>>> CS 7840 Soft Computing
>>> Basic Statistics

Name: Dr. Sanghui Han

Date: December 31, 2021

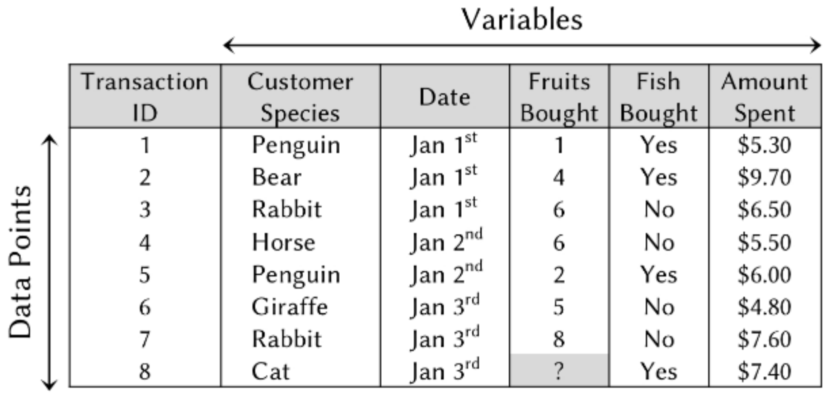
>>> Agenda

1. Basics
2. Clustering
3. Regression Analysis
4. Unconstrained Optimization
5. Dimension Reduction Methods

>>> Data

* All data consists of data points and variables.
* Variables are also known as attributes, features, or dimensions.
* There are four main types of variables:
   1. Binary - Two possible options.
   2. Categorical - More than two options, but finite.
   3. Integer - Represented by whole number, and can be infinite.
   4. Continuous - Most detailed, representing any number system (real, imaginary, ect.).
* There are ways to handle missing data:
   1. Approximation - Simplest way to handle missing data. Replace with mode or median.
   2. Computed - More time consuming and complex, but more accurate. Use advanced algorithms such as supervised learning to estimate based on trends.
   3. Removal - Last resort when no other option is available.

>>> Example


Variables
Data Points

Transaction ID	Customer Species	Date	Fruits Bought	Fish Bought	Amount Spent
1	Penguin	Jan 1st	1	Yes	\$5.30
2	Bear	Jan 1st	4	Yes	\$9.70
3	Rabbit	Jan 1st	6	No	\$6.50
4	Horse	Jan 2nd	6	No	\$5.50
5	Penguin	Jan 2nd	2	Yes	\$6.00
6	Giraffe	Jan 3rd	5	No	\$4.80
7	Rabbit	Jan 3rd	8	No	\$7.60
8	Cat	Jan 3rd	?	Yes	\$7.40

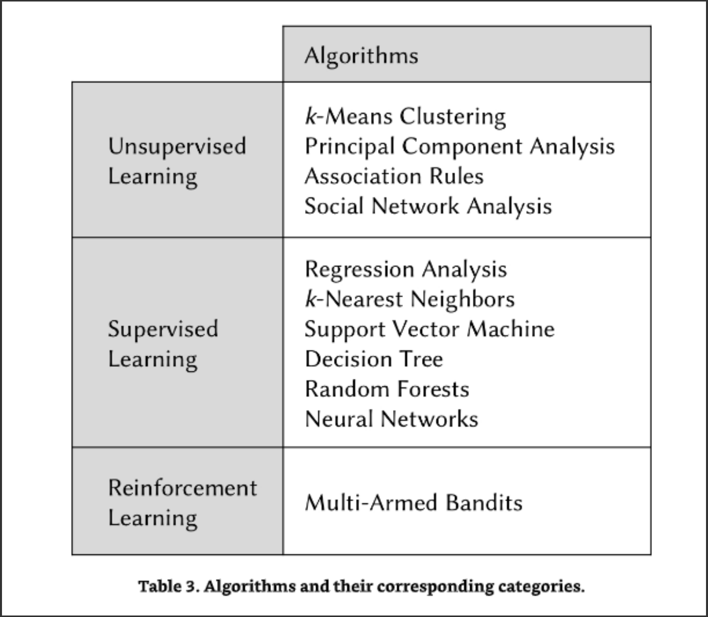
Table 1. Imaginary dataset of grocery transactions from animals shopping at a supermarket. Each row is a transaction, and each column provides information on the transactions.

>>> Algorithm Selection

We cover four types of algorithms:

* Unsupervised Learning: Find pattern that exist in data.
  1. Clustering
  2. Association
* Supervised Learning: Use known patterns in data to make predictions.
  1. Classification
  2. Regression
* Reinforcement Learning: Find patterns, then make predictions, and improve them with more information. Hybrid of supervised and unsupervised learning.
  1. Policy optimization or iteration
  2. Q-learning or value-iteration
  3. Model-based Methods
* Semi-supervised learning will be covered at end after Evolutionary Computation.

>>> Table of Algorithms



	Algorithms
Unsupervised Learning	*k*-Means Clustering Principal Component Analysis Association Rules Social Network Analysis
Supervised Learning	Regression Analysis *k*-Nearest Neighbors Support Vector Machine Decision Tree Random Forests Neural Networks
Reinforcement Learning	Multi-Armed Bandits

Table 3. Algorithms and their corresponding categories.

>>> Unsupervised Learning

Also called self-organized learning, this type of learning does not require labeled training data.

* It is a task-independent measure of what the network learns and the free parameters are optimized according to this measure.
* There are two types of unsupervised learning problems:
  1. Clustering - find inherent clusters in the data.
  2. Association - find rules that describe large portions of the data.
* The goal is to find the underlying structure of the data without knowing the answers and so new classes are created automatically.

>>> Supervised Learning

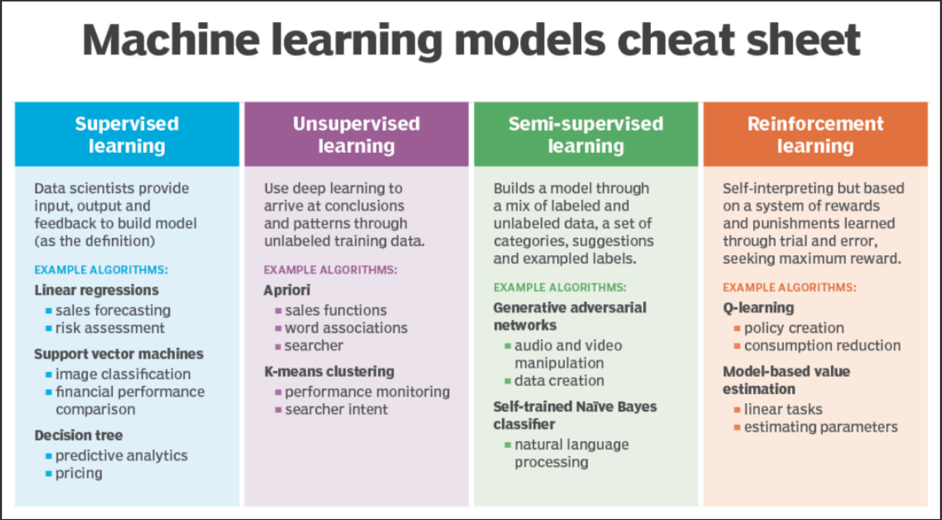
This is a learning process that requires labeled training data.

* The input-output examples serve as a priori knowledge of the environment.
* While there are examples of the desired input and output (training data), the environment is still unknown.
* There are two pain categories of supervised learning problems:
    1. Classification - fit data into known categories.
    2. Regression - data are real values, and we fit an equation to the values.
* The network parameters are adjusted as new training data becomes available.
* The error signal is the difference between the true output and the actual output.

>>> Reinforcement Learning

* Also called semi-supervised learning, this is a hybrid learning process.
* There are many different algorithms that can be roughly classified into three methods:
  1. Policy optimization or iteration - maps rules or policies to actions which provides the additional information for the data grouping.
  2. Q-learning or value-iteration - measures the value of an action after policies are mapped.
  3. Model-based Methods - use models to choose the optimal actions.
* Input-output mapping is through continuous interaction of the learning system with its environment to minimize index of performance.

>>> Machine Learning Models Cheat Sheet



Machine learning models cheat sheet

Supervised learning

Data scientists provide input, output and feedback to build model (as the definition)

EXAMPLE ALGORITHMS:

Linear regressions
- sales forecasting
- risk assessment

Support vector machines
- image classification
- financial performance comparison

Decision tree
- predictive analytics
- pricing

Unsupervised learning

Use deep learning to arrive at conclusions and patterns through unlabeled training data.

EXAMPLE ALGORITHMS:

Apriori
- sales functions
- word associations
- searcher

K-means clustering
- performance monitoring
- searcher intent

Semi-supervised learning

Builds a model through a mix of labeled and unlabeled data, a set of categories, suggestions and exampled labels.

EXAMPLE ALGORITHMS:

Generative adversarial networks
- audio and video manipulation
- data creation

Self-trained Naïve Bayes classifier
- natural language processing

Reinforcement learning

Self-interpreting but based on a system of rewards and punishments learned through trial and error, seeking maximum reward.

EXAMPLE ALGORITHMS:

Q-learning
- policy creation
- consumption reduction

Model-based value estimation
- linear tasks
- estimating parameters

>>> Parameter Tuning

* Parameters are options in an algorithm's settings.
* Different algorithms have different parameters for tuning.
* Selecting the right parameters can find a good fit without overfitting or underfitting the data.
  * Overfitting data makes the model too sensitive to outliers. This yields highly accurate results for data used, but difficult to generalize for future data. This happens when a model is too complex.
  * Underfitting data makes the model likely to neglect significant trends, and make predictions less accurate.
  * Well tuned parameters balance the model to make accurate prediction while ignoring outliers.
* A way to keep a model's complexity in check is with a penalty parameter or regularization. This accounts for the model's complexity and accuracy in optimizing its original parameters.

>>> Parameter Tuning Example

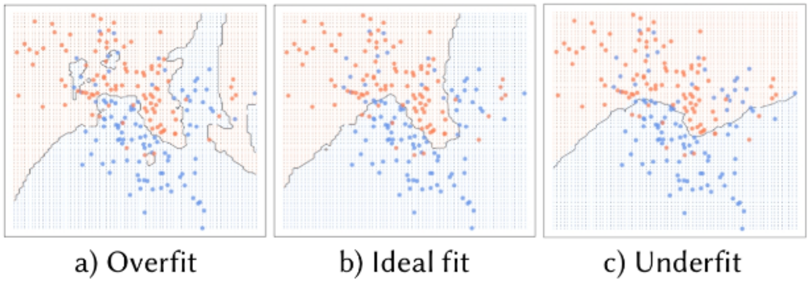
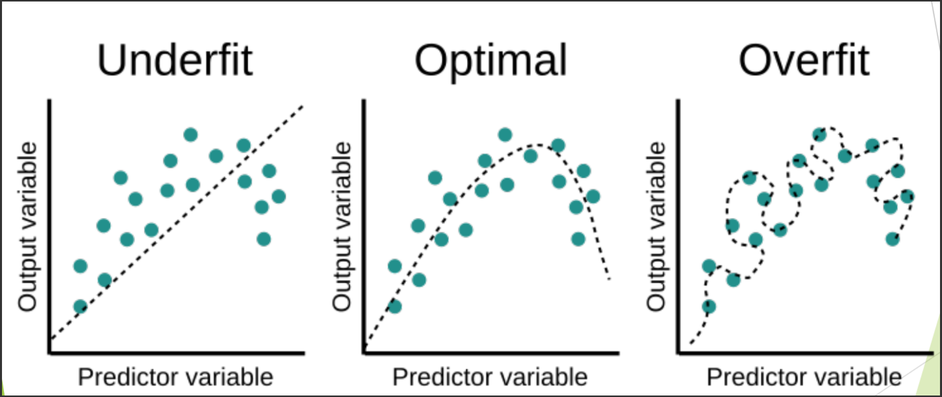


Figure 2. Comparison of prediction results from the same algorithm fitted with different parameters.

>>> A Simpler Parameter Tuning Example



>>> Evaluation Metrics

* Once model is built, it needs to be evaluated for accuracy.
* There are several ways to calculate the accuracy of a model.
* The simplest way is to calculate the percentage of correct predictions.
* Another method is to calculate the confusion matrix.
* The Root Mean Squared Error is used for predictions with continuous values.
* Validation is an assessment of how accurate a model is in predicting new data.
* Cross-validation is using the available data, and splitting it into several segments to test the model repeatedly.

>>> Clustering Example

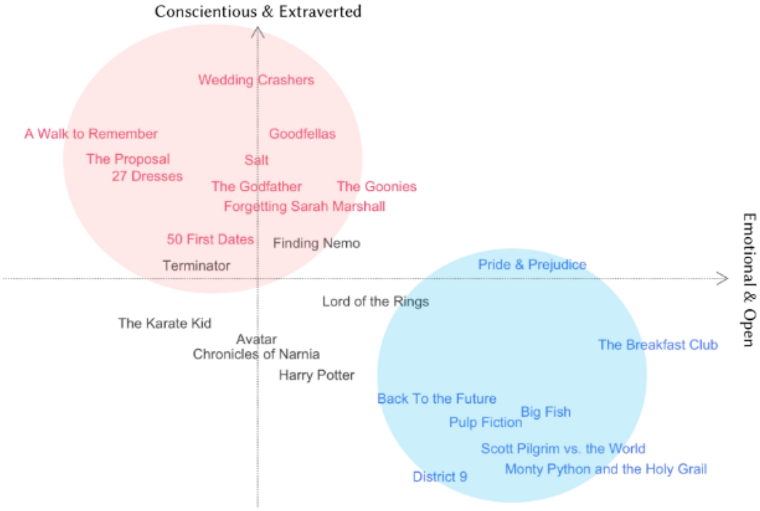


Figure 1. Personality profiles of movie fans.

>>> k-Means Clustering

* Clustering is an unsupervised learning method that partitions the data into specified number of groups to minimize the error in the data from the mean of the clusters.
* k-Mean Clustering is a popular algorithm due to its simplicity and effectiveness.
* The algorithm is iterative with the following steps:
  1. Select a random k points in the data.
  2. Calculate the distance between the centers and each data point.
  3. Assign the data point to the group with the minimum distance from the center.
  4. Recalculate the cluster center based on the data assigned to each group.
  5. Iterate steps 2 and 4 until there is no change in the data point assignments.

>>> k-Means Illustration

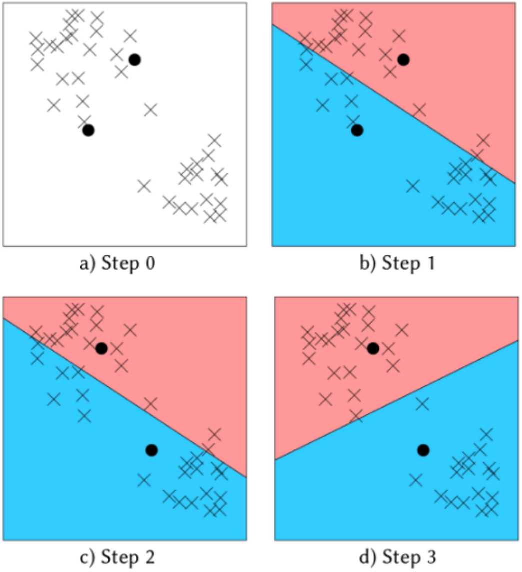


Figure 3. Iterative process in *k*-means clustering.

>>> Recursive Least Squares Estimation

* The k-Means Clustering Algorithm requires recursive computation of the least squares error.
* We can treat the least squares estimation as a recursive computation of the weight vector of the output layer in terms of a discrete time value n.

$$\mathbf{R}(n)\hat{\mathbf{w}}(n) = \mathbf{r}(n) \qquad n = 1, 2, ... \tag{1}$$

* The K-by-K correlation function of the hidden layer output is defined by $\mathbf{R}(n)$.

$$\begin{aligned} \mathbf{R}(n) &= \sum_{i=1}^{n} \mathbf{\Phi}(\mathbf{x}_i)\mathbf{\Phi}^T(\mathbf{x}_i) \qquad where \\ \mathbf{\Phi}(\mathbf{x}_i) &= [\phi(x_i, \mu_1), ...\phi(x_i, \mu_K)] \qquad and \\ \phi(x_i, \mu_j) &= exp\left(-\frac{||x_i - \mu_j||^2}{2\sigma_j^2}\right) \qquad j = 1, 2, ...K \end{aligned} \tag{2}$$

>>> Recursive Least Squares Estimation

* The K-by-1 cross-correlation vector between the desired response $d(i)$ at the output and the hidden-unit output is defined by $\mathbf{r}(n)$.

$$\mathbf{r}(n) = \sum_{i=1}^{n} \mathbf{\Phi}(\mathbf{x}_i) d(i) \tag{3}$$

* The unknown weight vector $\hat{\mathbf{w}}(n)$ is solved recursively.

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{R}^{-1}(n)\mathbf{\Phi}(n)\alpha(n) \tag{4}$$

$$\alpha(n) = d(n) - \mathbf{w}(n-1)\mathbf{\Phi}^T(n)$$

* The new term $\alpha(n)$ is the prior estimation error which is based on the old estimate of the weight vector.

>>> Linear Regression Model

* Linear regression is a way to find the trend line or best-fit line of data that minimizes error.
* This is popular because it is easy to understand and you can build on predictors.
* Regression analysis allows us to account for multiple predictors and compare strengths of each predictor.
* The prediction error is the distance each point is away from the trend line.
* Calculating the slope of the trend line that minimizes the prediction error is the primary goal of linear regression models.
* Finding the appropriate predictors is also a key task to find the best trend line.

>>> Trend Line and Prediction Error Example

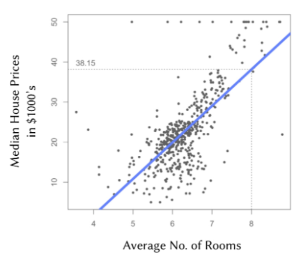


Figure 1. House price against number of rooms.

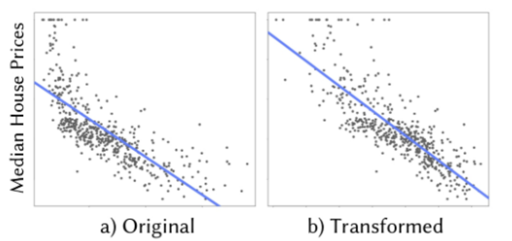


Figure 2. House price against proportion of neighborhood with low income.

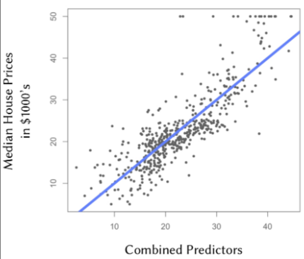


Figure 3. House price against a weighted combination of number of rooms and neighborhood affluence.

	Prediction Error (in $1000's)
No. of Rooms	4.4
Neighborhood Affluence	3.9
No. of Rooms & Neighborhood Affluence	3.7

Table 1. Average prediction errors from using the three trend lines.

>>> Gradient Descent

* Another method for optimizing parameters when they cannot be derived directly is gradient descent.
* Gradient descent is one of the most common method for optimizing parameters and prediction models.
* Its an iterative method take takes an initial set of weights or regression coefficients and applies them to every data point to make prediction.
* For each predictor, the weight is called the correlation coefficient.
* Correlation coefficients provide direction and magnitude of the predictor and has values between -1 and 1.
* At each step, the algorithm determines which direction provides the steepest descent, and re-calibrates the weights in that direction.

>>> Gradient Descent Example

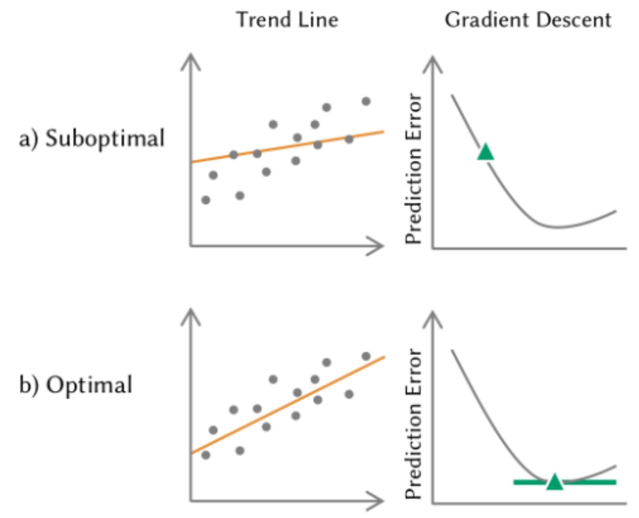


Figure 4. How a trend line approaches optimality via gradient descent.

>>> Correlation Coefficient Example

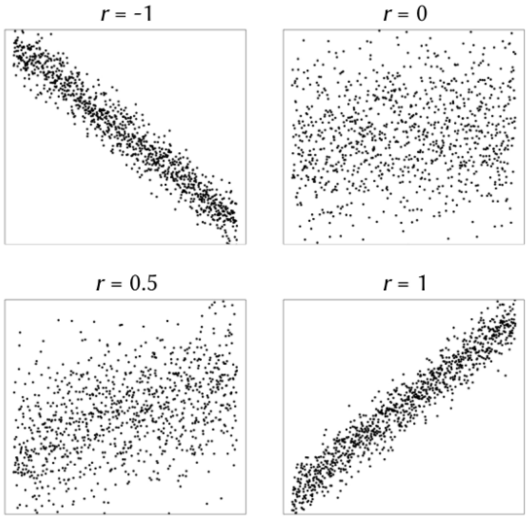


Figure 6. Examples of data spread corresponding to various correlation coefficients.

>>> Steepest Descent

* Successive adjustments applied to the weight vector $\mathbf{w}(n)$ are in the direction of steepest descent, or opposite to the gradient vector $\nabla\mathcal{E}(\mathbf{w})$.

$$\mathbf{g} = \nabla\mathcal{E}(\mathbf{w}) \tag{5}$$

* The algorithm can be described by

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta\mathbf{g}(n)$$

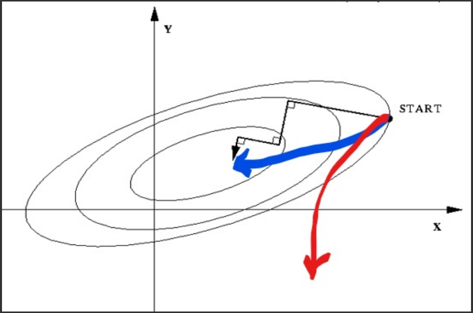
  where η is the step size, or learning-rate, and $\mathbf{g}(n)$ is the gradient vector evaluated at point $\mathbf{w}(n)$.

* In the iterative process going from n to $n+1$, the algorithm applies the correction

$$\nabla\mathbf{w}(n) = \mathbf{w}(n+1) - \mathbf{w}(n) = \eta\mathbf{g}(n)$$

  to find the optimal solution $\mathbf{w}^*(n)$.

>>> Steepest Descent Illustration



* If η is small, it is overdampened, then the trajectory follows a smooth path. Shown in blue.
* If η is large, it is underdampened, and has a zig-zag trajectory. Shown in black.
* If η exceeds a critical value, then the algorithm becomes unstable. Shown in red.

>>> Newton's Method

* The Newton's Method of optimization is a more elaborate technique which minimizes the quadratic approximation of the cost function $\mathcal{E}(\mathbf{w})$.
* It uses the second-order Taylor series expansion of the cost function around $\mathbf{w}(n)$.

$$\begin{aligned} \nabla\mathcal{E}(\mathbf{w}(n)) &= \mathcal{E}(\mathbf{w}(n+1)) - \mathcal{E}(\mathbf{w}(n)) \\ &\approx \mathbf{g}^T(n)\nabla\mathbf{w}(n) + \frac{1}{2}\nabla\mathbf{w}^T(n)\mathbf{H}(n)\nabla\mathbf{w}(n) \end{aligned} \tag{6}$$

* Here $\mathbf{g}(n)$ is the M-by-1 gradient vector of $\mathcal{E}(\mathbf{w})$ evaluated at $\mathbf{w}(n)$, and $\mathbf{H}(n)$ is the m-by-m Hessian of $\mathcal{E}(\mathbf{w})$.
* This method assumes the cost function is differentiable and the second derivative exists, and assumes $\mathbf{H}(n)$ is a positive definite matrix for all n.

>>> Newton's Method Continued

* The Hessian of $\mathcal{E}(\mathbf{w})$ is defined by

$$\mathbf{H} = \nabla^2\mathcal{E}(\mathbf{w}) = \begin{bmatrix} \frac{\delta^2\mathcal{E}}{\delta w_1^2} & \frac{\delta^2\mathcal{E}}{\delta w_1 \delta w_2} & \cdots & \frac{\delta^2\mathcal{E}}{\delta w_1 \delta w_M} \\ \frac{\delta^2\mathcal{E}}{\delta w_2 \delta w_1} & \frac{\delta^2\mathcal{E}}{\delta w_2^2} & \cdots & \frac{\delta^2\mathcal{E}}{\delta w_2 \delta w_M} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\delta^2\mathcal{E}}{\delta w_M \delta w_1} & \frac{\delta^2\mathcal{E}}{\delta w_M \delta w_2} & \cdots & \frac{\delta^2\mathcal{E}}{\delta w_M^2} \end{bmatrix} \tag{7}$$

* If these assumptions hold true, the cost is minimized by solving

$$\mathbf{g}(n) + \mathbf{H}(n)\nabla\mathbf{w}(n) = \mathbf{0} \tag{8}$$

$$\nabla\mathbf{w}(n) = -\mathbf{H}^{-1}(n)\mathbf{g}(n) \tag{9}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \nabla\mathbf{w}(n) = \mathbf{w}(n) - \mathbf{H}^{-1}(n)\mathbf{g}(n) \tag{10}$$

* Newton's method converges quickly, and does not have the zig-zag behavior.

>>> Gauss-Newton Method

* Often there is no guarantee that all values of the Hessian are positive definite. In these cases, the Gauss-Newton method is used to handle the computational complexities without compromising its convergence behavior.
* For this method, the cost function is expressed as the sum of errors squared.

$$\mathcal{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n} \mathbf{e}^2(i) \tag{11}$$

* Here $\mathbf{e}(n)$ is the error vector:

$$\mathbf{e}(n) = [e(1), e(2), ..., e(n)]^T \tag{12}$$

>>> Gauss-Newton Method Continued

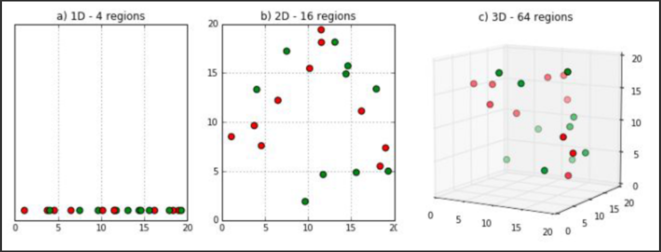
* In order to minimize the error, we calculate the Jacobian of $\mathbf{e}(n)$.
* The Jacobian $(\mathbf{J}(n)$ is the partial derivative of $\mathbf{e}(n)$ with respect to $\mathbf{w}(n)$ and the transpose of the m-by-n gradient matrix $\nabla\mathbf{e}(n)$.
* The recursive equation to minimize error is

$$\mathcal{E}(\mathbf{w}) = \frac{1}{2}\left\{\sum_{i=1}^{n} e^2(i) + \delta||\mathbf{w} - \mathbf{w}(n)||^2\right\} \qquad (13)$$

  where $\mathbf{w}(n)$ is the current value of the weight vector $\mathbf{w}(i)$.
* The sum of $e^2(i)$ is dependent on the training data.
* The Euclidean norm $||\mathbf{w} - \mathbf{w}(n)||^2$ depends on the filter structure and acts as a stabilizer.

>>> Curse of Dimensionality

* What makes machine learning powerful is its ability to absorb data with high dimensions. However, how many dimensions are necessary and how many just cause problems?
* When dimensions increase, the volume of data increases and makes it sparse.
* The amount of training data needed for reliable results increases exponentially.



>>> Principal Component Analysis

* Principal Component Analysis (PCA) is a technique to find the underlying components of a dataset with high dimensionality that has the most information.
* It is one of the most popular dimension reduction methods.
* It assumes the dimensions with the highest variability, or the most spread out, has the most information.
* If more than one variable is used to calculate the PCA, the dimensions need to first be standardized.
* One of the parameters for tuning in a predictive model is the selection of how many principal components to use in the model.
* PCA always generated orthogonal principal components.

>>> PCA Mechanics

* PCA Uses linear fitting and Singular Value Decomposition (SVD) to project the higher dimensions into a lower dimension.
* The projection depends on the global linearity of the data, so noise-adjusting is a common pre-processing step.
* Assume $\mathbf{X}$ is a random vector in the dataset with zero mean:

$$\mathbb{E}[\mathbf{X}] = \mu = \mathbf{0} \qquad \textit{where} \qquad \mathbf{X} = [X_1, X_2, ..., X_p]^T \tag{14}$$

* If it does not have zero mean, then we do noise-adjusting to make it so.
* The linear combination of its components is

$$Y_1 = \mathbf{a_1}^T\mathbf{X} = a_{11}X_1 + a_{21}X_2 + ... + a_{p1}X_p \tag{15}$$

* The first principal component is the linear combination of Y_1 that maximizes the variance subject to the constraint $\mathbf{a_1}^T\mathbf{a_1} = 1$ so $\mathbf{a_1}$ has length 1.

>>> Finding the Principal Components

* The variance-covariance of $\mathbf{X}$ is

$$\mathbf{\Sigma} = Var(\mathbf{X}) = \mathbb{E}[(\mathbf{X} - \mu)(\mathbf{X} - \mu)^T] \tag{16}$$

* The eigenvalues of $\mathbf{\Sigma}$ are $\lambda_1, \lambda_2, ..., \lambda_p$, and the eigenvectors normalized by their respective eigenvalues are $\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_p$. Here the eigenvalues are ordered so that $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p \geq 0$.
* The jth principal component is

$$Y_j = \mathbf{e}_j^T \mathbf{X} \tag{17}$$

and

$$Var(Y_j) = \mathbf{e}_j^T \mathbf{\Sigma} \mathbf{e}_j, \qquad j = 1, 2, ..., p \tag{18}$$

* The principal components are uncorrelated since $Cov(Y_i, Y_j) = \mathbf{e}_i^T \mathbf{\Sigma} \mathbf{e}_j = \lambda_j \mathbf{e}_i^T \mathbf{e}_j = 0$ for $i \neq j$.

>>> Independent Component Analysis

* Since the requirement for each component to be orthogonal can be restrictive, we can use an alternate techniques known as Independent Component Analysis.
* ICA allows components to be non-orthogonal, but prevents overlap in the information they contain.
* ICA can reveal unique information in the dataset and looks beyond variability in determining the components.
* So why is PCA so much more popular than ICA?
    * PCA assumes each data point is an independent random process with normal distribution.
    * ICA works for data when this assumption does not hold.
    * ICA uses the inverse principal component matrix to make the data statistically independent.

>>> ICA Mechanics

* For a dataset $\mathbf{X}$, the random vector components are $\mathbf{S} = S_1, S_2, ...S_p$.
* We find a linear transformation $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_p)$ so that $\mathbf{S} = \mathbf{W}\mathbf{X}$.
* The independent components are $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_p)$ and the observed variables of the dataset $\mathbf{X}$ can be expressed as

$$X_i = a_{i,1}S_1 = a_{i,2}S_2 + ... + a_{i,p}S_p \tag{19}$$

* The mixing matrix $\mathbf{A}$ and the random vector components $\mathbf{S}$ are estimated by adaptively calculating the cost function of $\mathbf{W}$ that maximizes the non-guassianity of $S_i = \mathbf{w}_i^T\mathbf{X}_i$ or minimizes mutual information.

>>> PCA and ICA Illustration

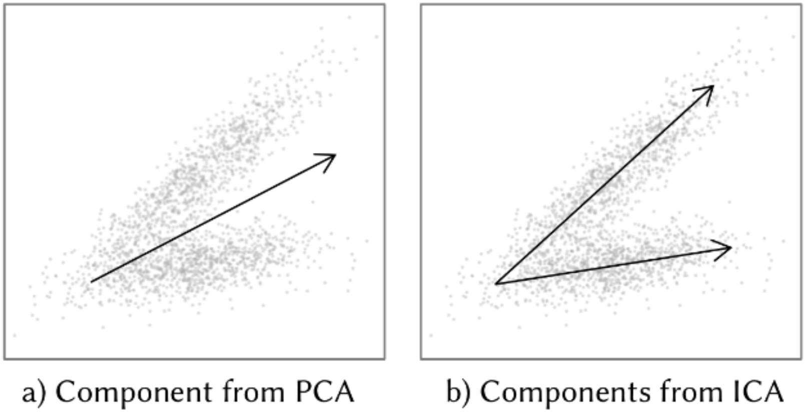


Figure 9. Comparison of how PCA and ICA identify important components.

So why is PCA so much more popular than ICA?