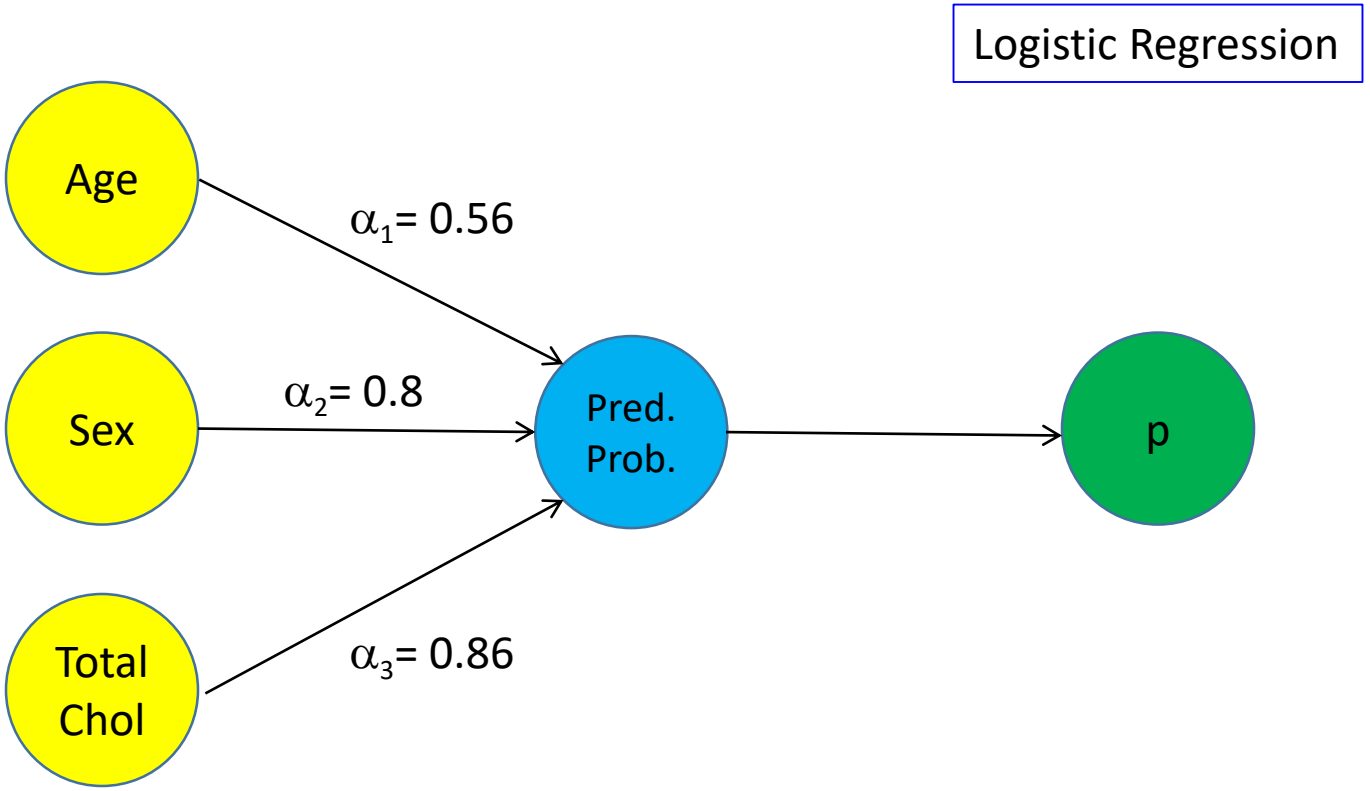
Sex – AUC=0.97

Smoking status – AUC=0.72

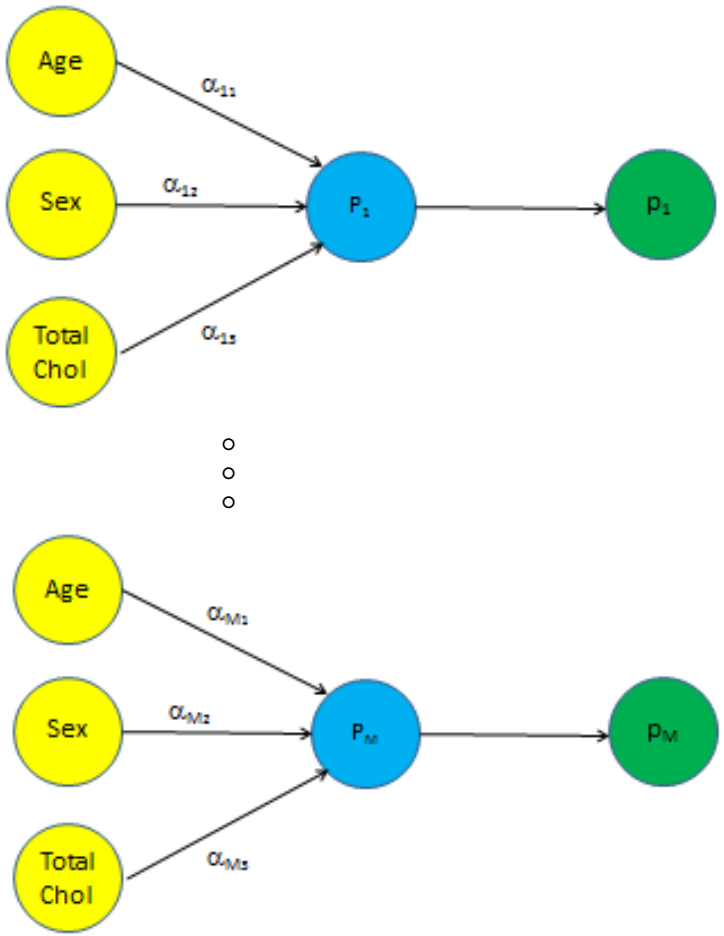
SBP – Mean absolute error of 11.2 mmHg

Poplin, Ryan, et al. "Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning." *Nature Biomedical Engineering* 2.3 (2018): 158.

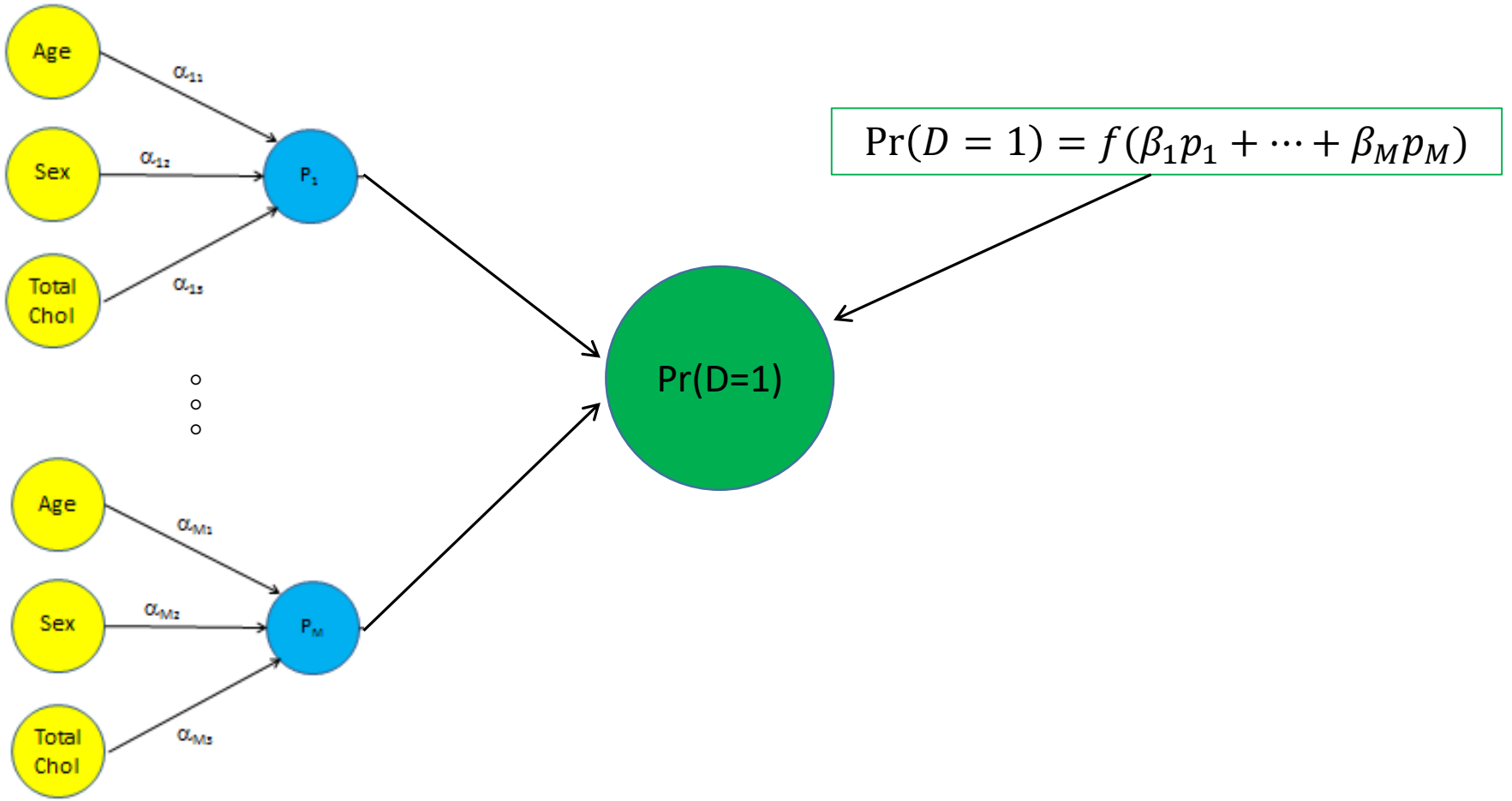
Logistic Regression



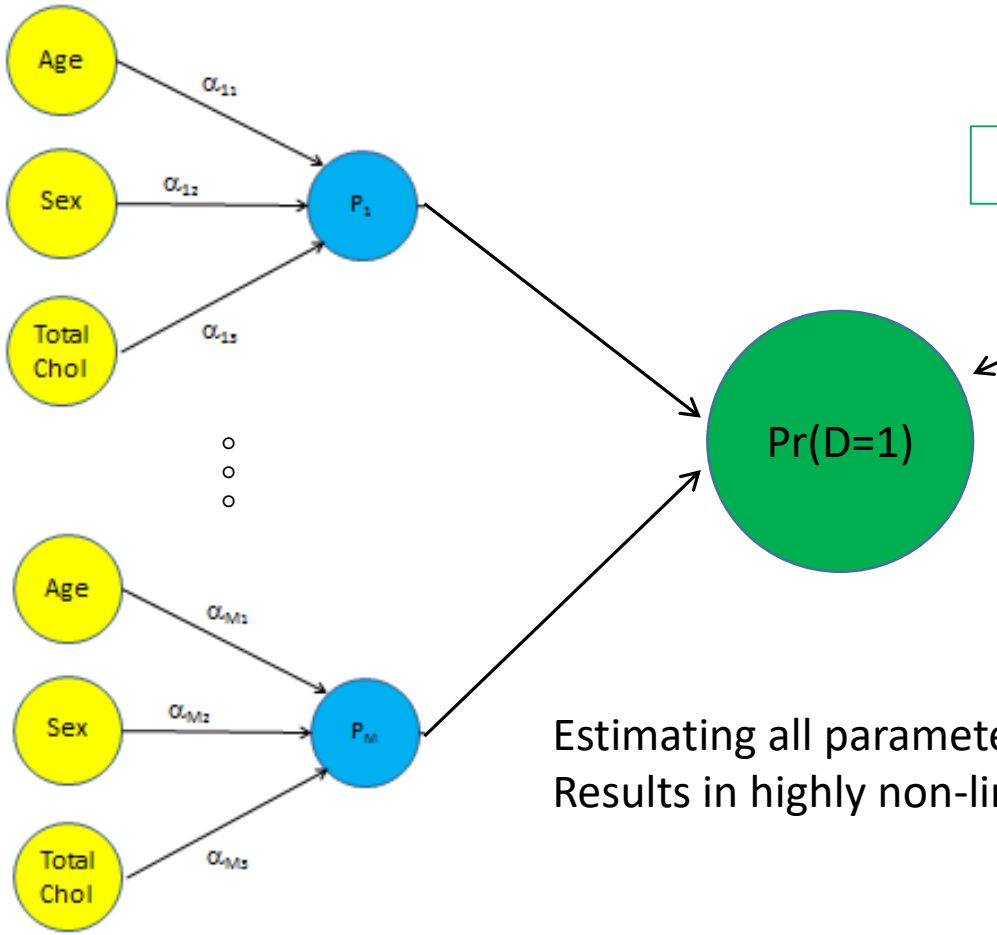
Neural Networks



Neural Networks



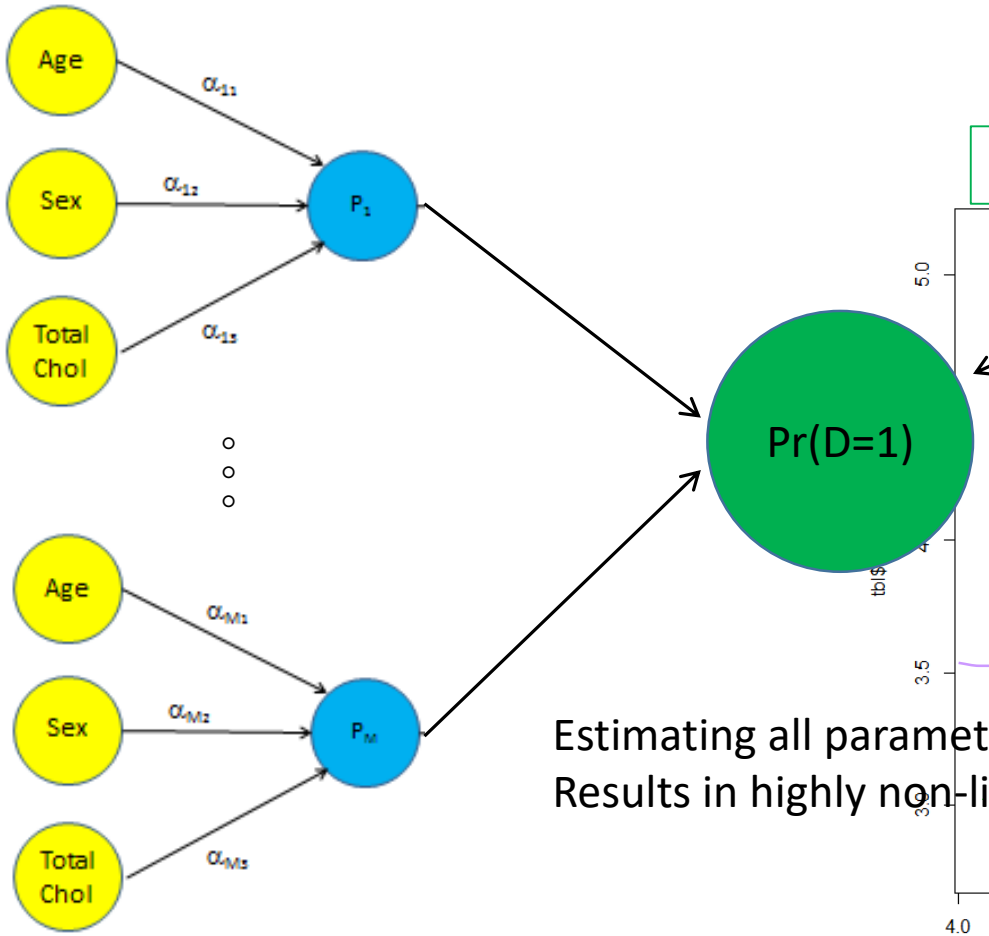
Neural Networks



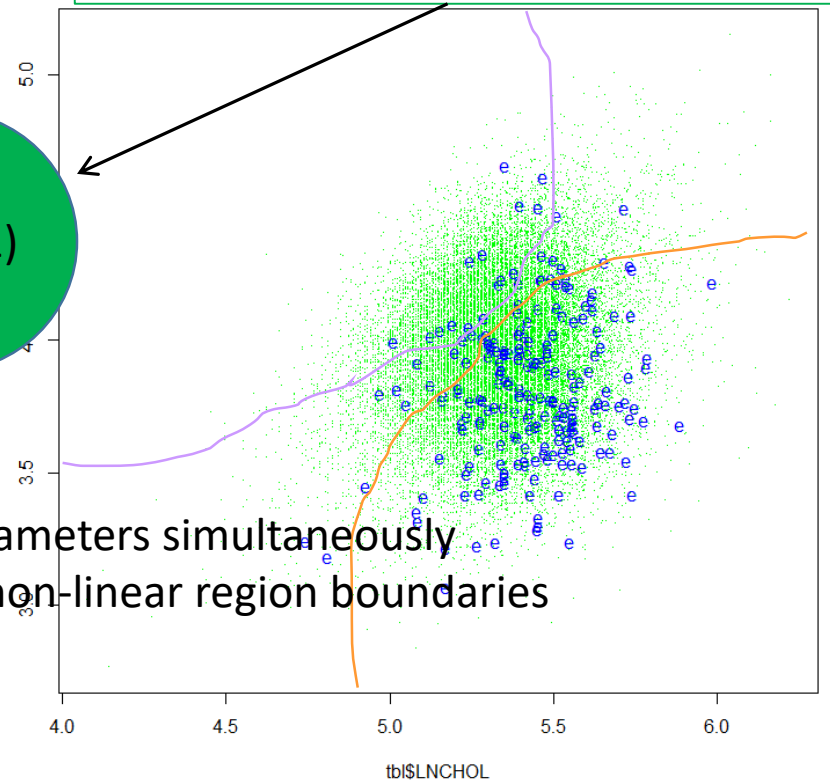
$$\Pr(D = 1) = f(\beta_1 p_1 + \cdots + \beta_M p_M)$$

Estimating all parameters simultaneously
Results in highly non-linear region boundaries

Neural Networks

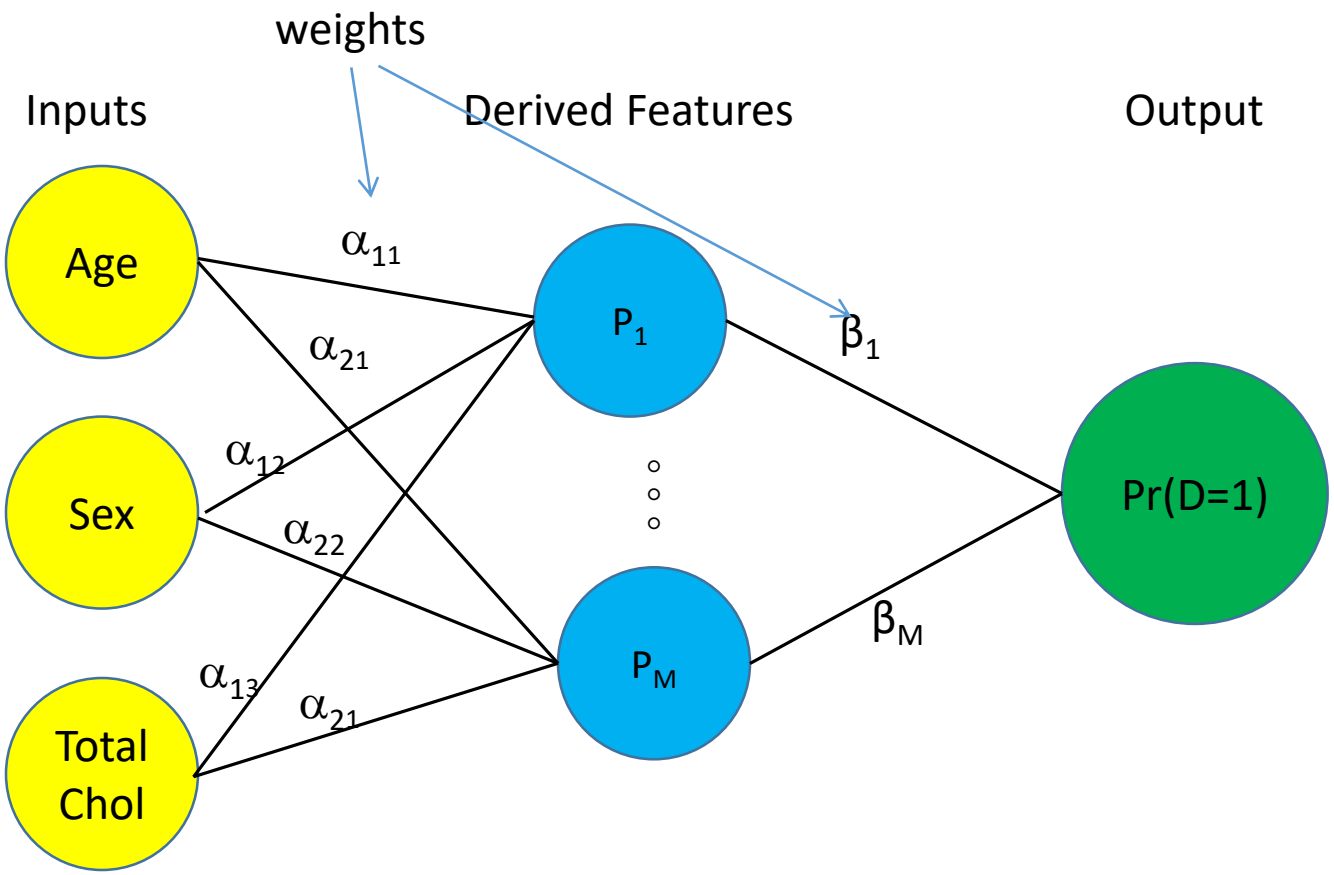


$$\Pr(D = 1) = f(\beta_1 p_1 + \cdots + \beta_M p_M)$$

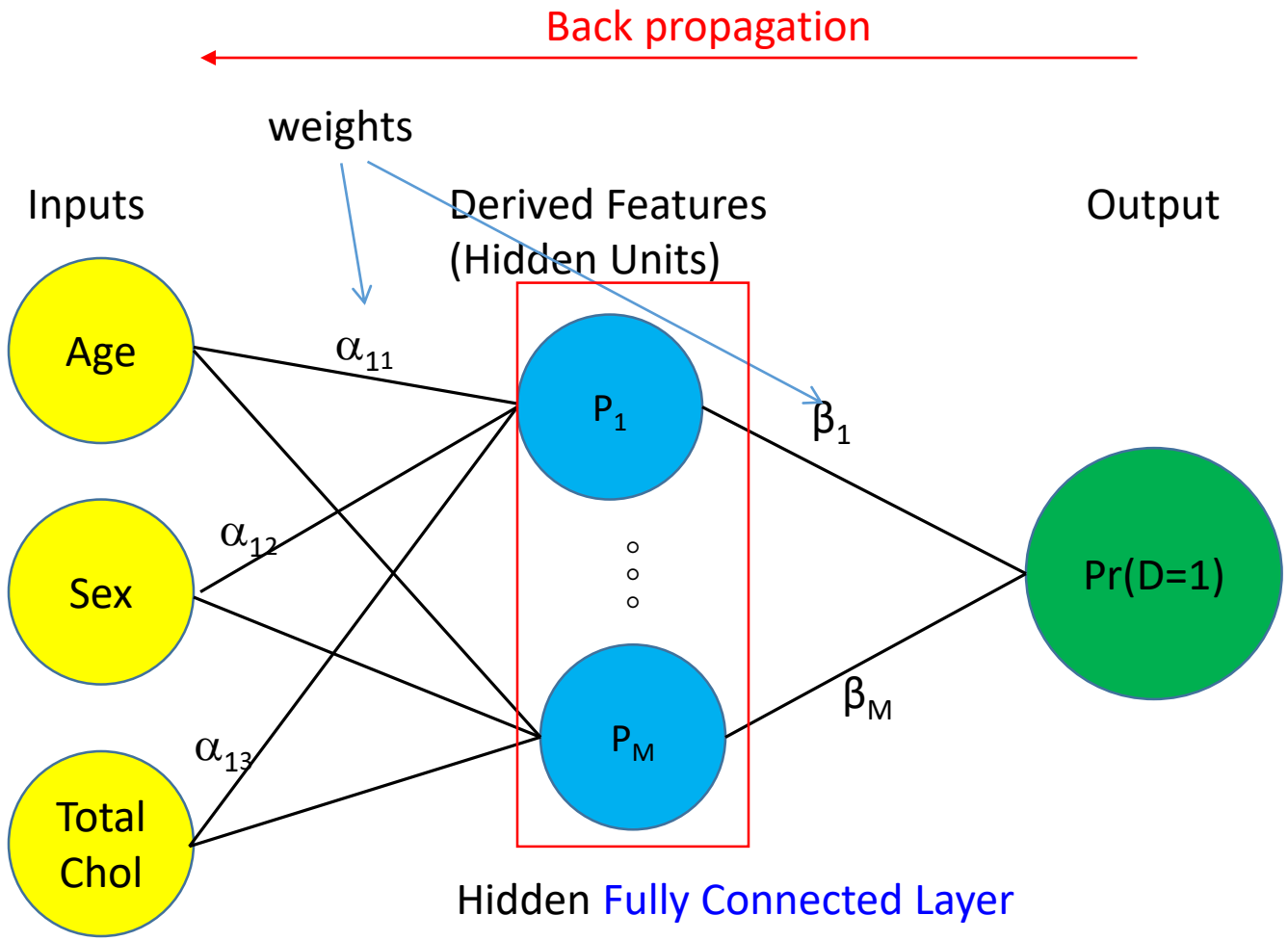


Estimating all parameters simultaneously
Results in highly non-linear region boundaries

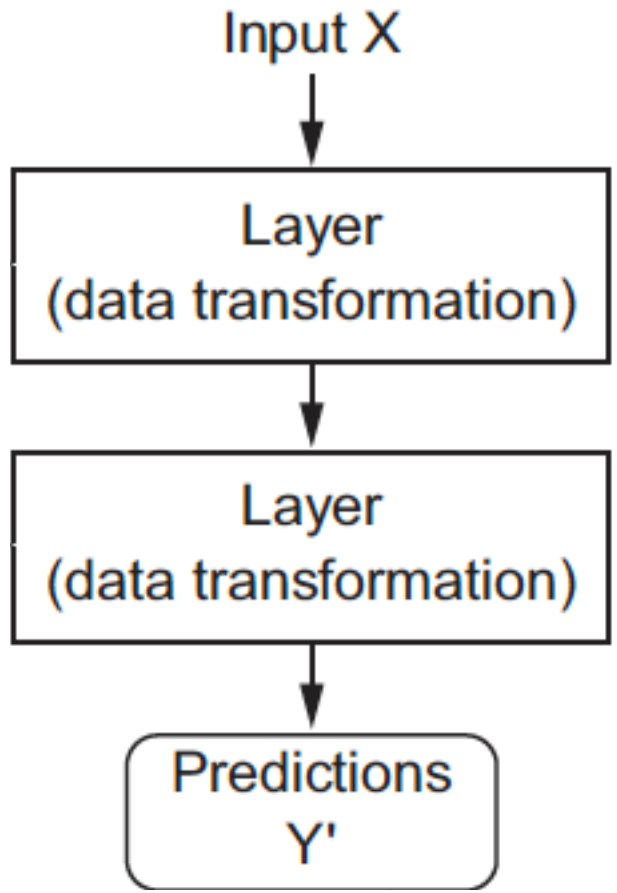
Neural Networks

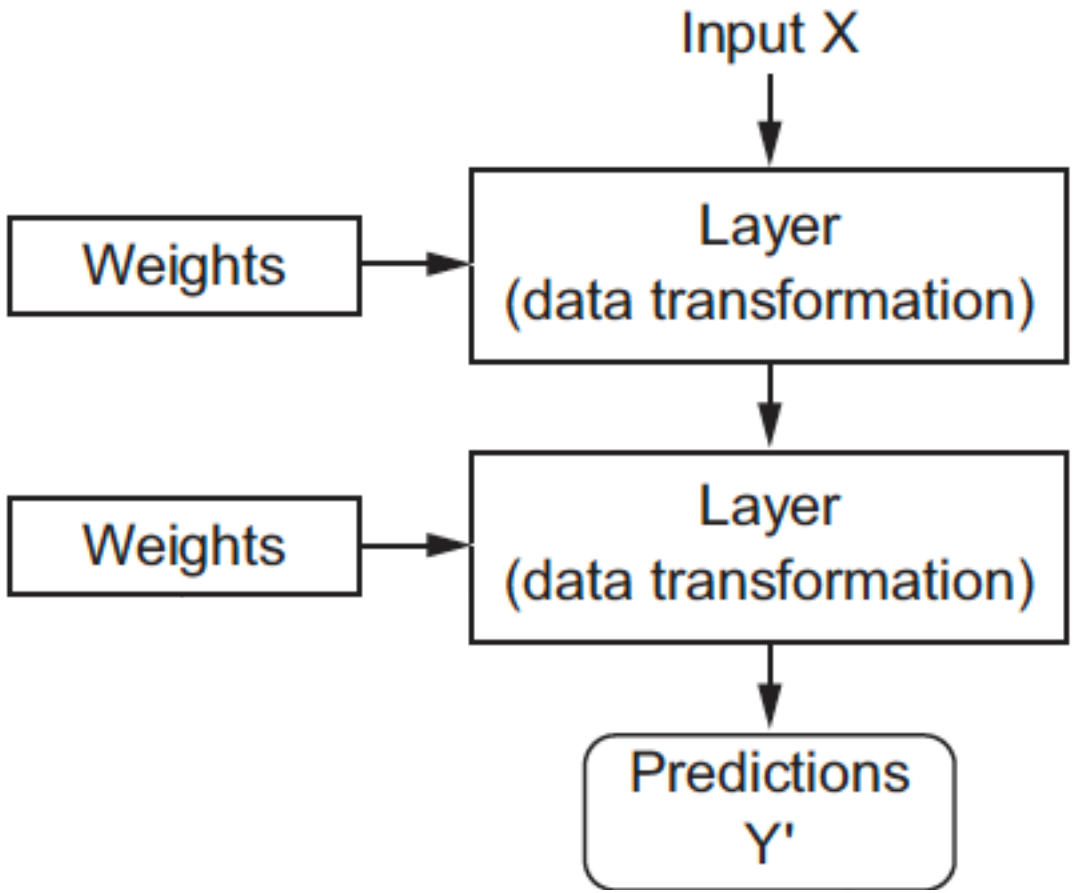


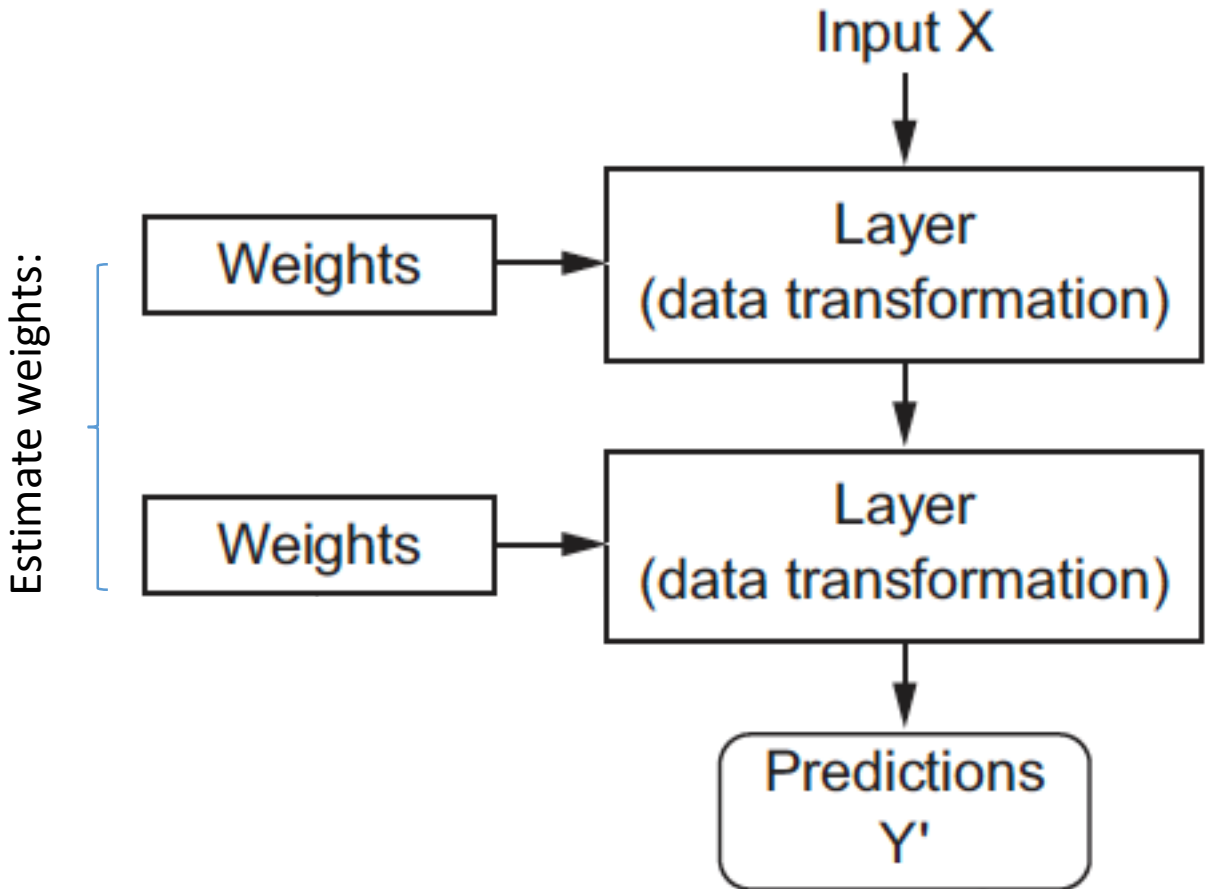
Short review of predictive models: Neural Networks

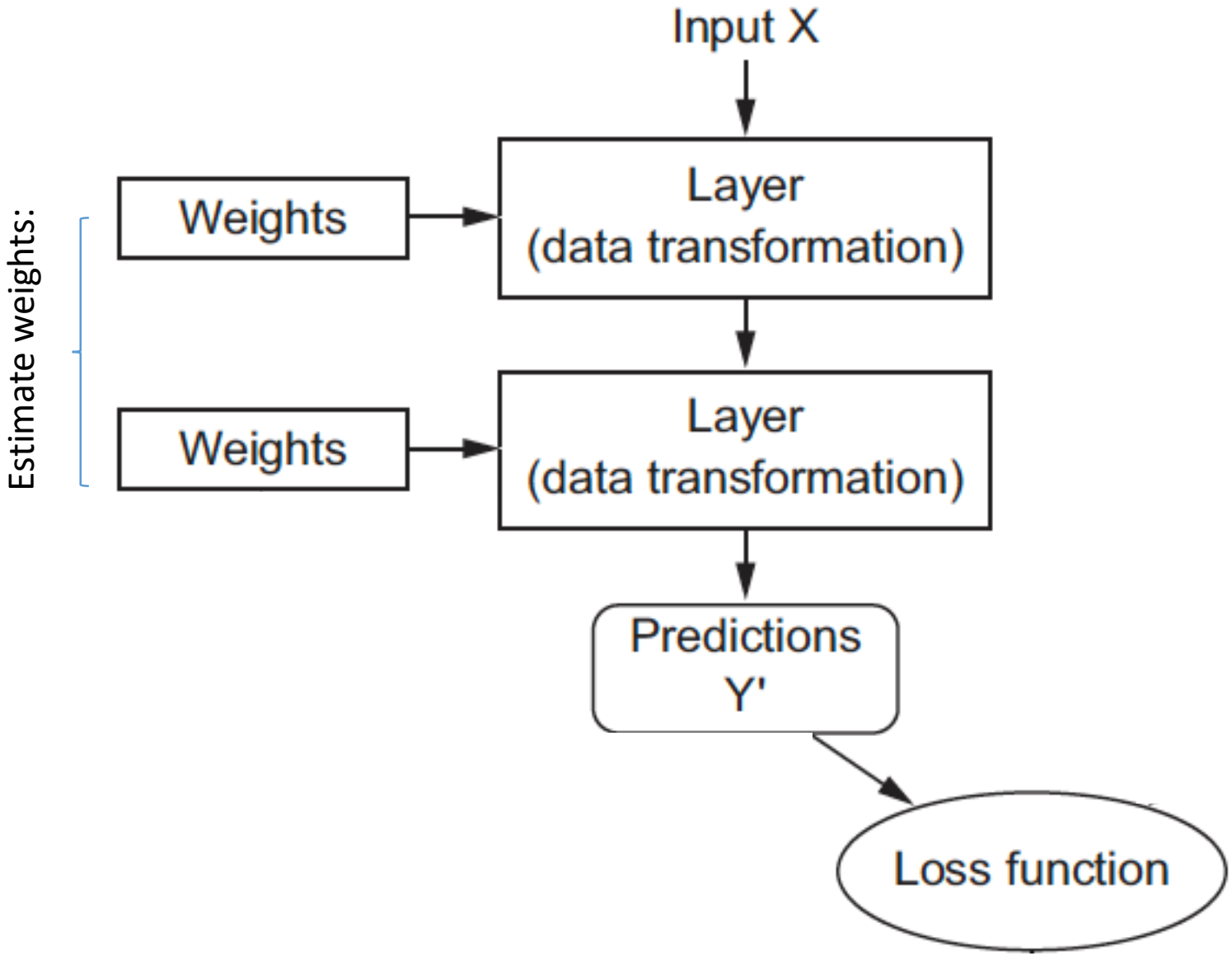


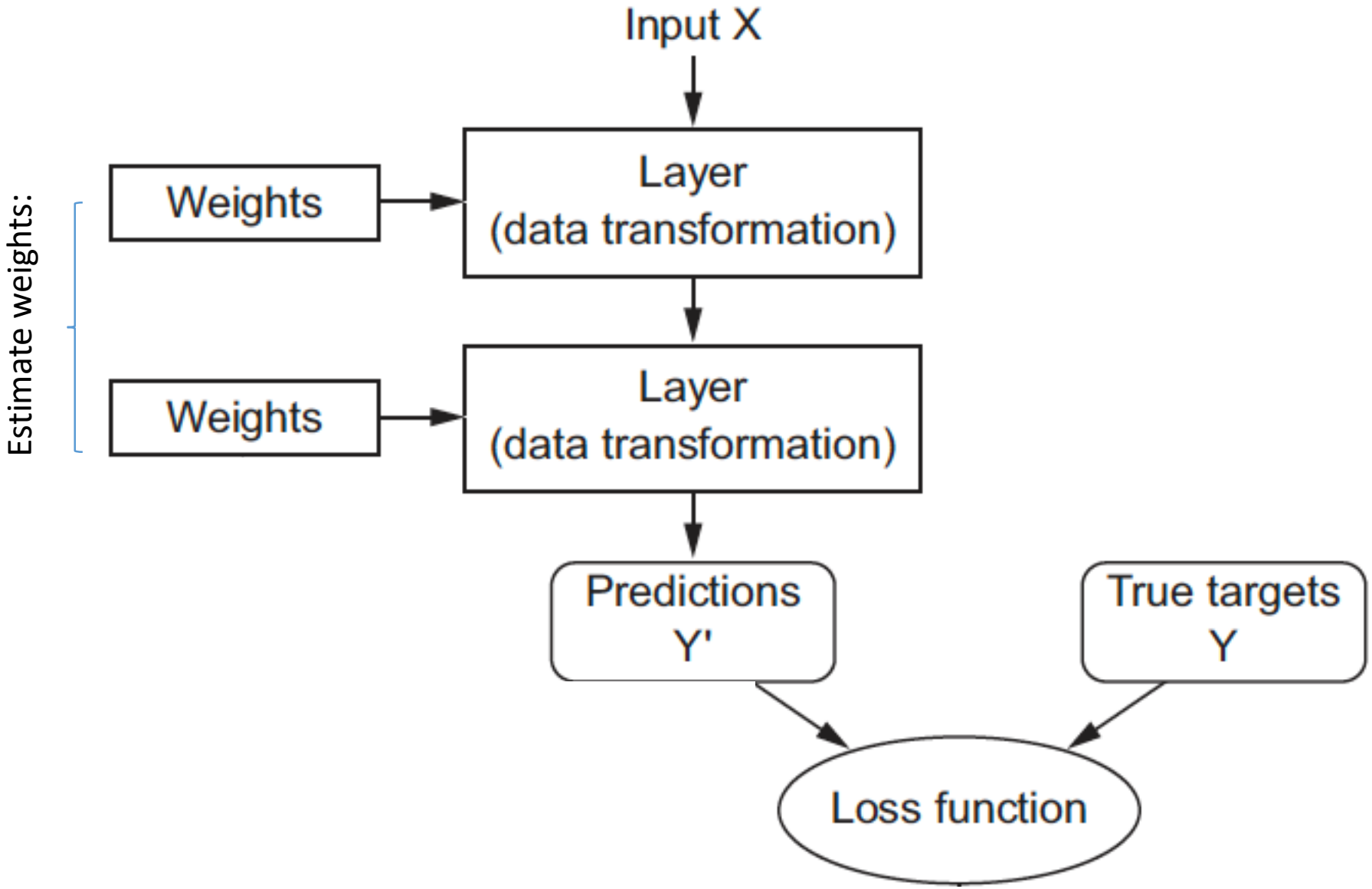
Might have more than one hidden layer

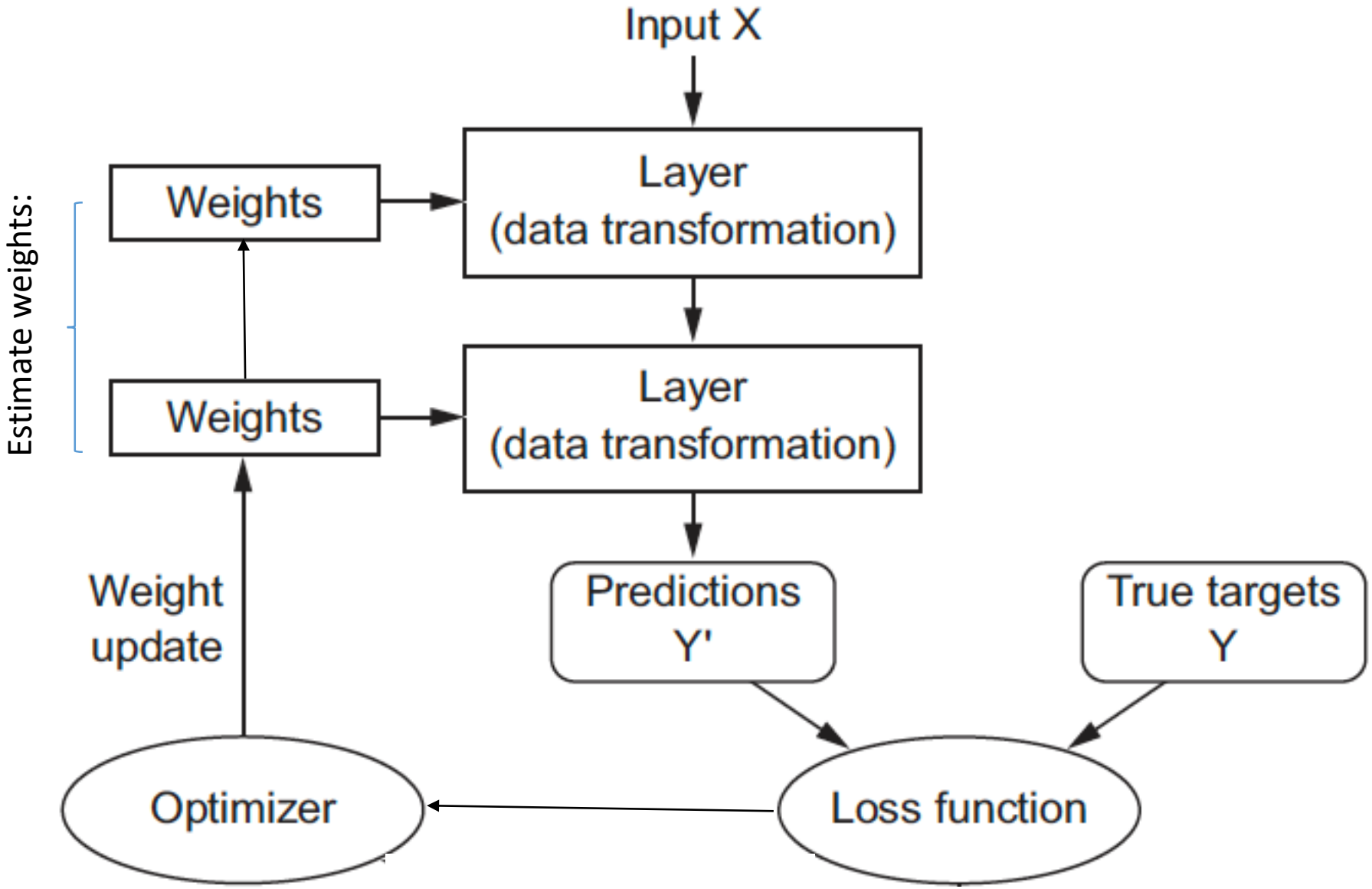

Input X
Layer
(data transformation)
Layer
(data transformation)
Predictions
Y'

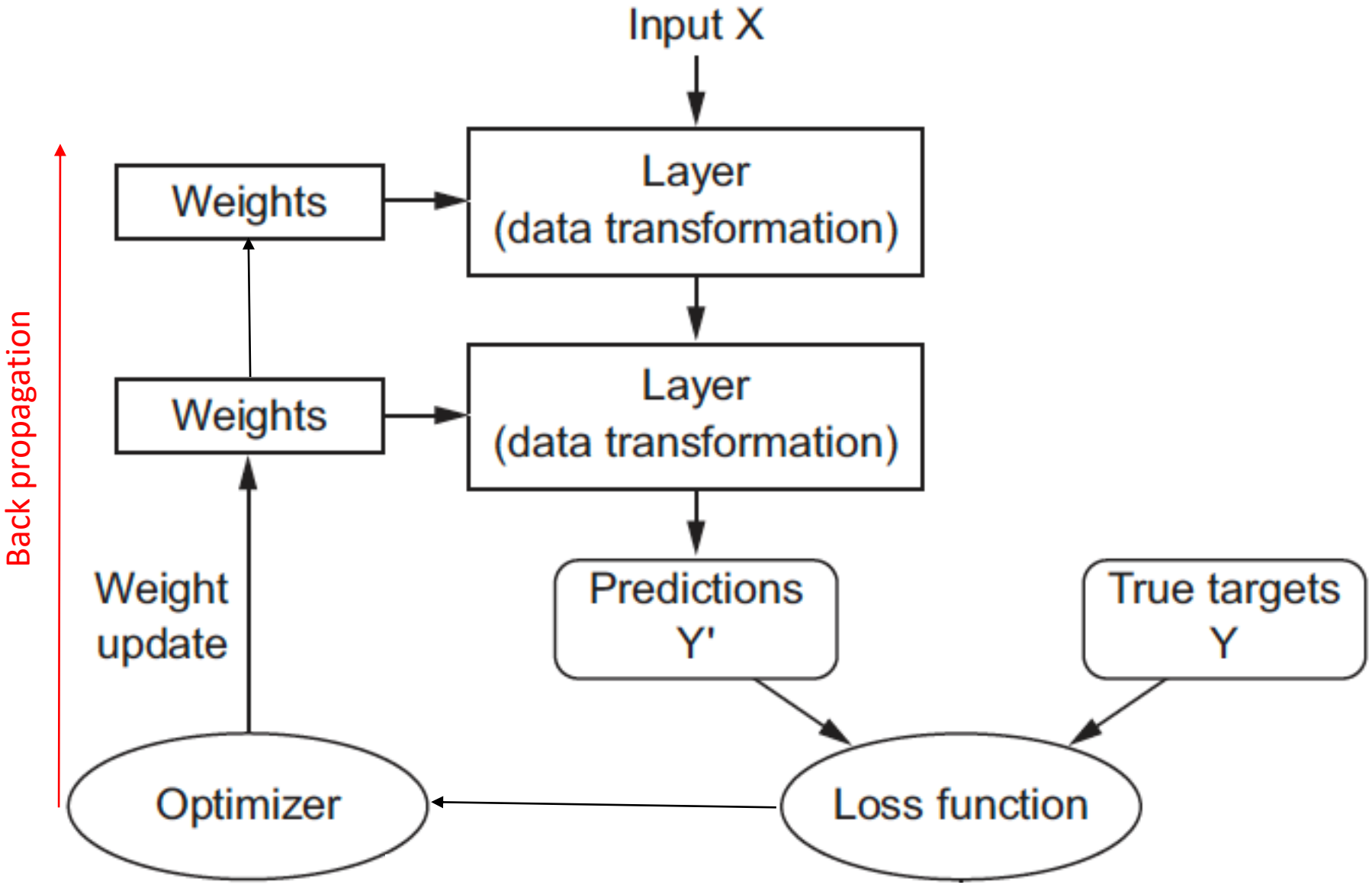

Input X
Weights
Layer
(data transformation)
Weights
Layer
(data transformation)
Predictions
Y'

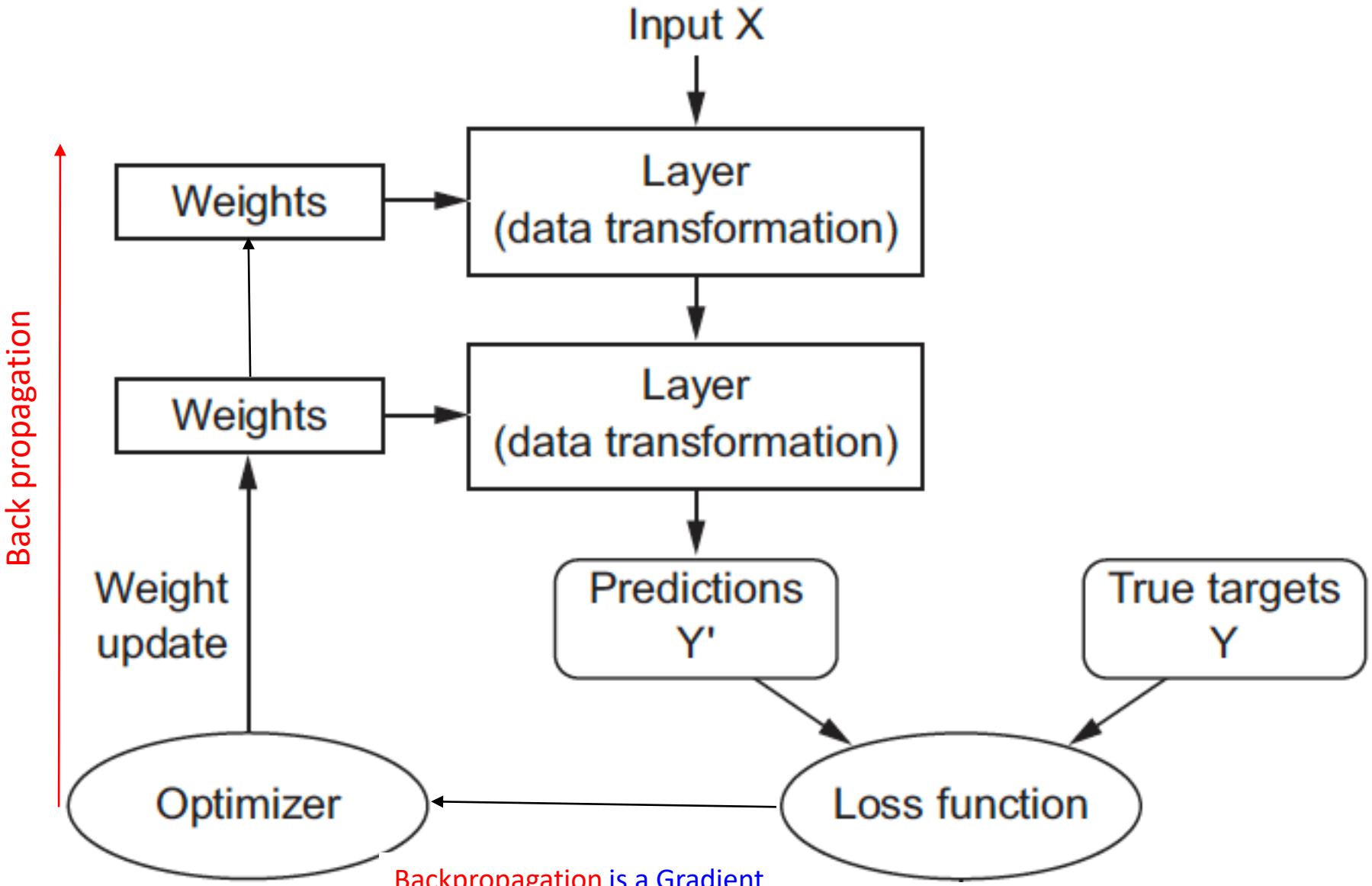

Input X
Weights
Layer
(data transformation)
Weights
Layer
(data transformation)
Predictions
Y'
Estimate weights:


Input X
Weights
Layer
(data transformation)
Weights
Layer
(data transformation)
Predictions
Y'
Loss function
Estimate weights:


Input X
Weights
Layer
(data transformation)
Weights
Layer
(data transformation)
Predictions
Y'
True targets
Y
Loss function
Estimate weights:

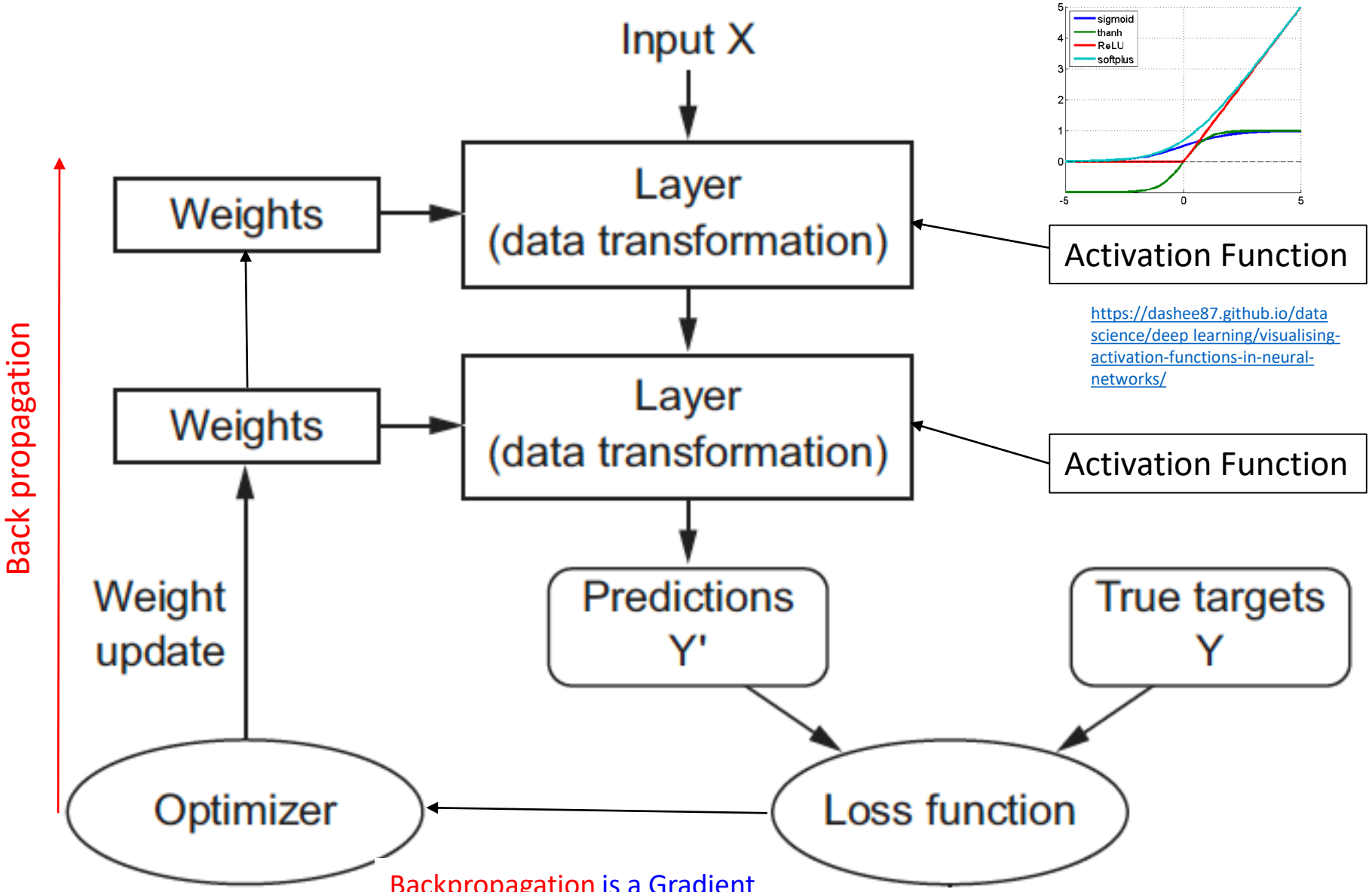

Input X
Weights
Layer
(data transformation)
Estimate weights:
Weights
Layer
(data transformation)
Weight
update
Predictions
Y'
True targets
Y
Optimizer
Loss function


Input X
Weights
Layer
(data transformation)
Weights
Layer
(data transformation)
Predictions
Y'
True targets
Y
Loss function
Optimizer
Weight
update
Back propagation



Backpropagation is a Gradient Descent Optimization by Stochastic Gradient Descent or RMSprop or other methods

http://ruder.io/optimizing-gradient-descent/

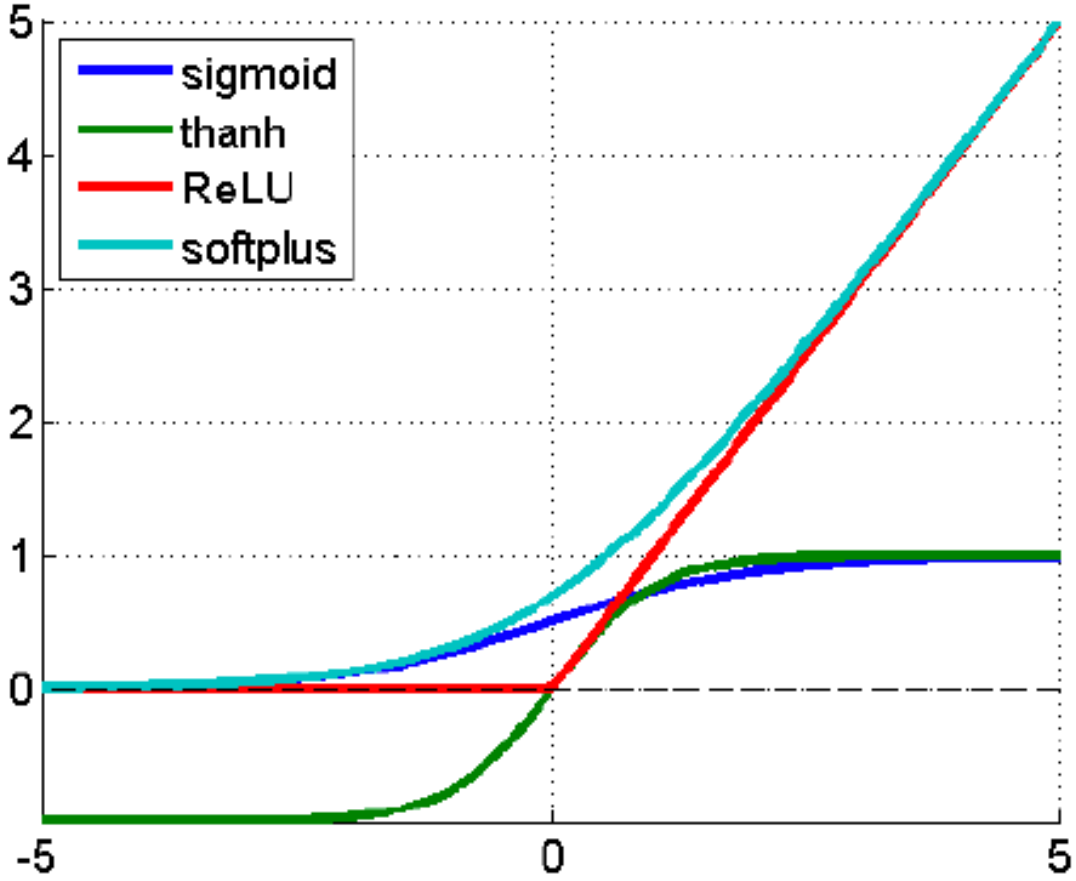


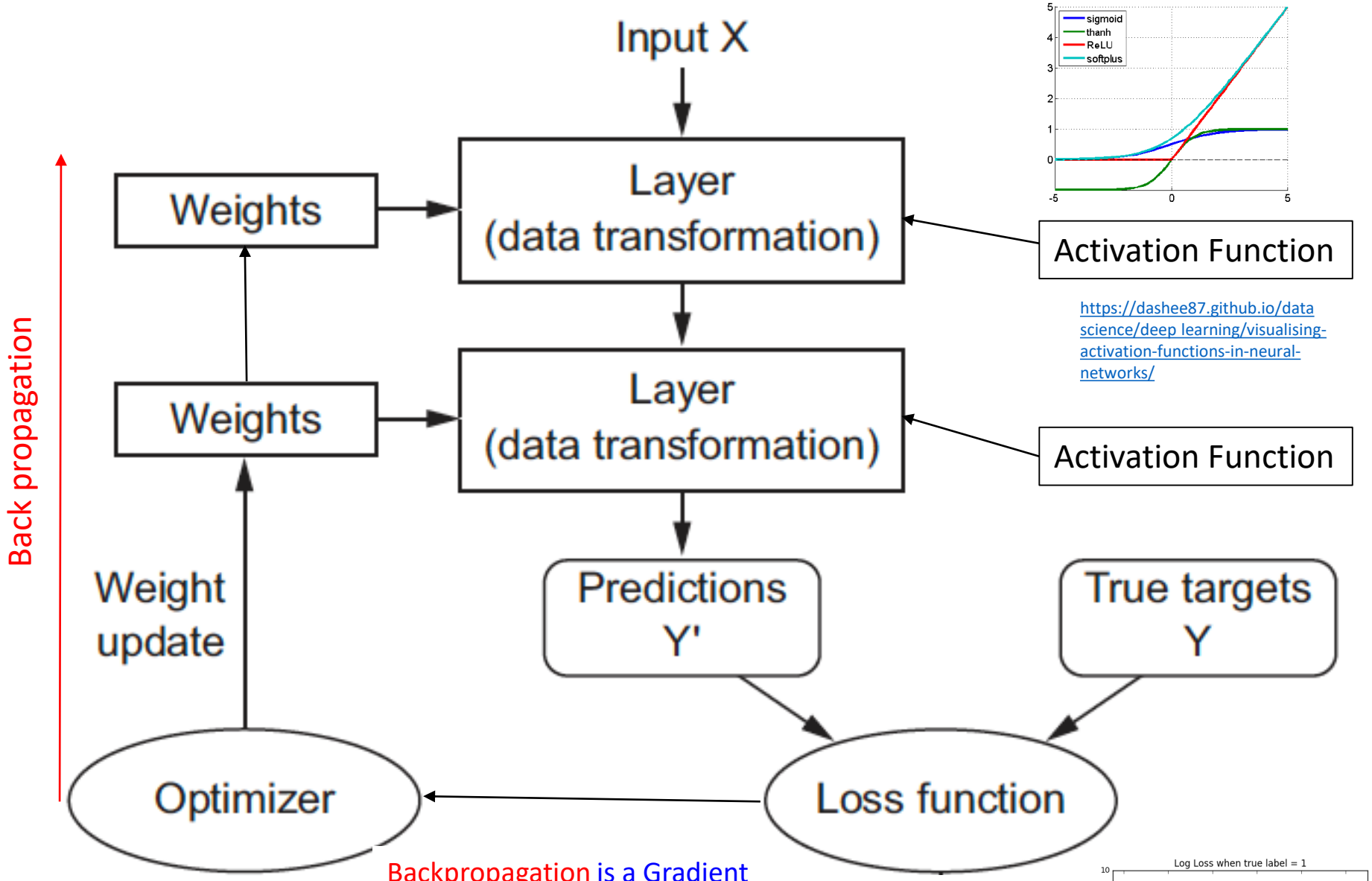
https://dashee87.github.io/data science/deep learning/visualising-activation-functions-in-neural-networks/

Backpropagation is a Gradient Descent Optimization by Stochastic Gradient Descent or RMSprop or other methods

http://ruder.io/optimizing-gradient-descent/

Activation Function – adding non-linearity to the model

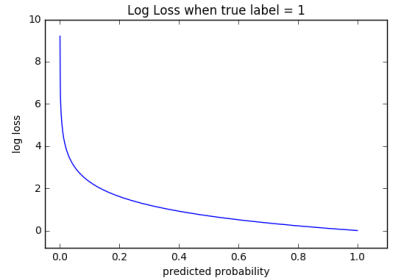




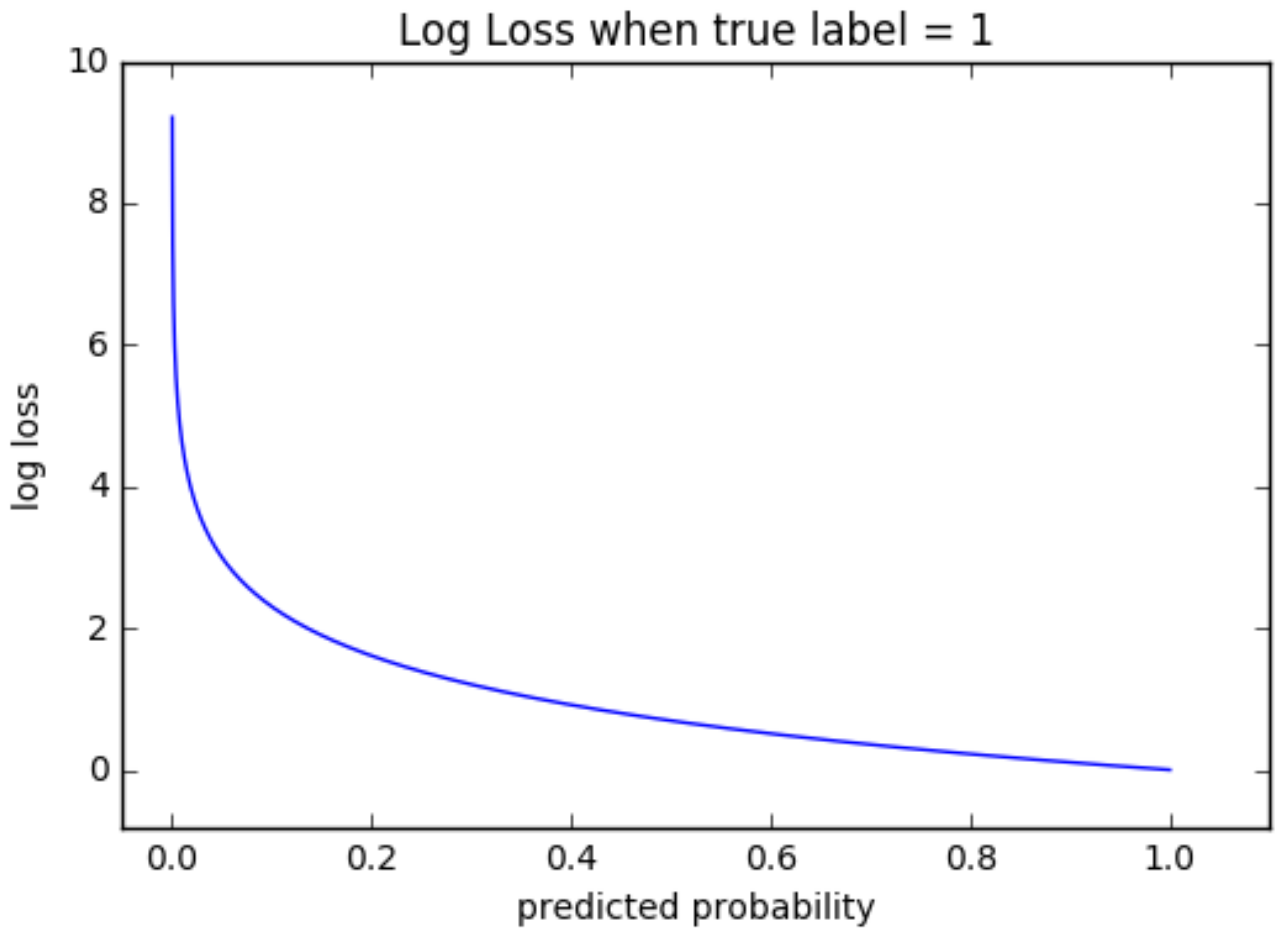
https://dashee87.github.io/data science/deep learning/visualising-activation-functions-in-neural-networks/

Backpropagation is a Gradient Descent Optimization by Stochastic Gradient Descent or RMSprop or other methods

http://ruder.io/optimizing-gradient-descent/



Cross Entropy Loss



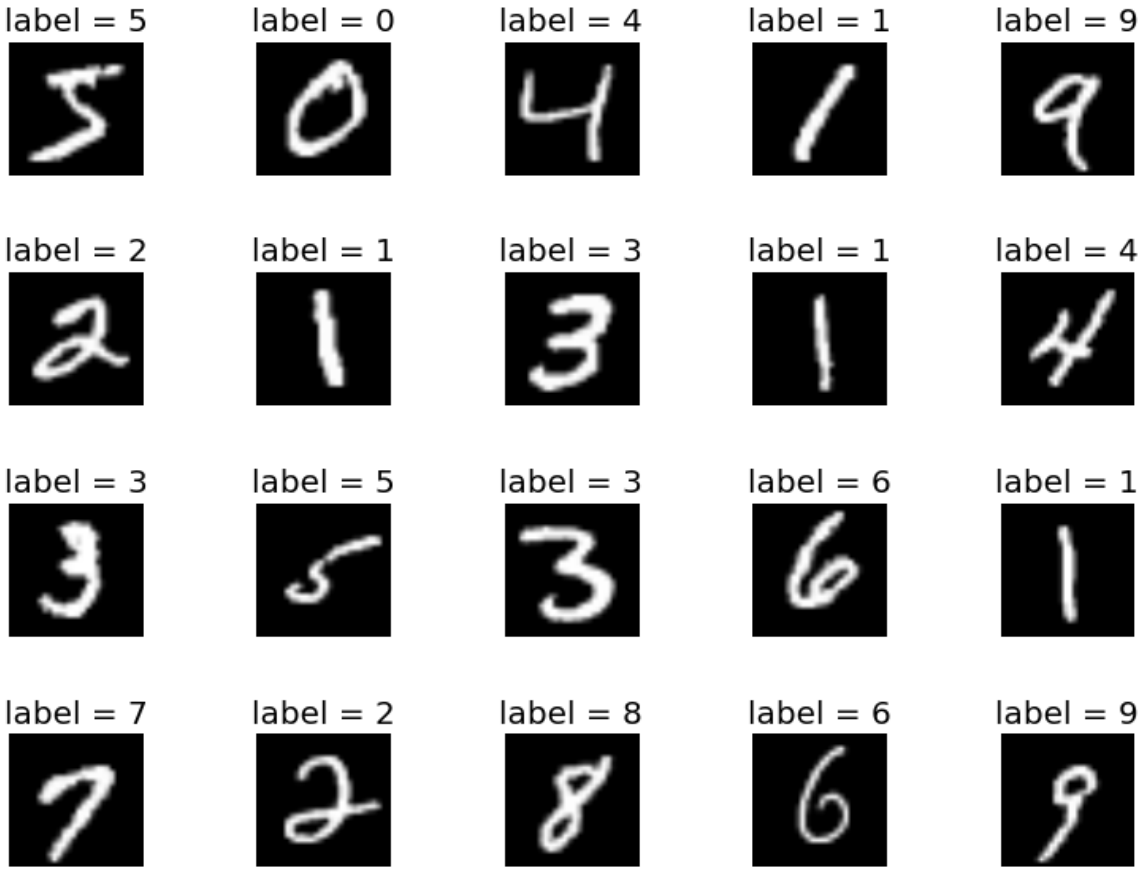
Neural Networks

How the weights are estimated?

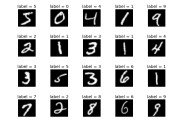
- Start with some set of weights as initial starting values
- Calculate predicted probabilities for one observation in your data and identify incorrect classifications.
- Do "Back Propagation": change more those weights which are responsible for incorrect classification of a given observation
- Repeat for other observations until overall error is minimized
- One pass through data is called an "epoch"
- Duration of the training is measured in epochs
- Also use weight decay: allow algorithm dynamically adjust the complexity of the network by setting some weights to zero.

Backpropagation is a Gradient Descent Optimization by Stochastic Gradient Descent or RMSprop or other methods

MNIST dataset used for image recognition



The MNIST database contains 60,000 training images and 10,000 testing images
MNIST is Modified National Institute of Standards and Technology database



For 1st batch

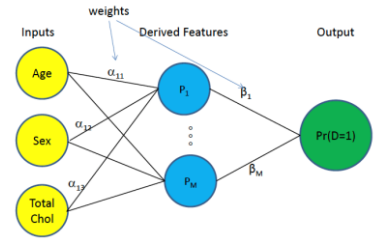
use random weights

Batch 1…1000

Training Instance 1…6

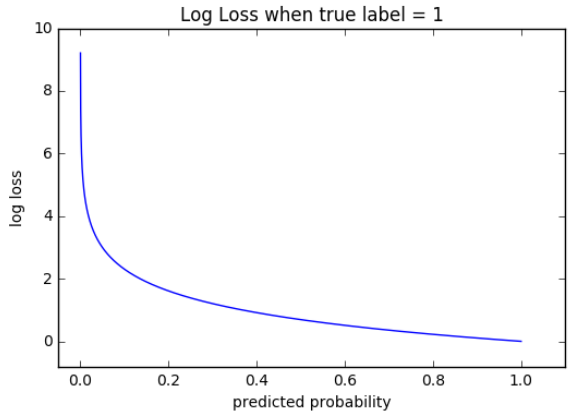
label = 5

Run through the Network, Predict the Label, Calculate the Loss

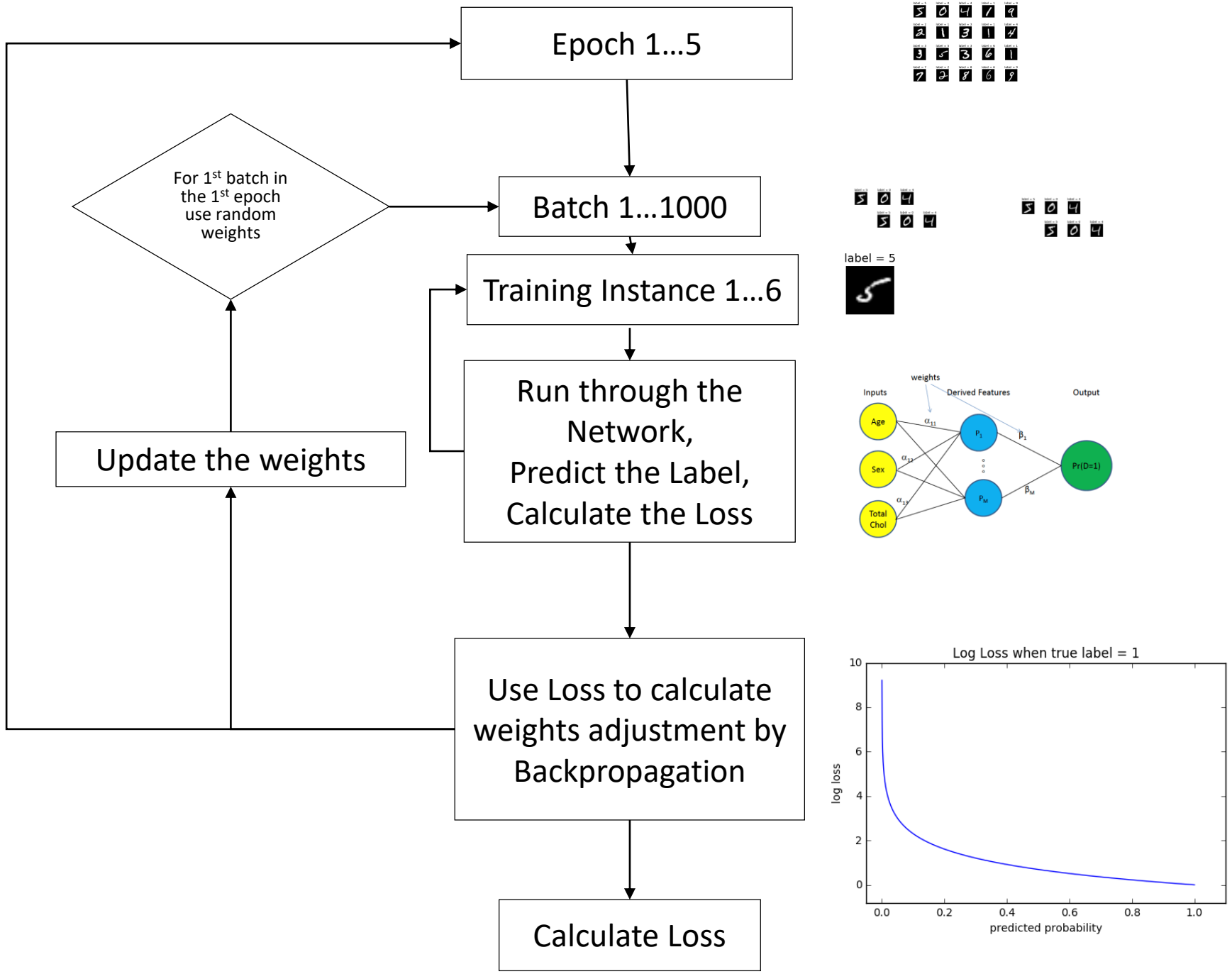


Update the weights

Use Loss to calculate weights adjustment by Backpropagation

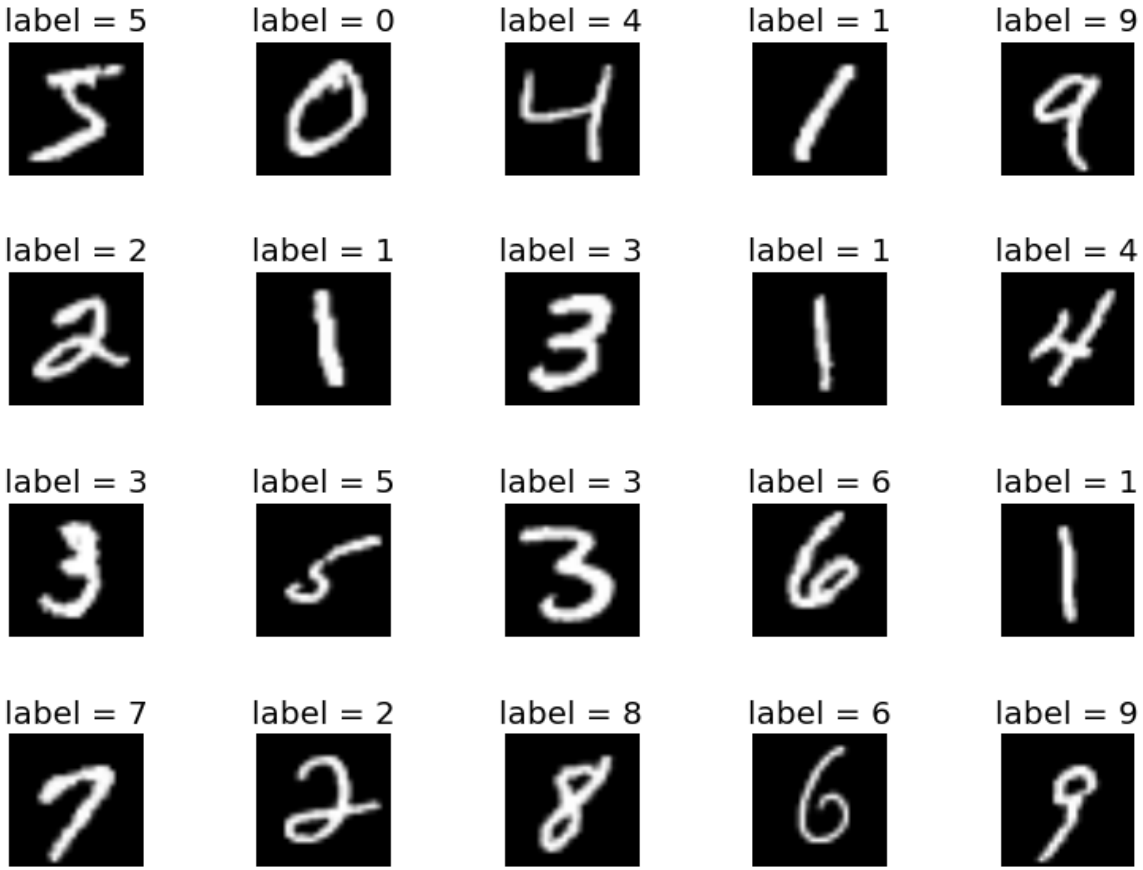


Calculate Loss


Epoch 1…5
For 1st batch in the 1st epoch use random weights
Batch 1…1000
Training Instance 1…6
Run through the Network, Predict the Label, Calculate the Loss
Update the weights
Use Loss to calculate weights adjustment by Backpropagation
Calculate Loss
label = 5
weights
Inputs
Derived Features
Output
Age
Sex
Total Chol
Pr(D=1)
Log Loss when true label = 1
log loss
predicted probability
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10

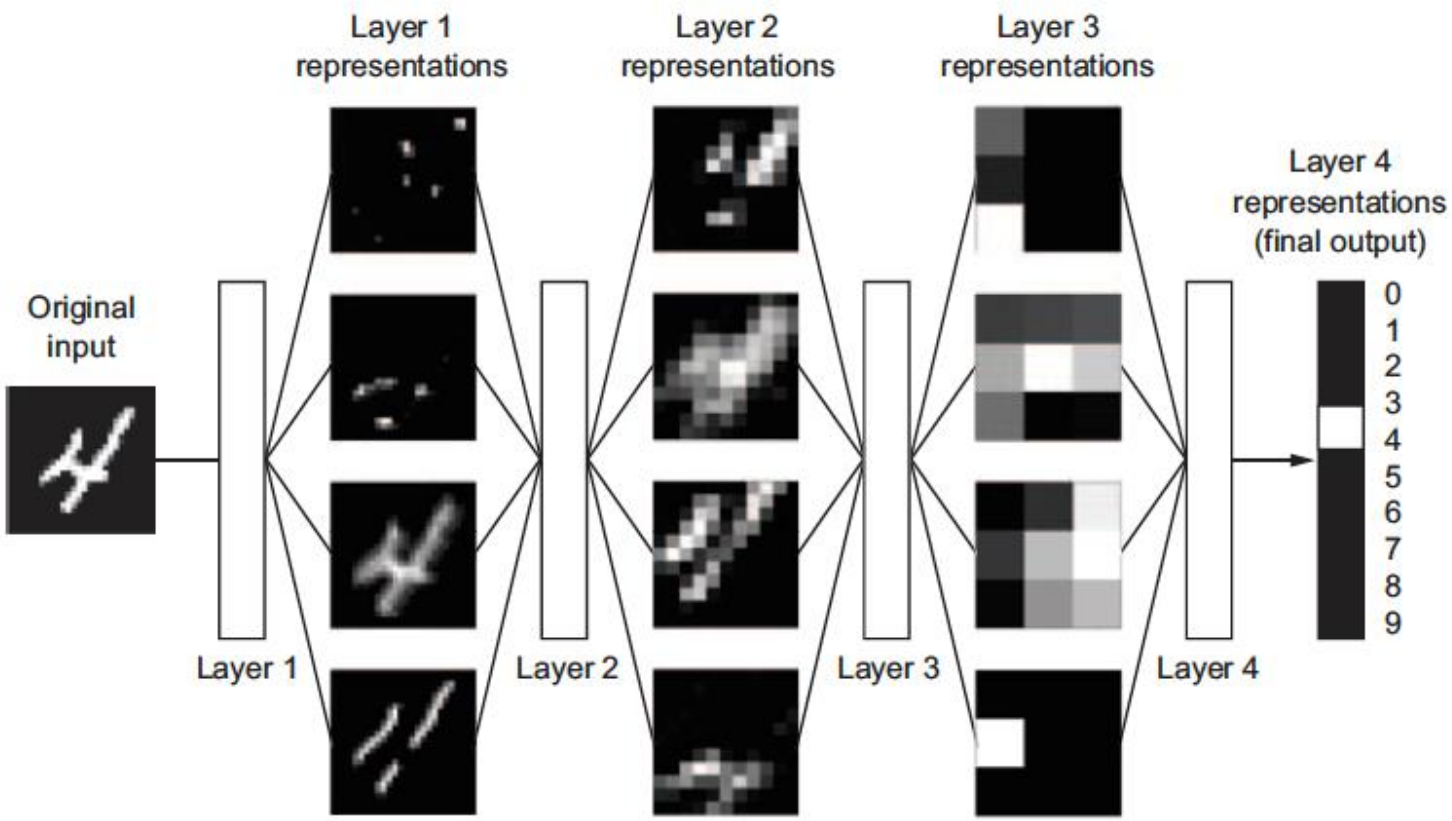
Convolutional Neural Networks (CNN) for Image Recognition

MNIST dataset used for image recognition

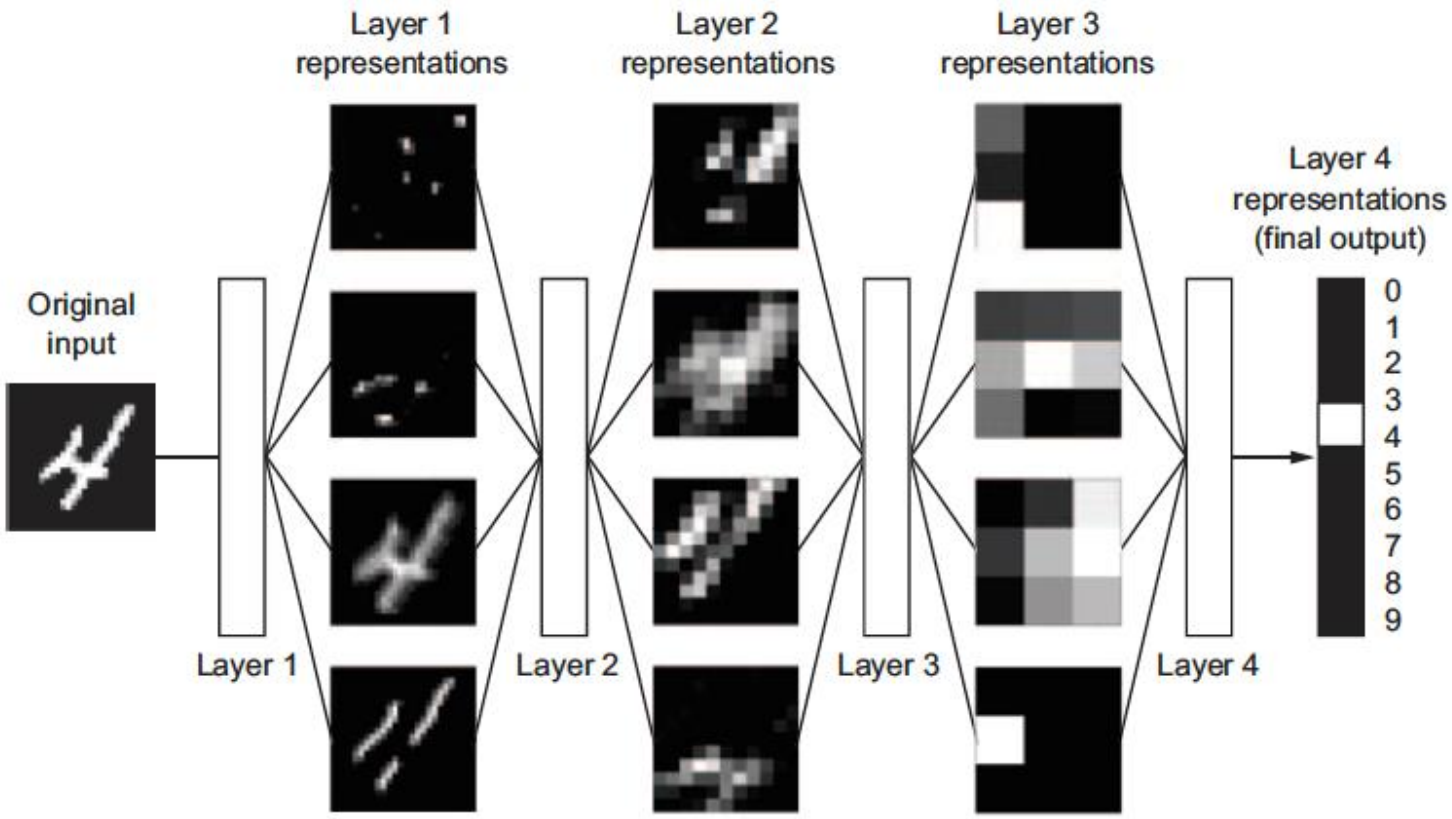


The MNIST database contains 60,000 training images and 10,000 testing images
MNIST is Modified National Institute of Standards and Technology database

Convolutional Neural Networks (CNN) for Image Recognition



Convolutional Neural Networks (CNN) for Image Recognition



Convolution Layers

Dense Layer = Fully Connected Layer

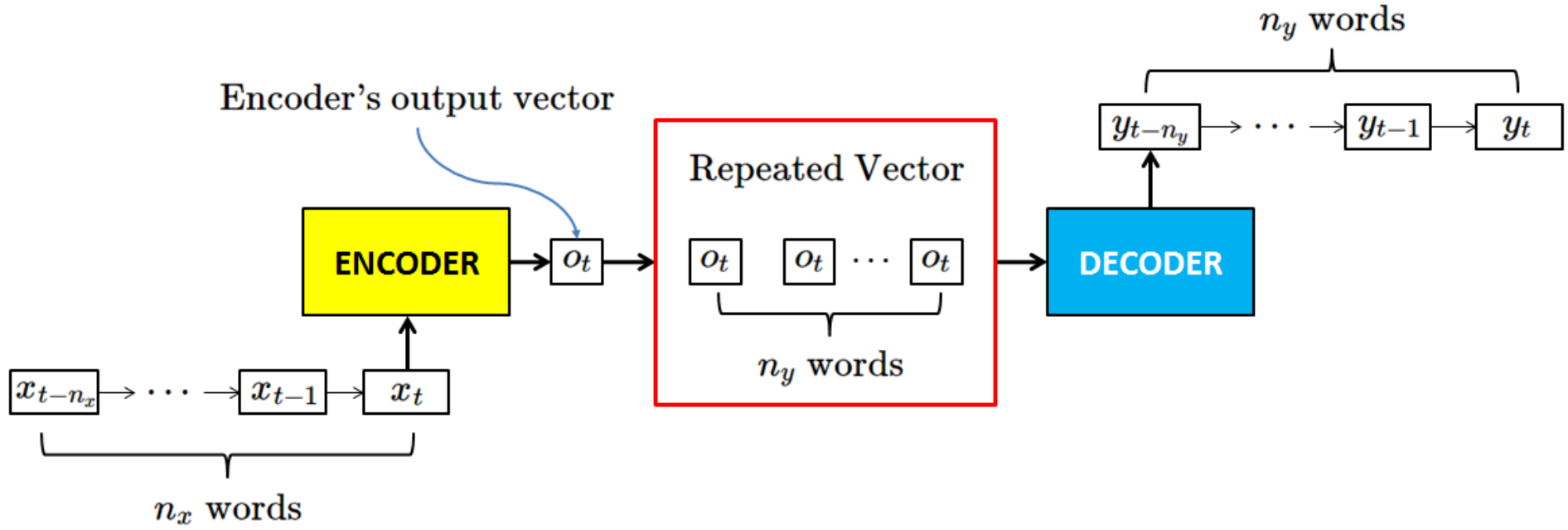
Convolution is a function used in Signal Processing. In most CNNs convolution is equivalent to filtering.

Image of a cat modified by the first convolutional layer.
We can interpret it as an eye filter.

More about Convolutional Neural Networks:
https://www.youtube.com/watch?v=aircAruvnKk

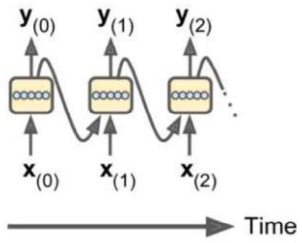
Some Other Types of Neural Networks

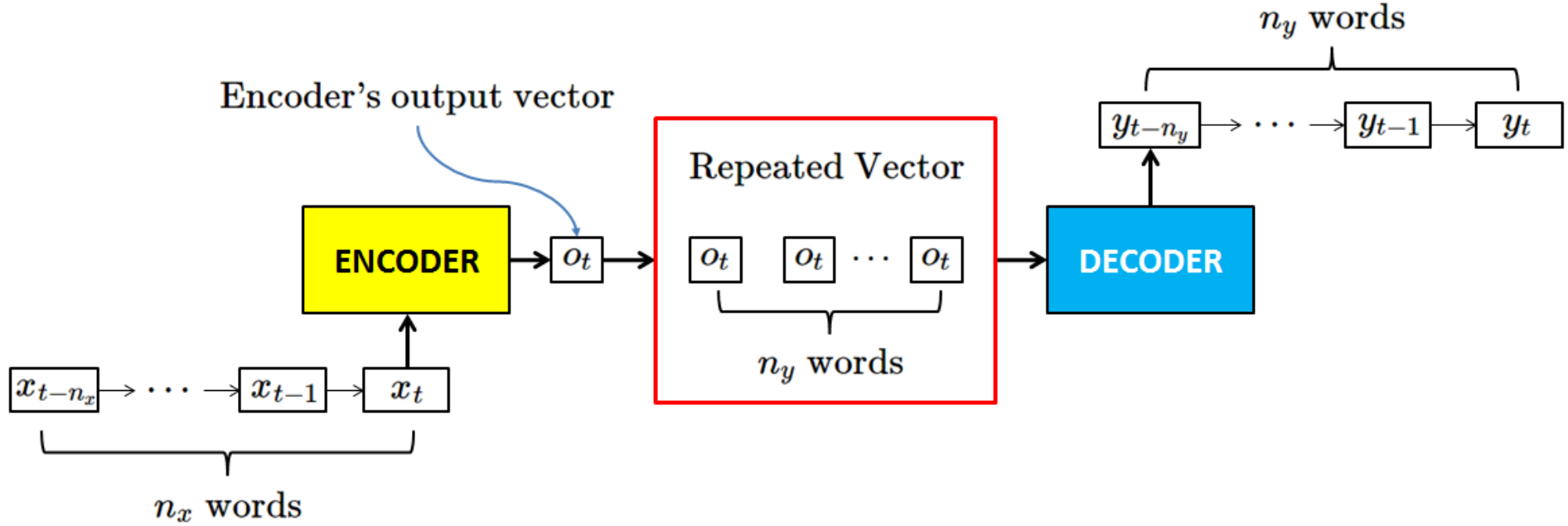
- Recurrent Neural Networks: speech recognition, language translation



Some Other Types of Neural Networks

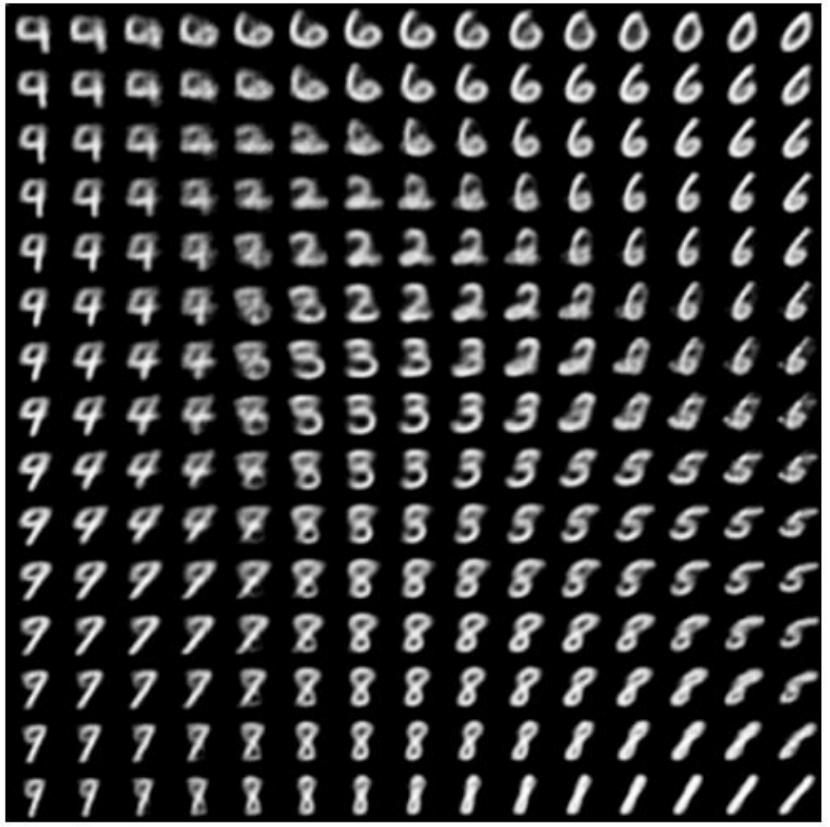
- Recurrent Neural Networks: speech recognition, lan slation



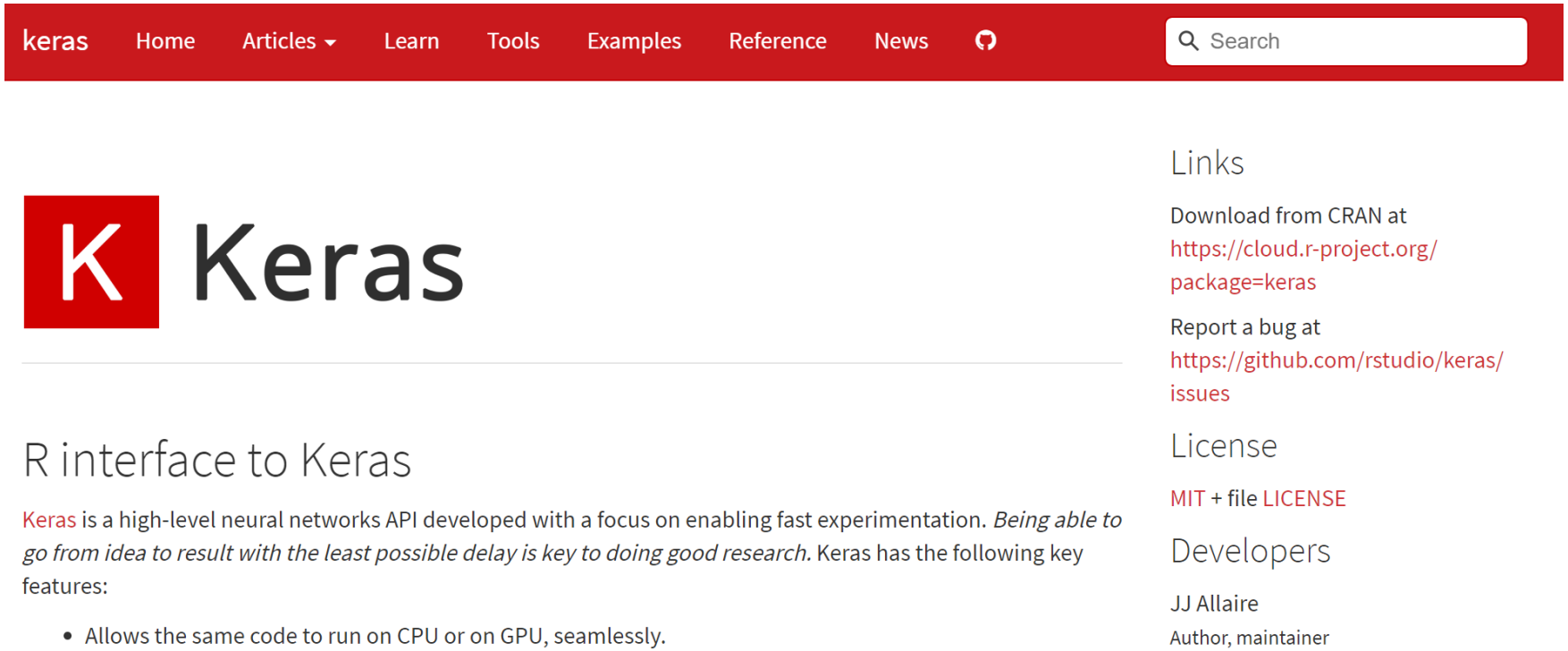


Some Other Types of Neural Networks

- Autoencoders: unsupervised learning, finding common classes, image generation

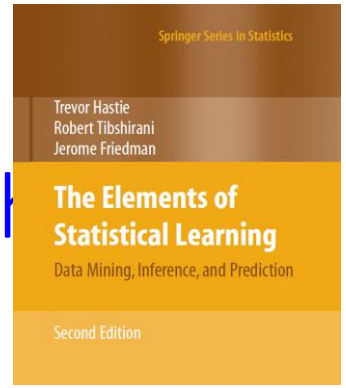


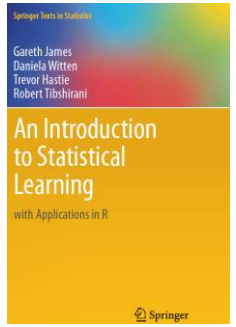
Further Reading



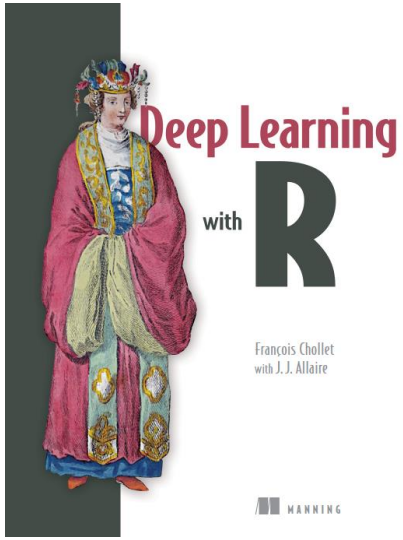
https://keras.rstudio.com/
https://conda.io/docs/user-guide/install/index.html
https://medium.freecodecamp.org/why-you-need-python-environments-and-how-to-manage-them-with-conda-85f155f4353c

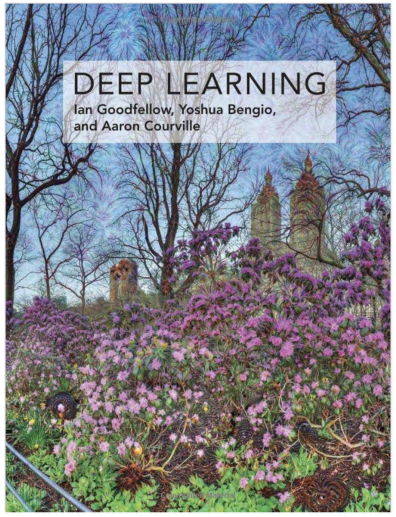
Furth ing





- http://statweb.stanford.edu/~tibs/ElemStatLearn/index.html
- http://www-bcf.usc.edu/~gareth/ISL/





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

Thank you!