Machine Learning Library in DRIP

Lakshmi Krishnamurthy

v0.68 25 March 2015

Markov Models

Markov Property

1. Definition: The state of the system at time t_i only depends on the state at t_{i-1} (of course, in addition it may also depend on additional external shocks during this instant).
2. Semi-Markovian Smoothener: In this case, the state at t_i actually depends on the states at $\{t_j\}_{j=i-N}^{i}$, i.e., smoothening needs to occur across the last N observations.
   - In lagging Markovian system, the state at t_i depends only on the lagging state t_{i-N}.

Markov Chains

1. Definition: Markov definition indicates strict immediate prior dependence. The sequence of Markov realizations constitutes the Markov chain.
2. Example of Markov Chain: Language parsing using tagging is a great example (Sequence Labeling (Wiki)). Here the whole context ends up being parsimoniously restricted to the neighborhood tags per each parsed fragment. The tags can, of course, become a sequence, and therefore a sizeable list/chain – a Markov chain.
3. Bayesian Estimation/Updates as Markov Chain Update: The single posterior Bayesian estimate depends only on the prior. Thus, the Markov property is maintained. Further, the sequence of Bayesian posteriors automatically constitutes a Markov chain.
4. "Model" in Bayesian vs. "Control" in Markov Filtering: Bayesian models are analogously identical to controls in Markov filtering, as in:

- $Bayesian Posterior = Model \otimes Bayesian \Pr ior$
- $Markov Filtering Update = Model Update \otimes Control Update$

The difference between them is that, since Bayesian analysis deals with state variate distributions (posterior and prior), the operator $\otimes$ ends up becoming a convolution. In Markov filtering updates, the operation $\otimes$ is a linearized additive.

5. Entities in the Markov Chain: The entities that constitute the Markov chain are all inferred/predicted, i.e., they forma sequence of entities that sequentially form the basis for the next stage of prediction.
6. Markov Random Field: These are also called Markov network. It is just a multi-dimensional Markov chain. The inference techniques used result in multi-dimensional joint distributions.

Classification of the Markov Models

1. Markov Model Classification Table: This classification comes from Wikipedia (Markov Model (Wiki)). Markov chains as described earlier correspond to models for fully observable and autonomous systems.

System Observability and Type	Fully Observable	Partially Observable
Autonomous	Markov Chain	Hidden Markov Model
Controlled	Markov Decision Process	Partially Observable Markov Decision Process

2. Partially Observable, Autonomous System: These systems are modeled using Hidden Markov Models, which employ state inference/estimation techniques from a sequence of observations. The following algorithms fall into this category:
   - The Viterbi algorithm will infer the most likely corresponding sequence of states.

- The forward algorithm will infer the probability of a given sequence of observations.
- The Baum-Welch will infer the starting probabilities, the transition functions, and the observation functions.

3. Controlled Markov System: Control is always specified through a controlled action vector, which implies that the optimal control vector parameters need to be extracted. In general, this extraction occurs by using non-linear iterative fixed point finder approaches.
4. Fully Observable, Controlled System: In these systems, the state transitions are observed from the current state to which an action vector is applied. These systems are modeled using the Markov Decision Process, which employ a policy of actions that maximize a chosen utility function with respect to some targeted rewards.
    - These are closely related to reinforcement learning and Kalman filtering/extraction formulation, and the action policy parameters are computed using iteration and other related methods.
5. Partially Observable, Controlled System: These systems are also modeled using the Markov Decision Process. These are known to be NP-complete, but recent approximations complete (Kaelbling, Littman, and Cassandra (1998)) have improved their tractability to make them useful for a variety of applications such as controlling robots, etc:

Monte Carlo Markov Chains (MCMC)

1. Definition: MCMC methods are a class of algorithms for sampling a probability distribution based on constructing a Markov chain that has the desired distribution that, after a large number of steps, the generated distribution becomes the desired distribution (Markov Chain Monte Carlo (Wiki)).
2. MCMC Entity Distribution vs. Target Distribution: Given that the steps correspond to a Markov chain in the theoretical sense, the chain of steps can be constructed to form the TRUE target distribution.

- MCMC Entity Generation Rule => As long as the proposed rule is guaranteed to be able to sample the target distribution (either as a consequence of direct algorithms, or using sufficient statistics proxies), the MCMC entity generation walks will correspond to the equilibrium distribution.

3. MCMC Mixing Time: This is the time needed to converge to the stationary distribution within acceptable error (it is measured in the number of steps, or more accurately, in cops or cost-of-operations). Lesser number of steps indicates that the corresponding algorithm has a rapid mixing time.

MCMC for Multi-dimensional Integrals

1. Base Algorithm: Here, an ensemble of MCMC invocations (called walkers) moves around the variate space semi-randomly, looking for a place with a "high-enough" contribution (Gill (2008), Robert and Casella (2004)).
2. Conventional Random-Walker Integral vs. MCMC Integral: Random samples in a conventional random-walker integrand are generated statistically independently, whereas those generated in MCMC random walks are auto-correlated (i.e., due to the Markov nature, the current walker position determines where it is walk to next). The only constraint is that the generated walk-coordinates maintain the target distribution, which, as we've seen, is easy to do.
3. Random Walker Sampling Algorithms:
   - Metropolis-Hastings Algorithm => This algorithm generates a random walk using a proposal density, and a method for rejecting the proposed moves.
     - The multiple-try Metropolis algorithm is a variant on the above that allows multiple trials at each variate point. It allows the algorithm to take larger steps in each direction, and is particularly useful when dealing with large dimensions.
   - Gibbs Sampling => In this algorithm, all the conditional samples of the target distribution are generated precisely, thereby without need for algorithmic tuning.

- Slice Sampling => This algorithm samples the distribution by sampling uniformly the region under its density plots. In practice, it alternates between the "uniform vertical" and the "uniform horizontal" slice defined by the current vertical position.

4. MCMC Semi Random-Walker Integration Algorithms: MCMC Semi Random-Walker Integration Algorithms prevent the walker from doubling back. These are harder to implement, but result in faster convergence.
5. MCMC Semi Random-Walker Sampling Algorithms:
    - Successive Over-relaxation => The Monte-Carlo version of successive over-relaxation sometimes avoids the random walks, thereby improving upon the Gibbs sampling.
    - Hybrid Monte-Carlo => Hybrid Monte-Carlo algorithms avoid random walks by introducing an auxiliary momentum vector and Hamilton dynamics (where the potential energy is a function of the target density).
        - Hybrid Monte-Carlo vs. Simulated Annealing => Both of these are techniques borrowed from statistical physics and applied in optimal algorithmic walk/search.
    - Targeted Slice Sampling => This uses a modification in the slice sampling technique to avoid random walks.
    - Self-targeting Candidates => Langevin MCMC and other methods (Stramer and Tweedie (1999)) that rely on gradient/Hessian (and other higher order Jacobian descents) of the log posterior avoid random walks by making proposals that are likely to be in the directions of higher probability.
    - Changing Dimensions => Reversible-jump algorithm is a variant on the Metropolis-Hastings algorithm that allows proposals that alter the dimensionality of the variate space. This comes of particular use when performing Gibbs sampling over non-parametric Bayesian models, where the number of mixing components/clusters has to be automatically inferred from the data.
        - Sliced reversible jumps => Once the dimensions are altered, reversing along a variate may be needed. Using this technique in conjunction with sliced/sampling and other methods may quicken the convergence.

Generative and Discriminative Models

Generative Models

1. Definition: Here, the joint distribution between the observations and the hidden states are modeled. Another way of saying this is that the a priori distribution of the hidden states is first generated from the transition probabilities; the conditional distribution of the observations given the states (i.e., the emission probabilities) is then generated.
2. Observation generation Process: This can also be a two-stage process that is generated using Gaussian mixtures. First one Gaussian distribution among the a candidate list is picked; then the probability of a given set of observations is generated.
3. Generative Model Kick-off: The first few steps of a generative model may essentially be seeded off using a discriminant type parameter space initialization to kick start the run.
4. Parameter-Sparse and Model-Rich: Given that there could be times where the generative models work off of limited data, it needs to be more model intensive (and correspondingly parameter space sparse) – thereby lending itself well to be able to model joint state/observation distributions.
5. Measurements in Generative Models: Measurements steer state estimates closer to "true" estimates, but do not entirely determine it, partly because of measurement uncertainty.

Discriminant Models

1. Definition: Here the conditional distribution of the hidden states is generated directly given the observations, rather than modeling the joint state/observation distributions.

2. Advantages of Discriminant Models - #1: Arbitrary features (i.e., functions) of the observations can be modeled by injecting domain specific knowledge targeted at the problem at hand (essentially via the predictor-response transform). Further, straightforward features/correlations/combinations of any observation set may be modeled directly.
3. Advantages of Discriminant Models - #2: The observation features need not be statistically independent of either the states, or among themselves.
4. Disadvantages of Discriminant Models:
    a. Given the limited nature of the model parameters, the types of priors applied on the hidden state models are limited.
    b. Further, these models cannot be used to predict the probability of an arbitrary state/observation sequence.

Examples of Discriminant Approaches

1. Maximum Entropy Markov Models (MEMM): This models the continuous distribution of states from observations using logistic regressions.
2. Linear Chain Conditional Random Field (CRF): This discriminant model uses undirected graphical model of the Markov variables (also referred to called Markov Random Field) as opposed to the directional graphical model used in MEMM-type approaches. Therefore CRF/MRF do not suffer from label bias, increasing the chances for accuracy. However, they are computationally slower.
3. Factorial Hidden Markov Model: Here, a single observation can be conditioned on the corresponding hidden variables of a set of K independent Markov chains (Ghahramani and Jordan (1997)). Thus, if there are N states per each chain, the Viterbi learning algorithm attains a complexity of $\Theta\left(N^{2K}\right)\times T$, where T is the number of observations.

- Exact solution for the factorial HMM may be achieved by using the junction-tree algorithm (with a complexity of $\Theta\left(N^{K+1}\right)\times K\times T$), but approximation techniques such as variational approaches may also be used.

4. K-Level Markov Tree: Here, the state at i depends on $i-1,\ldots,i-K$. Thus, hidden state is not strictly Markovian anymore, but a Markov tree; again, the complexity is $\Theta\left(N^{K}\right)\times T$ for K adjacent states and T total observations.
5. Triplet Markov Models: Here, the process model is augmented to model data specificities. The Theory of Evidence and Triplet Markov Models (Pieczynski (2002), Pieczynski (2007)) are now unified to be able to fuse data in a Markovian context (Bouderan, Monfrini, Pieczynski, and Aissani (2012), Lanchantin and Pieczynski (2005)) and to model non-stationary data (Bouderan, Monfrini, and Pieczynski (2012)).
    - It is also possible that Triplet Markov Models are the ones that deal explicitly with disequilibrium/non-stationary, and/or time evolving Markov models.

Differences Between Generative and Discriminant Models

1. Differences in the Approach Philosophy: Discriminant philosophy appears to be "collect data first, characterize relationships later". Generative is collect – characterize – collect –characterize - …
2. Accommodating State Evolution: The generative models may have to accommodate state evolution as well, in addition to modeling the steady state (although modeling of the evolutionary dynamics may be limited). Discriminant models appear to be primarily for steady-state/constitutive modeling.
3. Unconditional vs. Conditional Worlds: The Regression/Discriminant Models operate in a world where the observations are the only given, therefore they restrict their exploration to that conditional world. Generative Hidden Markov models do not.

Therefore, these models cover a wider state space, i.e., they explore vaster axiomatic informational frameworks.

- The generative models treat the conditional world merely as a constraint they have to accommodate, and the constraint may be hard/soft (depending upon the confidence of the observations). However, from a formulational point of view, it is important to reiterate that both discriminant and generative models are about stationary state characterization, or dynamic equilibrium (as reflected in the state transition probabilities etc:). Within this framework, generative models do accommodate richer modeling paradigms.

4. Unifiability: Given that generational models can build steady state constructs off of other latent drivers, notions such as unified approaches are more naturally handled than they are in discriminant models.
5. State Structure Estimation: Estimation of both the emission probabilities and the state transition probabilities may be possible in both frameworks. However, in discriminant/regression frameworks, these are characterized "as is", whereas in generative frameworks the parameter inference is part of a bigger story.
    - Discriminant models typically simply regress the latent state against the observation measures. Generative models infer the hidden state from the observation measures as well as past history. Further, in the generative models, the regression is to primarily establish the constitutive state relationship between an inelastic state predictor and an elastic state quantification metric.
6. Comparison in the Context of Supervised vs. Unsupervised: Usage of discriminant algorithms in the learning phase is highly useful, as they uncover the relationships among the predictors/responses across the feature contexts. Once these relationships are uncovered, however, the stage is set for the synthesis of a broader set of rules to construct the generative models that generate the joint distributions.

Supervised Learning

Introduction

1. Supervised Learning as a Calibration Exercise: One very valid view is that any form of calibration is supervised learning in that it deciphers the "optimal learning algorithm" (Mohri, Rostamizadeh, and Talwalkar (2003)). Therefore, as such considerations such as bias/variance trade-offs will apply (they do, for example, for all regressions – logistic or real-valued).
2. Human-Animal Concept Learning: Psychological learning (which may or may not be concept learning) is the best example of supervised learning (Supervised Learning (Wiki)). This may still be a giant hash-map, in which case the "indexer" algorithm is the calibration routine!
3. Challenges with Hand-Labeling: How do you do that for a humongous input data set? Something like Penn Tree-bank is painful handcrafted, but how do we do this for finance data? Are there effective, general-purpose feature extractors (thus, this in itself will become the supervised set in part!)?
4. Effectiveness: Vast sets of inputs data may effectively, through classification/calibration/other supervised learning paradigms, be "digested" onto learned fragments - specific to practice domains at hand – used for future predictions.

Supervised Learning Practice Steps

1. Step #1 => Training Example Types: Physical/Cognition level identification of the training data types could be heterogeneous (i.e., combination of elaborate data and/or single characterizer) using a weight-based state.
2. Step #2 => Gathering Training Set: The training set should be a representation of the real-world, as well contain the cross-validatable groups. Many require some

automated algorithm for hand-labeling (esp. if the data is humongous). In all, this stage pairs the cognition/physical input groups with their label names.

3. Stage #3 => Feature Representation: Convert the set of input data into a machine-represented feature vector (again heterogeneous). Balance must be achieved between too grainy and too broad, i.e., balance between curse of dimensionality and inaccuracy.
4. Step #4 => Learning Algorithm Choice: Determination of the learning algorithm/physical model also requires making decisions on the algorithm design choice, parameters, complexity, flexibility, and performance.
5. Step #5 => Run setup: Group the training inputs into cross validation/GCV subsets. This helps:
   - Identify the base parameter set
   - Bayesian parameter distribution.
   - If the input is rich enough, it also helps determine the Bayesian hyper-parameters.

   Attention needs to be paid to over-fitting and inaccuracy at this stage as well.
6. Step #6 => Execution/Post-Execution: This stage is also called active learning. After the run, you may need to re-adjust the calibrated state to improve after additional observations (say using Kalman Filter). Further, this improvement may take the form of enhancements to parameters, Bayesian distributions, and/or hyper-parameters.

Challenges with the Supervised Learning Practice

1. Bias-Variance Trade-off: Typically lowering the bias with regard to one training group renders higher variance across the other variance groups. Optimizing for the variance across all groups inevitably ends up increasing the bias. Therefore, calibration algorithms must optimize for a combination of both (Geman, Bienenstock, and Doursat (1992), James (2003)).
2. Function Complexity vs. Sufficiency of Training Data:
   - Complicated function + lots of data => Need optimized bias/variance algorithms.
   - More complex function + less training data => Need lower bias functions.

- Less Complex function => Use higher bias, and lower variance functions, no matter what amount of training data you have.

3. Dimensionality of the Input: The challenge is to avoid over-fitting in the case of high dimension inputs. Choices are:
   - Reduce the input features to a lower feature dimensionality space.
   - Hand eliminate instances of lower relevance
   - Since this is in essence similar to measurement noise, but effectively introduced due to ignoring input dimensions, this is also called deterministic noise. Techniques described in Brodely and Friedl (1999) and Smith and Martinez (2011) help identify noisy training examples and remove them.
4. Stochastic Noise in Training Output: Need to distinguish between the supervisory signal (the real response class), the supervisory target (the biased target output value), and the supervisory noise (the measurement error between the supervisory signal and the measured output). If you can characterize the nature of measurement error, use that to reduce variance (and therefore increase the bias!).
5. Training Data Heterogeneity: Many algorithms work on only one type of data, so the heterogeneous training data need to be transformed into the appropriate type (e.g., onto real-valued type for the distance classifier, logistic regression, etc:). Some techniques such as decision trees automatically handle heterogeneous training data.
6. Data Redundancy: Redundant (e.g., highly correlated) inputs result in poor performance on some algorithms (e.g., linear/logistic regressions, distance methods). Regularization helps transform the input.
7. Modeling Interactions and non-linearities: Non-interacting linear methods work well under the following situations:
   - Contributions from the individual factors/features are independent.
   - The features may be dependent, but may be transformed using a straight non-linear transformation, in which case the transformation must be specified to avoid deterministic noise.

   If the above situations do not apply, algorithms such as neural nets and decision trees work better, as they uncover the relationships more effectively.

Formulation

1. Nomenclature: The Training Set: $\{(x_1, y_1),...,(x_N, y_N)\} \Rightarrow \{X, Y\}$. The learning algorithm seeks a function $g : X \to Y$ where g is part of the space of hypothesis functions G. Alternately, define $f : X, Y \to R$ where f is part of the space of scoring functions F. Then g is the function returning the value of y that gives the highest score, i.e., $g(x) = \arg\max_y f(x, y)$.
2. Choice of g / f: Typically g is chosen from the discriminant family that uses a conditional probability model $g(x) = P(y \mid x)$ (say linear/logistic regression), whereas f is chosen from the generative family that is modeled using the joint probability $f(x) = P(x, y)$ (e.g., naïve Bayes', LDA).
3. Choice of the Loss Function: Assume that $\{X, Y\}$ is an i.i.d. sample. Fitness of data is estimated by minimizing a specified loss function L defined from $L : Y \times Y \to R^{\geq 0}$.
4. Approaches for Loss Function Minimization from Data:
   - Empirical Loss Risk Minimization seeks the function that best fits the data, i.e., the low bias solution.
   - Structural Loss Risk Minimization seeks the function that controls the bias/variance trade-off (Vapnik (2000)).
5. Empirical Loss Minimization: Define $R_{emp} = \frac{1}{N} \sum_i L(y_i, g(x_i))$. The minimization of R_{emp} chooses the corresponding g. If g is the conditional probability distribution, and L is the negative log likelihood, i.e., $L(y, g(x)) = -\log P(y \mid x)$, then this corresponds to an MLE approach.
   - Drawbacks => When G contains many candidate functions, or the training set is not sufficiently large, empirical loss minimization leads to high variance and poor generalization, i.e., pure training data memorization/over-fitting.

6. Structural Loss Minimization: This aims to prevent over-fitting by incorporating a regularization penalty $C(g)$. For instance, when g may be expressed as a combination of linear basis functions, i.e., $g(x) = \sum_{i=0}^{n-1} \beta_i h(x)$, the penalty may assume the following forms: a) The L_2 or Euclidean Norm => $\sum_{i=0}^{n-1} \beta_i^2$; b) The L_1 Norm => $\sum_{i=0}^{n-1} |\beta_i|$; c) The L_0 Norm => The number of non-zero β_i's.
   - Formulation => The challenge is to get the g that minimizes $J(g) = R_{emp}(g) + \lambda C(g)$, where λ is the bias-variance trade-off tuner ($\lambda = 0$ corresponds to pure empirical risk with low bias and high variance; $\lambda \to \infty$ corresponds to pure structural risk minimizer with high bias and low variance).
   - Bayesian Interpretation of Structural Risk Minimization => $C(g)$ may be interpreted as $-\log P(g)$ where $P(g)$ is the joint probability, and the corresponding $J(g)$ may be interpreted as the posterior probability of g. Thus, empirical risk minimizer is frequentist, whereas structural risk minimizer is Bayesian.
7. Generative Training: This corresponds to the special case $f(x, y) = P(x, y)$, and $L \Rightarrow \sum_i \log P(x_i, y_i)$. Often these are simpler and computationally easier.

Statistical Classification

1. Classification vs. Clustering: Classification is the term used in the supervised learning context, whereas clustering is classification in the unsupervised learning context. Since there is no absolute context to go by in the case of unsupervised learning context, all the unsupervised clustering algorithms only uncover "similar" instances using similarity metric (e.g., distance metric).
2. Machine Learning as a Dimension Reduction Process: In one sense machine learning is used primarily to reduce the observation space to the "lower dimension" parameter/rules space. However, it may proceed by employing additional dimension reduction approaches internally, e.g., reduce the class features from a higher to a lower dimension space.
3. Binomial Frequentist Classification: Fisher's (Fisher (1936), Fisher (1938), Gnanadesikan (1977)) work depended on the assumption that the output probabilities were multivariate on the input features, and extracted the classification/clustering rules.
4. Multinomial Frequentist Classification: Fisher's approach was extended to multinomial output states (Gnanadesikan (1977), Rao (1952), Har-Peled, Roth, and Zimak (2003)), and further extensions to non-linear rules were done in Anderson (1958), essentially using classifications based off of the adjusted Mahalanobis distance.
5. Multinomial Bayesian Classification: This incorporates distributions of feature populations (Binder (1978)). Approximations to the clustering rules were introduced in Binder (1981) to enable tractability. MCMC, ABC, and later computational developments reduced the need for these approximations.

Linear Discriminant Analysis

Introduction

1. Definition: LDA and Fisher Linear Discriminant are related statistical methods that find a linear combination of features to identify two/more classes of objects/events underlying the data (Linear Discriminant Analysis (Wiki)).
2. Comparison LDA and related Methods:
    - LDA => Real-valued Predictors and Categorical Responses (Fisher (1936), McLachlan (2004)).
    - ANOVA => Categorical Predictors and Real-valued Responses (Wetcher-Hendricks (2011)).
    - Linear Regression => Real-valued Predictors and Real-valued Responses (Fisher (1936), McLachlan (2004)).
    - Logistic Regression => Real-valued Predictors and Categorical Responses (Fisher (1936), McLachlan (2004)).
    - Discriminant Correspondence Analysis => Categorical Predictors and Categorical Responses (Abdi (2007), Perriere and Thioulouse (2003)).
3. LDA vs. Component/Factor Analysis:
    - LDA explicitly models class differences, whereas PCA does not (Martinez and Kak (2001)).
    - Factor Analysis and PCA => Here there is no real distinction between predictors and responses, since that is not the primary purpose.

Setup and Formulation

1. Core Assumption for the 2 Class Formulation: The conditional probability of realization of each class is multinomial Gaussian on the predictors (Venables and

Ripley (2002)): $P(\vec{x} \mid y=0) = \exp\left[-\frac{(\vec{x} - \vec{\mu}_{y=0})^2}{[\sigma^2]_{y=0}} \right]$, and similar expression for $P(\vec{x} \mid y=1)$. Here $\vec{x}$ is the feature vector of the predictor ordinates, $\vec{\mu}_{y=0,1}$ is the mean of the feature vector for each class, and $[\sigma^2]_{y=0,1}$ is the covariance of the feature vector for each class.

2. LDA Likelihood Ratios: $-\log\left\{ \frac{P(\vec{x}_p \mid y=0)}{P(\vec{x}_p \mid y=1)} \right\} = -\left\{ \frac{(\vec{x}_p - \vec{\mu}_{y=0})^2}{2[\sigma^2]_{y=0}} - \frac{(\vec{x}_p - \vec{\mu}_{y=1})^2}{2[\sigma^2]_{y=1}} \right\}$.

   Further, LDA assumes that $[\sigma^2]_{y=0} = [\sigma^2]_{y=1} = [\sigma^2]$, i.e., the feature set variance is homoscedastic.

3. Threshold-based Homoscedasticity: By applying a) the homoscedasticity assumption, b) the full rank assumption, and c) the chance that the points belonging to class 0 result from exceeding a threshold, we get the probability of belonging to class 0 as $\frac{\vec{x}_p.(\vec{\mu}_{y=0} - \vec{\mu}_{y=1})}{[\sigma^2]} > c$, where c is the threshold (or its transformation).

4. Hyper-plane and Hyper-surface View: The above is a simple linear relation among the feature set co-ordinates, and as such identifies the hyper-plane dividing the classes. Removing the homoscedasticity assumption sets the divider to be a hyper-surface.

Fisher's Linear Discriminant

1. Definition: Fisher (Fisher (1936)) defined the separation between the classes 0 and 0 to be the ratio of the variance between the classes to the variance within the classes at the vector plane $\vec{w}$: $S = \frac{\sigma^2_{Between}}{\sigma^2_{Within}} = \frac{\sum_i [\vec{w}_i (\vec{\mu}_1 - \vec{\mu}_0)]^2}{\sum_i \left[\vec{w}_i^T \left({\sigma_0}^2 + {\sigma_1}^2 \right) \right] \vec{w}_i}$. This may be viewed as the classifier's signal-to-noise ratio.

2. Rao Multi-Class LDA: This is also referred to as Canonical Discriminant Analysis for multiple classes. Rao (Rao (1948)) generalized Fisher's LDA by defining the clas variability using the sample covariance of the class means as

   $\Xi_b = \frac{1}{C}\sum_{i=1}^{C}(\mu_i - \mu)(\mu_i - \mu)^T$. The class separation is then defined as $S = \frac{\vec{w}^T \Xi_b \vec{w}}{\vec{w}^T \Xi \vec{w}}$.

   Thus, when $\vec{w}$ is an eigenvector of $\Xi_b^{-1}\Xi$, the corresponding separation produces the eigen-value.

3. Cross Validation Type Classification: For multiple classes, they may be partitioned, and the LDA applied individually to each partition. This provides either the C classes, or the $\frac{C(C-1)}{2}$ re-combined class set for the final classification.

4. Essence of the Fisher/Rao Approach: This approach is much more geometric/ loose cognitive appeal oriented rather than the full-fledged LDA rigor developed earlier. Of course, by applying the appropriate tweaks/resets in Fisher's formulation, the LDA approach formulation may be recovered.

5. Sample Size Impact on Fisher/LDA: Specific approaches are needed for dealing with small sample sizes. These include LDA on a reduced sub-space projection (Yu and Yang (2001)), or regularized discriminant analysis (Friedman (1989)) – the latter uses a shrinkage estimator of the covariance matrix, i.e., $\Xi_{Shrunk} = (1-\lambda)\Xi + \lambda I$ where λ is the shrinkage intensity/regularization parameter (Ahdesmaki and Strimmer (2010)).

6. Extension of LDA to multiple Classes: The "one against the rest" approach may be applied class-by-class. This implies that the threshold of the original LDA partition shifts with each cross run. This also provides a natural way to accommodate multinomial ordinal classification.

Quadratic Discriminant Analysis

1. QDA/Quadratic Classifier Definition: As noted earlier, heteroscedasticity of predictor variances across classes implies that the partition surface is quadratic/conic (Quadratic Discriminant Analysis (Wiki)).
2. Transformed Basis Predictors: The predictors may be combined to produce a bigger set of convolved basis function predictors of the same order, i.e, the set of $\{x_1, x_2, x_3\}$ can be convolved to $\{x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3\}$ as the fresh basis set for order 2.
3. Circular Special Case of QDA: This corresponds to introducing only the quadratics terms $\{x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3\}$ without the cross terms $\{x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3\}$. This has proven to be the optimal compromise between extending the classifier's representation power and controlling the risk of over-fitting (the Vapnik-Chervonenkis dimension) (Cover (1965), Ridella, Rovetta, and Zunino (1997)).

Logistic Regression

Introduction

1. Definition: Probabilitis classification model used for predicting the outcome of a categorical dependent variable (e.g., class label) based on one/more of the predictor variable features (Logistic Regression (Wiki), Bishop (2006), Bhandari and Joensson (2008)).
2. Popular Applications: TRISS (Boyd, Tolson, and Copes (1987)), Voting Behavior (Harrell (2001)), Survival Analysis for Process/Product/System (Strano and Colosimo (2006), Palei and Das (2009)).

Formulation

1. Setup using a Logit Link Function: Probability of the observed instance characterized by the feature vector $\vec{x}$ belonging to the TRUE (i.e., 1) class is $\pi(\vec{x}) = \frac{\exp\left[\sum_{i=0}^{n-1} \beta_i f_i(\vec{x})\right]}{\exp\left[\sum_{i=0}^{n-1} \beta_i f_i(\vec{x})\right] + 1}$. The logistic function (also called the link function) (Hosmer and Lemeshow (2000)) is defined as $g(\vec{x}) = \log \frac{\pi(\vec{x})}{1-\pi(\vec{x})} = \sum_{i=0}^{n-1} \beta_i f_i(\vec{x})$.
2. MLE based Estimation: Since the estimation here is going to be non-linear (i.e., not closed form/quasi-analytic in the coefficients), iterative non-linear methods (such as iteratively re-weighted least squares (IRLS), quasi-Newton (e.g., L-BGFS) need to be used (Menard (2002)).
3. Challenges with MLE:

- Large number of predictors to cases ratio => Rule of thumb is that logistic regression models require a minimum of 10 events per variable (Peduzzi, Concato, Kemper, Holford, and Feinstein (1996)).
- Multi-collinearity among the predictors => In this case the coefficients remain unbiased, but the errors increase, making the convergence less likely. This situation may be identified by regressing the predictors among each other.
- Sparseness in data => This refers to a large proportion of empty cells with zero count (log of zero is undefined). To remedy this, categories may be collapsed in a meaningful way, or a harmless constant may be added to all the cells.
- Complete Separation => In this case, the predictors deterministically predict 1/0 without stochasticity. Such a situation may indicate a data formulation error.

4. Minimum Chi-squared Estimator for Grouped Data: In grouped data, you use the fraction of 1/0 per each group of the feature vector, and the minimum chi-squared involves using weighted least squares to estimate a linear model of the logit (Greene (2003)).

Goodness of Fit

1. Deviance – Definition: Since sum of squares penalty does not work well in the logistic regression setup, deviance (D) as defined below (Cohen, Cohen, West, and Aiken (2002)) is used: $D = -2\ln\left[\frac{Likelihood_{FittedModel}}{Likelihood_{SaturatedModel}}\right]$. Here the "saturated model" is the model with a theoretically perfect fit. This is also referred to as the likelihood-ratio test.
2. NULL Model vs. Fitted Model Deviance:
   - NULL Model => Model with no predictors (only intercept).
   - Fitted Model => Model with one/more predictors.
   - $D_{NULL} = -2\ln\left[\frac{Likelihood_{NULLModel}}{Likelihood_{SaturatedModel}}\right]$.

- D_{Fitted} is the same as before. The NULL model provides a baseline on top of which to evaluate a valid predictor/response model.
- $D_{Fitted} - D_{NULL} = -2\ln\left[\frac{Likelihood_{FittedModel}}{Likelihood_{NULLModel}}\right]$.

3. Chi Squared Testing: In linear regression, non-significant chi-square values indicate little unexplained variance, whereas significant chi-square values indicate significant unexplained variance. Likewise, in logistic regression, significantly smaller fitted deviance than NULL deviance indicates significantly improved fit (this is analogous to the F-test used in linear regression (Cohen, Cohen, West, and Aiken (2002))).
4. Pseudo-R^2 Metrics of Goodness of Fit:
   - $R_L^2 = \frac{D_{NULL} - D_{MODEL}}{D_{NULL}}$. This is also called the likelihood ratio R^2, and is the most commonly used goodness of fit metric, but the drawback is that it is not monotonic with the logit.
   - Cox/Snell R^2 is proportional to the logit, but reaches a maximum at 0.75 (which occurs when the maximum variance is 0.25). Nagelkerke R^2 removes this limitation (Cohen, Cohen, West, and Aiken (2002)), but exhibits wider departure with R_L^2.
   - Since by construction logistic regression is heteroscedastic, all these goodness of fit metrics are called pseudo-R^2, as pure R^2 is of questionable value in this.
5. Hosmer-Lemeshow Test: This method uses a test statistic that asymptotically follows a χ^2 distribution to assess whether or not the observed event rates match the exected event rates in sub-groups of the model population.
6. Significance of Coefficients: The $\frac{d\,\mathrm{Response}}{dCoefficient}$ sensitivity metric of linear regression is analogous to the $\frac{dLogit}{dCoefficient}$ sensitivity metric of logistic regression (Cohen, Cohen, West, and Aiken (2002)). In addition, the significance attributed to the individual predictors is assessed using the likelihood ratio test (discussed above) or the Wald statistic (discussed below).

- The likelihood ratio deviance test for goodness of fit discussed above may also be applied for the basis functions one-at-a-time to incrementally assess the impact in order. The validity of these so-called hierarchical/stepwise assessment is examined in Hosmer and Lemeshow (2000), Menard (2002), and Cohen, Cohen, West, and Aiken (2002).

7. Predictor Significance Assessment using Wald Statistic: The Wald Statistic for the coefficient β_j is defined as $W_j = \frac{\beta_j^2}{\varepsilon_j^2}$ where ε_j is the error estimate on β_j. The Wald Statistic is an asymptotic χ^2 distribution. Unlike the deviance likelihood ratio, the Wald Statistic is exposed in statistical packages such as SPSS, SAS, R, etc: However the limitations with this are:
   - When the regression coefficients are large, the corresponding error for a given Wald Statistic also tend to be large, thereby increasing the chances of Type-II errors;
   - Wald Statistic tends to be biased when the data is sparse (Menard (2002), Cohen, Cohen, West, and Aiken (2002)).

Mathematical Setup

1. Base Mathematical Setup: The probability of the given categorical outcome is the consequence of the Bernoulli process conditioned on the realizations of the predictor ordinates and the category under consideration, i.e., $Y_i \mid \{x_{ij}\}_{i=1}^{m} \approx Bernoulli(p_i)$, or, alternatively, $E\left[Y_i \mid \{x_{ij}\}_{i=1}^{m}\right] = p_i$.

2. As a Generalized Linear Model: Here

$$\log it\left(E\left[Y_i \mid \{x_{ij}\}_{i=1}^{m}\right]\right) = \log it(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \sum_{l=0}^{n-1}\beta_l f_l(\vec{x}), \text{ thus}$$

$$E\left[Y_i \mid \{x_{ij}\}_{i=1}^{m}\right] = p_i = \log it^{-1}\left(\sum_{l=0}^{n-1}\beta_l f_l(\vec{x})\right) = \frac{1}{1+\exp\left(\sum_{l=0}^{n-1}\beta_l f_l(\vec{x})\right)}.$$

3. As a Latent Variable Model: $Y_i^* = \sum_{l=0}^{n-1}\beta_l f_l(\vec{x}) + \varepsilon$ where $\varepsilon \approx Logistic(0,1)$, and $Y_i = 1$ if $Y_i^* > 0$ (i.e., $\varepsilon < -\sum_{l=0}^{n-1}\beta_l f_l(\vec{x})$) and $Y_i^* < 0$ otherwise. This formulation can be shown to be identical to both the base setup as well as the GLM setup. Further the logistic function is symmetric with a peak around the mean just like Gaussian, but accommodates fatter tails, so it is beter for calibrations.

4. As a 2-way Latent Variable Model: Here we have $Y_{0,i}^* = \sum_{l=0}^{n-1}\beta_{0,l} f_l(\vec{x}) + \varepsilon_0$ and $Y_{1,i}^* = \sum_{l=0}^{n-1}\beta_{1,l} f_l(\vec{x}) + \varepsilon_1$ where $\varepsilon_0 \approx EV_0(0,1)$, $\varepsilon_1 \approx EV_1(0,1)$, and EV is a standard type-1 extreme value distribution with $\varepsilon_1 - \varepsilon_0 \approx Logistic(0,1)$. Then the postulate $Y_{0i} = 1$ if $Y_{0i}^* > Y_{1i}^*$ and 0 otherwise can be shown to be equivalent to the 1-way latent variable formulation above. As may be seen, the advantage of the 2-way formulation is that it can be extended to multi-way/multi-class.

5. Log-Linear Model: For a class C model, we may set:

   - $\ln \Pr(Y_i = C) = \sum_{l=0}^{n-1}\beta_{c,l} f_l(\vec{x}) - \ln Z$
   - Summing up across all classes, we get $\Pr(Y_i = C) = \frac{\exp\left(\sum_{l=0}^{n-1}\beta_{c,l} f_l(\vec{x})\right)}{\sum_i \exp\left(\sum_{l=0}^{n-1}\beta_{c,l} f_l(\vec{x})\right)}$. Thus, this produces a normalized logistic multinomial class setup.

6. As a Single-Layer Perceptron: Also called a single-layer ANN, here the Bernoulli probability p_i may be specified as $p_i = \frac{1}{1+\exp\left(-\sum_{l=0}^{n-1}\beta_{c,l} f_l(\vec{x})\right)}$. This computes a single layer of continuous output in place of a step function. The derivative of p_i with respect to $\vec{x}$ is computed from the general form $y = \frac{1}{1+\exp\left(-f\left(\vec{X}\right)\right)}$. This choice makes this ANN identical to logistic regression, and thus maybe used in back-propagation and easy derivative extraction from $\frac{dy}{d\vec{X}} = y(1-y)\frac{df}{d\vec{X}}$.
7. As Bernoulli-distributed Binomial Data: Each $Y_i \approx Binomial(n_i, p_i)$, where Y_i is the number of successes observed. Then

$$p_i = E\left[\frac{Y_i}{n_i} \mid \vec{X}_i\right] \Rightarrow \log it\left(E\left[\frac{Y_i}{n_i} \mid \vec{X}_i\right]\right) = \log it(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \sum_{l=0}^{n-1}\beta_l f_l(\vec{x}).$$

Equivalently, $\Pr ob\left[Y_i = y_i \mid \vec{X}_i\right] = C_{y_i}^{n_i} p_i^{y_i}(1-p_i)^{n_i-y_i}$ where p_i is given from above.

Bayesian Logistic Regression

1. Philosophy: Here the prior distributions are placed on the regression coefficients, which can be problematic for the logistic distribution, as conjugates for them is hard to extract.
2. Solution Approaches:
   - Perform a MAP point estimate in place of a full-fledged MAP.
   - Employ the MCMC/Metropolis-Hastings variants with the heavy-tailed multi-variate candidate distributions found by matching the mode/curvature at the normal approximation to the posterior – and then use the student-t distribution with a low degree of freedom (Bolstad (2010)). Conjugates – further, this approach may be used to sample arbitrary distributions too.

- Use a latent variable model and approximate the logistic and/or extreme value distributions using a more tractable distribution, e.e., student-t or even a mixture of normal distribution (or even a probit distribution).
- Use a Laplace distribution in place of the posterior distribution (Bishop (2006)). This approximates the posterior with a Gaussian (which is not too good), but the posterior mean and variance may be estimated, and schemes such as variational Bayes (Bishop (2006)) may also be used.

Logistic Regression Extensions

1. Multinomial Logit/Polytomous Regression: Multi-way categorical dependence variables with unordered values (also called "classification").
2. Ordered Logit: Handles Ordinal Dependent (i.e., ordered) values
3. Mixed Logit: Allows for Correlations among the choices of the dependent variables.

Model Suitability Tests with Cross Validation

1. Base Methodology: Use "one-against-the-rest" selected cross-validation (Myers and Forgy (1963), Mark and Goldberg (2001)). The cross-validation sample is referred to as the "holdout".
2. Suitability Estimation for Binary Logistic Regression: Cross-examine and validate actual and predicted values. Use the following definitions (TRUE/FALSE is defined with respect to the Holdout):
   - True Negative (TN) => Prediction – FALSE and Holdout - FALSE
   - False Negative (FN) => Prediction – FALSE and Holdout - TRUE
   - False Positive (FP) => Prediction – TRUE and Holdout - FALSE
   - True Positive (TP) => Prediction – TRUE and Holdout – TRUE

- Accuracy => Fraction of Observations with Correct Prediction = $\frac{TP + TN}{TP + FP + TN + FN}$
- Precision => Fraction of Correctly Predicted Positives = $\frac{TP}{TP + FP}$
- Negative Predictive Value => Fraction of Correctly Predicted Negatives = $\frac{TN}{TN + FN}$
- Recall/Sensitivity => Fraction of TRUE Holdout's Correctly Predicted = $\frac{TP}{TP + FN}$
- Specificity => Fraction of FALSE Holdout's Correctly Predicted = $\frac{TN}{TN + FP}$

Multinomial Logistic Regression

Introduction

1. Logistic Regression Calibration Data Sufficiency: Given that the logit itself is a continuous function of the predictor, you need a sufficient number of observations per each predictor ordinate instance.This can happen if you specify a bucket (infinitesimal or otherwise) and proxy the predictor ordinate instance to a central measure, and the reponse to the logit of the probability. Of course, all other observations regarding data sufficiency for this and notes regarding non-linearity of the calibrations still hold.
2. Multinomial Logistic Regression Definition: This is a losgistic regression model allowing more than two categorical outcomes (Greene (1993)). This is also referred to as softmax or multinomial logit regression.
3. Normalization Constraint for Multinomial Logistic Regression: This would imply that the logit of the last class is determined ENTIRELY from the remaining predictor<->class representations; therefore, no exogenous spline based basis functional specification is possible for this.
4. Ordered Multinomial Logistic Regression: Roving/pregressing thresholds are a natural fit for ordered multinomial logits. LDA supports this by construction; vanilla multinomial logistic regression may be extended relatively easily to do this.

Setup and Formulation

1. Assumption of IIA (Independence of Irrelavant Alternatives): This assumption states that the odds of preferring one class over another does not depend on the presence of other “irrelevant” alternatives. If violation of this occurs, other models such as nested logit or multinomial probit may be used.

2. Independent Binary Regressors: Consider a conditional universe where the outcome subset is (j,k), where k is the reference/pivot class. The logit P_j is given from $\ln \frac{\Pr(y=j \mid \vec{x})}{\Pr(y=k \mid \vec{x})} = \sum_{i=0}^{n-1} \beta_{ij} f_i(\vec{x})$. Thus, this introduces separate sets of regression coefficients, one for each j.

   - Exponentiating the above, we get $\Pr(y=j \mid \vec{x}) = \Pr(y=k \mid \vec{x}) = \exp\left[\sum_{i=0}^{n-1} \beta_{ij} f_i(\vec{x})\right]$.
   - Summing the probabilities to unity, i.e., $\sum_{j=1}^{K} \Pr(y=j \mid \vec{x}) = 1$, we get $\Pr(y=k \mid \vec{x}) = \frac{1}{1 + \sum_{j \neq k} \exp\left[\sum_{i=0}^{n-1} \beta_{ij} f_i(\vec{x})\right]}$.
   - The fact that we used multiple regression runs off of a single pivot reveals the reliance on the assumption of IIA.

3. Log Linear Model: Setup the class probability as $\ln[\Pr(y=j \mid \vec{x})] = \sum_{i=0}^{n-1} \beta_{ij} f_i(\vec{x}) - \ln Z$. Summing the probabilities again yields $Z = \sum_j \exp\left[\sum_{i=0}^{n-1} \beta_{ij} f_i(\vec{x})\right]$ where Z is referred to as the partition function. Z is not a function of the observation set – it is a function of $\vec{x}$ and β_{ij}.

4. Softmax Function: $Soft\max\left(k, \{x_i\}_{i=0}^{n-1}\right) = \frac{\exp(x_k)}{\sum_{i=0}^{n-1} \exp(x_i)}$. The impact of exponentiating x_i is to exaggerate the difference between x_i and the others. Thus, $Soft\max\left(k, \{x_i\}_{i=0}^{n-1}\right) \to 0$ if $x_i << \max\left(\{x_i\}_{i=0}^{n-1}\right)$, and $Soft\max\left(k, \{x_i\}_{i=0}^{n-1}\right) \to 1$ if $x_i \sim \max\left(\{x_i\}_{i=0}^{n-1}\right)$. Thus, softmax can be used to construct a weighted average that behaves as a smooth function (differentiable easily, etc:) as an approximation to the non-smooth function $\max\left(\{x_i\}_{i=0}^{n-1}\right)$, i.e., $f\left(\{x_i\}_{i=0}^{n-1}\right) = \max\left(\{x_i\}_{i=0}^{n-1}\right) \approx \sum_{i=0}^{n-1} x_i Soft\max\left(k, \{x_i\}_{i=0}^{n-1}\right)$.

5. Latent Variable Model: For each data point p and outcome j, there exists a continuous latent variable distributed as $Y_{p,j}^* = \sum_{i=0}^{n-1} \beta_{ij} f_i(\vec{x}) + \varepsilon_j$, where $\varepsilon_j \sim EV_1(0,1)$, s atandard type-1 extreme value distribution. $Y_{p,j}^*$ may be thought of as the utility associated with the data point p choosing an outcome j, where some randomness is used to account for the unmodeled factors.

- Actual Outcome Probability => The actual outcome j occurs only when $Y_{p,j}^* > Y_{p,k}^*$ for all $j \neq k$. Thus,

$$\Pr(Outcome \to j) = \Pr(Y_{p,j}^* > Y_{p,k}^* \forall k \neq j) = \Pr\left(\sum_{i=0}^{n-1} (\beta_{ij} - \beta_{ik}) f_i(\vec{x}) > \varepsilon_j - \varepsilon_k \forall k \neq j \right)$$

.

- Nature of $\varepsilon_j - \varepsilon_k$ => Since both ε_j and ε_k are $EV_1(0,1)$, by definition $\varepsilon_j - \varepsilon_k$ then becomes $\varepsilon_j - \varepsilon_k \sim Logistic(0,1)$. In fact, the generalized $\varepsilon_j - \varepsilon_k$ may also be shown to be scale-separable, i.e., $\varepsilon_j - \varepsilon_k \sim bLogistic(0,1) \Rightarrow Logistic(0,b)$.

Non-Parameteric Classification Algorithms

Decision Trees and Decision Lists

1. Decision Tree: A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility (Decision tree (Wiki), Yuan and Shaw (1995)).
2. Drawback of Decision Trees: General inability to handle multi-valued, complex, and uncertain paths; in particular, for data including categorical variables with different number of levels, information gain in decision trees are biased in favor of those attributes with more levels (Deng, Runger, and Tuv (2011)).
3. Decision Lists: Decision lists are representations for Boolean functions. They are more expressive than conjunctions/disjunctions, but are typically less expressive than the disjunctive/conjunctive normal forms (Decision List (Wiki), Rivest (1987)).
    - In particular, decision lists are useful for efficient attribute learning (Klivans and Servedio (2006)).

Variable Bandwidth Kernel Density Estimation

1. Definition: Variable bandwidth kernel density estimators are a form of kernel density estimator in which the size of the kernels used in the estimate are varied dependeing upon on either the location of the samples or the location of the test point (Variable Kernel Density Estimation (Wiki), Terrell and Scott (1992)).
2. Setup: Given a set of samples $\{\vec{x}_i\}$ we aim to estimate the point density $P(\vec{x})$ at the test point $\vec{x}$, i.e., $P(\vec{x}) \approx \frac{W}{nh^D}$ where $W = \sum_{i=0}^{n-1} w_i$ and $w_i = K\left[\frac{\vec{x} - \vec{x}_i}{h}\right]$. Here n is the number of samples, h is the width, and D is the number of dimensions in $\vec{x}$. If K is chosen to be linear in $\vec{x}$, it may be imagined as a simple, linear band-pass filter.

3. Balloon Estimation: In Balloon Estimation, the kernel width is varied to make it proportional to the density at the test point $\vec{x}$, i.e., $h \approx \frac{g}{\left[nP(\vec{x})\right]^{1/D}}$, where g is a constant. This results in a constant W across samples, and produces a generalization of kNN - i.e., a uniform kernel function returns the unbiased kNN evaluator (Mills (2011)).
4. Pointwise Estimation: Here, the kernel width is altered with respect to the location (Terrell and Scott (1992)). For multivariate distributions, h can be varied on the shape - not just the size.
5. Variable Kernel Density in Statistical Classification: In this approach, the probability distribution functions of each class is computed separately with its own kernel, custom bandwidth, etc. (Taylor (1997)). In an alternate approach, the sum of each class is divided across the sample variate space, i.e., $P(j,\vec{x}) \approx \frac{1}{n}\sum_{i=0,c_i=j}^{n-1} w_i$ where $c_i = j$ is the class of the ith sample.
    - Decision Boundary => Say you are classifying for classes 1 and 2 using any smooth kernel (say Gaussian) – this makes sure that the estimates of joint or conditional probabilities both continuous and differentiable. The border is searched by zderoing the difference between the conditional probability as: $R(\vec{x}) = \frac{P(2|\vec{x}) - P(1|\vec{x})}{P(2|\vec{x}) + P(1|\vec{x})}$. Any 1-D root finding algorithm for $R(\vec{x}) = 0$ establishes the border straddling samples.
    - The Classification Work-out =>
        - Spot the point index closest to $\vec{x}$ as $j = \underset{i}{\arg\min} \left|\vec{b}_i - \vec{x}\right|$
        - The gradient shift at j is $p = \left(\vec{x} - \vec{b}_j\right)\nabla_{\vec{x}} R(\vec{x})_{\vec{x}=\vec{b}_j}$
        - The estimated class then would be $c = \frac{1}{2}\left[3 + \frac{p}{|p|}\right]$.

k-Nearest Neighbors Algorithm

1. Definition: *kNN* is a non-parametric method for classification and regression, which predicts object values and class memberships based on the k closes training examples in the feature space (K-Nearest Neighbor (Wiki), Altman (1992)).
2. *kNN* Motivation: *kNN* is a type of instance-based learning or lazy learning where the function is only approximated locally (e.g., using majority voting) and all computation is deferred until classification.
3. *kNN* Regression: Here the property value of the object is the average of its k nearest neighbors, perhaps with their corresponding weights applied (i.e., $\frac{1}{d}$ inverse distance metric).
4. *kNN* Implict Decision Boundary: Since *kNN* rules implicitly compute the decision boundary, it is sensitive to the local structure of the data. Further since it is possible to compute the decision boundary explicitly and efficiently (SVM techniques do it in a non-probabilistic sense), the classification computational complexity is a function of the decision boundary computation complexity (Bremner, Demaine, Erickson, Iacono, Langerman, Morin, and Touissant (2005)).
5. *kNN* Distance Metric: Euclidean distance metric is commonly employed for real-valued continuous feature vector classifications. For discrete variables, alternatives such as the overlap metric (e.g., the Hamming distance) may be used. Classification accuracy maybe improved significantly if the distance metrics are learned using specialized algorithms such as Large Margin Nearest Neighbor or Neighborhood Component Analysis.
6. Drawbacks of the Majority Voting Technique: Classification skew may be introduced owing to potential dominance of one of the candidate classes – which, in turn, may be addressed by distance weighting techniques (Coomans and Massart (1982)).
7. Classification Accuracy Improvement through Improved Data Representation: A good instance for such a representation would be the self-organizing map (SOM). In an SOM, each node is a representative center of a cluster of similar points (i.e., each node would be a source density field). *kNN* may then be applied to the SOM.

8. Choice of k in kNN: While large k reduces the impact of noise/variation in the classification, it ends up making the boundaries less distinct and more diffuse (Everitt, Landau, Leese, and Stahl (2011)). Appropriate k for the problem set may be chosen using targeted heuristic techniques (such as hyperparameter optimization).
9. Noisy/Irrelevant Feature Reduction: Since kNN accuracy maybe severely degraded by the presence of noisy/irrelevant features, much effort has been put into reducing them. Typical approaches include:
    a. Use of evolutionary algorithms to optimize feature scaling (Nigsch, Bender, van Buuren, Tissen, Nigsch, and Mitchell (2006)).
    b. Scaling the features by using mutual information across the training data and the training classes (this approach must be similar to cross validation).
10. Binary kNN Classification (2 Class Classification): Here it is useful to choose k to be an odd number to avoid ties. An empirically optimal value for k maybe found out using the bootstrap method (Hall, Park, and Samworth (2008)).
11. kNN Customization using Feature Scaling: Although the kNN approach is treated as a non-parametric method, it may be customized by working out an optimal k. This is indeed the case, for example, in metric-based classifications, SOM approaches, noisy/irrelevant feature reduction, etc:
12. Efficient kNN Approaches: Naïve kNN approaches that aim to compute distance metrics to all the data points are often computationally intractable, so approaches seek to use a "small enough" k. The consequence of these approaches is that they reduce the number of distance metric and/or feature scaling evaluations performed. This is the reason approaches such as variable bandwidth kernel density balloon estimators with a uniform kernel are among the most common kNN approaches (Terrell and Scott (1992), Mills (2001)).
13. kNN Asymptotic Error Rates: The kNN approach has strong consistency results. As sample size approaches infinity, the kNN approach produces an error rate that is less than twice the best possible error rate achievable - viz. the Bayes' error rate (Covert and Hart (1967)) for some value of k. Further improvements may be possible through the use of proximity graphs (Touissant (2005)).

14. Feature Extraction/Reduction Example using the *kNN* Approach: An example of a typical computer vision computation pipeline for face recognition using *kNN* includes the feature extraction and dimension reduction pre-processing steps (this is implemented in OpenCV) is a s follows:
    a. Haar Face Detection.
    b. Mean-Shift Tracking Analysis.
    c. PCA or Fischer LDA Projection onto the Feature Space.
    d. *kNN* Classification.
15. *kNN* Curse of Dimenstionality: For high dimensions, Euclidean and other similar distance metrics become unhelpful and/or computationally infeasible, so dimension reduction preceedes *kNN* (Beyer, Goldstein, Ramakrishnan, and Shaft (1999)).
16. Low-Dimensional Embedding: Here feature extraction and dimension reduction are combined together in one step using PCA/LDA/CCA as a pre-processing step befor applying *kNN* (Shaw and Jebara (2009)).
17. High Dimensional Tick Dataset: For very high dimensional real-time/tick datasets (e.g., when performing similarity search on live video streams, DNA data, or high-dimensional time series), running a fast approximate *kNN* search using locality sensitive hashing, random projections (Bingham and Mannila (2001)), sketches (Shasha (2004)), or other high dimension similarity search techniques from the VLDB tool-box may be the only feasible option.
18. *kNN* Data Reduction: Since only some points are needed for accurate classification (these are called prototypes), they may be identified as follows:
    a. Select the class outliers – the training data those are incorrectly classified by *kNN* for a given k.
    b. Separate the remainder into two sets:
        i. The prototypes that are to be used for classification decisions
        ii. The absorbed points that can correctly classified by *kNN* using prototypes, and therefore be removed from the training set
19. Reasons for Outliers: Outlier is an instance in the training data that is surrounded by instances of other classes. Outliers may occur due to:
    a. Random Error

b. Insufficient training examples of this class (an isolated example ossurs instrad of a cluster)
c. Missing important (classes may be separated in dimensions that we do not as yet know about)
d. Too many training instances of other classes (unbalanced classes) that create a hostile background for the given small class

20. $(k,r)NN$ Class Outlier: Given 2 numbers $k > r > 0$, a training example is called a $(k,r)NN$ class outlier if its k nearest neighbors include more than r examples of other classes. Class Outliers should be detected and separated.

21. Condensed Nearest Neighbor (CNN) Algorithm: This is also called the Hart's algorithm (Hart (1968)), CNN selects a set of prototypes U from the training set such that $1NN$ with U can classify the examples almost as accurately as $1NN$ does with the full data set.
    a. Hart's Algorithm => Given a training set X CNN works with full iterative scans:
        i. Scan all elements of X looking for element x whose nearest prototype from U has a different label than that of x.
        ii. Move x from X onto U.
        iii. Repeat until there are no more prototypes to add to U.

22. Mirkes' Border Ratio: For the scan above, it is efficient to scan the training examples in the order of decreasing border ratio (Mirkes (2011)), defined as $a(x) = \frac{\|x'-y\|}{\|x-y\|}$ where $\|x-y\|$ is the distance between x and y where y has a different color than x, and $\|x'-y\|$ is the distance between x' and y where y has a different color than x'. This ratio stays inside the bracket $[0,1)$, as $\|x'-y\|$ never exceeds $\|x-y\|$.

23. kNN Regression: This consists of a typical inverse distance weighted kNN algorithm:
    a. Compute the Euclidean/Mahalanobis distance from the query instance to the labeled instances.

b. Order the labeled instances by their distance metrics.
c. Find a heuristically optimal k based on RMSE, and using cross validation.
d. Compute the inverse distance weighted average among the k nearest multivariate neighbors.

24. *kNN* Accuracy Validation: This is typically done using either a confusion matrix (for binary classifications) or a matching matrix (for multi-class classifications). Likelihood ratio tests are also often applied.

Perceptron

1. Definition: Peceptron is an algorithm for supervised classification of the input to one of the several possible binary outputs (Perceptron (Wiki), Rosenblatt (1957) , Rosenblatt (1958) , Rosenblatt (1962)).
2. Perceptron as a Linear Classifier: The perceptron makes its prediction based on a linear predictor function that combines a set of weights with a feature vector describing a given input state using the delta rule.
3. Perception as an Online Algorithm: The learning algorithm is an online algorithm, as it processes elements in the training set one at a time.
4. Setup: It is a binary classifier that maps the input feature vector $\vec{x}$ to an output value $f(\vec{x})$ – a single binary value, as $f(\vec{x}) = \begin{cases} 1 & \vec{w}.\vec{x} + b > 0 \\ 0 & otherwise \end{cases}$ where $\vec{w}$ is a real-valued vector of weights, and b is the bias term that is independent of the input. Spatially, the bias alters the position, but not the orientation, of the decision boundary.
5. Linearly Separable Learning Set: The perceptron learning algorithm does not terminate if the learning set is not linearly separable, and then the classification will not be achieved (e.g., Boolean XOR). The solution spaces of decision boundaries for all the binary functions and their learning behaviors are studied in Liou, Liou, and Liou (2013).
6. From an ANN perspective: A perceptron may ne viewed as an artificial neuron that uses the Heaviside step function as the activation function – this is also referred to as

the single-layer perceptron (as opposed to a multi-layer perceptron, which is just a neural network). As a linear classifier, single layer perceptron is the simplest feed forward neural network.

a. The multi-layer perceptron learning algorithm => Here, a hidden layer exsist, so that the algorithms such as back-propagation should be used (Minski and Papert (1969)). Alternatively, methods such as delta-rule can be used if the perceptron function is non-linear and differentiable. Finally, each neuron in the ANN operates independently of the others – so the learning outputs may be considered in isolation.

7. Perceptron Setup: $y = f(\vec{z})$ is the learning perceptron function for the input vector $\vec{z}$. The bias b is assumed to be zero here, and $\{\vec{D}\} \Rightarrow \{(\vec{x}_j, d_j)\}_{j=1}^{s}$ is the training set of S samples. $\vec{x}_j$ is the feature vector for sample j (with $i = 0,...,n-1$ feature components). $\vec{w}(t) = \{w_i(t)\}_{i=0}^{n-1}$ corresponds to the feature vector weights at time t. α is the learning rate with the restriction $0 < \alpha \leq 1$.

8. The Algorithm:

a. Initialization => Initialize the weight vector and the error threshold γ; weights may be initialized to zero, or a small random value.

b. Update $y_j(t)$ => For each example j in our training set $\{D\}$, compute the output $y_j(t)$ from the following:

$y_j(t) = f\lfloor \vec{w}(t).\vec{x}_j \rfloor = f\lfloor \vec{w}_0(t)x_{j,0} + \vec{w}_1(t)x_{j,1} + ... + \vec{w}_{n-1}(t)x_{j,n-1} \rfloor$, which can be generalized to automatically incorporate the plane intercept bias onto the feature vector set.

c. Update the weights => $w_i(t+1) = w_i(t) + \alpha\lfloor d_j - y_j(t) \rfloor x_{j,i}$ for all $0 \leq i < n$.

d. Iterate until convergence => Repeat the past two steps until the error $\frac{1}{S}\sum_{j=1}^{S}\lfloor d_j - y_j(t) \rfloor \leq \gamma$ (or bail if the pre-determined number of steps have been reached). The previous two steps immediately update the weights to the given pair rather than wait until all the pairs have undergone the steps.

9. Separability: The training set $\{D\}$ is said to be linearly separable if the positive examples can be separated from the negative examples using a hyperplane – that is, there exists a positive constant vector $\vec{\zeta}$ AND a weight vector $\vec{w}$ such that $(\vec{w}.\vec{x}_j + b)d_j > \zeta_j$ fo all j.
    a. Linear Separable Convergence from Below => From below, along the iteration, the weight vector always gets adjusted by a bounded amount in the direction it has negative dot product with, thus getting bounded above by $\Theta(\sqrt{t})$, where t is the number of steps.
    b. Linear Separable Convergence From Below => The weight vector gets bounded from below by $\left(\frac{2\kappa}{\zeta}\right)^2$, where κ is the maximum norm of the input vector (Novikov (1962)).
    c. Combining Above and Below => Combining the observations listed in a) and b), linearly separable data setds cause the perceptron algorithm to converge after a finite number of steps. In fact, these steps and their convergence characteristics are really just the single-layer ANN analogue of matrix linear reduction techniques (such as Gauss-Jordan, Newton's, or Caylet-Hamilton convergence techniques), and as such, all the criteria that cause non-convergence apply here too.
10. Invariance of the Decision Boundary to the Scaling of the Weight Vector: The decision boundary of the perceptron is invariant to the scaling of the weight vector, i.e., a perceptron trained with an initial weight vector $\vec{w}$ and a learning rate α behaves identically to a perceptron trained with with an initial weight of $\frac{\vec{w}}{\alpha}$ and learning rate unity.
11. Perceptron Variants:
    a. Gallant's Pocket Algorithm => The pocket algorithm with ratchet (Gallant (1990)) improves efficiency by returning the best solution in the pocket rather than the latest solution. This may also be used for non-linearly separable data-sets, where the aim is to identify the solution with the least misclassification.

b. Perceptron with the Largest Separation Margin => Also referred to as to the perceptron of optimal stability, these techniques employ iterative training and optimization schemes such as the Min-Over algorithm (Krauth and Mezard (1987)) or the AdaTron algorithm (Anlauf and Biehl (1989)). The perceptron of optimal stability, together with the kernel trick, serve as the conceptual foundations for the support vector machines techniques.

c. α -Perceptron => This technique uses a pre-processing layer of random fixed weights alongwith a set of feature thresholded units. Among other applications, these perceptron variants are used to classify analogue patterns by projecting them onto the binary space (after digitizations, of course).

d. Multi-layer Perceptrons => By adding extra non-linear layers between the inputs and the outputs, one can separate all data, and can model any well-defined function according to an arbitrary precision given enough training data. This generalization is called the multi-layer perceptron.

12. High-Dimension Separability: "High dimensions" refers to high feature vector dimensions. Sure enough, if there are as many feature vectors as samples, one WILL achieve complete separation – through straightforward I/O mapping!

13. Higher Order Networks ($\sigma - \pi$ Units): These techniques help address non-linear perceptron functions without using multiple layers. Here each element of the input vector is extended with a pairwise combination of multipled inputs (aka the quadratic classifier). This may also be applied to the multi-layer network.

14. "Best Classifier": Remember that the best classifier need not classify all the training data perfectly (i.e., it need not always uncover a hyperplane boundary). As an example, given the contraint that the input data each come from its own Gaussian distribution, the logistic regression/LDA might tur out be optimal classifiers.

15. Multi-class Perceptron: The multi-class perceptron may be defined as follows: Given an input feature vector $\vec{x}$ and an output class vector $\vec{y}$, the feature representation function $\vec{f}(\vec{x}, \vec{y})$ maps each possible input/output pair to a finite-dimensional real-valued feature vector. The score that results from the feature vector/weight vector dot

product is used to choose among the many possible outputs, i.e.,

$\hat{y} = \arg\max_{y} \left[\vec{f}(\vec{x}, \vec{y}).\vec{w} \right]$.

16. Iterative Multi-class Perceptron Learning: As in the binary classification, multi-clas classifications again iterate over training instances, predicting an output for each example, leaving the weights unchanged when the predicted output matches the target, and changing them when the predictions don't match: $w_{t+1} = w_t + f(x, y) - f(x, \hat{y})$. As expected, this reduces to the original binary classifier when x is a real-valued vector, y is chosen from $(0,1]$, and $f(x, y) = xy$.
17. Efficient Feature-Score Calculations: Efficient algorithms are available to calculate the feature score $\hat{y} = \arg\max_{y} \vec{f}(x, y).\vec{w}$ under specific problem spaces:
    a. In NLP for tasks such as part-of-speech tagging and syntactic parsing (Collins (2002)).
    b. Large-scale machine learning in a distributed computing setting (McDonald, Hall, and Mann (2010)).
18. Perceptron Extensions: Margin Bound guarantee for the Kernel Perceptron algorithm was proposed initially by Aizerman, Braverman, and Rozonoer (1964), and extended to the general non-separable case by Freund Shapire (1998), Freund and Shapire (1999), and Mohri and Rostamizadeh (2013).

Support Vector Machines (SVM)

1. Definition: SVM Model is a representation of the training data as points in space, mapped such that the examples of the distinct categories are separated by a clear gap that is as wide as possible (Support Vector Machine (Wiki), Cortes and Vapnik (1995)).
2. SVM Kernel Trick: In addition to performinglinear classifications, SVM's can efficiently perform a non-linear classification using the kernel trick (i.e., mapping their inputs to high-dimensional space using custom kernel basis functions.

a. The kernel function is essentially the same as a basis function, i.e., the feature vector set appears only as arguments to the kernel function during the entire formulation, thus the earlier spline basis function analysis we did should be usable here.

3. Hyperplane Construction: An SVM constructs a hyperplane or a set of hyperplanes in a high or infinite dimensional space, which may be used for classification, regression, or other tasks.
   a. Intuition behind the Hyperplane Construction => A good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (the so-called function margin), since, in general, larger the function margin, lower the generalization eror of the classifier (Preuss, Teukolsky, Vetterling, and Flannery (2007)).
   b. Alternate names for the hyperplane => This is also called the maximum margin hyperplane (for a p-dimension feature vector, we will need a $(p-1)$ dimensional hyperplane). The corresponding linear classifier is then referred to as the maximum margin classifier, or the perceptron of optimal stability.
4. Binary Classification Linear SVM: Given a sample set $D \Rightarrow \{(x_i, y_i): x_i \in R^p, y_i \in [-1,1]\}_{i=1}^n$, our intent is to find out the maximum margin hyperplane that separates $y_i = 1$ from $y_i = -1$, i.e., $\vec{w}.\vec{x} - b = \pm 1$. The corresponding offset that we chose to maximize is $\frac{b}{\|\vec{w}\|}$, i.e., we seek to minimize $\|\vec{w}\|$. More formally, we seek to minimize $\|\vec{w}\|$ in $(\vec{w}, b)$ subject to the constraint $y_i(\vec{w}.\vec{x} - b) \geq 1$.
5. The Primal Form: Here we minimize $\frac{1}{2}\|w^2\|$ subject to the constraint $y_i(\vec{w}.\vec{x} - b) \geq 1$ using the method of Lagrange multipliers. We compute $\arg \underset{\vec{w},b}{\min} \underset{\alpha \geq 0}{\max} \left\{\frac{1}{2}\|w^2\| - \sum_{i=1}^{n} \alpha_i [y_i(\vec{w}.\vec{x} - b) - 1]\right\}$, which automatically implies that we ignore all contributions from $y_i(\vec{w}.\vec{x} - b) - 1 > 0$, as their corresponding $\alpha_i = 0$.
   - Solution for the Primal Form => The stationary Karush-Kahn-Tucker condition implies that the solution can be expressed as a linear combination of the training

vectors: $\vec{w} = \sum_i \alpha_i y_i \vec{x}_i$. The corresponding $\vec{x}_i$'s are called the SUPPORT VECTORS, as they lie at the margin and satisfy $y_i(\vec{w}.\vec{x} - b) \equiv 1$. This allows one to define b as $b = \vec{w}.\vec{x} - y$, or more generally, $b = \frac{1}{N_{SV}} \sum_{i=1}^{N_{SV}} (\vec{w}.\vec{x}_i - y_i)$ where N_{SV} is the number of the support vector samples.

6. The Dual Form: Writing the classification rule in its unconstrained dual form reveals that the maximum margin hyperplane (and therefore the classification task) is a function of only the support vectors, the set of training points that lie on the margin. Using $\vec{w} = \sum_i \alpha_i y_i \vec{x}_i$ reduces the optimization task to

$$\min_{\alpha_i} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \vec{x}_i \vec{x}_j \right\} = \min_{\alpha_i} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \kappa(\vec{x}_i, \vec{x}_j) \right\}$$ subject to the constraint $\alpha_i \geq 0$. The kernel, therefore, is $\kappa(\vec{x}_i, \vec{x}_j) = \vec{x}_i.\vec{x}_j$.

   a. $b = 0$ is referred to as the biased SVM, and $b \neq 0$ as the unbiased SVM.

7. SVM Soft Margin: If there exsis no hyperplane such that the "yes" and the "no" categories cannot be split, the soft margin method will choose a hyperplane that separates the example set as cleanly as possible, while still maintaining the distances to the nearest cleanly split examples. Thus, this modified maximum margin allows for mislabeled examples (Cortes and Vapnik (1995)).

8. Soft Margin Slack Variable: The non-negative slack variable ξ_i measures the degree of miscalculation of the data x_i such that $y_i(\vec{w}.\vec{x}_i - b) \geq 1 - \xi_i$ for all $1 \leq i \leq n$. The objective function now has an additional term that penalizes non-zero ξ_i, and the optimization is a trade-off between a large margin and a small penalty error.

9. Soft-Margin Optimization Setup: If the penalty function is linear, the optimization problem becomes $\arg \min_{\vec{w}, \vec{\xi}, b} \left\{ \frac{1}{2} \left\| w^2 \right\| + C \sum_{i=1}^{n} \xi_i \right\}$ subject to $y_i(\vec{w}.\vec{x}_i - b) \geq 1 - \xi_i; \xi_i \geq 0$ for all $1 \leq i \leq n$. The corresponding Lagrangian objective function becomes

$$\arg \min_{\vec{w},\vec{\xi},b} \max_{\vec{\alpha},\vec{\beta}} \left\{ \frac{1}{2}\left\| w^2 \right\| + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i\left(\vec{w}.\vec{x} - b\right) - 1 + \xi_i \right] - \sum_{i=1}^{n}\beta_i\xi_i \right\}; \alpha_i, \beta_i, \xi_i \geq 0$$

.

10. Soft Margin – Dual Form: This is similar to the non-soft margin formulation.

    Minimize $\min_{\alpha_i} \left\{ \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \kappa\left(\vec{x}_i, \vec{x}_j\right) \right\}$ subject to the constraints $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{n}\alpha_i y_i = 0$. For linear separation, the kernel $\kappa\left(\vec{x}_i, \vec{x}_j\right)$ is still the dot product $\vec{x}_i.\vec{x}_j$

    .

    a. The key advantage of the linear penalty is that the slack variables vanish from the dual formulation, with constant C appearing purely as an additional constraint on the Lagrange multipliers. Nonlinear penalty functions have been used particularly to reduce the effect of outliers on the classifiers, but unless care is taken this problem can become non-convex, thus making it difficult to find a global solution.

11. Nonlinear SVM Classification: Boser, Guyon, and Vapnik (1992) extend the kernel trick originally proposed by Aizerman, Braverman, Emmanuel, and Rozonoer (1964) to create nonlinear classifiers – the formulation is more or less identical to that of the linear case, with the key difference being that the kernel basis function is not $\vec{x}_i.\vec{x}_j$ anymore – it becomes a nonlinear kernel function. This allows the algorithm to fit the maximum margin hyperplane in the transformed feature space.

    a. The Transformed Feature Space => The transformation is nonlinear, and the transformed space is high-dimensional. Thus, though the classifier is a hyperplane in the high-dimensional feature space, it may be nonlinear in the original input space.

12. The Nonlinear Kernel Basis Function:

    a. Properties => In all the cases (including the linear basis), the kernel basis function is symmetric in $\vec{x}_i$ and $\vec{x}_j$, and in all cases except the Gaussian radial basis, it only depends on the $\vec{x}_i.\vec{x}_j$ dot product (in the case of Gaussian radial

basis function, there is additional dependence on the self dot-product, or the Euclidean/Mahalanobis distance of the given feature vector).

b. Usage => More generally, the kernel basis function is related to the feature vector transform $\varphi(\vec{x}_i)$ as $\kappa(\vec{x}_i, \vec{x}_j) = \varphi(\vec{x}_i).\varphi(\vec{x}_j)$. $\vec{w}$ is evaluated in the transformed feature vector space as $\vec{w} = \sum_i \alpha_i y_i \varphi(\vec{x}_i)$. Dot products involving $\vec{w}$ for classification can be computed again using the kernel trick, i.e., $\vec{w}.\varphi(\vec{x}) = \sum_i \alpha_i y_i \kappa(\vec{x}_i, \vec{x})$. However, in general, there does not exist a $\vec{w}'$ such that $\vec{w}'.\varphi(\vec{x}) = \kappa(\vec{w}', \vec{x})$ (this straight scaling, however, is valid in the linear case).

c. Kernel Basis Function: Gaussian Radial Basis => If the kernel used is a Gaussian radial basis function, the corresponding feature space is a Hilbert-space of infinite dimensions. Maximum margin classifiers are well-regularized, so infinite feature vector dimensions do not spoil the results. The kernel Gaussian radial basis function is $\kappa(\vec{x}_i, \vec{x}_j) = e^{-\gamma \left\| \vec{x}_i - \vec{x}_j \right\|^2}$ for $\gamma > 0$. Of course γ is also expressed often as $\gamma = \frac{1}{2\sigma^2}$.

d. Kernel Basis Function: Homogenous Polynomial Basis => $\kappa(\vec{x}_i, \vec{x}_j) = (\vec{x}_i.\vec{x}_j)^d$.

e. Kernel Basis Function: Inhomogenous Polynomial Basis => $\kappa(\vec{x}_i, \vec{x}_j) = (1 + \vec{x}_i.\vec{x}_j)^d$.

f. Kernel Basis Function: Hyperbolic Tangent Radial Basis => $\kappa(\vec{x}_i, \vec{x}_j) = \tanh(c + k\vec{x}_i.\vec{x}_j)$ where $c < 0, k > 0$.

13. <u>SVM vs. Other Classifiers</u>: SVM belongs to the family of generalized linear classifiers. They correspond to a special case of Tikhonov regularization that simultaneously minimized the empirical classification error and maximizes the geometric margin, and are therefore refered to maximum margin classifiers (Meyer, Leisch, and Hornik (2003)).
14. <u>SVM Parameter Selection</u>:

a. Effectiveness of the SVM depends on the kernel basis function choice, the kernel parameters, and the soft margin parameter C. For instance, for the Gaussian kernel with parameter γ, the best combination of γ and C may be selected by a grid search of exponentially growing sequences of C/γ, i.e., $C \in \{2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}\}$ and $\gamma \in \{2^{-15}, 2^{-13}, ..., 2^{1}, 2^{3}\}$.

b. Further, each combination of the parameter choices is checked using cross validation, and the parameters with the cross-validation accuracy are picked. The final model, which is used for testing and classifying new data, is then trained on the entire training set using the selected parameters (Hsu, Chang, and Lin (2003)).

15. Drawbacks of the SVM Approach:
    a. Uncalibrated Class Membership Probabilities
    b. As is, it is directly applicable only to two-class classification problems. Additional algorithms to reduce the multi-class classification may need to be applied.
    c. Parameters of the solved model are hard to interpret.
16. Multiclass SVM: As outlined in Hsu and Lin (2002) and Duan and Keerthi (2005), the choices are:
    a. One vs. All => This is a winner take all strategy where the classifier with the highest score wins (for this, it is important the output functions be calibrated to produce comparable scores).
    b. One vs. One => Here, the classification is done by a max wins voting strategy in which every classifier assigns the instanceto one of two classes, after which the assigned class is increased by one vote, and finally the class with most votes wins.
17. Other Multiclass Extensions:
    a. Directed Acyclic Graph SVM (Platt, Christianini, and Shawe-Taylor (2000)).
    b. Error correcting output codes (Dietterich and Bakiri (1995)).
    c. Crammer and Singer (2001) propose a multiclass SVM which casts the classification problem into a single optimization problem rather than

decomposing it into multiple binary classification problems (Lee, Lin, and Wahba (2001), and Lee, Lin, and Wahba (2004)).

18. Transduction SVM: These extend the traditional SVM treatment so that they could handle labeled data in semi-supervised learning by following the principles of transduction. In effect, the formulation extends the training data $\{\vec{D}\}$ with the unlabeled set $\{\vec{D}'\}$ and optimizes the calibration across both sets (Joachims (1999)).
    a. Setup => The following primal optimization problem statement sets it up:

    Minimize $\frac{1}{2}\|w^2\|$ in $(\vec{w}, b, \vec{y}^*)$ subject to the constraints $y_i(\vec{w}.\vec{x}_i - b) \geq 1, \forall i$ and $y_j^*(\vec{w}.\vec{x}_j - b) \geq 1, \forall j$ with $y_j^* \in \{-1,1\}$.

19. SVM Regression (SVR): For the regression extension SVM, the model produced by SVR depends only on the subset of the training data (just like SVM classification), as the cost function for building the model ignores any training data close to the model prediction within a threshold ε (Drucker, Burges, Kaufman, Smola, and Vapnik (1997)). Suykens and Vandewalle (1999) lay out a least squares version of SVR.
20. Maximim Margin Hyperplane Solutions:
    a. There exists several specialized algorithms for quickly solving the QP problem that arises from the SVM formulation, mostly relying on heuristics for breaking the problem down into smaller, more manageable chunks. A common mehod is Platt's Sequential Minimal Optimizer (SMO) which breaks the problem down into 2D sub-problems that may be solved analytically, eliminating the need for a numerical optimization algorithm.
    b. Another approach is to use an interior-point method that employs Newton-like iterations to find a solution to the Karush-Kahn-Tucker conditions of the primal and the dual problems (Ferris and Munson (2002)). Instrad of solving a sequence of broken-down problems, this approach directly solves the problems as a whole. To avoid solving a linear system involving a large kernel matrix, a low rank approximation to the matrix is often used in the kernel trick.

Gene Expression Programming (GEP)

1. Definition: GEP is an evolutionary algorithm that creates computer programs and models. The programs are complex tee structures that learn and adapt by changing their sizes, shapes, and composition.
2. GEP as a Genotype-Phenotype Algorithm: The GEP programs are encoded in simple linear chromosomes of fixed length; thus, just like a genotype-phenotype system, GEP benefits from a simple genome to keep and transmit genetic information, and a complex phenotype to explore the environment and adapt to it.
3. Evolutionary Algorithms: These us populations of individuals, select individuals according to fitness, introduce genetic variation one/more genetic operators, and have been used in optimization problems since the 1950's (Box (1957)), Friedman (1959), Rechenberg (1965), and Mitchell (1966)).
4. GEP as a Member of the Family of Evolutionary Algorithms: GEP is closely related to genetic algorithms agd genetic programming (Ferreira (2001)). From genetic algorithms GEP inherits the notion of linear chromosomes of fixed length; and from genetic programming it inherited the expressive parse trees of various sizes and shapes.
5. Multigenic Genotype/Phenotype: The linear chromosomes work as genotypes, and the parse trees as phenotypes, thereby creating a genitype/phenotype system that is multigenic, thus encoding multiple parsed trees in each chromosome. This indicates the programs created by GEPs are composed of multiple-parse trees. Further, since these parse-trees themselves are the result of gene expression, they are called as expression trees.
6. The Encoding – The Genotype: Genotypes consist of one/more fixed length genes, which encode expression trees of different sizes and shapes. The expressions contain encoding for both the operators $(\times,\div,+,-....)$ and the variables/constants (e.g., $L+a-baccd...$). They may also possess an implied execution pipeline order (BODMAS etc)
7. Expression Trees - The Phenotype: These interpret the genome specified above and expand them out into expression trees. Thus, to move from the genotype (the coding)

to the phenotype (the expression), you need a pre-agreed dictionary, and possibly an assembly rule, since the terminals assembly is non-unique.

   a. The genome k-expressions above are also referred to as k-expressions (or Karva expressions).

8. k-Expressions: The k-expressions of GEP correspond to regions of genes that WILL get expresses – there may be sequences of the genes that may not be expressed at all, and the reason for that is to provide a buffer of terminals s that all k-expressions encoded in GEP genes always correspond to valid programs/expressions.
9. Gene Head/Tail: The gene head is used to encode the functions and variables needed to solve the problem at hand, whereas the tail is used to a) encode variables, and b) provide a reservoir of terminals that ensures that the fixed length gene is error free.
   a. GEP Genes Tail Length => $t = h(n_{MAX} - 1) + 1$ where h is the length of the head, and n_{MAX} is its maximum arity.
10. Multigenic Chromosomes: Each gene is composed of one/more genes, the output of each of which can be combined in some way. Each output is referred to as a sub-ET (sub expression tree).
11. Gene Output Linkages: Although the output will have to be a primitive, there are no restrictions on them, thus they may be linked using any combination of primitives (mean, median, average etc) in a number of ways. They can also be evolved, of chosen in an ad hoc manner (Ferreira (2002a), Ferreira (2006a)).
12. Homeotic Genes: These control the interactions of the different sub-ET's, i.e., determine which sub-ET's are called upon and in which order, in which cell (the driver program), as well as the nature of connections the ET's establish with neach other.
   a. Homeotic Genes and Cellular Systems => The homeotic genes are organized identically to that od the other genes, except their heads contain linking functions and a special kind of terminal – the genic terminal – that represents normal genes. Expressions of the normal genes result in different sub-ET's (also referred to as ADF's – automatically defined functions), with different evolutionary linkages, etc.

13. Multicellular Programs: These are composed of more than one homeotic gene, and the multigenic output of each of homoetic gene results in different sub-ET's and their linkages, each of which may be custom evolved.
14. Additional Gene Domains: In addition to the typical head/tail domains, you may have additional domains used for maintaining/learning/calibrating/process tuning the constants used in finding a solution.
15. Examples of the Numerical Constants Employed in Domain Usage:
    a. As weights/factors in a function approximation problem (the GEP-RNC algorithm)
    b. As weights/thresholds of a neural network (GEP-NN)
    c. Numerical Constants in a Decision Tree (GEP-DT algorithm)
    d. Weights used in a Polynomial Induction
    e. Random numerical constants to discover parameter values in a parameter optimzation task
16. Basis Gene Expression Algorithm:
    a. Select Function Set
    b. Select Terminal Set
    c. Load Dataset for Fitness Evaluation
    d. Create Chromosomes of initial Population randomly
    e. For each Program in the Population:
        i. Express the Chromosome
        ii. Execute the Program
        iii. Evaluate the Fitness
    f. Verify the Termination Condition
    g. Select the Programs
    h. Replicate the Selected Programs to form the next Population
    i. Modify Chromosomes using Genetic Operators
    j. Go back to Step e.
17. Preparation for the Execution: Steps a through e are just the preparatory steps needed for e through i. The key step of initial population occurs using random elements of function/terminal sets.

18. Population of Programs: Like other evolutionary programs, GEP works with individual populations (i.e., populations of computer programs – the organism is treated as a program!) From the initial population, descendents are evolved via selection or genetic modification. In the genetype/phenotype system of GEP, only the simple linear chromosomes of individual programs need to be evolved, as this localizes the structural soundness and eventually syntactically correct programs.
19. Fitness Functions and the Selection Environment: Selection environments are akin to training sets, and the fitness functions determine the penalty cost. The fitness of a program depends on both the cost function and the training data.
20. Selection Environment: This contains the training data/records as well as the fitness cases, the selection environment should represent the problem cases well, as well being well-balanced, should not be too large, but large enough to enable good generalization and validation.
21. Fitness Functions for Regression Analysis (Continuous Inference): Due to the continuous nature, direct comparison metrics are possible. Any of the appropriate Bayes' loss functions would be a good candidate for fine granularity/smoothness to the solution space.
    a. Multi-target Regression Fitness Function => While fitness functions based purely on R^2 and correlation β are smooth, they work best as combinations of multiple metrics (i.e., ones that control coarseness of fit, say less than 10% of estimated samples out of the range, quality of approximation/shape preservation, etc)
22. Fitness Functions for Classification:
    a. Fitness Functions Based on Confusion Matrix => Confusion matrix alongwith the appropriate smoothener creates efficient/effective fitness functions. Examples of smootheners include F-measures, Jaccard similarity, Matthews correlation coefficient, and cost/gain matrix that combines the costs/gains assigned to the 4 different confusion classifications.
    b. Fitness Functions based on the Solution Space => These fitness explore the structure of the classification model itself (which includes the domain, the

range, the distribution of the model output, and the classifier margin). For instance, one can combine:

i. Measures based on confusion matrix with those based on the mean squared error between the rae model outputs and the actual values
ii. F-measure and the R^2 for the model and the target
iii. Cost/gain Matrix with the correlation coefficient
iv. Functions that expose model granularity with metrics based on the area under the ROC curve and the rank measure

23. Fitness Functions for Logistic Regression: Here the confusion matrix is combined with the joint metric that uses model probabilities and measures probabilities.
24. Fitness Functions for Boolean Problems: Here the cost/gain confusion matrix in conjunction with the deterministic/hard hit rates is the only real option
25. Selection and Elitism: Roulette-wheel selection according to fitness and the luck of the draw is the most popular. Combining roulette-selection with cloning the best program of each generation guarantees that the very best traits are not lost (this is called simple elitism).
26. Repreduction with Modification: Program repreduction first involves selection, then replication of the genome sequence. Genome modification is not required for reproduction, but without it adaptation and evolution won't take place.
27. Selection for Replication: Selection operator (say, a copy instruction) selects programs for the replication operator to copy (selection typically occurs vis simple elitism). For evolution to occur, replication is implemented with a few random errors thrown in through genetic operators such as mutation, recombination, transposition, inversion, and many others.
28. Mutation: Mutation is the most important genetic operator (Ferreira (2002b)), and changes genomes by changing one element in it by another. Accumulation of many small changes over time creates great diversity.
    a. Unconstrained Mutation => In GEP, mutation is totally unconstrained, which means that in each gene domain, any domain symbol can be replaced by another. For example, in the heads of genes, any function can be replaced by a terminal or another function (regardless of the number of arguments in the

new function), and a terminal may be replaced by a function or a new terminal.

29. Recombination: Recombination typically involves two parent chromosomes to create two new chromosomes by combining different parts from the parent chromosomes. As long as the parent chromosomes are aligned and the exchanged fragments are homologous (i.e., they occupy the same position in the chromosomes), the new chromosomes created by recombination will always encode syntactically correct programs.
    a. Recombination crossover implementation types =>
        i. Changing the number/combination of parents
        ii. Changing the split points
        iii. Changing the order of the fragments to exchange
30. Transposition: This involves introduction of an insertion sequence into the chromosome. The insertion sequence may be chosen from anywhere in the chromosome, but is inserted only at the head. This ensures that all domains, including the tail domain sequences, result in error-free programs.
    a. Transposition Implementation Methods => Transposition needs to preserve the structure and the length, so it needs to be implemented in the following ways. Both methods can be implemented to operate within chromosomes, or within even a single gene.
        i. Create a shift at the insertion site, followed by a delete at the end of the head
        ii. Overwrite the local sequence at the target site (thus, making it easier to implement)
31. Inversion: Inversion inverts a small sequence within a chromosome, and is especially powerful for combinatorial optimzation (Ferreira (2002c)). In GEP it can be easily implemented in all gene domains, and the produced off springs are always syntactically correct. For any gene domain, a sequence (ranging from at least two elements to as big as the domain itself) is chosen at random within that domain, and is then inverted.

32. Other Genetic Operators: The possibilities are endless – examples include one-point/two-point/gene/uniform recombinations, gene/root/domain-specific transposition, domain-specific mutation, etc: All are easily implemented and widely used.
33. GEP-RNC Algorithm: Here the GEP's use an extra domain for accommodating the random numerical constants (RNC) – the DC domain. Special DC-specific genetic operators allow for efficient calculation of the RNC's among the individual programs. Also, a special mutation operator allows for the permanent introduction of variation onto the RNC set.
34. GEP-NN:
    a. Motivation => Typical neural networks consists of three different classes of units – the input units, the hidden units, and the output units. An activation pattern presented at the input spreads forward through the hidden layers to the output. The activation input is amplified by the link-specific weight set, and is then thresholded/transmitted through the link set to the output.
    b. Parameters => The parameters are essentially the amplication weights and the corresponding thresholds, each of which may be populated in a GEP algorithm from a random initial population and then evolved.
    c. The Algorithm => Here the network architecture is encoded in the head/tail domain (Ferreira (2006b)). The head contains special encodings (or functions in the GEP-NN terminology) that activate the hidden/output units, and terminals that represent the input units. The tail contains only of the terminal units.
    d. Encoding the weights/thresholds => Besides the head/tail domains, GEP-NN uses two extra domains - D_w and D_t - the encode the weights/thresholds of the NN. D_w follows the tail with length $d_w = hn_{MAX}$, and D_t follows D_w with length $d_t = t$.
35. GEP-DT:
    a. Structure and Motivation => Decision Trees (DT) have three types of nodes – the root nodes, the internal nodes, and the leaf/terminal nodes. Roots/internal

decision nodes represent the test conditions for the different attributes/variables in the dataset. Leafs specify the class labels that terminate across the tree paths (Ferreira (2006c)).

b. GEP-DT Algorithm => Select an attribute for the root node (starting with random initial population and an eventual evolution) and use the nomial/numeric DT genesis algorithm to drive the classification label output.

c. DT Encoding => Rules for encoding/evolving the attributes are no different than that of GEP-RNC domain localization etc: in that the extra domain encodes the numerical constants and input labels corresponding to the attributes.

36. Performance: As is evident, beyond the encoding representation and the pipeline evolution strategies, there are no significant performance (as you would think) with GEP, and this is borne out by tests on GEP-DT (Oltean abd Grosan (2003)).

Hidden Markov

HMM State Transition/Emission Parameter Estimation

1. State Estimation Definition: State estimation is the full parameterization of the functional relationship between one/more elastic constitutive predictor(s) and one/more elastic response variables. Typically, the constitutive predictor(s) are not stochastic; the response variables may/may not be stochastic.
   - Representation vs. Transformation => Representation makes sense when dealing with state estimation, as it captures an inherent relationship (i.e., an relationship that already exists) between a hidden state quantification metric and the predictor. Transformation happens when a quantification metric manifests itself as an observation manifest metric. Representation and transformation may share similar function forms, but the conceptual notions underlying them are distinct.
   - Incorporating Dynamics vs. History-based Evolution => In HMMs, dynamics still refers to modeling dynamic equilibrium – otherwise there no real "stasis" oriented state to calibrate/uncover/characterize.
2. Undirected Graphical Relationship of the HMM States: Given the nature of the transition probability matrix (potentially non-zero, i.e., dense), HMM essentially models essentially an undirected graphical relationship. However, directionality may be imposed using non-zero unidirectional state transition matrix. Of course, a single HMM state inference run may use the full suite of the observation set – while the still maintaining the state updates Markovian (for e.g., this property makes HMM filtering/smoothing use the entire observation set).
3. Categorical Observable HMM Emission Parameters Analysis: If the observed categorical variable takes any of the M discrete values, and if there are N HMM parameters, then the number of emission parameters is $N(M-1)$ ($M-1$ instead of M since the emission probabilities from each state sum up to 1).

4. Continuous Real-Valued Observable HMM Emission Parameters Analysis: If there are M continuous states for each of the N hidden states' emission set, you then have one mean and $\frac{M(M+1)}{2}$ covariance entries across all the states' emission sets. Thus, the total is $N\left\{M + \frac{M(M+1)}{2}\right\} \Rightarrow N\left\{\frac{M(M+3)}{2}\right\} \cong \Theta\left(NM^2\right)$.
    - One way to reduce the number of parameters in the previous case is to assume that the emission states are independent, in which case the total number of emission parameters comes to NM. The State Transition matrix will still be a $N \times N$ matrix with $N^2 - N \Rightarrow N(N-1)$ entries (the $-N$ term is due to the sum of the probabilities being 1).
5. Continuous State Space/Continuous Observation HMM: Kalman filtering is a simple example where the state estimation from observation is a simple linear inference, and therefore tractable. If either the state evolution or the observation transform is non-linear, then the UKF/EKF/Particle Filter techniques are to be used.
6. Auto-regressive Nature of Markovian States: Markov states are, by definition, auto-regressive. Thus, the auto-regression covariance matrix corresponds to the state transition matrix.
7. A Note on Inferring the Past vs. Predicting the Future: You may infer the past quantification metric, as well as predict the future quantification metric/manifest measure. Therefore, in that sense, inference/prediction is relative only to the current time (and using earlier/later information).
8. i.i.d. vs. HMM: HMM's are not quiet i.i.d's, since the state jumps depend on the current state. In a history transition sense, the order of history dependence goes as i.i.d -> HMM -> k-level HMM -> Volterra series …

HMM Based Inference - Applications

1. Formulation Treatment: Details in Baum and Petrie (1966), Baum and Eagon (1967), Baum and Sell (1968), Baum, Petrie, Soules, and Weiss (1970), Baum (1972), Stratonovich (1960).
2. Inferring the Probability of an Observed Sequence: Given the model parameters, the probability of observing the sequence $\vec{Y} = \{y(0),...,y(L-1)\}$ is given as $P(\vec{Y}) = \sum_{\vec{X}} P(\vec{Y} \mid \vec{X}) P(\vec{Y})$ where $\vec{X}$ captures all the possible hidden sequences.

   Dynamic Programming and Forward Algorithm solves this.
3. Filtering: Given the model parameters and a sequence of observations, the task is to compute the distribution $P(\vec{X}(t) \mid y(0),...,y(t))$ over the hidden states of the last latent variable at the end of the sequence. Again Forward Algorithm solves this.
4. Smoothing: This is the same as filtering, with the difference being that the task is now to compute the distribution $P(\vec{X}(k) \mid y(0),...,y(t))$ where $k < t$. Again Forward-Backward Algorithm solves this.
5. Most Likely State Sequence: This addresses the joint probability of the entire sequence of hidden states that generated the specified sequence of observations. This is solved effectively using the Viterbi algorithm, and requires finding a maximum over all possible state sequences.
6. Statistical Significance: Re-casting the MLE statement above as: What is the probability that a sequence drawn from some NULL distribution will have a HMM emission probability at least as large as that of a particular sequence (using forward algorithm, see Newberg (2009))? What is the probability that a sequence drawn from some NULL distribution will have a maximum state sequence probability at least as large as that of a particular sequence (using Viterbi algorithm)?
7. Parameter Learning: Given an output sequence (or a set of sequences), what is the optimal set of state transition and emission/output probabilities (using MLE)?
   - Both Baum-Welch and Baldi-Chauvin algorithms are special case locally optimal MLE's. Exact solution to parameter learning is not available through any known tractable algorithm.

Non-Bayesian HMM Model Setup

1. Stage #1: The prior state distribution is assumed to uniform over all possible states (i.e., the starting state is not specified). Further, in typical HMM cases, the categorical states are computed from the real valued states.
2. Stage #2:
   - $N \Rightarrow$ The Number of States (the number of discrete state realizations possible).
   - $T \Rightarrow$ The Number of Observations – one measurement per each time step.
   - $\theta_{i=1,\ldots,N} \Rightarrow$ Emission Parameter for all observations associated with state i. Each $\theta_{i=1,\ldots,N}$ could be an array in itself, if the observation space is discrete (which is the case for the typical HMM).
   - $\phi_{i=1,\ldots,N;\, j=1,\ldots,N} \Rightarrow$ Probability of transition from state i from state j.
   - $\Phi_{i=1,\ldots,N} \Rightarrow$ N dimensional vector for each of $\phi_{i=1,\vee j}$. Sums to Unity.
   - $x_{t=1,\ldots,T} \Rightarrow$ State at each of the time instants t.
   - $y_{t=1,\ldots,T} \Rightarrow$ Observation at each of the time instants t.
   - $F(y \mid \theta) \Rightarrow$ Probability distribution of an Observation, parameterized on θ (Gaussian or Categorical).
   - $x_{t=2,\ldots,T} \Rightarrow$ Categorical value computed from $\phi(x_{t-1})$.
   - $y_{t=1,\ldots,T} \Rightarrow$ Observation computed from $F[\theta(x_t)]$.
3. Setup with Real-Valued/Gaussian Observations:
   - $N \Rightarrow$ The Number of States (the number of discrete state realizations possible).
   - $T \Rightarrow$ The Number of Observations – one measurement per each time step.
   - $\phi_{i=1,\ldots,N;\, j=1,\ldots,N} \Rightarrow$ Probability of transition from state i from state j.
   - $\Phi_{i=1,\ldots,N} \Rightarrow$ N dimensional vector for each of $\phi_{i=1,\vee j}$. Sums to Unity.
   - $\mu_{i=1,\ldots,N} \Rightarrow$ Mean of Observations associated with state i.
   - $\sigma^2{}_{i=1,\ldots,N} \Rightarrow$ Variance of Observations associated with state i.
   - $x_{t=1,\ldots,T} \Rightarrow$ State at each of the time instants t.

- $x_{t=2,\ldots,T} \Rightarrow$ Categorical value computed from $\phi(x_{t-1})$.
- $y_{t=1,\ldots,T} \Rightarrow$ Observation at each of the time instants t.
- $y_{t=1,\ldots,T} \Rightarrow \mathrm{N}[\mu(x_t), \sigma^2(x_t)]$.

Notice that there is no θ here, since the observations are real-valued.

4. Setup with Categorical Observations:
   - $N \Rightarrow$ The Number of States (the number of discrete state realizations possible).
   - $T \Rightarrow$ The Number of Observations – one measurement per each time step.
   - $\phi_{i=1,\ldots,N;\, j=1,\ldots,N} \Rightarrow$ Probability of transition from state i from state j.
   - $\Phi_{i=1,\ldots,N} \Rightarrow$ N dimensional vector for each of $\phi_{i=1,\vee j}$. Sums to Unity.
   - $V \Rightarrow$ Dimension of the categorical variables – e.g., size of the word vocabulary, number of financial regimes etc:
   - $\theta_{i=1,\ldots,N;\, j=1,\ldots,V} \Rightarrow$ Probability of state i observing observation item j.
   - $\theta_{i=1,\ldots,N} \Rightarrow$ V dimension vector, composed of $\theta_{i=1,\ldots,V}$ - must sum to Unity.
   - $x_{t=1,\ldots,T} \Rightarrow$ State at each of the time instants t.
   - $x_{t=2,\ldots,T} \Rightarrow$ Categorical value computed from $\phi(x_{t-1})$.
   - $y_{t=1,\ldots,T} \Rightarrow$ Observation at each of the time instants t.
   - $y_{t=1,\ldots,T} \Rightarrow$ Observation computed from $\theta(x_t)$.

Bayesian Extension to the HMM Model Setup

1. The Framework: All are essentially same as those for the non-Bayesian setup, except for the new formulations below, based on the specified hyper-parameters:
   - $\alpha \Rightarrow$ Shared Hyper-parameters for the Emission Parameters.
   - $\beta \Rightarrow$ Shared Hyper-parameters for the Transition Parameters.
   - $H(\theta \mid \alpha) \Rightarrow$ Prior Probability Distribution of Emission Parameters parameterized on α.

- $\theta_{i=1,\ldots,N} \Rightarrow H(\alpha)$; H is the conjugate of F.
- $\phi_{i=1,\ldots,N} \Rightarrow$ Made from $Symmetric\_Dirichlet_N(\beta)$.

2. Setup with Real-Valued/Gaussian Observations:
   - $N \Rightarrow$ The Number of States (the number of discrete state realizations possible).
   - $T \Rightarrow$ The Number of Observations – one measurement per each time step.
   - $\phi_{i=1,\ldots,N;\, j=1,\ldots,N} \Rightarrow$ Probability of transition from state i from state j.
   - $\Phi_{i=1,\ldots,N} \Rightarrow$ N dimensional vector for each of $\phi_{i=1,\vee j}$. Sums to Unity.
   - $\mu_{i=1,\ldots,N} \Rightarrow$ Mean of Observations associated with state i.
   - $\sigma^2{}_{i=1,\ldots,N} \Rightarrow$ Variance of Observations associated with state i.
   - $x_{t=1,\ldots,T} \Rightarrow$ State at each of the time instants t.
   - $y_{t=1,\ldots,T} \Rightarrow$ Observation at each of the time instants t.
   - $\beta \Rightarrow$ Concentration Hyper-parameter controlling the Density of the Transition Matrix.
   - $\mu_0, \lambda \Rightarrow$ Shared Hyper-parameters of the Means for each State.
   - $\nu, \sigma_0{}^2 \Rightarrow$ Shared Hyper-parameters of the Variances for each State.
   - $\phi_{i=1,\ldots,N} \approx Symmetric\_Dirichlet_N(\beta)$.
   - $x_{t=2,\ldots,T} \approx$ Categorical value computed from $\phi(x_{t-1})$.
   - $\mu_{t=1,\ldots,N} \approx \mathrm{N}\left[\mu_0, \lambda\sigma_i^2\right]$.
   - $\sigma^2{}_{i=1,\ldots,N} \approx Inverse\_Gamma\left[\nu, \sigma_0{}^2\right]$.
   - $y_{t=1,\ldots,T} \approx \mathrm{N}\left[\mu(x_t), \sigma^2(x_t)\right]$.
3. β as a Concentration Parameter: β controls the density of the transition matrix. With high β ($\beta >> 1$), the probabilities of transitioning to the other states are higher, making the HMM process more random. Low β ($\beta << 1$) causes greater state localization, and thinner outbound transition, thereby making the HMM less random.
4. Priors for Categorical Distribution: Dirichlet distribution is a natural choice, as it automatically serves as a conjugate pair for any categorical distribution. Symmetric

Dirichlet used across multiple categorical variates is analogous to uniform distributions used in real-valued priors.

5. Setup with Categorical Observations:
   - $N \Rightarrow$ The Number of States (the number of discrete state realizations possible).
   - $T \Rightarrow$ The Number of Observations – one measurement per each time step.
   - $\phi_{i=1,\ldots,N;\, j=1,\ldots,N} \Rightarrow$ Probability of transition from state i from state j.
   - $\Phi_{i=1,\ldots,N} \Rightarrow$ N dimensional vector for each of $\phi_{i=1,\vee\, j}$. Sums to Unity.
   - $V \Rightarrow$ Dimension of the categorical variables – e.g., size of the word vocabulary, number of financial regimes etc:
   - $\theta_{i=1,\ldots,N;\, j=1,\ldots,V} \Rightarrow$ Probability of state i observing observation item j.
   - $\theta_{i=1,\ldots,N} \Rightarrow$ V dimension vector, composed of $\theta_{i=1,\ldots,V}$ - must sum to Unity.
   - $x_{t=1,\ldots,T} \Rightarrow$ State at each of the time instants t.
   - $y_{t=1,\ldots,T} \Rightarrow$ Observation at each of the time instants t.
   - $\alpha \Rightarrow$ Shared Concentration Hyper-parameter θ for Each State.
   - $\beta \Rightarrow$ Concentration Hyper-parameter controlling the Density of the Transition Matrix.
   - $\phi_{i=1,\ldots,N} \approx Symmetric\_Dirichlet_N(\beta)$.
   - $\Theta_{i=1,\ldots,V} \approx Symmetric\_Dirichlet_V(\alpha)$.
   - $x_{t=2,\ldots,T} \approx$ Categorical value computed from $\phi(x_{t-1})$.
   - $y_{t=1,\ldots,T} \approx$ Observation computed from $\theta(x_t)$.
6. Two-Level Bayesian HMM: The purposes are to a) independently control the overall density of the transition matrix, and b) independently control the target densities of states to which the transitions are likely (i.e., the density of the prior distribution of states in any particular hidden variable – this serves as the initial estimate for the next stage). In both cases, this is accomplished by maintaining ignorance over which specific states are more likely to be transitioned into starting from a given state.
   - The following set of 2-level parameter models with non-uniform priors can be learned with Gibbs' sampling or enhanced Expectation Maximization algorithms.

- Formulation => Here, we enhance the fields β and $\Phi_{i=1,\ldots,N}$ with:
    - $\gamma \Rightarrow$ Concentration Hyper-parameter controlling how many states are intrinsically linked.
    - $\beta \Rightarrow$ Concentration Hyper-parameter controlling the Density of the Transition Matrix.
    - $\eta \Rightarrow$ N Dimensional Vector Probabilities, specifying the intrinsic probability of a given State.
    - $\mathrm{N} \approx Symmetric\_Dirichlet_N(\gamma)$.
    - $\Phi_{i=1,\ldots,N} \Rightarrow Dirichlet_N(\beta, N, \gamma)$.
- Given that the Dirichlet process is the conjugate of unknown infinite state of categorical variables, multi-level Dirichlet models are also called hierarchical Dirichlet HMM (HDP-HMM) or Infinite Markov Models.
- Two-level priors with both concentration parameters set to produce sparse distributions are used in unsupervised part-of-speech tagging, for e.g., where some parts occur more frequently than others.

HMM in Practical World

1. Role for HMM based Algorithmization: Full-scale HMM appear to deployed much more after the process is a) either matured, or b) is in the final stages of maturity (e.g., speech processing) where the characteristic regime transitions/state switches are all well tested and studied – in other words, it lends itself well to post-supervised processes.
2. Applications Developed as a Practice:
    - Biological Sequences (Bishop and Thompson (1986)) and Bio-informatics (Durbin, Eddy, Krogh, and Mitchison (1999)).
    - Gesture Recognition (Starner and Pentland (1995)).
    - Metamorphic Virus Detection (Wong and Stamp (2006)).
    - Musical Score Following (Pardo and Birmingham (2005)).

- Partial Discharges (Satish and Gururaj (2003)).
- Protein Folding (Stigler, Ziegler, Giesecke, Gebhardt, and Rief (2011)).
- Speech Recognition (Baker (1975), Jelinek, Bahl, and Mercer (1975), Huang, Jack, and Ariki (1990), Huang, Acero, and Hon (2001)).
- Speech Tagging (Rabiner (1989)).

Markov Random and Condition Random Fields

Introduction and Background

1. Multi-dimensional Graph/Lattice View of MRF/CRF: In MRF, the state response variables are laid out in an n-D graph/lattice. In addition, in the case of CRF, the realizations at the response lattices are conditional on the corresponding observations.
2. Random Fields as an Enhancement to the Old Regression Relations: In traditional regression frameworks, you have $y_i = f(x_i) + \varepsilon$. In Conditional Random Fields extension to Markov Random Fields formulation, these change to $y_i = f(y_{i-1}, x_i) + \varepsilon$, where y_{i-1} in the feature function is used to accommodate the Markov nature of the regression.
    - Discriminant models as State-Space based multivariate regression approaches => That is another way of looking at its, since after all they generate conditional distributions (both y_i and y_{i-1} can be vectors).
    - Non-trivial observation/feature functions => Further, the enhanced regression may be formulated as $y_i = f_{y_{i-1}}(x_i) + \varepsilon$, indicating that the feature function can in itself be dependent on the (Makovian) history.
3. Markov Random Field Motivation: This is undirected discriminant approach, and thus can represent cyclic relationships unlike directed graphs. However, it cannot represent induced/directed dependencies.
4. Conditional Random Field Motivation: CRF is a variant of MRF where the random variable at each graph node is conditioned upon the set of global observations $\{O_i\}$. The feature function for MRF maps $\{O_i\}$ to the given feature clique (defined below).
5. n-Way Property of MRF: Undirected lattice background (e.g., Ising models, see Kindermann and Snell (1980)) are MRF's prototypical settings. Thus its commonly used in image processing (image restoration, image retrieval, image registration,

image segmentation, and image completion), and computer vision (texture synthesis, resolution, matching, and retrieval) (Li (2009), Rue and Held (2005)).

- Labeling is a very common application scenario, particularly for CRF. Here, the predictor is a word/speech, and the response could the corresponding word/speech type.
- Other customized uses for CRF include computer vision (He, Zemel, and Carreira-Perpinnan (2004)), Shallow Parsing (Sha and Pereira (2003)), and Named Entity recognition (Settles (2004)).

6. Markovian on State only: Just like HMM, states represented by the random fields are Markovian. Thus, a single state inference run may use the full suite of the observation set – while the still maintaining the state updates Markovian.
    - Difference with HMM => In typical HMM setup, the observation/feature functions and/or the state transition matrix are independent of the observation set. In MRF/CRF, however, this is not the case (as seen before).

MRF/CRF Axiomatic Definition/Properties

1. MRF Definition: Given an undirected graph $G = (V, E)$, the set of random variables $X = (X_v)_{v \in V}$ indexed by v form a MRF with respect to graph G if they satisfy the following three Local Markov properties (Markov Random Field (Wiki)):
    - Pair-wise Markov Property => Any two non-adjacent variables are conditionally independent given all the other variables, i.e., $X_u \perp X_v \mid X_{v \setminus \{u,v\}}$ if $\{u,v\} \notin E$ ($\perp$ is the symbol for independence).
    - Local Markov Property => A variable is conditionally independent of all other variables given its neighbors, i.e., $X_u \perp X_{v \setminus ClosedNeigh(v)} \mid X_{Neigh(v)}$ where $Neigh(v)$ is the set of neighbors of v and $ClosedNeigh(v)$ is the closed neighborhood of v.

- Global Markov Property => Any two subsets of variables are conditionally independent given a separating subset, i.e., $X_A \perp X_B \mid X_S$ where every path from any node in set A to a node in set B passes through a node in set S.

2. Markov Property Strength: The above three Markov properties are NOT equivalent to each other at all (i.e., none of them telescope any other). In other of strength precedence, the Local Markov Property is stronger than the pair-wise one, but is weaker than the Global Property.
3. Gaussian MRF: A multivariate normal distribution forms an MRF with respect to the graph $G = (V, E)$ if the missing edges correspond to zeros on the precision matrix (the inverse covariance matrix) $X = (X_v)_{v \in V} \sim \mathrm{N}(\mu, \Sigma)$ such that $(\Sigma^{-1})_{uv} = 0$ if $\{u, v\} \notin E$.
4. CRF Definition: For observations $\vec{O}$ and the Random State Variables $\vec{X}$, CRF is defined as (Lafferty, McCallum, and Pereira (2001)): Given an undirected graph $G = (V, E)$ such that $\vec{X} = (\vec{X}_v)_{v \in V}$ such that $\vec{X}$ is indexed by the vertices of G. Then $(\vec{X}, \vec{O})$ is a conditional random field when the random variables X_v, conditioned on $\vec{O}$, obey the following Markov property with respect to the graph: $P(X_v \mid \vec{O}, X_w : w \neq v) = P(X_v \mid \vec{O}, X_w : w \sim v)$ where $w \sim v$ indicates that w and v are neighbors (Conditional Random Field (Wiki)).
   - In other words, CRF is an undirected graphical discriminant model whose nodes can be divided into exactly 2 disjoint sets - $\vec{X}$ and $\vec{O}$ (these two graphs are unconnected). Then conditional distribution $P(\vec{\mathrm{X}} \mid \vec{\mathrm{O}})$ is then modeled.
   - Another view – CRF is essentially an extension of logistic regression applied to sequential data – this form makes is particularly amenable for use in NLP.
5. Higher Order CRF: Higher-order CRF relaxes the single look-back MRF requirement, thereby using the fixed observation set with look-back K. K is typically kept <= 5 to reduce computational costs. Large margin models such as SVM are alternatives to CRF.
   - Semi-Markov CRF employs variable length look-back segmentations (Sarawagi and Cohen (2005))), thereby retaining the power of higher-order CRF to model deep-range dependencies at a reasonable computational cost.

- Linear-chain CRF can be used effectively in conjunction with parallelization (Lavergne, Cappe, and Yvon (2010)).

Clique Factorization

1. Definition: In certain cases (Moussouris (1974) identifies an exception), it may be possible to express the joint distribution probability into a distribution over the "feature base", i.e., the joint probability density may be expressed as $\prod_i P(x_i) \Rightarrow \prod_C \phi_C(x_C)$, where ϕ_C is the probability of a particular configuration in the feature space. ϕ_C is called the factor potential or clique potential.
2. Logistic Formulation using the Clique Factorized Representation:
   - Nomenclature:
     - k => Element index into the clique configuration space (i.e., the clique cardinality).
     - N_k => The number of State Entities in Configuration k.
     - $w_{i,k}$ => Weight of the State Entity in Configuration k, defined as $w_{i,k} = \log\{\phi(C_{i,k})\}$, where $C_{i,k}$ is the i[th] configuration in clique k.
     - $f_{i,k}$ => The Feature Function Indicator in the clique configuration, defined as $f_{i,k}(x_{[k]}) = 1$ if state i is part of the clique configuration C_k, and zero otherwise.
     - $\aleph$ => The universe of the relevant states across all the cliques under consideration.
   - The normalized joint distribution now is $P(X = x) = \frac{1}{Z}\exp\left\{\sum_k {w_k}^T f_k(x_{[k]})\right\}$, where ${w_k}^T f_k(x_{[k]}) = \sum_{i=1}^{N_k} {w_{i,k}}^T f_{i,k}(x_{[k]})$ and $Z = \sum_{x \in \aleph} \exp\left\{\sum_k {w_k}^T f_k(x_{[k]})\right\}$.

- The probability expression above is also called Gibbs' measure, with the only restriction being that, in the partition expression for $\aleph$, there can be no ZERO contributions.

3. Value behind the Partition-Function based Logistic Representation: These formulations derive direct intuitions from statistical mechanics, and therefore ease the computations of expectations for various metrics. For instance, by adding a driving force term J_v for each vertex v in the graph, we may differentiate it to get the expectation as follows:
   - $Z\left[\vec{J}\right] = \sum_{x \in \aleph} \exp\left\{ \sum_k w_k{}^T f_k\left(x_{[k]}\right) + \sum_v J_v X_v \right\}$ => $\left\langle X_v \right\rangle = \frac{1}{Z\left[\vec{J}\right]} \left| \frac{\partial Z\left[\vec{J}\right]}{\partial J_v} \right|_{J_v = 0}$ provides the expectation of X_v.
   - Further, $\rho\left[X_u, X_v\right] = \frac{1}{Z\left[\vec{J}\right]} \left| \frac{\partial^2 Z\left[\vec{J}\right]}{\partial J_u \partial J_v} \right|_{J_u = 0, J_v = 0}$ provides the expectation of the pair-wise correlation function.

Inference in MRF/CRF

1. Alternate Terminology in Use: Most of this involves the observation set, so applicable more to CRF.
   - Model Training => Learn the Conditional distribution $\left(\vec{X}, \vec{O}\right)$ between $\vec{X}$ and $\vec{O}$ using the feature/observation functions from the input corpus of data.
   - "Inference" => Determine the probability of a given label sequence $\vec{X}$ from $\vec{O}$; this implicitly requires learning/calibration to have been already done.
   - Decoding => Determine the most likely label sequence $\vec{X}$ from $\vec{O}$ (honestly, this is one of the prediction operations).
2. Exact Inference in MRF: If we use exact inference as is done in a Bayesian network, it is easy to calculate the conditional distribution of a set of nodes $V' = \left\{v_1, ..., v_i, ..., v_n\right\}$

given another set $W' = \{w_1,...,w_j,...,v_n\}$ by summing over all $u \notin V', W'$. However, this is #P-complete, and is computationally intractable.

- Here V' and W' can be current and the Markov prior states, respectively. Given that this is an MRF framework, observations do not come in here explicitly.
- CRF Inference/Learning => Using MLE, if all state nodes have exponential family distributions and they are all observable/observed, the optimization is convex; therefore, gradient descent, Quasi-Newton etc: maybe used (Sutton and McCallum (2010)).
- CRF Unobserved Inference => If some of the state variables are unobservable, exact inference becomes intractable, and the approximate methods outlined earlier become useful.

3. Approximate Inference: For pure MRF, MCMC, loopy belief propagation, MRF subclasses (using trees – for e.g., Chow-Liu tree) have more feasible polynomial time inference.
    - In case of CRF, approximate inference techniques also include mean field inference, linear programming relaxations, etc:

Maximum Entropy Markov Models (MEMM)

1. Definition: MEMM's are also referred to as Conditional Markov Model. They are discriminant graphical models for sequence labeling that combine the features of HMM and maximum entropy models (Maximum Entropy Markov Model (Wiki)).
2. Sequence Generation given Observations: The probability of generating a sequence S given the observation set O is given as $P(S_1,...,S_n \mid O_1,...,O_n) = \prod_{t=1}^{n} P(S_t \mid S_{t-1}, O_t)$.

   This arises out of the Markov property, namely that the probability of transitioning to a particular label depends ONLY on a) the observation at that location, and b) the earlier label.
3. Clique based Reduction (Berger, Pietra, and Pietra (1996)):

   $P(S_t \mid S_{t-1}, O_t) = \frac{1}{Z(S_{t-1}, O_t)} \exp\left[\sum_a \lambda_a f_a(S_t, O_t)\right]$ where $Z(S_{t-1}, O_t)$ is the normalizer,

   and $f_a(S_t, O_t)$ is the real-valued or categorical-valued feature function.
   - Inference of λ_a => Given that λ_a is extracted out of a calibration process, the generalized iterative scaling technique (Darroch and Ratcliff (1972)) is a popular one if the observations are available. Variant of the Baum-Welch algorithm (McCallum, Freitag, and Pereira (2000)) has been proposed if the training data has missing or incomplete, and has been applied for speech tagging (Toutanova and Manning (2000)) and information extraction (McCallum, Freitag, and Pereira (2000)).
4. Optimal Label Sequence Extraction: Given the observation set O, a variant of the Viterbi algorithm may be used to compute the forward probability:

   $\alpha_{t+1}(S) = \sum_{s' \in S} \alpha_t(s') P(s' \mid S, O_{t+1})$.
5. Drawbacks of MEMM: Two key shortcomings: a) Label bias associated with completely ignoring low-entropy state transitions (Lafferty, McCallum, and Pereira

(2001)), and b) Since the training/calibration occur only with regards to the last known label, MEMM does not work well if there is label uncertainty (Bottou (1991)).

6. Advantage over HMM: Given that it is an undirected discriminant model, MEMM freedom in the choice of features and their functions (same as CRF). HMM relies upon the observations independence, while MEMM does not.
    - Advantage over CRF => Training in MEMM is significantly more efficient, since there is a straightforward one-to-one mapping between the transition probability and the MEMM probability distribution. In HMM/CRF, typically a variant of the forward/backward algorithm needs to be used for the training, which can be expensive.

Probabilistic Grammar and Parsing

Parsing

1. Definition: Parsing is also called syntactic analysis, and is the process of analyzing a string of symbols according to the rules of a formal grammar (Parsing (Wiki)).
2. Traditional Sentence Parsing: This emphasizes the importance of divisions in sentence construction (such as subject/predicate etc), and is done with the help of sentence diagrams.
    - Traditional Method-Clause Analysis is essentially a grammatical exercise to break down the speech form and function and to identify the syntactic relation between the parts. This is, of course, very intricate for highly inflected languages (i.e., languages that employ frequent conjugations/declensions).
    - Problems with human languages => Structural ambiguity, i.e., "man bites dog" in language #1 is the same as "dog bites man" in language 2, i.e., the interpretation needs to rely on a bigger context established by the language grammar rules, which are often hard to clearly elicit.
    - Challenges with a single grammatical framework => Even if one grammar is chosen, the grammar parsing system must be established, with the typical choices being lexical functional grammar parser (which is NP-complete!), head-driven grammar parsing (which can get very complex), shallow parsing (which only identifies the boundaries between the major sentence constituents), and any of these augmented by the dependency grammar parsing, etc:
3. Computational Linguistic Parsing: This is done using a formal computer analysis on a word string and its constituents, resulting in a parse tree that displays the syntactic relation, semantic content, and other information between those words.
    - Modern statistical parsing => Modern statistical parsers are trained on a corpus of data, using a context-based frequency of occurrence metric, with enhancements to

incorporate probabilistic context free grammars, lexical statistics, and smoothing to avoid over-fitting.

4. Psycholinguistic Parsing: This refers to the way humans analyze a sentence/word in terms of its grammatical constituents, identifying the parts of speech and their syntactic relations etc: These techniques are used by psychologists to describe language comprehension and to provide cues to help speakers interpret/fix garden-path (i.e., wrongly structured) sentences.
    - Psycholinguistic parsing evaluates the meaning of the sentence according to the rules drawn by the inferences made from each word in the sentence.
    - Psycholinguistic parsing is also incremental in that the interpretation/structure is constructed right through the processing, and expressed in terms of the partial syntactic structure, with care to avoid garden-pathing.
5. Programming Language Parsing: This refers to the syntactic analysis of the input computer code into its component parts to facilitate the tasks of compilers/interpreters.
    - Steps:
        - Lexical analyzer breaks the input into tokens
        - Syntactic analyzer (true parser) verifies the validity of the sequence to check if they constitute a valid expression (done using the grammar rules)
        - Semantic analyzer converts the sequence into the appropriate operational/execution units (e.g., generates machine code, or the interpretation execution).
    - The typical output of syntactic parsing is the parse tree (either as an abstract syntax tree or as an equivalent hierarchical structure (Tree Structure (Wiki))).

Parser

1. Context-free Parsing Grammar: Context-free grammars are limited in the extent to which they can be expressive of all the language requirements; however, they are

simpler than context-oriented grammars to need to choose in/switch out of the context (based upon an additionally established criterion).

2. Formal Definition: The task of a parser is to determine of and how the input can be derived from the start symbol of the grammar. It is conducted in essentially 2 ways – top-down parsing, and bottom-up parsing.
3. Terminology and Nomenclature:
    - LL => Left-to-tight token processing, left-most derivative
    - LR => Left-to-tight token processing, right-most derivative
    - LALR => Look-ahead LR parser
    - Production => Production is the result of conversion of a single token into its corresponding part-of-speech.
    - Derivation => A set of productions together constitute a derivation or a parse. Typically, notions of syntactic validity are enforced at this stage, as the "set" may correspond to a syntactic isolation unit (e.g. a "sentence" in a natural language, or a "statement" in programming language).
4. Top-down Parsing: Top-down parsing attempts to find the left-most derivations of an input stream by searching for the parse trees using a top-down expansion of the given formal rules. Tokens are consumed left to right (Aho, Sethi, and Ullman (1986)).
5. Bottom-up Parsing: Here, the parser typically attempts to locate the most basis elements, followed by the elements containing these, etc: This is also referred to a shift-reduce parsing. LR parsers are examples of bottom-up parsers.
    - Enhanced top-down parsing => A new family of sophisticated algorithms for top-down LL parsing of ambiguous, context-free grammars (Frost, Hafiz, and Callaghan (2007), Frost, Hafiz, and Callaghan (2008)) accommodate ambiguity and left-recursion in polynomial time, and polynomial-size representations of potentially exponential number of parse trees (again in polynomial time). These algorithms can produce both left-most and right-most derivations of a given input.
6. Look-ahead Parsing: Look-ahead parsing establishes the maximum number of incoming tokens that a parser can use to decide which rule it can invoke. It is represented as LALR (number of tokens).

Context-Free Grammar (CFG)

1. Probabilistic/Stochastic Context-Free Grammar (PCFG/SCFG) Definition: In this context-free grammar, each production is augmented with a production/translation probability, and thus the probability of the entire derivation becomes a product of the individual productions used in the derivation. Alternately, this probability is indicative of the probability that the derivative is consistent with the given grammar.
2. Estimation of the Probability of most likely Derivation: A variant of the CYK algorithm finds the Viterbi parse of the sequence for a given SCFG – the Viterbi parse that is most likely derivative/parse of the sequence given the SCFG.
3. Probability of all the Generatable Sets associated with the given Sequence: Use the inside-outside algorithm, a variant of the forward-backward algorithm. This may be used in conjunction with expectation-maximization algorithms to learn the maximum likelihood probabilities of a SCFG based off of the given set of training sequences (Ayciena (2005)).
4. CFG as a Natural Language Model: Natural Language grammars, conceived as sets of production rules (syntaxed, say, with the Backus-Naur form), are absolute (Chomsky (1957)), but inadequate (except in programming language type situations).
    - PCFG/SCFG as a way to handle Natural Language Complexities => CFG's are one way to ensure that more than one production rule may apply to a given sequence of words. CFG-based approach is better than approaches that use fixed-set production rules for the following reasons:
        - Having a deep-set production rule proliferate (to add more fixed rules as necessary) makes it difficult to manage (despite using rule precedence hierarchy).
        - Results in over-generation (i.e., generation of valid yet highly unlikely structures).
        - Prevents parsing without context, thereby cannot apply CFG's.

- How CFG handles this => CFG's address the issues above by assigning a probability to the given derivation, and working out a winner-takes-all (i.e., the most likely) interpretation.

5. Learning Grammar for SCFG: Learning happens from large corpus of annotated texts. Typical sources include the Penn Tree-Bank that trained the Stanford Statistical Parser (Klein and Manning (2003)), Speech Recognition Tuner (SCFG parsing implementation is available in Beutler, Kaufman, and Pfister (2005)).
6. RNA Sequence Mapping: Nucleotide secondary structure base-pairing may be represented with the grammar $S \rightarrow aSu \mid cSg \mid gSc \mid uSa$ (this admits only wholly complementary regions of canonical pairs $a-u$ and $c-g$ (Durbin, Eddy, Krogh, and Mitchison (1999), Eddy and Durbin (1994))). Thus, given a genome sequence, SCFG may use this grammar to find the RNA genes (e.g., see Stochastic Context-Free Grammar (Wiki)).
7. SCFG in Psycholinguistics: SCFG based models (Horning (1969) and Clark (2001)) have helped revise earlier nativist views that language is hard-wired in humans at birth (Gold (1967) and Chomsky (1980)).
8. Probabilistic Grammars in Cognitive Plausibility: Probabilistic versions of minimalist grammars (Hale (2006)) have helped evaluate psycholinguistic models and assess product difficulties associated with different syntactic structures (e.g., accessibility hierarchy for relative clauses).

Tensors and Multi-linear Subspace Learning

Tensors

1. Definition: Tensors are geometric objects used to describe linear relationships between scalars/vectors/other tensors. Order/Degree/Rank of a tensor is the dimensionality needed to represent the tensor (e.g., scalars are tensors of rank zero). See Reich (1994), Hamilton (1854-1855), Voigt (1898), Pais (2005), Goodstein (1982), and Tensor (Wiki) for the origin and the varied uses of tensors.
2. Families of Predictor Ordinates: Each family of the spanning set of predictor ordinates of the tensor corresponds to a single predictor ordinate dimension of the tensor. For example, 3D pixel predictor ordinate families may consist of a) x, y, and z location co-ordinate family, and b) RGB color family of predictors.
3. Covariant vs. Contravariant Tensor Components: If a tensor component transforms as the tensor itself, it is called a covariant component, and is represented using a subscript index. If a component transforms as the inverse of the tensor, it is called a contravariant component of the tensor, and is represented using a superscript. For instance, in a 2D tensor matrix, the row indices would be covariant and the column indices would be contra-variant.
    - Family Spanning Set #1 <-> Family Spanning Set #2 Transforming Tensor => This is a transforming tensor, with source family spanning set #1 acting as the covariant basis component tensor, and the destination family spanning set #2 acting as the contra-variant basis component tensor.
4. Expressing Covariant and Contravariant Tensors as Multi-dimensional Arrays: The transformation law for an order m tensor with n contra-variant indices and $m-n$ covariant indices is given as (Marion and Thornton (1995), Sharpe (1997), Griffiths (1999)) $\hat{T}^{i_1,\ldots,i_n}_{i_{n+1},\ldots,i_m} = \left(R^{-1}\right)^{i_1}_{j_1}\ldots\left(R^{-1}\right)^{i_n}_{j_n}\left(R\right)^{j_{n+1}}_{i_{n+1}}\ldots\left(R\right)^{j_m}_{i_m}\hat{T}^{j_1,\ldots,j_n}_{j_{n+1},\ldots,j_m}$. This expression uses the Einstein notation to sum over $j_1,\ldots,j_n$ (i.e., the summation is implicit).

5. Expressing Tensors as Field Transformation Jacobians (or Tensor Fields): Here, the transformation law is expressed in terms of the partial derivatives of the co-ordinate functions $\bar{x}(x_1,...,x_k)$, thereby defining the transformation in terms of the Jacobian (Kline (1972), Curbastro (1892)) as

$$\hat{T}^{i_1,...i_n}_{i_{n+1},...,i_m}(\bar{x}_1,...,\bar{x}_k) = \frac{\partial \bar{x}^{i_1}}{\partial x^{j_1}} ... \frac{\partial \bar{x}^{i_n}}{\partial x^{j_n}} \frac{\partial x^{j_{n+1}}}{\partial \bar{x}^{i_{n+1}}} ... \frac{\partial x^{j_m}}{\partial \bar{x}^{i_m}} \hat{T}^{j_1,...,j_n}_{j_{n+1},...,j_m}(x_1,...,x_k).$$

6. Expression as Tensor Products or Multi-linear Maps: A type (m,n) tensor is defined as an element in the tensor product of vector spaces (Hazelwinkel (2001a)) as $T \in \{V \otimes ... \otimes V\} \otimes \{V^* \otimes ... \otimes V^*\}$, where the first term in the bracket spans n copies of the vector space V, and the second term in the bracket spans m copies of the vector space V^*. Thus, if $\hat{v}_i$ is the basis of $\vec{V}$ and $\hat{w}_j$ is the basis of $\vec{W}$, then $\vec{V} \otimes \vec{W}$ has the natural basis $\hat{v}_i \otimes \hat{w}_j$.
7. Expressed using Penrose Graphical Notation: In the diagrammatic notation (Penrose (2007), Wheeler, Misner, Thorne (1973)), tensor symbols are replaced by shapes, and indices by lines and curves!
8. Tensor Operations:
   - Tensor Product => $S(l,k) \otimes T(n,m) = S \otimes T(l+n, k+m)$
   - Tensor Contraction => This reduces the tensor order by 2, i.e., $S(l,k) \Rightarrow S(l-1,k-1)$
   - Raising an Index => Changes a Covariant Index to a Contravariant Index.
   - Lowering an Index => Changes a Contravariant Index to a Covariant Index.
9. Infinite Dimensional Tensors: Infinite dimensional tensors may be generalized via the tensor product of Hilbert spaces (Segal (1956)), or by extending multi-maps that employ infinite-dimensional vector spaces and their algebraic duals (this is automatically achieved by using infinite-dimensional Banach manifolds and their continuous duals (Abraham, Marsden, and Ratiu (1988), Lang (1972))).

10. Tensor Density: A tensor with density r transforms like an ordinary tensor under co-ordinate transformations (Hazelwinkel (2001b)), with the exception that is scaled by the determinant of Jacobian to the power r (i.e., $\left\| \frac{\partial \hat{T}}{\partial \bar{x}} \right\|^r$).

Multi-linear Subspace Learning

1. Definition: Multi-linear Subspace Learning (MSL) aims to learn a specific part of a large space of multi-dimensional objects having a desired property. Essentially, MSL is a dimension reduction technique for finding low dimension representation with preferred characteristics, without resorting to vectorization (Lu, Plataniotis, and Venetsanopoulus (2011), He, Cai, and Niyogi (2005), Multi-linear Subspace Learning (Wiki)).
2. MSL as a Generalization of PCA: MSL may also be viewed as a higher order PCA/CCA/LDA (Vasilescu and Terzopoulus (2007)).
3. Challenges with Vectorized Linear Subspace Learners: Since they represent input data as vectors and attempt to solve for optimal mapping to the lower dimensional space, vectorized subspace learners become:
   - Inadequate when dealing with massive multi-dimensional data.
   - Need estimation of a large number of parameters.
   - Break down the structure and correlation inherent in the original data (Yan, Xu, Yang, Zhang, Tang, and Zhang (2005), Lu, Plataniotis, and Venetsanopoulus (2008)).
4. MSL vs. Tensor Decomposition: Both are similar in that both use multi-linear algebra (Kolda, and Bader (2009)), but differ in that while tensor decomposition focuses on factor analysis, MSL focuses on dimensionality reduction.
5. Multi linear Subspace Definition: This simply refers to the multi-linear projection that maps the input tensor data from a higher dimensional space to a lower dimensional space (Hitchcock (1927)).

6. Tensor-to-Tensor Projection (TTP): This is a direct projection of a high dimensional tensor to a low dimensional tensor of the same order, using N projection matrices for the order N tensor – it is simply an extension of HOSVD (Tucker (1966), Lathauwer, Moor, and van de Walle (2000a)).
7. Tensor-to-Vector Projection (TVP): This is a projection from a high dimensional tensor to a low dimension vector, and is also referred to as a Rank-1 projection. A TVP of a tensor to a P-dimensional vector consists of P projections from the tensor to a scalar – each projection is called an EMP (elementary multi-linear projection) (Carroll and Chang (1970), Harshman (1970)).
8. MSL Solution Approach: N set of parameters need to solved, one per each mode. The following sub-optimal approach is followed (Kroonenberg and de Leeuw (1980), Lathauwer, Moor, and van de Walle (2000b)):
    - Initialize a set of projections in each mode.
    - Fix all but one projection, and solve for that fixed projection.
    - Perform mode-wise optimization until convergence.
9. Multi-linear Extensions to PCA:
    - TTP based MPCA (Lu, Plataniotis, and Venetsanopoulus (2008))
    - TVP based uncorrelated MLPCA (UMPCA) (Lu, Plataniotis, and Venetsanopoulus (2009b)).
10. Multi-linear Extensions to LDA:
    - TTP based Discriminant Analysis with Tensor Representation (Yan, Xu, Yang, Zhang, Tang, and Zhang (2005)).
    - TTP based General Tensor Discriminant Analysis (GTDA) (Tao, Li, Wu, and Maybank (2007)).
    - TTP based Uncorrelated Multi-linear Discriminant Analysis (UMLDA) (Lu, Plataniotis, and Venetsanopoulus (2009a)).
11. Multi-linear Extensions to CCA:
    - TTP based Tensor Canonical Correlation Analysis (TCCA) (Kim and Cipolla (2009)).
    - TVP based Multi-linear Canonical Correlation Analysis (MCCA) (Lu (2013)).

Multi-linear PCA

1. Definition: MPCA is a mathematical procedure that uses multiple orthogonal transformations to convert a set of multi-dimensional objects into another set of multi-dimensional objects of lower dimension.
2. Purpose: The aim is to capture as high a variance as possible, accounting for as much of the variability in the data as possible, subject to the constraint of mode-wise orthogonality.
3. PCA as Response De-convolution: Given an array of responses, one way to look at PCA/ICA is as a process to extract/infer the individual, independent predictor drivers from the observation responses.
4. MPCA vs. Regular PCA: PCA needs to reshape the multidimensional object into a vector, while MPCA works directly on the multi-dimensional objects through mode-wise processing. For example, PCA converts a 100x100 image into a vector of size 10,000x1, while MPCA processes the same 100x100 using 2 100x1 vectors. Thus, MPCA results in savings of 50 in processing time in this case.
5. Use of MSL Techniques in MPCA: Like MSL, MPCA uses tensor based dimensionality reduction techniques (Lu, Plataniotis, and Venetsanopoulus (2008)), therefore MSL approaches such as Tucker decomposition (Tucker (1966)), HOSVD (Lathauwer, Moor, and van de Walle (2000a)), and best-rank (R1, R2, …,, RN) approximation of higher-order tensors (Lathauwer, Moor, and van de Walle (2000b)) are all applicable here.
6. The MPCA Algorithm: MPCA performs feature extraction by determining the multi-linear projection that captures most variations in a given mode. It works on centered data, and typically uses the altering least squares (ALS) approach (Kroonenberg and de Leeuw (1980)), by alternating across each mode.
    - ALS decomposes the original tensor into multiple projection sub-problems, each of which is a classical PCA, and therefore easily solved.
7. Retention of the Feature Correlations: Because of the tensor-to-tensor nature of the transformation, MPCA features are not uncorrelated in general, although the

transformation in each mode is orthogonal (to generate uncorrelated features UMPCA (Lu, Plataniotis, and Venetsanopoulus (2009b)) is used).

8. Feature Selection in MPCA: While conventional classifiers often use only vector features, specialized tensor feature selection to enhance/improve MPCA performance maybe used in specific situations (examples are supervised discriminant MPCA feature selection for object recognition (Lu, Plataniotis, and Venetsanopoulus (2008)), and unsupervised MPCA feature selection for visualization tasks (Lu, Eng, Thida, and Plataniotis (2010)).
9. MPCA Extensions:
    - Uncorrelated MPCA (UMPCA) (Lu, Plataniotis, and Venetsanopoulus (2009b))
    - Boosting + MPCA (Lu, Plataniotis, and Venetsanopoulus (2009c))
    - Non-negative MPCA (NMPCA) (Panagakis, Kotropoulus, and Arce (2010))
    - Robust MPCA (RMPCA) (Inoue, Hara, and Urahama (2009))
    - More extension are detailed in (Lu, Plataniotis, and Venetsanopoulus (2011))

Pattern Recognition

Introduction

1. Purpose: Pattern Recognition is the process associated with assignment of the label to a given input value/value set, e.g.;
   - Classification => Each input value is assigned to one among the given set of classes (e. g., Spam/non spam classes).
   - Regression => Assign a real-value to an output from an input stream.
   - Sequence Labeling => Assign a class to each member of the input sequence of values (e.g., part-of-speech tagging).
   - Parsing => Assign a parse-tree to an input sentence, describing the syntactic structure of the sentence.
2. Pattern Recognition vs. Pattern Matching: Pattern Recognition algorithms aim to provide a reasonable answer for all possible inputs and to perform the "most likely" matches on the inputs, taking into account their statistical variation. Pattern matching looks for exact matches in the input against pre-existing patterns (e.g., in regular expression parsing/matching).

Supervised vs. Unsupervised Pattern Recognition

1. Learning Procedure for Supervised Learning: The learning procedure generates a model that meets possibly conflicting objectives: a) Perform as well as possible on the hand-labeled training data, and b) Generalize as well as possible to new data.
2. Supervised vs. Unsupervised Terminology Clarification: Unsupervised equivalent of classification is called clustering, based on the notion that clustering into groups results from say, using a distance-based metric.

3. Instance of Input Data: This is the same as one part of input data that is described by a vector of features, and is used to fully characterize that instance. Features could be categorical, ordinal (e.g., fist, second etc), integer/real valued.

Probabilistic Pattern Recognition

1. Advantages of Probabilistic Pattern Matching:
   - These output a probabilistic confidence value with each answer choice.
   - These may choose N best outcomes and their confidence values, ranked according to their inference probabilities (esp. if N is small, as in classification).
   - Chained probabilities may be readily incorporated into larger learning tasks in such as way as to either discard a choice, so as to avoid error propagation.
   - Class Probability as a Performance Metric: In addition, a) they are not affected by the relative class sizes (Mills (2011)), and b) it imposes no penalties for simply re-arranging classes.
2. Feature Selection: Feature Selection attempts t prune out redundant and/or irrelevant features (Clopinet and Elisseeff (2003)). Complexity here arises from the need to process the entire $m^n - 1$ feature power-set, given m realizations per feature. While branch-and-bound algorithms (Foroutan and Sklansky (1987)) may reduce the complexity, they become rapidly untenable for large n (Kudo and Sklansky (2000)).
3. Feature Extraction: Feature Extraction is an alternate to Feature Selection. Here the raw feature vectors are first transformed to reduce the dimensionality and the redundancy using PCA/ICA variants. The features after feature selection may appear very different from the ones before. Feature selection may still follow feature extraction.

Formulation of Pattern Recognition

1. Formal Supervised Pattern Recognition Problem Statement: Given an unknown function $g : X \to Y$ (the ground truth) that maps input instances $x \in X$ to output labels $y \in Y$, along with training data $D = \{(x_1, y_1),...,(x_n, y_n)\}$ assumed to represent accurate examples of the mapping, produce a function $f : X \to Y$ that approximates g as closely as possible.
2. "As Closely As Possible": In decision theory, this is defined by specifying a loss function that assigns a specific value to the "loss" resulting from the production of an "incorrect" label. The goal is minimize the risk/loss function.
3. Generative vs. Discriminant Probabilistic Pattern Recognition: Discriminant pattern recognition estimates the probability $p(label \mid \vec{X}, \theta) = f(\vec{X}, \theta)$ of the label given the observations, where $\vec{X}$ is the feature vector over the input observations, and θ is the parameterization. With generative pattern recognition, the inverse probability of the observations given the labels $p(\vec{X} \mid label)$ is first estimated, and then combined with the prior probability $p(label \mid \theta)$ using Bayes' rule as

   $p(label \mid \vec{X}, \theta) = \frac{p(\vec{X} \mid label) p(label \mid \theta)}{\sum_{L \in \{AllLabels\}} p(\vec{X} \mid L) p(L \mid \theta)}$ for discrete labels, and

   $p(label \mid \vec{X}, \theta) = \frac{p(\vec{X} \mid label) p(label \mid \theta)}{\int_{L \in \{AllLabels\}} p(\vec{X} \mid L) p(L \mid \theta) d\theta}$ for continuous labels.
4. MLE Inference for θ: Here, θ is estimated basically as the point-value maximum of $\theta^* = \overset{\arg\max}{\theta} p(\theta \mid \vec{D})$ where $p(\theta \mid \vec{D}) = \left[\prod_{i=1}^{n} p(y_i \mid x_i) \right] p(\theta)$. This is still Bayesian, except that here θ is replaced by its point-value maximum θ^*.
5. Generative vs. Discriminant Probabilistic Pattern Recognition: Discriminant pattern recognition estimates the probability $p(label \mid \vec{X}, \theta) = f(\vec{X}, \theta)$ of the label given the observations, where $\vec{X}$ is the feature vector over the input observations, and θ is the parameterization. With generative pattern recognition, the inverse probability of the observations given the labels $p(\vec{X} \mid label)$ is first estimated, and then combined with

the prior probability $p(label \mid \theta)$ using Bayes' rule as

$$p(label \mid \vec{X}, \theta) = \frac{p(\vec{X} \mid label)p(label \mid \theta)}{\sum_{L \in \{AllLabels\}} p(\vec{X} \mid L)p(L \mid \theta)} \text{ for discrete labels, and}$$

$$p(label \mid \vec{X}, \theta) = \frac{p(\vec{X} \mid label)p(label \mid \theta)}{\int_{L \in \{AllLabels\}} p(\vec{X} \mid L)p(L \mid \theta)d\theta} \text{ for continuous labels.}$$

6. MLE Inference for θ: Here, θ is estimated basically as the point-value maximum of $\theta^* = \overset{\arg\max}{\theta} p(\theta \mid \vec{D})$ where $p(\theta \mid \vec{D}) = \left[\prod_{i=1}^{n} p(y_i \mid x_i)\right] p(\theta)$. This is still Bayesian, except that here θ is replaced by its point-value maximum θ^*.

7. Bayesian Approach: This integrates over all possible θ, weighted according to the posterior. Bayesian lets you explicitly specify $p(\theta)$ (presumably by drawing into past experience), from $p(Label \mid \vec{X}) = \int_{\theta} p(Label \mid \vec{X}, \theta)p(\theta \mid \vec{D})d\theta$.

8. Pattern Recognition Applications:
   - Automatic Recognition of Sub-topic Images/Hand-writing (Duda, Hart, and Stork (2001), Milewski and Govindaraju (2008), Brunelli (2009)).
   - Identification/Authentication, e.g., License Plate Recognition, Finger-printing, face detection/verification.
   - Medical diagnostics, e.g., PAPNET/Tumor Screening.
   - Defense Navigation/Guidance, Target Recognition (Egmont-Peterson, de Ridder, and Handels (2002)).
9. Specialized Psychology Applications: As related to psychology and perception, pattern recognition may be understood as being multi-staged:
   - Stage #1 => This is consists of template matching, where the incoming stimuli are compared with templates (i.e., patters used to produce items of the same proportion) in the long-term memory, AND
   - Stage #2 => If there is a template match, the secondary feature detection models are triggered.

Pattern Recognition Practice SKU

1. Model Selection: Identify stochastic relations between the feature vectors and the categories to be predicted. This brings both model selection and feature selection into focus (Wolpert (2001)). In general, fewer the parameters, the better.
2. Model Determinants: In addition to the simplicity and the performance of the algorithms and the models, the following criteria are also important:
    - Parametric Prior Distribution
    - Whether the classifier can cope with the missing features
    - Whether changes in class prior probabilities can be incorporated
    - Speed of the trainer, and the corresponding memory required
    - Parallelizability
    - Closeness of resemblance/proxying of human perceptions
    - Any target-specific features (i.e., in Visual Pattern Recognition, variations are needed for color, rotation, scale etc:)
3. Optimal Bayes' Classifier: The theoretically optimal Bayes' classifier minimizes the loss risk-function. When all types of mislabeling are associated with equal loss intensities (i.e., outcome A becoming B is as undesirable as outcome B becoming A), the Bayes' classifier with the minimal error rate (on the given training set) is the optimal one for the classification task.
    - Given that the error term is a convolution of a) the error magnitude/loss intensity for a given vector feature configuration, and b) the probability of the feature vector configuration, in general it is unknown what the optimal classifier type and the parameter set are. However, upper bound on the error-rate may be worked out in specific classifier schemes (e.g., in the case of K-nearest neighbor).
4. Supervised Classification Pattern Recognition Practice Steps (Pattern Recognition (Wiki)):
    - Separate the available data, at random, into a training set and a test set. Test set is used only for the final performance evaluation of the trained set.

- Experiment by training a number of classification algorithms, including parametric (e.g., discriminant analysis, multinomial classifier (Glick (1973))), and non-parametric algorithms (K-nearest neighbor, support vector machines, feed-forward neural net, standard decision-tree, etc:).
- Test distribution assumptions of the continuous features – including distributions per category (Gaussian?).
- Which subset of the feature vector contributes most to the discriminative performance of the classifier?
- Work out the detailed confidence intervals for the error-rates and the class-predictions (McLachlan (2004)).
- White box vs. black box considerations may render specific classifiers unsuited for the task.

Kalman Filtering

1. Original Formulation: Despite the name, Kalman filtering had already been formulated by others – see Lauritzen (1981), Lauritzen (2002), Stratonovich (1959a), Stratonovich (1959a), Stratonovich (1959a), and Stratonovich (1959a). Kalman Filter (Wiki) contains additional references.
2. Atypical Applications: In addition to the most applied areas such as guidance, navigation, and control of vehicles such as spacecraft/aircraft, and computer vision, Kalman filtering also applied in structural macroeconomic models (Strid and Walentin (2009), Andreasen (2008)), tracking and vertex fitting of charged particles in particle detectors (Fruhwirth (1987)), and human sensorimotor processing (Wolpert (1996)). Of particular relevance to illiquid trading may be the application of Kalman Filtering for the recovery of sparse, dynamic signals using restricted isometry and probabilistic recovery (Carmi, Gurfil, and Kanevsky (2010) and Vaswani (2008)).
3. Kalman Filtering vs. Hidden Markov Models: While there is a lot of similarity, there are a few critical differences. First, the hidden state variables in Kalman filtering are continuous, whereas in HMM they are discrete. However, the HMM can represent an arbitrary distribution for the state variables, whereas Gaussian noise models are used in Kalman filtering. The parallel between the state equations using HMM and the Kalman filtering treatments is compared in Hamilton (1994) and Roweis and Ghahramani (1999).
    - Markov Processes vs. Filtering => Although filtering processes seem to be treated in conjunction with HMM, filtering processes do not need to be Markov at all. However, there may be an impact on real-time applicability for non-Markov systems.
    - Extended Markov Systems => Perhaps using a few additional past observations may still not overtly compromise the speed of computation of what was a Markov process, but may improve the estimation quality (for e.g., using the limited Volterra expansions). This is NOT the same as Kalman smoothing, however.

4. Dempster-Shaefer Theoretical Underpinning: Under the Dempster-Shaefer theory, each state equation/observation is the result of a linear belief function, and the Kalman filter results under the special case of combining the linear belief functions on a Markov tree.
5. Definition: The Kalman Filter, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, containing noise (random variations), and produces estimates of unknown state variables that tend to be more precise than those based on single measurement alone.
6. State Definitions:
   - $\hat{x}_{k|k-1} \Rightarrow$ This is the a priori state estimate at time k given all the observations to the instant $k-1$.
   - $\hat{x}_{k|k} \Rightarrow$ This is the a posteriori state estimate at time k given all the observations to the instant k, and after applying the appropriate gain adjustment to the measurement.
   - $x_k \Rightarrow$ The actual state at time k.
7. Covariance Definitions:
   - $P_{k|k} = Covariance\left(x_k - \hat{x}_{k|k}\right)$
   - $P_{k|k-1} = Covariance\left(x_k - \hat{x}_{k|k-1}\right)$
   - $S_k = Covariance\left(\tilde{y}_k\right)$. $\tilde{y}_k$ will be defined later.
8. A Priori State Estimation Model of the Kalman Filter: $\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k + w_k$
   - F_k is the state transition model applied to the previous a posteriori state estimate $\hat{x}_{k-1|k-1}$.
   - B_k is the control-input model that is applied to the control vector u_k.
   - w_k is the process noise that zero-mean multivariate normal with co-variance Q_k.
   - The corresponding a priori error covariance matrix estimate is $P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$.
9. Measurement of the True State: $z_k = H_k x_k + v_k$

- H_k is the observation model operator that maps the true state space into the observed space.
- v_k is the observation noise that is a zero-mean Gaussian white noise with co-variance R_k.
- z_k is the observation/measurement of the true state x_k.

10. Assumption of the Variable Independence in the Kalman Filter: The initial state and the noise vector at each step $\{x_0, w_1, ...w_k, v_1, ..., v_k\}$ are all mutually independent.
11. Innovation/Update Phase of the Kalman Filter: The innovation of the measurement residual is $\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1}$. This residual is generated by applying the measurement operator on the a priori state estimate $\hat{x}_{k|k-1}$.
12. Methodology Separation in the Kalman Filter: The Kalman Filtering methodology seeks to make the a priori state estimate and the a priori error covariance estimate separately from the a posteriori state estimate and the a posteriori error covariance estimate.
13. A Posteriori State Estimation: The a posteriori state estimate is typically expressed in terms of the Kalman gain K_k, i.e., $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$.
    - Observation weighting vs. Model weighting through Kalman Gain => Expanding out $\tilde{y}_k$, we get $\hat{x}_{k|k} = (I - K_k H_k)\hat{x}_{k|k-1} + K_k z_k$. To uncover the intuition behind this, set $H_k = I$, to get $\hat{x}_{k|k} = (I - K_k)\hat{x}_{k|k-1} + K_k z_k$. Thus, here the unit gain $K_k = 1$ weights $\hat{x}_{k|k}$ toward the measurement z_k, whereas the zero gain $K_k = 0$ weights the a posteriori estimate towards the model.
14. Kalman Filter Invariants:
    - $\langle x_k - \hat{x}_{k|k} \rangle = \langle x_k - \hat{x}_{k|k-1} \rangle = 0$
    - $\langle \tilde{y}_k \rangle = 0$
15. The Optimization Step in Kalman Gain: Optimization is really only applied to the a posteriori error covariance, because that is where you can optimize using the Kalman gain.
16. A Posteriori Covariance Formulation:

- $P_{k|k} = \text{covariance}\left[x_k - \hat{x}_{k|k}\right] = \text{covariance}\left[x_k - \left\{\hat{x}_{k|k-1} + K_k \tilde{y}_k\right\}\right]$
- $P_{k|k} = \text{covariance}\left[x_k - \left\{\hat{x}_{k|k-1} + K_k\left(z_k - H_k \hat{x}_{k|k-1}\right)\right\}\right]$
- $P_{k|k} = \text{covariance}\left[x_k - \left\{\hat{x}_{k|k-1} + K_k\left(H_k \hat{x}_{k|k} + v_k - H_k \hat{x}_{k|k-1}\right)\right\}\right]$
- $P_{k|k} = \text{covariance}\left[\left(I - K_k H_k\right)\left(x_k - \hat{x}_{k|k-1}\right)\right] + \text{covariance}\left[K_k v_k\right]$
- Here we have used the fact that $\text{covariance}(A - B) = \text{covariance}(A) + \text{covariance}(B)$ if $\text{covariance}(AB) = 0$.
- $P_{k|k} = \left(I - K_k H_k\right)\text{covariance}\left[\left(x_k - \hat{x}_{k|k-1}\right)\right]\left(I - K_k H_k\right)^T + K_k\,\text{covariance}\left[v_k\right]K_k^{\,T}$
- $P_{k|k} = \left(I - K_k H_k\right)P_{k|k-1}\left(I - K_k H_k\right)^T + K_k R_k K_k^{\,T}$

17. A Priori/A Posteriori Prediction/Update Summary:
    - A Priori State Prediction => $\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k + w_k$.
    - A Priori State Error Covariance Estimation => $P_{k|k-1} = F_k P_{k-1|k-1} F_k^{\,T} + Q_k$.
    - A Posteriori State Prediction/Update => $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$.
    - A Posteriori State Error Covariance Estimate => $P_{k|k} = \left(I - K_k H_k\right)P_{k|k-1}\left(I - K_k H_k\right)^T + K_k R_k K_k^{\,T}$. This form of a posteriori state error covariance is called the Joseph's form. It is true for any Kalman system, not just the optimal system.
18. Re-expansion and Formulation of the Kalman Gain:
    - $P_{k|k} = P_{k|k-1} - K_k H_k P_{k|k-1} - P_{k|k-1} H_k^{\,T} K_k^{\,T} + K_k H_k P_{k|k-1} H_k^{\,T} K_k^{\,T} + K_k R_k K_k^{\,T}$
    - $P_{k|k} = P_{k|k-1} - K_k H_k P_{k|k-1} - P_{k|k-1} H_k^{\,T} K_k^{\,T} + K_k S_k K_k^{\,T}$, where we use the fact that $S_k = H_k P_{k-1|k-1} H_k^{\,T} + R_k$. Remember that S_k is the variance of $\tilde{y}_k$, i.e., $S_k = \left\langle \tilde{y}_k^{\,2} \right\rangle$. Further, notice the similarity between S_k and $P_{k|k-1}$, where we have switched H_k for F_k and R_k for Q_k - this results from identically analogous input drivers.
19. Minimization of the A Posteriori Error Vector: Minimization of $x_k - \hat{x}_{k|k}$ is the same as minimization of the trace of $P_{k|k}$.
20. Optimal Kalman Gain Formulation:

- Trace Minimization of $P_{k|k}$ => $\frac{\partial Trace\left(P_{k|k}\right)}{\partial K_k} = -2\left(H_k P_{k|k-1}\right)^T + 2K_k S_k$
- $\frac{\partial Trace\left(P_{k|k}\right)}{\partial K_k} = 0 \Rightarrow K_k S_k = \left(H_k P_{k|k-1}\right)^T = P_{k|k-1} H_k{}^T$ where we've used the fact $P_{k|k-1} = P_{k|k-1}{}^T$. Thus, the optimal Kalman Gain is $K_k = P_{k|k-1} H_k{}^T S_k{}^{-1}$.
- $\frac{\partial^2 Trace\left(P_{k|k}\right)}{\partial K_k{}^2} = S_k > 0.$ Thus, the optimal K_k corresponds to the desired minimum.
- Corresponding Optimal A Posteriori Covariance => $P_{k|k} = \left(I - K_k H_k\right) P_{k|k-1}$.
- Optimal Kalman Gain is Inversely Proportional to the Observation Noise Variance => Thus, more weight (i.e., less gain) is applied to the predicted estimate in cases where the measurement noise variance is high.
- High Gain => Closer to measurements, therefore the filter becomes more responsive, but also jumpy (non-smooth).

21. Estimation of the Noise Covariances: Estimation of R_k and Q_k is often difficult. A recent promising method is the method of Autocovariance Least Squares, as laid out in Rajamani (2007), Rajamani and Rawlings (2009), and the reference on Autocovariance Least Squares.
    - Remember that auto-regression is done on the innovation residual, as that is where the "update" rubber meets the "measurement" road. To eliminate/reduce bias, the expectation is that the innovation residual should be composed of evenly spread white noise.
    - Further, the sensitivity Jacobian of the Filter to the noise covariances indicate how robust the estimator is to the potentially misspecified noise and other statistical parameters (Anderson and Moore (1979)).
22. Filter Performance Estimation: Optimal performance of Kalman Filter can be cross-checked with the quality of white noise generated by the innovation sequence (see e.g., Matisko and Havlena (2012)).

23. Numerical Stability of the Filtering Algorithm Sequence: If the process noise Q_k is small, round-off problems result in $P_{k|k-1}$ becoming non-positive-definite. Algorithms to improve stability include:
    - Cholesky Factorization => Decompose P as SS^T - this ensures that P stays positive definite.
    - Upper Triangular Decomposition => Decompose P as UDU^T. This uses less storage and computation than SS^T. Algorithms using UDU^T for Kalman filtering are widely available (Bierman (1977), Thornton (1976)).
    - LDL^T / LU Decomposition => Possibly the most efficient and robust of the lot, with specific pivoting and conditioning passes applied (Thornton (1976), Bar-Shalom, Li, and Kirubarajan (2001), Golub and Van Loan (1996), Higham (2002)).
24. Bayes' Factor Analysis of Predict/Update: Bayes' Factor based analysis of the Kalman predict/update and eventual parameter inference occurs at the measurement stage, thereby introducing/associating the minimal error Optimal Kalman Gain at that stage (see e.g., Masreliez and Martin (1977)).
    - Bayesian Estimation vs. Filtering Estimation => Bayesian treatment deals with the hidden parametric uncertainty (given a set of uncertain observations), whereas filtering treatments deal with de facto observation/measurement uncertainty (in order to be able imply the set of hidden states and their uncertainties).
    - Bayesian A Priori Estimation => This is a simple, straight-up MLE pass-through, as there is neither an explicit dependence on the hidden state, nor is there any measurement involved. Joint a priori predict inference is simply a result of MLE'izing the normalized convolution of the model-transformed a posteriori updated state for the prior step and the model uncertainty.
    - A Posteriori Probabilistic Estimate => This is a convolution of the measurement uncertainty and the a priori state predict, given the observations. This therefore lends itself perfectly to standard Bayesian analysis (in fact, the whole suite, including ABC via sufficient statistics), and also eventually to the optimal gain extraction.

25. Information Vector Approach to Kalman Filtering:

- The main steps in this approach are the following. The main advantage of this approach is that N successive measurements may be trivially summed up in the information space.
    - Transform the State Space variables onto the Information Space.
    - Perform the update/predict in the Information Space.
    - Revert back to the State Space.
- Information Vector Definitions:
    - A Priori Information Matrix => $Y_{k|k-1} = P_{k|k-1}^{-1}$.
    - A Priori Information State Vector => $\hat{y}_{k|k-1} = P_{k|k-1}^{-1} x_{k|k-1}$.
    - A Posteriori Information Matrix => $Y_{k|k} = P_{k|k}^{-1}$.
    - A Posteriori Information State Vector => $\hat{y}_{k|k} = P_{k|k}^{-1} x_{k|k}$.
    - Measurement Information Matrix => $I_k = H_k^T R_k^{-1} H_k$.
    - Measurement State Information Vector => $i_k = H_k^T R_k^{-1} z_k$.
- Predict/Update Phase:
    - Information Matrix Update => $Y_{k|k} = Y_{k|k-1} + I_k \Rightarrow Y_{k|k} = Y_{k|k-1} + \sum_{j=1}^{n} I_{k,j}$.
    - Information State Vector Update =>

      $\hat{y}_{k|k} = \hat{y}_{k|k-1} + i_k \Rightarrow \hat{y}_{k|k} = \hat{y}_{k|k-1} + \sum_{j=1}^{n} i_{k,j}$.
- State Space Extraction from the Information Matrix/Vector:
    - $M_k = \left[F_k^{-1}\right] Y_{k-1|k-1} F_k$.
    - $C_k = M_k \left[M_k + Q_k^{-1}\right]^{-1}$.
    - $L_k = I - C_k$.
    - $Y_{k|k-1} = L_k M_k L_k^T + C_k Q_k^{-1} C_k$.
    - $\hat{y}_{k|k-1} = L_k \left[F_k^{-1}\right]^T \hat{y}_{k-1|k-1}$.

Continuous Time Kalman Filtering

1. Discrete/Continuous Linear Filter Variants:
   - Kalman => Discretized Model Prediction, and Discretized Measure Updates.
   - Bucy => Continuous Model Prediction, and Continuous/Simultaneous Measure Updates.
   - Hybrid => Continuous Model Prediction, and Discretized Measure Updates.
2. Kalman-Bucy Filter:
   - The continuous time Kalman Filter is expressed using the state equations (Bucy and Joseph (2005), Jazwinski (1970)) $\frac{\partial}{\partial t}x(t) = F(t)x(t) + B(t)u(t) + w(t)$ and $z(t) = H(t)x(t) + v(t)$, and $Q(t)$ and $R(t)$ are the intensities of the white noises $w(t)$ and $v(t)$.
   - In the continuous time Kalman Filtering equations, since there is no explicit distinction between the predict step and the update step, the covariance of the noise process $R(t)$ is the same as the covariance of the innovation residual $\tilde{y}(t) = z(t) - H(t)\hat{x}(t)$ (Kailath (1968)).
   - $\frac{\partial}{\partial t}\hat{x}(t) = F(t)\hat{x}(t) + B(t)u(t) + K(t)[z(t) - H(t)\hat{x}(t)]$
   - $\frac{\partial}{\partial t}P(t) = F(t)P(t) + P(t)F^T(t) + Q(t) + K(t)H(t)K^T(t)$ where the optimal gain is $K(t) = K(t)H^T(t)R^{-1}(t)$. This equation is called the Riccati equation.
3. Hybrid Kalman Filter: Used for e.g., in physical systems, which typically are continuous-time models, while discrete-time measurements are taken for estimation.
   - $\frac{\partial}{\partial t}x(t) = F(t)x(t) + B(t)u(t) + w(t)$, and $w(t) \sim N(0, Q(t))$.
   - $z_k = H_k x_k + v_k$ and $v(t) \sim N(0, R(t))$.
   - Predict Phase:

- $\frac{\partial}{\partial t}\hat{x}(t) = F(t)\hat{x}(t) + B(t)u(t)$; start with $\hat{x}(t_{k-1}) = \hat{x}_{k-1|k-1}$, and compute $\hat{x}_{k|k-1} = \hat{x}(t_k)$.
- $\frac{\partial}{\partial t}P(t) = F(t)P(t) + P(t)F^T(t) + Q(t)$; start with $P(t_{k-1}) = P_{k-1|k-1}$, and compute $P_{k|k-1} = P(t_k)$.

- Update Phase: Same as continuous-time Kalman filter. The optimal Kalman Gain results are re-capped below:
  - $K_k = P_{k|k-1}H_k^T\left[H_k P_{k|k-1}H_k^T + R_k\right]^{-1}$
  - $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\left[z_k - H_k\hat{x}_{k|k-1}\right]$
  - $P_{k|k} = (I - K_k H_k)P_{k|k-1}$

Non-linear Kalman Filtering

1. Definition: In non-linear Kalman filters, the state transition and the observation models are not linear, but differentiable, i.e.,
   - $x_k = f(x_{k-1}, u_k) + w_k$
   - $z_k = h(x_k) + v_k$
2. Extended Kalman Filter: The non-linearity may be handled by computing the evolution Jacobian at each step. Using these Jacobian tensors essentially linearizes the non-linear functions f and h around their current estimates.
   - Disadvantages of extended Kalman Filter approach:
     - Highly non-linear functions f and h give poor linearization performance/accuracy.
     - Jacobian calculations can become time consuming, although it may be significantly improved (in some cases) using the automatic differentiation techniques.

3. Unscented Kalman Filter (UKF): The challenges above are overcome by picking a set of points (called sigma points) around the mean. These points are propagated through non-linear functions, from which the sample mean and covariance are recovered more accurately (Julier and Uhlmann (1997)).
   - Predict Phase => The state and the covariance estimators are augmented with the process noise:
     - $\hat{x}_{AUG,k-1|k-1} = \left[\hat{x}_{k-1|k-1}{}^{T} \quad \left\langle w_k{}^{T} \right\rangle\right]^{T}$
     - $P_{AUG,k-1|k-1} = \begin{bmatrix} P_{k-1|k-1} & 0 \\ 0 & Q_k \end{bmatrix}^{T}$
   - UKF Sigma Points Generation for Predict => The $2L+1$ predict sigma points are generated as:
     - $\chi_{0,k-1|k-1} = x_{AUG,k-1|k-1}$
     - $\chi_{i,k-1|k-1} = x_{AUG,k-1|k-1} + \left[\sqrt{(L+\lambda)P_{AUG,k-1|k-1}}\right]_i$ for $i = 1,\ldots,L$
     - $\chi_{i,k-1|k-1} = x_{AUG,k-1|k-1} - \left[\sqrt{(L+\lambda)P_{AUG,k-1|k-1}}\right]_{i-L}$ for $i = L+1,\ldots,2L$
     - Here the subscript $i, i-L$ refers to the column $i, i-L$ of the Cholesky decomposition of $(L+\lambda)P_{AUG,k-1|k-1}$, i.e.,

       $(L+\lambda)P_{AUG,k-1|k-1} = \left[\sqrt{(L+\lambda)P_{AUG,k-1|k-1}}\right]\left[\sqrt{(L+\lambda)P_{AUG,k-1|k-1}}\right]^{T}$.
   - UKF Sigma Points Propagation and Re-combination:
     - Compute the a priori sigma points from their a posteriori counterparts computed in the previous step: $\chi_{i,k|k-1} = f\left(\chi_{i,k-1|k-1}\right),\ i = 0,\ldots,2L$.
     - Re-combine the a priori sigma points to estimate the predicted state and covariance:
       - $\hat{x}_{k|k-1} = \sum_{i=0}^{2L} W_{i,s}\chi_{i,k|k-1}$.
       - $P_{k|k-1} = \sum_{i=0}^{2L} W_{c,s}\left[\chi_{i,k|k-1} - \hat{x}_{i,k|k-1}\right]\left[\chi_{i,k|k-1} - \hat{x}_{i,k|k-1}\right]^{T}$.
     - The weights $W_{i,s}$ and $W_{c,s}$ are chosen to (see Wan and van der Merwe (2000)):

- Control the sigma point spread.
- Adapt to the x_k distribution.

- UKF Update Phase – A Priori State Augmentation => Again, the state and the covariance a priori estimates are augmented with the measurement noise:
  - $\hat{x}_{AUG,k|k-1} = \left[\hat{x}_{k|k-1}^{\ T} \ \left\langle v_k^{\ T} \right\rangle\right]^T$
  - $P_{AUG,k|k-1} = \begin{bmatrix} P_{k|k-1} & 0 \\ 0 & R_k \end{bmatrix}^T$
- UKF Update - Sigma Points Generation for Predict => The $2L+1$ predict sigma points are generated as:
  - $\chi_{0,k|k-1} = x_{AUG,k|k-1}$
  - $\chi_{i,k|k-1} = x_{AUG,k|k-1} + \left[\sqrt{(L+\lambda)P_{AUG,k|k-1}}\right]_i$ for $i = 1,...,L$
  - $\chi_{i,k|k-1} = x_{AUG,k|k-1} - \left[\sqrt{(L+\lambda)P_{AUG,k|k-1}}\right]_{i-L}$ for $i = L+1,...,2L$
  - Alternately, the UKF a priori sigma points themselves may be augmented as $\chi_{k|k-1} = \left[x_{k|k-1}^{\ T} \ \left\langle v_k^{\ T} \right\rangle\right]^T \pm \sqrt{(L+\lambda)R_{AUG,k}}$
- UKF Update – Final Phase => Project the sigma points through the observation function $\gamma_{i,k} = h(\chi_{i,k|k-1}),\ i = 0,...,2L$, and generate the following:
  - Predicted Observation $\hat{z}_k = \sum_{i=0}^{2L} W_{i,s}\gamma_{i,k}$
  - Predicted Measure Covariance $P_{\Xi_k\Xi_k} = \sum_{i=0}^{2L} W_{i,c}\left[\gamma_{i,k} - \hat{z}_k\right]\left[\gamma_{i,k} - \hat{z}_k\right]^T$
  - Predicted Measure Cross Covariance $P_{x_k\Xi_k} = \sum_{i=0}^{2L} W_{i,c}\left[\chi_{i,k|k-1} - \hat{x}_{k|k-1}\right]\left[\gamma_{i,k} - \hat{z}_k\right]^T$
  - Optimal UKF Kalman gain is computed from the alternate optimizing formulation as $K_k = P_{x_k\Xi_k} P_{\Xi_k\Xi_k}^{\ -1}$, and the corresponding state updates and covariance updates are computed from $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - \hat{z}_k)$ and $P_{k|k} = P_{k|k-1} - K_k P_{\Xi_k\Xi_k} K_k^{\ T}$.

4. Splined Kalman Filter: By using spline representations for both state variates and the covariance, we can almost certainly do a better estimation job than either EKF or UKF.

Kalman Smoothing

1. Optimal Fixed Lag Smoother: This provides an optimal estimate of $\hat{x}_{k-N|k}$ for a fixed lag N, using the measurements of z_1 to z_k. It does it one-pass-at-a-time for each time step, but does multiple passes in all.
2. Fixed Lag Smoother – State Equation:
   - $$\begin{bmatrix} \hat{x}_{t|t} \\ \hat{x}_{t-1|t} \\ . \\ . \\ . \\ \hat{x}_{t-N+1|t} \end{bmatrix} = \begin{bmatrix} I \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \hat{x}_{t|t-1} + \begin{bmatrix} 0 & I & 0 & \cdots & 0 \\ I & 0 & I & \cdots & 0 \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ 0 & 0 & 0 & \cdots & I \end{bmatrix} \begin{bmatrix} \hat{x}_{t-1|t-1} \\ \hat{x}_{t-2|t-1} \\ . \\ . \\ . \\ \hat{x}_{t-N+1|t-1} \end{bmatrix} + \begin{bmatrix} K_0 \\ K_1 \\ . \\ . \\ . \\ K_{N-1} \end{bmatrix} y_{t|t-1}$$
   - $\hat{x}_{t|t-1}$ => Estimated using the Standard Kalman Filter
   - $y_{t|t-1} = z(t) - H\hat{x}_{t|t-1}$ => Standard Innovation Residual
   - $\hat{x}_{t-i|t}$ => Estimate of the hidden/latent state at time $t-i$ given the observation at a later instant t
3. Fixed Lag Smoother – Gain Estimation: The optimal gains are given by the following equations for P_i and K_i: $P_i = P\left\{\left[F - KH\right]^T\right\}^i$, and $K_i = P_i H^T \left\{HPH^T + R\right\}^{-1}$. Here P and K are the prediction error covariance and the gain, respectively, of the standard Kalman Filter (i.e., they correspond to $P_{t|t-1}$).
4. Fixed Lag Smoother – Error Covariance: Define $P_i = \text{covariance}\left[\left(\hat{x}_{t-i} - \hat{x}_{t-i|t} \mid z_1,...,z_t\right)\right]$. The estimation $\hat{x}_{t-i}$ improves by the amount $P_i = P_{i|i-1} - \sum_{j=0}^{i}\left[P_j H^T \left\{HPH^T + R^{-1}\right\} H\left(P_i^T\right)\right]$.

5. Fixed Interval Smoother: The fixed interval smoother provides the estimate of $\hat{x}_{k|n}$ for $k < n$ using measurements from a fixed interval z_1 to z_n. This is also referred to as "Kalman smoothing".
6. Rauch-Tung-Striebel (RTS) Fixed Interval Smoother: This smoother uses 2 passes (Rauch, Tung, and Striebel (1965)).
   - The first pass is the regular forward Kalman pass. The filtered state estimates $\hat{x}_{k|k}$ and the covariance estimates $P_{k|k}$ are saved for the backward pass.
   - The backward pass computes the smoothed state estimate $\hat{x}_{k|n}$ and the covariance estimate $P_{k|n}$:
     - $\hat{x}_{k|n} = \hat{x}_{k|k} + C_k\left[\hat{x}_{k+1|n} - \hat{x}_{k+1|k}\right]$
     - $P_{k|n} = P_{k|k} + C_k\left[P_{k+1|n} - P_{k+1|k}\right]C_k^{\,T}$
     - $C_k = P_{k|k}F_k^{\,T}P_{k-1|k}^{\,-1}$
7. Fixed Interval Smoother: Modified - Bryson-Frazier Smoother: This technique also uses the regular Kalman filter for the forward pass, followed by a recursive backward pass that avoids finding the inverse of the covariance matrix.
8. Fixed Interval Smoother: Minimum-Variance Smoother: This is also a two pass smoother that achieves the theoretically best possible error performance using the variance error minimizer (Einicke (2006)).
   - The forward pass is on-step-ahead regular Kalman filter: $\hat{x}_{k+1|k} = F_k\hat{x}_{k|k-1} + K_k\left[z_k - H_k\hat{x}_{k|k-1}\right]$
   - Backward Pass => Set $\alpha_k = S_k^{-1/2}\left[z_k - H_k\hat{x}_{k|k-1}\right]$; then time-reverse α_k to compute β_k and the estimate $y_{k|N} = z_k - R_k\beta_k$.
   - Cross-check – Recovery of the Kalman Filter Formulation => Taking the causal part, i.e., setting $N = k$, you get $y_{k|k} = z_k - R_kS_k^{-1/2}\alpha_k$, which is the same as the original minimum variance Kalman filter.
   - Advantages of the minimum variance Smoother:
     - Underlying distribution need not be Gaussian (unlike RTS).

- Relatively easily extended to continuous time (Einicke (2007), Einicke (2009)). Continuous time uncertainty can be accommodated by adding a positive definite term to the Riccati equation (see Kalman-Bucy filter above) (Einicke, Ralston, Hargrave, Reid, and Hainsworth (2008)).
- The structure of α_k and α_k makes it relatively easily integratable with MLE-based state-space parameters.
- Relatively easy to incorporate step-wise linearization.

Particle Filtering

1. Definition: Simulation based model estimation technique – also called sequential Monte Carlo Method (Doucet, De Freitas, and Gordon (2001), Particle Filter (Wiki)).
2. State Space Evolution as Kolmogorov Density Estimation: Posterior probability may be modeled as a series histogram or a series of a set of weighted particles that evolve through time.
3. Advantages of using Particle Filtering:
   - It is faster than MCMC if the sampling/sample rejections are done well. The reason for this is that, while particle filtering computes $p(x_{k|k-1} \mid y_k)$, MCMC models attempt to compute $p(x_{k|k-1} \mid y_0,...,y_k)$.
   - It is easily extended to a wider variety of tractable priors/posteriors, as well as diverse types of state space functions.
   - With sufficient number of samples, particle filters can be made much more accurate than EKF/UKF.
4. Kalman vs. EKF/UKF vs. Particle Filtering vs. MCMC:
   - Kalman => HMM + Linear Model/Measurement Processes + Gaussian Error Terms
   - EKF/UKF => HMM + non-linear Model/Measurement Processes + Gaussian Error Terms
   - Kalman => HMM + non-linear Model/Measurement Processes + non-Gaussian Error Terms
   - MCMC => Hidden, non-Markov + non-linear Model/Measurement Processes + non-Gaussian Error Terms
5. Sufficient Statistics for Target Distributions: This is one relatively straightforward way to incorporate splining. Splines may also be used to represent the sampled, particle filtered Kolmogorov distributions.
6. Sequential Importance Re-sampling (SIR): Here the posterior filtering distribution $p(x_{k|k-1} \mid y_0,...,y_k)$ may be represented as a weighted set of P particles

$w_k : \{w_{k,0},...,w_{k,L},...,w_{k,P}\}$ where $L \in \{1,...,P\}$ (Gordon, Salmond, and Smith (1993)). SIR is also referred to as ***Sampling Importance Re-sampling***.

7. Practical SIR: In practice, however, given that $p(x_{k|k-1} \mid y_0,...,y_k)$ is not known, a proposal distribution $\pi(x_k \mid x_{L,0:k}, y_{0:k})$ is chosen instead. A sequential re-sampling algorithm will be needed to generate the appropriate weights.
   - Transition Density as the Proposal SIR Distribution => The discretized version of $\pi(x_k \mid x_{L,0:k}, y_{0:k})$ provides a set of weights, and is referred to as the importance function. The transition prior probability density distribution $\pi(x_k \mid x_{L,0:k}, y_{0:k}) = p(x_{k|k-1})$ forms a convenient importance function, as it is easy to draw particles (samples) it. It also has a few other advantages to be seen later.
   - SIR + Transition Density as Importance Function is also called bootstrap filtering or condensation (conditional density) algorithm.
8. SIR Re-sampling: Re-sampling is used to avoid degeneracy in the algorithm, i.e., avoiding the situation where all but one of the importance weights are almost zero. A variation of the algorithm, called stratified sampling (Kitagawa (1996)), is optimal in terms of variance.
9. SIR Algorithm:
   - #1 => Draw $L = 1,...,P$ samples from the proposal distribution $\pi(x_k \mid x_{L,0:k}, y_{0:k})$ and update the importance weights up to a normalizing constant: $\hat{w}_{L,k} = \hat{w}_{L,k-1} \frac{p(y_k \mid x_{L,k}) p(x_{L,k} \mid x_{L,k-1})}{\pi(x_k \mid x_{L,0:k}, y_{0:k})}$. Note that if we use the transition probability distribution $p(x_{L,k} \mid x_{L,k-1})$ as the importance function $\pi(x_k \mid x_{L,0:k}, y_{0:k})$, then the above reduces to $\hat{w}_{L,k} = \hat{w}_{L,k-1} p(y_k \mid x_{L,k})$.
   - #2 => Compute the normalized importance weights as $w_{L,k} = \frac{\hat{w}_{L,k}}{\sum_{j=1}^{p} \hat{w}_{j,k}}$, and the effective number of particles as $\hat{N}_{eff} = \frac{1}{\sum_{j=1}^{p} \hat{w}_{j,k}^{\,2}}$.

- #3 => If $\hat{N}_{eff} < \hat{N}_{Threshold}$, re-sample as follows:
    - Draw P particles from the current set of probabilities proportional to their weights.
    - Replace the current set with the new one.
    - Set $w_{L,k} = \frac{1}{P}$.

  In effect, this prunes out the low contributors.

10. Sequential Importance Sampling (SIS): This is the same as SIR, but without the re-sampling stage. It is also called the "direct version" – slight modification is needed in the algorithm to avoid re-sampling.
11. SIS Steps:
    - #1: Set $n = 0$, the number of particles counted so far. Uniformly choose an index L from $(1,...,P)$.
    - #2: Generate a test $\hat{x}$ from the distribution $p(x_{L,k} \mid x_{L,k-1})$, and generate the corresponding probability of a given $\hat{y}$ from $p(\hat{y} \mid \hat{x}) = p(y \mid x : y_k \mid \hat{x})$, where y_k is the kth measured value.
    - #3: Generate another uniform u in $[0,1]$. If $u > p(\hat{y})$, then re-generate another L and continue from #1. Otherwise, save $\hat{x}$ as $x_p(k \mid k)$ and increment n; continue till $n = P$, then quit. This technique is based off of rejection-sampling based optimal filter (Blanco, Gonzalez, and Fernandez-Madrigal (2008), Blanco, Gonzalez, and Fernandez-Madrigal (2010))
12. Particle Filter Extensions:
    - Auxiliary Particle Filter => Pitt and Shephard (1999)
    - Regularized Auxiliary Particle Filter => Liu, Wang, and Ma (2011)
    - Hierarchical/Scalable Particle Filter =>Canton-Ferrer, Casas, and Pardas (2011)
    - Rao-Blackwellized Particle Filter => Doucet, De Freitas, Murphy, and Russell (2000)
    - Econometric Particle Filter => Flury and Shephard (2008)

Unsupervised Learning

1. Definition: Unsupervised learning is the task of finding hidden structures in unlabeled data – the samples given to the learner are unlabeled, so there is no error or reward signal to evaluate a solution (Unsupervised Learning (Wiki)).
2. Techniques: Unsupervised learning is closely related to density estimation in statistics (Bishop and Jordan (2004)) and it encompasses many other techniques that seek to summarize and explain the key features of the data – e.g., data mining.
3. Approaches to Unsupervised Learning:
    a. Clustering (e.g., k-means, mixture models, hierarchical clustering)
    b. Hidden Markov Models
    c. Blind Signal Separation using Feature Extraction Techniques for Dimensionality Reduction (PCA, ICA, non-negative matrix factorization, SVD) (Acharyya (2008)).
4. Neural Network Models - SOM: The self-organizing map (SOM) is a topographic organization in which nearby locations in the map represent inputs in similar properties.
5. Neural Network Models - SRT: The Adaptive Resonance Theory (ART) Model allows the number of clusters to vary with the problem size and lets the user control the degree of similarity between members of the same clusters by means of a vigiliance parameter. ART networks are used for many pattern recognition tasks such as automatic target recognition and seismic signal processing (Carpenter and Grossberg (1988)).

Cluster Analysis

Introduction

1. Definition: Also called Clustering, it is the task of grouping a set of objects in such a way such that objects in the same group (cluster) are more similar to each other than to another in a different cluster.
2. Alternate Terms: Automatic Classification, Numerical Taxonomy, Botryology, and typological analysis (Tryon (1939), Bailey (1994)).
3. Cross Field Focus Differences: Differences in above terminology/focus is due to the eventual usage of the clustering results; in data mining, the resulting groups are the matter of interest, while in automatic classification, the resulting discriminative power is of interest (Cattell (1993)).

Cluster Models

1. Cluster Model/Conception: "Cluster Model" refers to the conceptual entity that defines a cluster unit (Estivill-Castro (2002), Cluster Analysis (Wiki)).
2. Types of Cluster Models: Following examples are givne for the corresponding cluster model:
    a. Connectivity Models => Hierarchical Clustering builds models based on distance Connectivity.
    b. Centroid Models => k-Means represents each cluster by its mean vector.
    c. Distribution Models => Clusters are modeled using statistical distributions, e.g., multi-variate normal distributions used by the Expectation Maximization Algorithm.
    d. Density Models => DBSCAN and OPTICS define clusters as connected dense regions in the data space.

e. Sub-space Models (also referred to as bi-clustering, co-clustering, or 2 mode clustering) => Here the clusters are modeled using both the cluster member characteristics and their relevant attributes.
f. Group Models => These models do not provide a refined mathematical model for the clustering – instead just provide the grouping information.
g. Graph-Based Models => These represent a clique, i.e., a sub-set of nodes in the graph such that every 2 nodes in the subset are connected by an edge to form a proto-typical cluster. Relaxations of the complete connectivity requirement (e.g., it may be stipulated that a fraction of the edges may be missing) are known as quasi-cliques.

3. Hard vs. Soft Clusters: In hard clustering, each object belongs to at most a single cluster. In soft clustering (also called fuzzy clustering), there exists a probability of a object belonging to a cluster.
4. Cluster Categorization Based on the Strength of Membership:
    a. Strict Partitioning Cluster => Here each object belongs to precisely one cluster.
    b. Strict Partitioning Cluster with Outliers => Objects can also belong to no clusters at all, and are considered outliers.
    c. Overlapping Clustering (also called alternative clustering or multi-view clustering) => While usually a hard cluster, objects may belong to more than one cluster.
    d. Hierarchical Clustering => Ovjects that belong to a child cluster also belong to a parent cluster.
    e. Sub-space Clustering => While typically an overlapping cluster, inside of a uniquely defined sub-space, clusters are not expected to overlap.

Connectivity Based Clustering

1. Definition: Also referred to hierarchical clustering, connectivity based clustering is based on the idea that "closer" objects form a cluster. Thus these algorithms form clusters by connecting objects inside of a distance metric.
2. Specification of the Connectivity Cluster: This cluster is described largely by the distance required to connect different parts of the cluster. Different cluster form at different max distance thresholds.
3. Connectivity Clustering – Linkage Criterion: In addition to the choice of the distance functions, the linkage criterion (an additional metric that links the elements of the cluster) needs to be specified. Popular choices are single-linkage clustering (here the linkage criterion is the minimum of the object distances), complete linkage clustering (the linkage criterion is the maximum of the object distances), and UPGMA (unweighted Pair Group Method with Arithmeic Mean, also known as average linkage clustering).
4. Hierarchical Cluster Formation Types: They are either agglomerative (starting with single elements and aggregating them into clusters) or divisive (Starting with the complete data set and dividing them into partitions).
5. Hierarchical Clustering Robustness and Uniqueness: Hierarchical clustering does not result in unique clusters – it depends on the starting point, and only produces a hierarchy from which additional decisions need to be made to fix down the clusters). Further, it is not robust to outliers, which results in additional clusters, or causes clusters to merge (this is called chaining phenomenan, esp. in single linkage clustering).
6. Hierarchical Clustering Performance: In general, the complexity associated is $O(n^3)$, thus making it too slow to be practical. Specialized situations produce $O(n^2)$ – e.g., for SLINK that uses single-linkage clustering (Sibson (1973)), or for CLINK that uses complete linkage clustering (DeFays (1977)).

Centroid Based Clustering

1. Definition: Here the clusters are represented by a central vector, which in itself may not be a member of any data set. When the number fo clusters is fixed to κ, κ-means clustering provides a formal definition as an optimization problem – find the κ clusters and assign the objects to the nearest center such that the distances (Euclidean/squared) are minimized.
2. κ-Means Optimization Problem: This is NP-hard, and only approximate solutions that uncover local optimum are used. The most well-known of these is the Lloyd's algorithm (Lloyd (1982)), and is often run multiple times with different randon initializations.
3. κ-Means Algorithm: Lloyd's algorithm is also referred to as the κ-Means algorithm. Variations on it include choosing the best of multiple runs, restricting the centroids to members of the data sets (κ-Medoids), choosing medians instead (κ-Means++), or allowing a fuzzy cluster assignment (fuzzy c-means). All of these variations may be applied concurrently.
4. Drawback of the κ-Means: Requires κ to be estraneously specified, often in advance. Further, this has a preference to generate clusters of similar size, as they always assign the object to the nearest centroid. Since κ-Means optimizes for the cluster centers (and not the cluster borders), it can produce incorrect borders.
5. Theoretical properties of κ-Means: Paritions data space into Voronoi polyhedral, and is conceptually close to nearest neighbor classification.

Distribution Based Clustering

1. Definition: Clusters are defined statistically: the representation here suits objects most likely to probabilistically belong to a specified distribution. Since a more complex distribution will explain/fit the data better, overfitting can become a challenge in this methodology.
2. Gaussian Mixture Cluster Models: Here the data set is modeled with a fixed number of Gaussian distributions to avoid over-fitting. These distributions are initialized randomly, and the parameters are iteratively optimized via the expectation

maximization algorithm.Connvergence is to a local optimum. For hard clustering, objects are assigned to the Gaussian ditribtuion they most likely belong to (this is not needed for soft clustering).

Density Based Clustering

1. Definition: Here the clusters are defined as areas of higher density than the remainder of the data set (Kriegel, Kroger, Sander, and Zimek (2011)). Objects in the soarse arreas (these are reuired to separate the clusters) are considered to be noise and border points.
2. DBSCAN: This is the most popular density based clustering method (Ester, Kriegel, Sander, and Xu (1996)). Similar to linkage based clustering, it is based on connecting points within certain distance thresholds. However, it connects only points that satisfy a density criterion (defined in the original variant as the number of objects within a specified radius). A cluster consists of all density-connected objects (which can form an arbitrary shape, in contrast to many other methods) PLUS all the objects within the original objects' range.
3. Properties of DBSCAN: Low complexity, requires a linear number of range queries, and it discovers results essentially in each run so that it doesn't need to be run multiple times (it is deterministic in the core and the noise points, but not at the border points).
4. Enhancement to DBSCAN - OPTICS: OPTICS (Ankerst, Breuning, Kriegel, and Sander (1999)) is a generalization of DBSCAN that removes the need for choosing an appropriate value for ε, and produces a hierarchical cluster result related to linkage clustering.
5. Enhancement of DBSCAN - DeLiClu: Density Link Clustering (DeLiClu – Achtert, Bohm, and Kroger (2006)) combines ideas from single-linkage clustering and OPTICS, thereby eliminating the ε parameter entirely, and offering performance improvements over OPTICS by using an R-tree index.

6. Drawback of DBSCAN - OPTICS: These need density drop to be able to detect the clusters, and end up performing poorly when the data is probabilistic (e.g., when the data uses Gaussian mixtures). A variant of DBSCAN called EnDBSCAN has been developed to detect intrinsic cluster structures prevelant in the majority of real-life data (Roy and Bhattacharyya (2005)).

Recent Clustering Enhancements

1. Performance Enhancements: Check out CLARANS (Ng and Han (1994)), BIRCH (Zhang, Ramakrishnan, and Livny (1996)), Zhang (1998), and Sculley (2010).
2. Clustering on Large Data Sets: With large data, there has been a willingness to trade semantic meaning of the generated clusters for performance. This has led to an array of pre-clustering techniques such as canopy clustering, which can process huge data sets efficiently. The resulting clusters are merely a pre-partitioning of the data that need to be analyzed using slower methods such as κ-Means (e.g., seed based clustering (Can and Ozkarahan (1990))).
3. High-Dimensional Clustering: Curse of dimensionality renders particular distance functions problematic. Therefore high-dimension clustering employs sub-space clustering, correlation clustering etc: that search for arbitrary rotated features. Sample algorithms are given in CLIQUE (Agrawal, Gehrke, Gunopoulus, and Raghavan (2003)) and SUBCLU (Kaling, Kriegel, and Kroger (2004)).
4. Density Subspace Clustering: A combination of the above techniques has been adopted in hierarchical subspace clustering. Combination of subspace clustering and DBSCAN/OPTICS has been used in HiSC (Achtert, Bohm, Kriegel, Kroger, Muller-Gorman, and Zimek (2006)) and DiSH (Achtert, Bohm, Kriegel, Kroger, Muller-Gorman, and Zimek (2007). Correlation clustering variants of the above include HiCO (Achtert, Bohm, Kroger, and Zimek (2006). Other enhancements include correlation connectivity enhancements in 4C (Bohm, Kaling, Kroger, and Zimek (2004)) and exploration of hierarchical density based correlation clusters (see ERiC – Achtert, Bohm, Kriegel, and Zimek (2007)).

5. Enhancements using Mutual Information: Examples include variation of information metric (Meila (2003)) with applications to hierarchical clustering (Kraskov, Stogbauer, Andrzejak, and Grassberger (2003)). Genetic information enables testing a wide variety of mutual information based fit functions (Auffarth (2010)). Finally, message-passing algorithms based on statistical physics has opened up a variety of clustering techniques (Frey and Dueck (2007)).

Internal Cluster Evaluation

1. Internal Cluster Evaluation - Objective: This carries out an evaluation of the clustering result based on the data set that was clustered in itself. Drawbacks with internal cluster evaluations are: a) High scores on internal metrics do not necessarily result in effective information retrieval approaches (Manning, Raghavan, and Schutze (2008)). Further, if the evaluation metric is similar to the clustering metric (e.g., say if both are distance based), this evaluation over-estimates the impact of the clustering algorithm.
2. Davies Bouldin Index:This is estimated using the formula $DB = \frac{1}{n}\sum_{i=1}^{n}\left\{\begin{matrix}max \\ i \neq j\end{matrix}\left(\frac{\sigma_i+\sigma_j}{d(c_i+c_j)}\right)\right\}$ where n is the number of clusters, c_i is the centroid of the cluster i, c_j is the centroid of the cluster j, σ_i is the average distance of all the elements in cluster i to its centroid c_i, and $d(c_i + c_j)$ is the distance between the centroids c_i and c_j. Algorithms that produce clusters with low intra-cluster distances (high intra-cluster similarity) and high inter-cluster distances (low inter-cluster similarity) have low index values. Thus algorithms that produce the smallest index value are the most effective ones by this metric (it is important to remember that the notion of "distance" needs to match between the clustering and the evaluation schemes – beyond that it is left open).
3. Dunn Index (Dunn (1974)): The Dunn index aims to identify dense and well-separated clusters, and is defined as the ratio between the minimal inter-cluster distance and the maximal intra-cluster distance: $D =$

$\underset{1 \le i \le n}{min} \left\{ \underset{1 \le j \le n; i \ne j}{min} \left[\frac{d(i,j)}{max_{1 \le k \le n} d'(k)} \right] \right\}$ where $d(i,j)$ is the distance between clusters i and j and $d'(k)$ is the measure of the intra-cluster distance in cluster k. Inter-cluster distance $d(i,j)$ may be measured in a number of ways – such as the distance between the centroids. Likewise, $d'(k)$ could be, say, the maximum distance between any pair of elements in cluster k. Thus, algorithms with a higher Dunn index value are more desirable.

External Cluster Evaluation

1. External Cluster Evaluation - Objective: This is more or less identical to cross validation or GCV, and the comparison is with hand labeled "cross validation" data set. Challenges with this approach include the choice of the cross validation sample (to reveal sub-structures embedded in the data set) (Farber, Gunnermann, Kriegel, Kroger, Muller, Schubert, Seidl, and Zimek (2010)).
2. Rand Measure (Rand (1971)): This measures how similar the cluster results are to the benchmark classsifications. It is the ratio of the correct decisions made by the algorithm to the total results (this comes from using a confusion matrix): $RI = \frac{TP+TN}{TP+TN+FP+FN}$. One challenge with this approach is that its weights FP and FN equally – the F-measure removes that.
3. F-Measure: Defining Precision Rate and the recall rate as $P = \frac{TP}{TP+FP}$ and $R = \frac{TP}{TP+FN}$ respectively, the F-measure weights FN through a $\beta \ge 0$ parameter as $F_\beta = \frac{(\beta^2+1).P.R}{\beta^2.P.R}$. When $\beta \to 0$, $F_0 \to P$, thus recovering the Rand index.
4. Pair Counted F-Measure: This is the F-Measure applied to the set of object pairs, where the object are paired with each other when they are part of the same cluster. This measure is able to compare clusterings with different numbers of clusters.
5. Fowlkes-Mallows Index (Fowlkes and Mallows (1983)): FM computes the similarity between clusters returned by the clustering algorithm and the benchmark classifications. Higher the FM, greater the similarity between the result cluster and

the benchmark cluster: $FM = \sqrt{\frac{TP}{TP+FP}.\frac{TP}{TP+FN}}$. In effect, FM is the geometric mean of the P and the R rates, while the F-measure may be viewed as their harmonic mean (Hubert and Arabie (1985)). P and R are also referred to as Wallace's B^I and B^{II} indices (Wallace (1983)).

6. Jaccard Index: This is simply the number of unique elements common to the two sets (the cluster set and the benchmark set) divided by the total number of unique elements in both sets: $J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP+FP+FN}$. Thus, $J(A,B)$ ranges from 0 (no common elements) to 1 (the sets are identical).
7. Mutual Information Based External Evaluation: This is an information theoretic measure of how much information is shared between a cluster and its ground truth classification that can detect a non-linear similarity between the two clusterings. Adjusted mutual information is the adjusted-for-chance variant of this that has reduced bias for varying cluster numbers.

Clustering Axiom

1. Formalism: A partition function F acts on a set S with $n \geq 2$ points (the points being labeled as $\{1,2,\ldots,n\}$ along with a number of clusters integer $k > 0$ and pair-wise distance among S. (These are the ONLY set of information available on S). The distance function $d: S \times S \to R$ has the property that for $i, j \in S$, $d(i,j) \geq 0$, and $d(i,j) = d(j,i)$. These properties ensure that the distance function is non-negative, symmetric, and that two points are identical ONLY if the distance between is zero.
2. Clustering Function: The clustering/partition function F takes a distance function d on $S \times S$ and an integer $k \geq 1$ to return a k-Partition of S, a collection of non-empty disjoint subsets of S whose union is S. Two clustering functions are equivalent if and only if they output the same partitioning on all values of d = and k.
3. Clusteing Axiom #1 – Scale Invariance: For any distance function d, number of cluster k, and scalar $\alpha > 0$, we should have $F(d,k) = F(\alpha.d,k)$. This simply states that the clustering function is immune to stretching/shrinking.

4. Cluatering Axiom #2 – k-Richness: For a fixed S and k, let $Range[F(\cdot, k)]$ be ths et of all possible outputs while varying d. Then, for any number of clusters k, $Range[F(\cdot, k)]$ is equal to the set of all the k-partitions of S. This property ensures that any partition $\mathbb{\Gamma}$ has associated with it (or extractable using a) distance metric d such that $F(d, k) = \mathbb{\Gamma}$.
5. Clustering Axiom #3 - Consistency: If the like partitions are blended together with their distances shrunk (expanded) and "unlike partitions" are altered such that their distances expand (shrink), the clustering function $F(d', k)$ should get back to the same partition as $F(d, k)$ (Kleinberg (2002)).
6. Sample Clustering Applications:
    a. In field robotics totrack objects and detect outliers for situational awareness (Bewley, Shekhar, Leonard, Upcroft, and Lever (2011))
    b. To find structural similarities by clustering chemical compounds into topological indices (Basak, Magnuson, Niemi, and Regal (1988))
    c. To find weather regimes or preferred sea-level pressure atmospheric patterns (Huth, Beck, Philipp, Demuzere, Ustrnul, Cahynova, Kysely, and Tveito (2008))

Mixture Model

Introduction

1. Definition: In statistics, a mixture model is a probabilistic model for representing the presence of sub-populations within an overall population, without requiring that an observed data set should identify the sub-population to which an individual observation belongs.
2. Components of the Mixture Model: See Mixture Model (Wiki):
    a. N random variables corresponding to observations, each assumed to be distributed according to a mixture of k components with each component belonging to the same parameter family of distributions (e.g., all normal, all Zipfan, etc.) but with different parameters.
    b. N corresponding random latent variables specifying the identity of the mixture components of each observation, each distributed according to a k-dimensional categorical distribution.
    c. A set of k mixture weights, each of which is a probability, and the sum of all of which come to 1
    d. A set of k parameters, each set specifying the parameter of the corresponding mixture component. For example, observations distributed according to a mixture of one-dimensional Gausssian distribution will have a mean and a variance for each component. Observations distributed according to a V-dimensional categorical distribution will have a vector of V probabilities, collectively summing to 1.

Generic Mixture Model Details

1. Bayesian Setting: In a Bayesian setting, the mixture parameters and the weights will themselves be random variables, and prior distributions are placed over these variables. The weights are typically viewed as a k-dimensional random vector drawn from a Dirichlet distribution (the conjugate prior of the categorical distribution), and the parameters will be distributed according to their respective conjugate priors.
2. The Basis Generic Parameter Mixture Model:
    a. k - The Number of Mixture Components
    b. N - The Number of Observations
    c. $\theta_{i=1,\dots,k}$ - Parameters of the Distribution of the Observation associated with Component i
    d. $\varphi_{i=1,\dots,k}$ - Mixture Weight, i.e., Prior Probability of a particular Component i
    e. $\vec{\emptyset}$ - k-dimensional vector composed of all the individual $\varphi_{i=1,\dots,k}$; must sum to unity
    f. $x_{i=1,\dots,k}$ - Observation i
    g. $z_{i=1,\dots,k}$ - Component of Observation i
    h. $F(x|\theta)$ - Probability Distribution of an Observation parametrized on θ
    i. $x_{i=1,\dots,k}$ - $Categorical(\emptyset)$
    j. $z_{i=1,\dots,k} - F\left(\theta_{z_i}\right)$
3. Bayesian Parametric Mixture Model:
    a. k, N, $\theta_{i=1,\dots,k}$, $\varphi_{i=1,\dots,k}$, $x_{i=1,\dots,k}$, $z_{i=1,\dots,k}$, $F(x|\theta)$ - As Above
    b. α - Shared Hyper-parameter for the Component Parameters
    c. β - Shared Hyper-parameter for the Mixture Weights
    d. $H(\theta|\alpha)$ - Prior Probability Distribution of Component Parameters, parametrized on α
    e. $\theta_{i=1,\dots,k}$ - $H(\alpha)$
    f. φ - $Symmetrical\_Dirichlet_k(\beta)$
    g. $x_{i=1,\dots,k}$ - $Categorical(\emptyset)$
    h. $z_{i=1,\dots,k}$ - $F\left(\theta_{z_i}\right)$
4. Nature of F and H: The above characterization uses F and H to describe arbitrary distributions over observations and parameters, respectively. Typically H will be a

conjugate prior of F. Most common choices for F are Gaussian for real-valued observations, and categorical for discrete observations.

5. Alternate Mixture Component Distributions:
    a. Binomial Distribution => Used to model the number of positive occurrences (successes, yes votes, etc.) given the total number of occurrences
    b. Multinomial Distribution => Similar to binomial distribution, but for counts of multi-way occurrences (e.g, yes/no/maybe in a survey)
    c. Negative binomial distribution, for binomial type observations, but where the quantity of interest is the number failures before a given number of successes occurs
    d. Poisson Distribution => For the number of occurrences of an event in a given period of time, for an event that is characterized by a fixed rate of occurrence
    e. Log Normal Distribution => For positive real numbers that are assumed to grow exponentially
    f. Multi-variate Normal Distribution (aka multivatiate Gaussian distribution) => For vectors of correlated outcomes that are individually Gaussian distributed
    g. A vector of Bernoulli distributed values => Corresponding to, for e.g., a black-white image, with each value representing a pixel.

Specific Mixture Models

1. Gaussian Mixture Model:
    a. k, N, $\varphi_{i=1,\dots,k}$, $\vec{\emptyset}$, $x_{i=1,\dots,k}$, $z_{i=1,\dots,k}$ - As above
    b. $\mu_{i=1,\dots,k}$ and ${\sigma^2}_{i=1,\dots,k}$ - Mean and Variance of Component i
    c. $\mu,\ \lambda,\ \nu, {\sigma_0}^2$ - Shared hyper-parameters
    d. $\mu_{i=1,\dots,k} \sim \aleph(\mu_0, \lambda{\sigma_i}^2)$; ${\sigma_{i=1,\dots,k}}^2 \sim Inverse\_Gamma(\nu, {\sigma_0}^2)$
    e. $\vec{\emptyset}$ - $Symmetrical\_Dirichlet_k(\beta)$; $z_{i=1,\dots,N}$ - $Categorical(\vec{\emptyset})$
    f. $x_{i=1,\dots,N} \sim \aleph(\mu_{z_i}, {\sigma^2}_{z_i})$

2. Multivariate Bayesian Gaussian Mixture Model: In a multivariate distribution (one modeling $\vec{x}$ with N random variables) one may a vector of parameters (such as several observations of a signal or patches within an image) using a Gaussian mixture model with a prior distribution on the vector of estimates given by $p(\vec{\theta}) = \sum_{i=1}^{k} \varphi_i \aleph(\overrightarrow{\mu_i}, \overrightarrow{\sigma_i}^2)$
3. Multivariate Bayesian Gaussian Estimation: To incorporate the above prior into a Bayesian estimation, the prior is multiplied with a known distribution $p(\vec{x}|\vec{\theta})$ of the data set $\vec{x}$ conditioned on $\vec{\theta}$ to be extended, thus resulting in $p(\vec{\theta}|\vec{x}) = \sum_{i=1}^{k} \tilde{\varphi}_i \aleph\left(\overrightarrow{\tilde{\mu}_i}, \overrightarrow{\tilde{\sigma}_i}^2\right)$.
4. Bayesian Multivariates Parameter Estimation: The new parameter set $\tilde{\varphi}_i, \overrightarrow{\tilde{\mu}_i}, \overrightarrow{\tilde{\sigma}_i}^2$ are then updated using the Expectation-Maximization algorithm (Yu (2012)). Although EM-based parameter updates are well-established, providing initial estimates for the parameters above is an area of active research. The formulation above yields a closed-form solution to the complete prior, and estimates of $\vec{\theta}$ are obtained by one of several standard Bayesian estimators (such as mean or maximization of the posterior distribution).
5. Patch-wise Multivariate Bayesian Gaussian: Such Bayesian estimates of Gaussian multivariates are useful for assuming path-wise shapes of images and clusters. For image representation, each Gaussian may be tilted, expanded, and warped according to $\overrightarrow{\tilde{\sigma}_i}^2$. A single Gaussian distribution of the set is fit to each patch (say 8×8 pixels) of image. Notably, any distribution of points around a cluster (e.g., k-means) may be accurately determined given enough Gaussian components, but rarely are $k > 20$ such components needed to accurately model a given image distribution or cluster of the data.
6. Categorical Mixture Model:
    a. k, N, $\varphi_{i=1,\ldots,k}$, $\vec{\emptyset}$, $x_{i=1,\ldots,k}$, $z_{i=1,\ldots,k}$ - As above
    b. V - Dimension of the Categorical Observations (e.g., size of the Word Volabulary)
    c. $\theta_{i=1,\ldots,k;j=1,\ldots,V}$ - Probability of Component i observing Item j

d. $\vec{\theta}_{i=1,\ldots,k}$ - Vector of Dimension , composed of $\theta_{i;1,\ldots,V}$; sums to unity

e. $x_{i=1,\ldots,k}$ - $Categorical(\emptyset)$

f. $z_{i=1,\ldots,k}$ - $Categorical\left(\theta_{z_i}\right)$

7. Bayesian Categorical Mixture Model:

   a. k, N, $\varphi_{i=1,\ldots,k}$, $\vec{\emptyset}$, V, $\theta_{i=1,\ldots,k;j=1,\ldots,V}$ - As represented as above in the non-Bayesian mixture model case

   b. α - Shared Concentration Hyper-parameter of $\vec{\theta}$ for each Component

   c. β - Concentration Hyper-parameter for $\vec{\emptyset}$

   d. $\vec{\emptyset} \sim Symmetrical\_Dirichlet_k(\beta)$; $\theta_{i=1,\ldots,k} \sim Symmetrical\_Dirichlet_V(\alpha)$

   e. $z_{i=1,\ldots,k} \sim Categorical\left(\vec{\emptyset}\right)$; $x_{i=1,\ldots,k} \sim Categorical\left(\theta_{z_i}\right)$

Mixture Model Samples

1. Financial Returns: Financial returns often behave differently in normal situations, and during crisis times. A mixture model (Dinov (2005)) for returns data seems reasonable. Sometimes the model used is a jump-diffusion model, or a model that is a mixture of two normal distributions.
2. Home Prices: This can assume that the prices are accurately described by a mixture model with K different components in one area, each distributed as a normal distribution with a corresponding unknown mean and variance.
3. Document Topics: This is also called a topic model. This assumes that a document is composed of N different words from a total vocabulary of size V where each word corresponds to one of K possible topics.

   a. Parameter Estimation => EM is applied to model tails and produce realistic results due to the excessive number of parameters. Additional assumptions are needed, e.g., a prior distribution is placed over the parameters describing the topic distribution where only a small number of words have significantly non-zero probabilities).

b. Topic Identity Constraint => Additional constraints placed over the topic identities of words to take advantage of the natural clustering include:
    i. Placing a Markov chain (i.e., the latent variables specifying the mixture component for each observation) on the topic identities (which are really just the cluster labels) represents the fact that nearby words belong to similar topics. This results in a hidden Markov model – specifically one where a prior distribution is placed over the state transitions that favor transitions that stay in the same state.
    ii. Natural Clustering Constraints => Another possibility to specify additional constraints is to model the topic identities using the latent Dirichlet allocation model, which devides the word set into D different documents and assumes that in each only a small number of topics can occur with any frequency.

4. Handwriting Recognition: Consider a $N \times N$ black-and-white (Bishop (2006)) that is a scan of a hand-written digit, i.e., $0 - 9$. We create a mixture model with $k = 10$ different components (one per digit number) where each component is a vector of N^2 Bernoulli distributions (one per pixel). This model can be trained with EM on an unlabeled set of hand-written digits, thus creating the clustering. The same model can then be used to recognize the digit of another image simply by holding the paraneters constant, computing the probability of realization of the new image for each possible digit, and returning the digit with the highest possibility.
5. Fuzzy Image Segmentation: In fuzzy/soft segmentation, any pattern can have "ownership" over any single pixel. If the patterns are Gaussian, fuzzy segmentation naturally results in Gaussian mixtures. Combined with other analytic/geometric tools (e.g., phase transitions over diffusive boundaries), such regularized mixture models could lead to more realistic and computationally efficient segmentation models (Shen (2006)).

Identifiability

1. Definition: Identifiability refers to existence of a unique characterization for any one of the models in the classs (family) being considered. Estimation procedure may not be well-defined, and asymptotic theory may not hold if a model is not identifiable.
2. Mathematical Specification: Consider a mixture of parametric distributions belonging to the same class. Let $J = \{f(\blacksquare, \theta) : \theta \in \Omega\}$ be the class of all the component distributions. Then the convex hull K of J defines the class of all the finite mixture distributions in J: $K = \{p(\blacksquare) : p(\blacksquare) = \sum_{i=1}^{n} a_i f_i(\blacksquare, \theta_i)\ ;\ a_i > 0;\ \sum_{i=1}^{n} a_i = 1\ ;\ f_i(\blacksquare, \theta_i) \in J\ \forall\ i, n\}$. K is said to be identifiable if all its members are unique, that is, given two members p and p' in K being mixtures of k distributions and k' distributions respectively in J, we have $p = p'$ if and only if, first, $k = k'$, and second, we can re-order the summations such that $a_i = a_i'$ and $f_i = f_i'$ for all i.
3. Parameter Estimation and System Identification: Typical approaches that use EM or MAP consider separately the questions of parameter estimation and system identification, that is to say, a distinction is made between the determination of the number and the functional forms of the components within the mixture, and the estimation of the corresponding parameter values.
4. Joint Parameter Estimation and System Determination: Notable departures to the separated system determination/parameter estimation methods are the graphical methods of Tarter and Lock (Tarter (1993)), more recently the minimum message length (MML) techniques (Figueiredo and Jain (2002)), and to some extent, the moment matching pattern analysis routines suggested by McWilliam and Loh (2008).

Expectation Maximization

1. Definition: This is seemingly the most popular technique used in determining the parameters of the mixture with an a priori given number of components. EM/MLE is of particular appeal for finite normal mixtures where closed-form expressions are

possible, an example of which is provided below (this is the Dempster-Shafer iterative algorithm):

a. $w_s^{j+1} = \frac{1}{N}\sum_{t=1}^{N} h_S^j(t)$

b. $\mu_S^{j+1} = \frac{\sum_{t=1}^{N} x(t) h_S^j(t)}{\sum_{t=1}^{N} h_S^j(t)}$

c. $\rho_S^{j+1} = \frac{\sum_{t=1}^{N}\left[x(t)-\mu_S^j\right]\left[x(t)-\mu_S^j\right]^T h_S^j(t)}{\sum_{t=1}^{N} h_S^j(t)}$

d. The corresponding posterior probabilities are $h_S^j(t) = \frac{w_s^j p_s\left(x(t),\mu_S^j,\rho_S^j\right)}{\sum_{i=1}^{n} w_i^j p_i\left(x(t),\mu_i^j,\rho_i^j\right)}$

2. State Evolution Updates: Using the above, on the basis of the current estimate for the parameters, the conditional probability for a given observation $x(t)$ being generated from state s is determined for each $t = 1, \ldots, N$, N being the sample size. The parameters are then updated such that the new component weights correspond to the average conditional probability, and component mean and co-variance is the component specific weighted average of the mean and the covariance of the entire sample.
3. EM Steps:
    a. The Expectation Step => With initial Guesses for the parameters of our mixture model, the "partial membership" in each constituent distribution is computed by calculating the expectation values for the membership variables of each data point. Thus given the data point x_j and a distribution Y_i, the membership value is given as $y_{ij} = \frac{a_i f_Y(x_j;\theta_i)}{f_X(x_j)}$.
    b. The Maximization Step => The mixing coefficients a_i and the means of the membership values over N data points are given as $a_i = \frac{1}{N}\sum_{j=1}^{N} y_{ij}$ The corresponding computed model parameters are computed using $\theta_i = \frac{\sum_{j=1}^{N} x_j y_{ij}}{\sum_{j=1}^{N} y_{ij}}$. With new estimates for a_i and θ_i, the expectation step is repeated until model parameters converge.
4. Advantages of the EM Approach: Dempster, Laird, and Rubin (1977) showed that each successive EM iteration will not decrease the likelihood, a property not shard by

other gradient based maximization techniques. EM also embeds within itself the constraints on the probability vector, and, for sufficiently large sample sizes, also maintains the positive definiteness of the covariance iterates. This is a key advantage, since explicitly constrained methods incur extra computational costs to check and maintain appropriate values.

5. Drawbacks of the EM Approach:
    a. EM is a first-order algorithm, so convergence is slow. Superlinear, second-order Newton, and quasi-Newton methods are therefore more preferred (although it needs to be noted that even if the convergence likelihood is rapid – which is what really counts – the convergence of the estimated parameters need not (Xu and Jordan (1996)).
    b. EM is a local maximizer (McLaughlan (2000)), and displays sensitivity to initial values. Approaches to overcoming these include evaluating EM at several points in the parameter space, but this can become computationally expensive too. Thus approaches such as annealing EM may be preferred – here the initial components are essentially forced to overlap, providing a less heterogenous basis for the initial guesses.
6. Minimum Message Length (MML) Algorithm: Figueiredo and Jain (2002) note that the convergence to meaningless parameter values obtained at the boundary (where regularity conditions breakdown) is frequently observed when the number of model components exceeds the optimal/true one. On this basis, they suggest a unfied approach to estimation and identification in which the initial n is chosen to greatly exceed the optimal value. This optimization routine is constructed via a minimum message length criterion that effectively eliminates a candidate component if there is insufficient information to support it. In this way it is possible to systematize reductions in n and consider estimation and identification jointly.

Alternatives to EM

1. Monte Carlo Markov Chain Alternative to EM: Here the mixture model paraneters can be deduced using posterior sampling. Essentially this will be an incomplete data problem whereby membership of the data points is the missing data. A 2-step iterative procedure known as Gibbs' sampling may be used.
    a. 2-step Gaussian MCMC => As in the EM case initial guesses for the parameters of the mixture model are made. Instead of computing partial memberships for each elemental distribution, a membership value for each data point is drawn from a Bernoulli distribution (that is, it will be assigned to either the first or the second Gaussian distribution). The Bernoulli parameter θ for the draw is determined for each data point on the basis of one of the constituent distributions. Draws from the distribution generate memnership associations for each data point. Plug-in estimators can them be used in the Maximization step of the EM to generate a new set of mixture model parameters, and the binomial draw is repeated.
2. Moment Matching Methods: In this approach, the parameters of the mixture are determined such that the composite distribution has moments matching the given value set (Wang (2001)). However, solutions to the moment equations may pose non-trivial computation challenges and other inefficiencies compared to EM (Day (1969)).
3. Spectral Methods - Applicability: Mixture model estimations are amenable to spectral methods if the data points x_i are points in high-dimensional real space, but the hidden distributions are known to be log-concave (such as Gaussian distribution or exponential distribution).
    a. Estimation Techniques => Spectral methods of learning mixgture models are based on the use of SVD of a matrix containing the data set. The idea of the technique is to consider the top k singular vectors where k is the number of distributions to be learned. The projection of each data point to a linear subspace spanned by those vectors groups those points originating from the same distribution close together, while the points from different distributions stay far apart.

   b. Robustness Property => One distinctive feature of the spectral method is that if the distributions satisfy a certain separation condition (e.g., they are not too close), the estimated mixure will be very close to the true one with very high probability.

4. Graphical Methods: In graphical approach to mixture identification a kernel function is applied to an empirical frequency plot to reduce the intra-component variance (Tarter (1993)). This helps more readily identify components with differing means. While this method does not require knowledge of the number or the basis functional form for the components, its success does rely on the choice of the kernel parameters, which to some extent implicitly embeds assumptions about the component structure.
5. Other non EM Methods: Some of the methods can even learn mixtures with heavy tailed distributions, including those with infinite variance. In this setting the EM approaches won't work, as the expectation step diverges due to the presence of the outliers.

Mixture Model Extensions

1. Bayesian Extensions: Additional Bayesian levels can be added to the model defining the mixture model. For example, in the common latent Dirichlet allocation topic model, the observations are sets of words drawn from D different documents, and the K mixture components represent topics that are shared across the documents. Each document has a different set of mixture weights which specify the topics prevelant in those documents. All sets of mixture weights share common hyper parameters.
2. Hidden Markov Extensions: Another common extension is to connect the latent variables defining the mixture component identities into a Markov chain instead of assuming that they are independent identically distributed random variables. This results in a hidden Markov model.
3. Application Areas: The advent of processing power has popularized EM/ML techniques (McLaughlan (1988)), and is now used vastly in areas such as agriculture, botany, economics, electrophoresis, finance, fisheries, genetics, geology, medicine,

paleontology, psychology, sedimentology, and zoology (Titterington, Smith, and Makov (1985)).

Deep Learning

Introduction

1. Definition: Deep learning comprises of a set of algorithms in machine learning that attempt to model high level abstractions in data by using architectures composed of multiple non-linear transformations (Deep Learning (wiki), Bengio, Courville, and Vincent (2013)).
2. Architectures for Learning Representations: Deep learning is part of a broader family of machine learning methods based on learning representations. These architectures include deep neural, convolutional deep neural networks, and deep belief networks.
3. Earlier Architectures: Original deep-learning architectures based on the standard ANN back-propagation algorithms (Werbos (1974)) and the neocognitron (Fukushima (1980)) fell out of favor due to the speed issues. Vanishing gradient problem was identified to be the key issue (Hochreiter (1991), Hochreiter, Bengio, Frasconi, and Schmidhuber (2001)), and ANN approaches were replaced by SVMs etc.
4. Many Layered Feed Forward ANN: Hinton (2007) showed how a many layered feed-forward neural network can be effectively pre-trained one layer at a time, treating each layer in turn as an unsupervised Boltzmann machine, and then using supervised back-propagation for fine-tuning.
5. Unsupervised Deep Hierarchies: Schmidhuber (1992) had already implemented a very similar idea for the more general case of unsupervised hierarchies of recurrent neural networks, demonstrating the benefits for speeding up supervised learning (Schmidhuber (2013)).
6. Usage of Deep Learning: Application areas such as computer vision and automatic speech recognition (ASR) use evaluation data sets such as MNIST and TIMIT respectively to improve deep learning applications. For instance, convolutional neural networks are widely used - although more in computer vision than in ASR.

7. Impact of Computational Advances in Deep Learning: Powerful GPUs highly suited for number crunching and matrix/vector math have enormously enhanced deep learning by reducing the training times from weeks to hours (Raina, Madhavan, and Ng (2009), Ciresan, Meier, Gambardella, and Schmidhuber (2010)).

Unsupervised Representation Learner

1. Deep Learning as a Representation Learner: Deep learning algorithms are based on distributed representations – whose underlying idea is that the observed data is generated as a result of many different factors on different levels. Deep learning adds the assumption that these factors are organized into multiple levels, corresponding to different levels of abstraction or composition.Varying numbers of layers and layer sizes can be used to provide different amounts of anbstraction (Bengio, Courville, and Vincent (2013)).
2. Hierarchical Explanatory Factors: Deep learning algorithms exploit the idea of hierarchical explanatory factors. Concepts are learned from other concepts, with the more abstract, higher level concepts being learned from the lower level ones. These architectures are often constructed using a greedy layer-by-layer method that models this data. Deep learning helps disentangle these abstractions and pick out the features that are useful for learning (Bengio, Courville, and Vincent (2013)).
3. Deep Learning as Unsupervised Learning: Many deep learning algorithms are actually framed as unsupervised learning problems. This ease of use in unlabeled inputs (very prevelant in real life situations) makes deep learning techniques widely applicable.

Deep Learning Using ANN

1. Motivation: Some of the most successful deep learning methods involve ANN (which was inspired by the 1959 biological model of the primary visual cortex as being

composed of 2 types of cells – the simple and the complex). Many ANN's may be viewed as cascading models (Reisenhuber and Poggio (1999)) of these cell types.

2. Deep Learning using Convolutional Neural Networks: Fukushima's Neocognitron (Fukushima (1980)) introduced the convolutional NN that is partially trained by unsupervised learning. LeCun, Boser, Denker, Henderson, Howard, Hubbard, and Jackel (1989) applied supervised back propagation to such architectures.
3. Challenges Training Deep ANN's: Hochreiter (1991) and Hochreiter, Bengio, Frasconi, and Schmidhuber (2001) identified the cause for the failure to be able to train deep ANN's from scratch as being due to the "vanishing gradient problem", which not only affects many-layered feed-forward networks, but also recurrent neural networks. The latter are trsained by unfolding them into very deep feed-forward networks, where a new layer is created for each time step of the input sequence processed by the network. As errors propagate layer-to-layer, they shrink exponentially with the number of layers.
4. Training Deep ANN's: Multi level Hierarchy of Networks: In this approach, the multi-level hierarchy of networks are pre-trained one level at-a-time through unsupervised learning, and are fine-tuned through back-propagation (Schmidhuber (1992)). Each level learns a compressed representation of the observations that is fed to the next level.
5. Training Deep ANN's - LSTM: Deep, multi-dimensional Long-Short Memory (LSTM) networks have been shown to be powerful in deep learning with many non-linear layers (Hochreiter and Schmidhuber (1997)). Applications to Connected Handwriting Recogition without any prior knowledge of the language to be learned for three different languages have also been demonstrated (Graves and Schmidhuber (2009), Graves, Liwicki, Fernandez, Bertolami, Bunke, and Schmidhuber (2009)).
6. Gradient Sign Learning: Sven Behnke (2003) relied only on the sign of the gradient (R prop) when training his Neural Abstraction Pyramid to solve problems like image re-construction and face localization.
7. Alternate Methods to Structure ANN Learning: These use unsupervised pre-training to structure a neural network, making it first learn generally useful feature detectors.

Then the network is trained further by supervised back-propagation to classify labeled data.

8. Hinton's Deep Learning Model: The deep learning model Hinton, Osindero, and Teh (2006) involves learning the distribution of the high-level representation using successive layers of binary/real-valued latent variables. It uses a restricted Boltzmann machine (Smolensky (1986)) to model each new layer of high-level features. Each new layer guarantees an increase on the lower-bound of the log-likelihood of the data if trained properly, thus improving the model. Once sufficient number of layers have been learnt, the deep architecture maybe used as a generative model byb reproducing the data when sampling down the model (an "ancestral pass") from the top-level feature activations. Hinton (2009) reports that these are more effective feature extractors over high-dimensional, structured data.
9. High Level Comcept Learning in Action: The Google Brain team led by Andrew Ng and Jeff Dean created a neural network that learned to recognize high-level concepts such as cats purely by watching unlabeled images taken from youtube videos (Markoff (2012), Ng and Dean (2012)).
10. Learning with Compute Power: Ciresan, Meier, Masci, Gambardella, and Schmidhuber (2010) showed that, despite the "vanishing gradient problem", superior processing power of GPUs makes back-propagation feasible for deep feed-forward neural networks with many layers. Their method outperformed all other learning techniques on the old famous MNIST handwritten digits problem of LeCun.
11. State-of-the-Art in Deep Learning: Currently the state-of the-art in feed forward networks alternates convolutional layers and max-pooling layers (Ciresan, Meier, Masci, Gambardella, and Schmidhuber (2011), Martines, Bengio, and Yannakakis (2013)) topped by several pure classification layers. Training is done without supervised pre-training. Applications that use this architecture are described in Ciresan, Meier, Masci, Gambardella, and Schmidhuber (2011), Ciresan, Meier, Masci, and Gambardella (2012), and Ciresan, Giusti, Gambardella, and Schmidhuber (2012).

12. Human Competitive ANN's: Above architecture was also among the first artificial pattern recognizer to achieve human-competitive performance on certain tasks (Ciresan, Meier, and Schmidhuber (2012)).

Deep Learning Architectures

1. Deep Neural Network: A Deep Neural Network (DNN) is an ANN with more than one hidden layer of units between the input and the output layers. Similar to shallow ANN's, a DNN can model complex non-linear relationships. The extra layers give it added levels of abstraction, thereby increasing its modeling capability. DNN's are typically designed as feed-forward networks, but recent research has successfully applied the deep learning architectures to to recurrent neural networks for applications such as language modeling (Mikolov, Karafiet, Burget, Cernocky, and Khudanpur (2010)).
2. Convolution DNN: Convolutional DNN (CNN) is used in computer vision where its success is well-documented (LeCun, Bottou, Bengio, and Haffner (1998)). More recently, CNN's are used for acoustic modeling of ASR with considerable improvements over previous models (Sainath, Mohamed, Kingsbury, and Ramabhadran (2013)).
3. DNN Formulation: A DNN can be discriminatively trained with the standard back-propagation algorithm. The weight updates can be done via stochastic gradient descent using the following equation: $\Delta w_{ij}(t+1) = \Delta w_{ij}(t) + \eta \frac{\partial C}{\partial w_{ij}}$ where η is the learning rate, and C is the cost function.
4. DNN Parameters: The choice of the cost function depends on factors such as the learning types (supervised, unsupervised, reinforcement etc.) and the activation function. For example, when performing supervised learning on a multi-class classification problem, common choice for the activation function and the cost function are the softmax and the cross entropy function, respectively.

Challenges with the DNN Approach

1. Overfitting: Overfitting happens when the DNN is allowed to model rare dependencies between units that occur in the trining data. DNN's are very prone to overfitting because of their added layers of abstraction, which can allow them to model rare dependencies in the training data. Regularization methods weight decay (l_2-regularization) or sparsity regularization (l_1-regularization) can be applied during training to help combat overfitting (Bengio, Boulanger-Lewandowski, and Pascanu (2013)). Finally, in drouput regularization, some units are randomly omitted from the hidden layers during training – this helps break those rare dependencies that occur in the data (Dahl, Sainath, and Hinton (2013)).
2. Computation Time: Back propagation and gradient descent have been the preferred training method for training these structures due to the ease of implementation and their tendency to converge to better local optima in comparison with other training methods. However, they can be computationally intensive, esp. when applied to DNN's.
3. Computation Details: There are many parameters to be considered with a DNN – such as the size (the number of layers, and the number of units per layer), the learning rate, the initial weights, etc. Sweeping through the parameter space may not be feasible for many tasks due to the time cost (Hinton (2010)). Various tricks such as 'mini-batchin" (where gradients on several training examples are computed at once, rather than individual examples (Hinton (2010))) have been shown to speed up computation.
4. GPU Power Deployment: The sheer processing power of GPU's has had the most significant impact, particularly in the matrix and the vector computation space. However, it is hard to make use of large cluster machines for training DNN's, so better methods of parallelizing the training are necessary.

Deep Belief Networks (DBN)

1. Definition: A DBN is a probabilistic generative model made up of multiple layers of hidden units. It can be looked at as a combination of simple learning modules that make up each layer (Hinton (2009)).
2. Training a DBN: A DBN can be used fo generatively pre-training a DNN by using the learned weights as initial weights. Back propagation or other discriminative algorithms can then be used for fine-tuning these weights. This is particularly useful in situations where limited training data is available, as poorly initialized weights can have significant impact on the performance of the final model. These pre-trained weights are in a region of weight space closer to optimal (than just random initialization). This enables improved modeling and faster convergence during fine-tuning (La Rochelle, Erhan, Courville, Bergstra, and Bengio (2007)).
3. DBN as Layers of RBM: A DBN can be effectively trained in an unsupervised, layer-by-layer manner where the layers are typically made up of restricted Boltzmann machines (RBM's). An RBM is an undirected, generative energy based model with an input layer and a single hidden layer. Connections only exist between the visible units of the input layer and the hidden units of the hidden layer; there are no visible-visible or hidden-hidden connections.
4. RBM Training: This is also known as Contrastive Divergence (CD, Hinton (2002)). This method provides an approximation to ML that would ideally be applied for learning the weights of a RBM (Hinton (2010)).
5. Weight Updates in RBM Training: In training a single RBM, weight updates are performed using gradient descent via the equation $\Delta w_{ij}(t+1) = \Delta w_{ij}(t) + \eta \frac{\partial log[p(\vec{v})]}{\partial w_{ij}}$. Here $p(\vec{v})$ is the probability of the visible vector $\vec{v}$, and is given by $p(\vec{v}) = \frac{1}{Z}\sum_h e^{-E(\vec{v},h)}$ where Z is the partition function normalizer, $E(\vec{v},h)$ is the energy function assigned to the state of the network – olwer energy indicates that the network is in a more desirable configuration.

6. Gradient in RBM Training: The gradient $\frac{\partial log[p(\vec{v})]}{\partial w_{ij}}$ has the simple form $\langle v_i h_j \rangle_{Data} - \langle v_i h_j \rangle_{Model}$ where the expectation are with respect to $p(\vec{v})$. The practical problemin sampling $\langle v_i h_j \rangle_{Model}$ is that this requires running alternating Gibbs' sampling for a long time. CD addresses this by running Gibbs' sampling for a smaller number of steps ($n = 1$ has been shown to perform well). After n steps, the data is sampled and used in place of $\langle v_i h_j \rangle_{Model}$ (Hinton (2010)).
7. The CD Procedure:
    a. Initialize the visible units to a training vector.
    b. Update the hidden units in parallel given the visible units from $p(h_j = 1|\vec{v}) = \sigma\left(b_j + \sum_i v_i w_{ij}\right)$. Here, σ represents the sigmoid function, whereas b_j is the bias of h_j.
    c. Update the visible units in parallel given the hidden units from $p(v_j = 1|\vec{h}) = \sigma\left(a_j + \sum_i h_i w_{ij}\right)$. a_i is the bias of v_i, and this step is called the reconstruction step.
    d. Re-construct the hidden units in parallel given the visible reconstructed units as in the previous step.
    e. Finally perform the weight update from $\Delta w_{ij} \propto \{\langle v_i h_j \rangle_{Data} - \langle v_i h_j \rangle_{Model}\}$, with the proportionality constant being the ANN update rate.
8. Stacking up RBM's: Once an RBM is trained, another can be stacked on top to create a multi-layer model. With each stack, the corresponding visible input layer can be initialized to a training vector, and values for the units in the already trained RBM layers are assigned using the current weights and biases.The final layer of the already trained layers is used as an input to the new RBM. The new RBM is trained as above, and the whoe process is repeated until a stpooing criterion is achieved (Bengio (2009)).
9. Effectiveness of the CD Algorithm: Despite CD's crude similarity to ML (CD has been shown to not follow the gradient descent of any function), empirical results have shown CD to be an effective method for use with deep training architectures (Hinton (2010)).

Convolutional Neural Networks (CNN)

1. Definition: A CNN is composed of one/more convolutional layers with fully connected layers on the top. The CNN uses tied weights and pooling layers. This architecture enables CNN's to take advantage of the 2D data of the input layer.
2. Advantages of a CNN: The CNN has demonstrated strengths in both image and speech applications. They may also be trained with the standard back propagation techniques. CNNs are easier to train than the other regular, deep, feed-forward ANN's and have much fewer parameters to estimate.

Deep Learning Evaluation Data Sets

1. TIMIT - ASR Evaluation Data Set: TIMIT is a common data set used for the evaluation of the Deep Learning Architecture. It contains 630 speakers from 8 major dialects of American English, with each speaker reading 10 different sentences (TIMIT (1993)). The small size allows many different configurations to be tried effectively.
2. MNIST Image Classification Data Set: MNIST is composed of hand-written digits and is composed of 60,000 training examples and 100,000 test examples. Similar to TIMIT, its smaller size allows multiple configurations to be tested. A comprehensive list of results on this set are presented in MNIST (2013), and the current best result is an error rate of 0.23% (Ciresan, Meier, and Schmidhuber (2013)).

Neurological Basis of Deep Learning

1. Neurological Computational Brain Learning: The neocortex appears to employ computational deep learning models using a hierarchy of filters, where each layer

captures some of the information of the operating environment and the passes the remainder (along with a modified base signal) to layers further up in the hierarchy (Blakeslee (1995), Elman (1996), Shrager and Johnson (1996), Quartz and Sejnowski (1997), Utgoff and Stracuzzi (2002)).

2. Deep Learning as a Result of Human Cognition: The theory of deep learning sees the co-evolution of culture and cognition as a fundamental condition of human evolution (Shrager and Johnson (1995), Bufill, Agusti, and Blesa (2011)).

Criticism of Deep Learning

1. Theoretical Foundations: Many criticism of deep learning stems from the lack of theory surrounding many of the methods. Most of the deep learning architecture are built around some form of gradient descent. Theoretical treatment of related algorithms such as CD are unclear (does it converge? Is it fast? Exactly what does it approximate?). Most confirmations are empirical, not theoretical.

Hierarchical Clustering

1. Definition: Hierarchicial Clustering is a Method of Cluster Analysis that seeks to build a hierarchy of clusters (Hierarchical Clustering (Wiki)). Strategies for hierarchical clustering fall into 2 categories:
    a. Agglomerative Clustering => This is a bottoms up approach; each element starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
    b. Divisive Clustering => This is a top down approach; all observations start in one cluster, and splits are done recursively as one moves down the hierarchy.
2. Complexity: General agglomeration complexity is $\sim O(n^3)$, making it too slow for large data sets. Divisive Clustering is worse, consuming $\sim O(2^n)$ for an exhaustive search. In certain special cases optimal agglomeration of $\sim O(n^2)$ is known, e.g., SLINK (Sibson (1973)) for single linkage clustering and CLINK (Defays (1977)) for complete linkage clustering.
3. Clustering Dissimilarity Metrics: In order to decide which clusters need to be combined (for agglomerative clustering) or where a cluster needs to be split (for divisive clustering), a measure of dissimilarity between the sets of observations is required. In most methods of hierarchical clustering this is achieved by using an appropriate metric (via a measure of distance between a pair of observations), and a linkage criterion that specifies dissimilarity of sets as a function of pairwise distances of observations between the sets. Some common distance metrics are given in DISTANCE (2009).
    a. Euclidean Distance => $\|a-b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$.
    b. Squared Euclidean Distance => $\|a-b\|_2{}^2 = \sum_i (a_i - b_i)^2$.
    c. Manhattan Distance => $\|a-b\| = \sum_i |a_i - b_i|$.
    d. Maximum Distance => $\|a-b\|_\infty = \max_i (|a_i - b_i|)$.
    e. Mahalanobis Distance => $\sqrt{(a-b)S(a-b)^T}$ where S is the covariance matrix.

f. Cosine Similarity => $\frac{a.b}{\|a\|\|b\|}$.

4. Metrics for Text/Non-Numeric Data: In this case metrics such as the Hamming distance or the Levenshtein distance or used.
5. Clustering Dissimilarity Linkage Criterion: This determines the distance between sets of observations as a function of the pairwise distances between the observations. Some commonly used linkage creiteion are (Szekely and Rizzo (2005), CLUSTER (2009)):
    a. Maximum or Complete Linkage Clustering => $max\{d(a,b): a \in A, b \in B\}$
    b. Minimum or Single Linkage Clustering => $min\{d(a,b): a \in A, b \in B\}$
    c. Mean or Average Linkage Clustering (UPGMA) => $\frac{1}{|A||B|}\sum_{a\in A}\sum_{b\in B} d(a,b)$
    d. Minimum Energy Clustering => $\frac{2}{nm}\sum_{i,j=1}^{n,m}\left\|a_i - b_j\right\|_2 - \frac{1}{n^2}\sum_{i,j=1}^{n}\left\|a_i - a_j\right\|_2 - \frac{1}{m^2}\sum_{i,j=1}^{m}\left\|b_i - b_j\right\|_2$
6. Other Linkage Criterion:
    a. Sum of all inter-cluster Variance
    b. Decrease in Variance for the Cluster being Merged – the Ward's Criterion (Ward (1963))
    c. Probability that the candidate clusters spawn from the same distribution function (V-linkage)
    d. The product of the in-degree and the out-degree in the k-nearest-neighbor graph (graph degree linkage (Zhang, Wang, Zhao, and Tang (2012))
    e. The increment of some Cluster descriptor (i.e., the metric defined for some measuring the quality of the cluster) after the merge/separation (Ma, Dirksen, Hing, and Wright (2007), Zhao and Tang (2008), Zhang, Zhao, and Wang (2013)).

k-Means Clustering

Introduction

1. Definition: k-Means Clustering is a method of vector quantization that aims to partition n observations into k clusters in which each observatione belongs to the cluster with the nearest mean, serving as a prototype of the cluster.
2. k-Means Complexity:It is NP-hard – however there exist efficient heuristic algorithms that converge quickly to a local optimum. These algorithms are similar to EM on Gaussian mixture Distributions.
3. Differences between k-Means and EM Algorithms: While both k-means and the EM algorithms cluster centers to model the data, k-means finds clusters of comparable spatial extent, while EM allows clusters to have different shapes.
4. Development History: The standard algorithm (Steinhaus (1957), MacQueen (1967)) was originally used in Bell Labs for pulse code modulation (Lloyd (1957), Forgy (1965)). An efficient version of this was created later by Hatrigan (1975) and Hartigan and Wong (1979).

Mathematical Formulation

1. Specification: Given a set of observations $\{\overrightarrow{x_i}\}_{i=1}^{n}$ where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k sets $(k < n)$ $\{\vec{S}\}_{j=1}^{k}$ so as to minimize the within-cluster sum of squares (WCSS): $\arg\min_{\vec{S}} \sum_{i=1}^{k} \sum_{x_j \in S_i} \left\|\overrightarrow{x_j} - \overrightarrow{\mu_i}\right\|^2$ where $\overrightarrow{\mu_i}$ is the mean of the points in $\overrightarrow{S_i}$.

The Standard Algorithm

1. The Layout: This is also referred to as the Lloyd's algorithm. Given an initial set of k-means $m_1^{(1)}, \ldots, m_k^{(1)}$, the algorithm proceeds by alternating between 2 steps – the assignment step and the update step (MacKay (2003)).
2. The Assignment Step: Assign each contribution to the cluster whose assignment yields the least WCSS. Since the sum-of-the-squares is the Euclidean distance, this algorithm intuitively chooses the nearest mean. Mathematically, this partitions the observations according to the Voronoi diagram generated by the means: $S_i^{(t)} = \left\{x_p: \left\|x_p - m_i^{(t)}\right\|^2 \leq \left\|x_p - m_j^{(t)}\right\|^2 \forall\, i, j; i \leq j \leq k\right\}$. x_p is assigned to exactly one $S_i^{(t)}$ (even if it is suitable to be assigned to more than to one).
3. The Update Step: Calculate the new means to be centroids of the observations in the new cluster: $m_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|}\sum_{x_j \in S_i^{(t)}} x_j$. Snce the arithmetic mean is a least squares estimator, this step also minimizes the WCSS objective.
4. k-Means Convergence: The aalgorithms is deemed to have converged when the assignments no longer change. Since both the steps optimize the WCSS objective, and there exist only a finite number of partitionings, the algorithm must convege to a local optimum (no guarantees of the global optimum).
    a. Convergence to the local optimum may depend on the initial clusters, so it is common to run it multiple times with different starting conditions (esp. since the algorithm is very fast). In the worst case, it can be very slow; in particular, it has been that exist certain set of starting points, even in 2 dimensions, on which the k-means takes an exponential time, i.e., $2^{\Omega(n)}$ to converge (Vattani (2011)).
5. Comparison with EM Algorithm: The assignment step is also referred to the expectation step, while the update step is referred to as the maximization step, making the k-means a variant of the generalized EM algorithm.
6. The Distance Metric: Given that the WCSS minimization occurs in both the steps, naively using other distance metrics (such as non-Euclidean metrics) will cause the

algorithm to not converge. Various modifications to k-means such as spherical k-means and k-medoids have been proposed to allow for using alternate distance metrics.

7. Drawbacks:
    a. Euclidean distance is used as the distance metric and variance is used as a measure of the cluster scatter (i.e., extraneously specified distribution scatter measures such as higher moments etc: are not specifically accommodated)
    b. The number of clusters k is an input parameter, and an inappropriate choice of k may lead to poor results. This makes it important to run diagnostic checks for determining the appropriate k.
    c. Convergence to a local minimum may produce counter-intuitive/wrong results.
8. Performance: Use of k-means has been successfully combined with simple, linear classifiers for semi-supervised learning in NLP (specifically for named entity recognition) (Lin and Wu (2009)), and in computer vision. Object recognition tests indicate that k-means has comparable performance with more sophisticated feature learners such as auto-encoders and RBMs (Coates, Lee, and Ng (2011)). However, it requires more data than for these methods to produce equivalent performance, since each data point only contributes to feature than multiple (Coates and Ng (2012)).

k-Means Initialization Schemes

1. The Standard Schemes: Commonly used initialization methods are the Forgy method and the Random Partition Method (Hamerly and Elkan (2002)). The Forgy method randomly chooses k observations from the data set, and uses them as the initial means. The random partition method first randomly assigns a cluster to each observation and then proceeds to the update step, thus computing the initial means to be centroid of the randomly computed points.
2. Comparison of Forgy and Random Partition: In effect, the Forgy method tends to spread the initial means out, while the random partition method places all of them

close to the center of the data set. The random partition method is preferred to the extensions of the standard k-means such as k-harmonic means or fuzzy k-means. For standard k-means and EM, the Forgy method is preferred.

k-Means Complexity

1. Generics: In general, finding the optimal solution to k-means clustering problems in d dimensions is:
    a. NP-Hard in the general Euclidean space even for 2 clusters (Aloise, Deshpande, Hansen, and Popat (2009), Dasgupta and Freund (2009))
    b. NP-Hard for k clusters even in a single plane (Mahajan, Nimbhorkar, and Varadarajan (2009))
    c. If k and d are fixed, the time for exactly solution is $\sim O(n^{dk+1} \log n)$ where n is the number of entries to be clustered (Inaba, Katoh, and Imai (1994))
    d. It is important to recall that all of these are heuristic algorithms that seek out a local optimum
2. Lloyd's k-mean Complexity: Lloyd's k-means algorithm has polynomial smoothed running time. It is shown (Arthur, Manthey, and Roeglin (2009)) that for an arbitrary set of n points in $[0, 1]^d$, if each point is independently pertured with a mean θ and a variance σ^2, the expected running time of k-means is bounded by $O\left(\frac{n^{34}k^{34}d^8 log^4\,(n)}{\sigma^2}\right)$, which is a polynomial in n, k, d and $\frac{1}{\sigma^2}$.
3. Other k-means Complexity Estimates: Estimates for other specific k-means configurations have been worked out, for e.g., Arthur and Bhowmick (2009) have shown that the running time for k-means on n points in the lattice $\{1, \dots, M\}^d$ is bounded by $O(dn^4M^2)$.

k-Means Variations

1. k-Medians Clustering: k-medians clustering uses the median in each dimension instead of the means, and in this way minimizes the L_1 norm (the Taxicab Geometry).
2. k-Medoids Clustering: Also called Partitioning Around Medoids (PAM), this uses the medoids instead of the means, and in this way minimizes the sum of distances for arbitrary distance functions.
3. Fuzzy C-Means Clustering: This is a softer version of the k-means, where each entry has a "degree" of belonging to each cluster.
4. Gaussian Mixtures: Clustering done with Gaussian mixture models trained using EM maintains the probabilistic assigments to clusters (instead of the deterministic assignments) alongwith multivariate distributions instead of a single set of means.
5. Starting Clusters: Several methods have been proposed to choose better starting clusters (e.g., k-means++).
6. Speeding up Iteration Steps: Kanungo, Mount, Netanyahu, Piatko, Silverman, and Wu (2002) propose a filtering algorithm that uses kd-trees to spped up each k-means step. Other methods to speed up k-means use the coresets (Frahling and Sohler (2006)) or the triangle inequality (Elkan (2003)).
7. Escaping Local Maxima: Escape local maxima by swapping points between the clusters (Hartigan and Wong (1979)).
8. Spherical k-means: The spherical k-means clustering is suitable for directional data (Dhillon and Modha (2001)).
9. Minkowski Weighting: The Minkowski weighted k-means deals with each irrelevant feature by assigning cluster specific weights to each feature (Amorim and Mirkin (2012)).

k-Means Applications

1. Application Range: Given that heuristic algorithms such as Lloyd's are easy to omplement even on large data sets, k-means is used widely in problems ranging from market segmentation, computer vision, geostatistics (Honarkhah and Caers (2010)),

astronomy, agriculture, etc. It is often used as a pre-processing step for other algorithms, for example, to find a starting configuration.

2. Vector Quantization: In computer graphics color quantization is the task of reducing the color palette to a fixed number of colors k. The k-means algorithm can be used for such quantization tasks and produces very competitive results. Other uses of vector quantization include non-random sampling, as k-means can be easily used to choose k different but prototypical objects from a large data set (possibly for further analysis).
3. Feature Learning: k-means clustering has been used as a feature learning (or dictionary learning) step in semi-supervised or unsupervised learning (Coates and Ng (2012)). The basic approach is to train a k-means clustering representation using the input training data that need not be labeled. Then, to project the input datum into the new feature space, we have a choice of encoding functions such as:
    a. Thresholded matrix-product of the datum with centroid locations
    b. Distance from the datum to each centroid
    c. Indicator function for the nearest centroid (Csurka, Dance, Fan, Willamowski, and Bray (2004), Coates and Ng (2012))
    d. Smooth transformation of the computed "distance" (Coates, Lee, and Ng (2012))
    e. Alternatively, by transforming the sample-cluster distance through a Gaussian RBF, one obains the hidden layer of a radial basis function network (Schwenker, Kestler, and Palm (2001))

Alternate k-Means Formulations

1. Generalized k-Means: k-means clustering, and its associated EM algorithm, is a special case of Gaussian mixture model – specifically, where th limit of covariances are diagonal, equal, and small (Press, Teukolsky, Vetterling, and Flannery (2007)). Another generalization is the k-SVD algorithm at estmates the data points as sparse linear combination of "codebook" vectors. K-means corresponds to the special case

of using a single codebook vector with unit weight (Aharon, Elad, and Bruckstein (2006)).

2. Mean-Shift Clustering: The basic means-shift clustering maintains a set of points the same size as the input data set. Initially, this set is copied from the input set. This set is then iteratively replaced by the mean of those points in the set that are within a given distance of that point. By contrast, k-means restricts this updated set to k points usually much less than the number of points in the input data set, and replaces each point in this set by the means of all points in the INPUT SET that are closer to the point than any other (e.g., within the Voronoi partition of each updating point).
    a. Variation => A mean shift algorithm that is more similar to k-means (called likelihood mean shift) replaces the set of points undergoing replacement by the mean of all points in the input set that are within a given distance of the changing set (Little and Jones (2011)).
    b. Properties => One of the advantages of mean shift clustering over k-means is that there is no need to choose the number of clusters, since mean shift is likely to find only a few clusters if only a small number exist. However, mean shift clustering can be slower than k-means, and still requires the selection of a bandwidth parameter. Mean shift also has soft variants like k-means.
3. PCA vs. k-means: It has been asserted (Zha, Ding, Gu, He, and Simon (2001), Ding and He (2004)) that the relaxed solution to k-means clustering, specified by the cluster indicators, is given by the PCA's principal components, and the PCA sub-space spanned by the principal directions is identical to the cluster centroid subspace. However, PCA being a useful relaxation of k-means is not a new result (Drineas, Frieze, Kannan, Vempala, and Vinay (2004)); further, it is straightforward to uncover counter-examples to the claim that the cluster centroid subspace is spanned by the principal components.
4. Bilateral Filtering: k-means implicitly assumes that the ordering of the inputs does not matter. The bilateral filter is similar to k-means and mean shift in that it maintains a set of data points that are iteratively replaced by their means. However, bilateral filtering restricts the calculation of the kernel-weighted mean to include only those points that are close in the ordering of the input data (Little and Jones (2011)). This

makes it applicable to problems such as image de-noising where spatial arrangements of pixels in an image is of critical importance.

5. Other similar Problem Spaces: The set of squared error minimizing cluster functions includes the k-medoid algorithm, an approach that forces the center point of each cluster to be one of the actual points, i.e., it uses medoids in place of centroids.

Correlation Clustering

1. Definition: Correlation Clustering provides a method for clustering a set of objects into the optimum number of clusters without specifying that number in advance (Becker (2005)).
2. Problem Description: In machine learning correlation clustering or cluster editing operates in a scenario where the representation of the relationships between the objects is what is known (in place of their actual representations).Given a signed graph $G = (V, E)$ where the edge labels indicate similarity/dissimilarity $(+/-)$ of the nodes, the task is to cluster the vertices so that similar objects are grouped together. Unlike other clustering algorithms, this does not require choosing the number of clusters in advance because the objective (which is to minimize the net disagreements) is independent of the number of clusters.
3. Imperfect Correlation Clustering: It may not be possible to find a perfect clustering where all the similar items are in one cluster, and the dissimilar ones are in different clusters. The problem of maximizing the agreements in NP-complete – the multi-way problem reduces to maximizing the weighted agreements, and the problem of partitioning into triangles can be reduced to the unweighted one (Garey and Johnson (2000)).
4. Constanct Factor Approximation Algorithm: Bansal, Blum, and Chawla (2004) discuss the NP-completeness proff and present constant factor algorithm and polynomial time approximation scheme to find the clusters in this setting. Ailon, Charikar, and Newman (2005) propse a randomnized 3-approximation algorithm for the above (Correlation Clustering (Wiki) contains the pseudo-code for this).
5. Optimal Number of Clusters Evaluation: Optimization of the correlation clustering functional is closely related to the well-known discrete optimization methods (Bagon and Galun (2011)). Bagon and Galun (2011)) also propose a probabilistic of the underlying implicit model that allows the correlation clustering to evaluate the underlying number of clusters. The analysis suggests that the functional assumes a

uniform prior over all partitions regardless of the number of clusters – thus, a uniform prior over the number of clusters emerges.

   a. Optimal Number of Clusters under specific scenarios => Bagon and Galun (2011)) propse several discrete optimization algorithms that scale gracefully with the number of elements (which can be over 100,000 variables for certain applications). They also evaluate the effectiveness of the recovery of the underlying number of clusters in a number of applications.
   b. They also evaluate the effectiveness of the recovery of the underlying number of clusters in a number of applications.

6. Feature Vector Correlation: In certain situations in data mining, correlations among feature vectors in high-dimensional space guide the clustering process. These correlations can be different in different cluster, therefore a global de-correlation cannot reduce this to typical uncorrelated clustering.
   a. Feature Vector Correlation Clustering => Correlations among the subsets of attributes results in different shapes of clusters. Therefore, the similarity between cluster objects needs to be defined by taking into account local correlation patterns. Thus, different types of correlation clustering (esp. high-dimensional clustering) and their relationships to standard correlation clustering techniques have been discussed in Bohm, Kailing, Kroger, and Zimek (2004), Zimek (2008), and Kriegel, Kroger, and Zimek (2009).

Kernel Principal Component Analysis (Kernel PCA)

1. Definition: Kernel PCA is an extension of the standard PC using the techniques of the kernel methods. With kernel PCA, the original linear operations of PCA are done in a reproducing kernel Hilbert space with a non-linear mapping (Kernel PCA (Wiki)).
2. Motivation: In general a set of N data points cannot be separated if $d < N$; they can almost always be separated in $d \geq N$ dimensions. That is, given N points $\{\overrightarrow{x_i}\}$, if we map them to an N-dimensional space with $\Phi(\overrightarrow{x_i})$ where $\Phi: R^d \rightarrow R^N$, it is easy to construct a hyperplane that divides the points into arbitrary clusters. This Φ creates a linearly independent set of vectors, so there is no covariance on which to perform eigen-decomposition explicitly as we would in linear PCA (Scholkopf, Smola, and Muller (1996)).
3. The "Kernel" Trick: In kernel PCA a non-trivial, arbitrary function Φ is "chosen" but never calculated explicitly, allowing for the possibility to use very high dimensional Φ's if we never have to actually evaluate the data in that space. Since we generally try to avoid working in the Φ-space (we call this the "feature space"), we create the $N \times N$ kernel $K = k(x, y) = [\Phi(x), \Phi(y)] = \Phi(x)\Phi^T(y)$ which represents the inner product space (i.e., the Gramian matrix) of an otherwise intractable feature space.
4. Feature Space Projections: Since we never actually work in the feature space, we do not need to compute its covariances and eigenvectors. This also implies thatb we do not explicitly compute the principal components themselves, but only compute the projections of our data onto these components. To evaluate the projection from a point in the feature space $\Phi(\vec{x})$ onto the k^{th} principal component, we compute it using $V_k^T \Phi(\vec{x}) = \left[\sum_{i=1}^{N} a_{i,k} \Phi(\overrightarrow{x_i})\right] \Phi(\vec{x})$.
5. Computation of the Kernel Coefficients: $\Phi(\overrightarrow{x_i})\Phi^T(\vec{x})$ is just the dot product, i.e., the elemets of the K matrix. Eigenization of $\Phi(\overrightarrow{x_i})\Phi^T(\vec{x})$ and the eventual normalization results in $a_{i,k}$.
6. Feature Space Mean Centering: Care must be taken regarding the fact that, whether or not $\vec{x}$ has zero-mean in the original space, it is not guaranteed to be centered in the

feature-space (whoch we never compute explicitly anyway). Since centering is required for PCA, we centralize $K \rightarrow K'$ using $K' = K - 1_N K - K 1_N + 1_N K 1_N$ where 1_N denotes a $N \times N$ matrix for which each element takes the value $\frac{1}{N}$. K' is used for the PCA above.

7. Kernel PCA Ranking: In linear PCA we use the eigenvalues to rank the eigenvectors based on how much variance is captured by each principal component on the given set of inputs points – this is used, amongst other things, for dimensionality reduction. No such ranking exists in kernel PCA, however.
8. Kernel PCA on Large Datasets: In praxtice, a large dataset leads to a large K, and storing those K components can become a problem. One way to deal with this is to cluster the large dataset, and replace the kernel with the means of the clusters. Since even this method may yield a large K, it is common to compute the top P eigen-vectors and eigen-values of K (bearing in mind that these do NOT necessarily translate into the corresponding maximal variance explainers of the original points).
9. Applications: Kernel PCA has been demonstrated to be useful for novelty reduction (Hoffmann (2007)) and image de-noising (Mika, Scholkpof, Smola, Muller, Scholz, and Ratsch (1999)).

Machine Learning

Calibration vs. Training

1. Nomenclature Fix: Training is a machine learning term with no explicit attempt at establishing/inferring states. However, calibration is in one sense a true state-inference technique, constructed with a view of incorporating the specific additional constraints:
    a. A unified latent state representation across the predictor ordinate range.
    b. Explicit mathematical associative link to one/more underlying drivers.
    c. Basis hypotheses set chosen more rigorously to adhere to the underlying "physics/neumenology".
    d. Volatility and State Dynamics are afforded significant consideration in the build-out of the inferred states.
2. The Conceptual Challenge: Currently state of the art for machine learning is that it ends up delegating the challenge of choosing the basis function set to the "physics" of the problem. Thus, by this very nature, it ends up getting more widely deployed for problems that already possess a sharp cognitive clarity (e.g., image recognition).
3. Calibrated Oriented Computational Analyses: Sensitivity computation, hedge estimation (where hedge is defined as the real world scaling/marking-to-market of latent state sensitivity), relative value computation, Custom Latent State scenario variation estimation, etc. are all primarily meaningful only in the calibration context. "Prediction" is the one operation that is meaningful to both training and calibration (although the inherent value on a single predicted value is of less epistemic significance in calibration, and opposed to training).

Ensemble Learning

Introduction

1. Definition: In statistics and machine learning ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any one of the constituent learning algorithms (Opitz and Maclin (1999), Polikar (2006), Rokach (2010)).
1. Differences with Statistical Mechanics: Unlike a statistical ensemble in statistical mechanics, which is usually infinite, a machine learning ensemble refers only to a concrete set of alternative models, but typically allows for much more flexible structure to exist between these alternatives.

Overview

1. Weak Learners vs. Strong Learners: Ensembles combine multiple hypotheses to form a better hypothesis – thereby presenting a technique for combining many weak learners to form a strong learner.
2. Ensembles vs. Multiple Classifier Systems: The term ***ensemble*** is reserved here for methods that generate multiple hypotheses using the same base learner. The broader term ***multiple classification systems*** also covers hybridization of the hypotheses not induced by the same base learner.
3. Motivation for Usage in Supervised Learning: Supervised learning algorithms search through a hypothesis space to locate a suitable hypothsis that has the appropriate prediction capabilities. Even if the hypothesis space contains hypotheses that are well-suited for a particular problem, it may be difficult to find a good (i.e., well-balanced) one.

4. Performance Advantages: Evaluating the performance of an ensemble typically requires more computation that evaluating the performance of a single model, so ensembles may be thought of as a way to compensate for poor learning algorithms at the expense of more computation. Fast algorithms such as decision trees are commonly used with ensembles (e.g., random forest), although slower algorithms can benefit from ensemble techniques as well.

Theoretical Underpinnings

1. Ensemble Hypotheses Space: Since an ensemble itself can be trained for predictions, it is a supervised algorithm in itself. However, as a trained ensemble, it represents a single hypothesis. The ensemble hypothesis space, therefore, is not necessarily contained within the space of the models from which it is built.
2. Ensemble Predictive Representation Flexibility: The above mentioned representation ability results in greater representation flexibility, at the cost of over-fitting the training data more than that for a single model. In practice, certain ensemble techniques (esp. bagging) tend to reduce problems related to over-fitting of the training data.
3. Predictor Diversity: Empirically, ensembles tend to yield better results when there is significant diversity among the models (Sollich and Krogh (1996), Kuncheva and Whitaker (2003)). Many ensemble methods therefore seek to promote diversity among the models they combine (Brown, Wyatt, Harris, and Yao (2005), Garcia-Adeva, Cervini, and Calvo (2005)).More random algorithms such as random decision trees can be used to produce a stronger ensemble than very deliberate algorithms such as entropy-reducing decision trees (Ho (1995)).
4. Strong Learners: Using a variety of strong learning algorithms, however, has been shown to be more useful than using techniques that "dumb down" the models to promote diversity (Gashler, Giraud-Carrier, and Martinez (2008)).

Ensemble Aggragator Types

1. Type 1: This is based purely on Bayes' Optimal Classification (see below). Aggregation here occurs exclusively across the hypotheses ensemble space, and training set remains fixed between classification runs. Examples of this type are Ensemble Averaging Techniques, BMA, BMC, certain BOM variants, and stacking.
2. Type 2: These are based off of cross-validation philosophy, and therefore involve generating subsets of re-samples and their eventual re-combination. Aggregation occurs purely across the re-sampling space and not the hypotheses space (oftern there will only be a single model). Examples of this type include bagging and boosting.
3. Type 3: This type is an extension of the two types above, and the expectation is that the aggregation should occur across both the hypothses space as well as the re-sample set according to $y = \underset{c_j \in C}{\arg max} \sum_{T_k \in T} \sum_{h_i \in H} P(c_j|h_i)P(T_k|h_i)P(h_i)$. The cross-validation based variants of BOM (e.g., gating based cross-validation selectors) could be applying the above.

Bayes' Optimal Classifier

1. Definition: This refers to an ensemble that contains all the hypotheses in the hypothesis space. On average (and over sufficient sample sizes) no other ensemble can outperform it (by construction), since this is the ideal ensemble that others attempt to mirror (Mitchell (1997)).
2. Conception Philosophy: Each hypothesis is given a vote proportional to the likelihood that the training data set would be sampled from the system if that corresponding hypothesis were true. This setup facilitates evaluation under the world where the hypothesis is valid. To accommodate training data set of finite size, the vote of each hypothesis is also multiplied by the prior probabilities of that hypothesis.
3. Formulation: $y = \underset{c_j \in C}{\arg max} \sum_i P(c_j|h_i)P(T|h_i)P(h_i)$ where y is the predicted class, c_j is the class instance j that is part of the bigger set of classes C, h_i is the hypothesis

i that is part of the hypothesis universe H; T is the set of all observations; $P(c_j|h_i)$ and $P(T|h_i)$ are the probabilities of realization of the class c_j and the observation set T given hypothesis h_i; and $P(h_i)$ is the probability of realization of the hypothesis h_i.

4. Hypothesis Space Optimality: The hypothesis represented by the Bayes Optimal Classifer is the optimal hypothesis in the ensemble space (the space of all possible ensembles consisting only of those hypothesis in H).
5. Implementation Challenges:
    a. Most interesting hypotheses space can be too large to iterate over, as required by $\arg max$
    b. Many hypothesis yield only a predicted class rather than the probability term $P(c_j|h_i)$
    c. Computing the unbiased estimate of the probability of training set given hypothesis (i.e, $P(T|h_i)$) is rarely feasible
    d. Estimating $P(h_i)$ is rarely possible

Bagging and Boosting

1. Boostrap Aggregating (Boosting): Bagging involves each model in the ensemble vote with equal weight (this could be $P(T|h_i)$ or $P(h_i)$ or both – i.e., a constant independent of h_i).
2. Model Diversity in Bagging: In order to promote model variance, bagging trains each model in the ensemble using a randomly drawn subset of the training data. As an example, random forest trees combine random decision trees with bagging to achieve very high classification accuracy (Breiman (1996)). Sahu, Runger, and Apley (2001) provide an interesting application of bagging in the unsupervised learning context.
3. Boosting: Boosting involves incrementally building an ensemble by training each new model instance to emphasize the training instances that the previous model set mis-classified. In some cases, boosting has been shown to yield better accuracy than bagging – however, it also contains a tendency to over-fit. The most common

boosting implementation is AdaBoost, although some newer algorithms are reported to achieve better results.

Bayesian Model Averaging (BMA)

1. Philosophy: BMA is en esemble technique that seeks to approximate Bayes' Optimal Classifier by sampling hypotheses from the hypothesis space, and combining them using Bayes' law (Hoeting, Madigan, Raftery, and Volinksy (1999)).
2. Methodology: Unlike the Bayes' Optimal Classifier, BMA implementation is practically achieved. Hypotheses are usually sampled (i.e., $P(h_i)$ determined) using a Monte Carlo sampling technique such as MCMC. Further, Gibbs' sampling may be used to draw hypothsis that is consistent with the distribution (i.e., $P(T|h_i)$). Wiki (Emsemble Learning) contains illustrative pseudo-code.
3. Asymptotic Error Rate: It has been shown that under certain circumstances, when the hypotheses are drawn according to BMA and averaged in accordance with Bayes' law, BMA has an error rate that is bounded to be within twice the error rate of the Bayes' Optimal Classifier (Haussler, Kearns, and Schapire (1994)).
4. Caveats: Despite the empirical correctness, BMA has been found empirically to promote more over-fitting in comparison to simpler techniques such as bagging (Domingos (2000)), thereby motivating the development of techniques such as Bayesian Model Combination (Minka (2002)).
5. Conceptual Shortcoming of BMA: As seen earlier, the use of Bayes' law to compute model weights necessitates the computation of each $P(T|h_i)$. Typically none of the models in the ensemble is the precise model from the training data is generated, so all of them correctly receive a value close to zero for this term ($P(T|h_i)$). Thus, in principle BMA would work well if the hypotheses ensemble set was wide enough to span the entire model space, but that is rarely possible. Consequently each pattern in the training data causes the ensemble weight to shift towards the model in the ensemble that is closes to the distribution of the training data.

6. Amelioration of this BMA Shortcoming: The possible weightings for an ensemble can be visualized as lying along the vertices of a simplex. In BMA, at each vertex of the simplex, all of the weights is given to the single corresponding model of the ensemble. BMA converges toward the vertex that is closest to the training data – this is what can be changed.

Bayesian Model Combination (BMC)

1. BMC as an Enhancement to BMA: Unlike BMA, BMC converges towards the point where the weightings distribution projects onto the simplex. In other words, instead of selecting the model closest to the distribution, BMC seeks the combination of models that is closest to the generating distribution.
2. BMC Algorithm: Instead of sampling each model in the ensemble individually as in BMA, BMC samples the space of possible ensembles (the space has model weightings drawn randomly from a Dirichlet distribution having uniform parameters). This modification overcomes the tendency of BMA to converge toward giving all of the weight to a single model.
3. BMC Effectiveness: Although BMC is computationally more expensive than BMA, it tends to yield dramatically better results over it, as well as over bagging (Montieth, Carroll, Seppi, and Martinez (2011).
4. BMC in Practice: The results of BMA can often be approximated by using cross-validation to select the best models from a bucket of models. Likewise, the results of BMC may be approximated by using cross-validation to select the bestensemble combination from a random sampling of possible weightings. Wiki (Ensemble Learning) contains sample BMC pseudo-code.

Bucket of Models (BOM)

1. Introduction: A "Bucket of Models" is an ensemble in which a model selection algorithm is used to choose the best model for each problem. When tested with only one problem, the BOM can produce no better results than the best model in the hypotheses set, but when evaluated across many problems, it will typically produce much better results on the average than any single model in the set.
2. BMO Cross-Validation Selection: The most common approach used for model selection is cross-validation (also called a "bake-off contest" – Wiki (Ensemble Learning) contains sample pseudo-code). This may be summed as: "Try them all with the training set, and pick the one that works best" (Zenko (2004)).
3. Gating Cross-Validation Selection: Gating is a generalization of the cross-validation based sample selection. It involves using a separate learning model to decide which of the models in the bucket is best-suited for the problem.
4. Gating Perceptrons: Often, a perceptron is used for the gating model. The perceptron is used to pick the best model, or it can be used to assign a set of linear weights to the predictions from each model in the basket.
5. Landmark Cross-Validation: When a bucket of models is used with a large set of problems, it may be desirable to avoid training some of the models that take a long time to train. Landmark learning is a meta-approach that seeks to address this problem. It involves first training only the fast but imprecise algorithms in the bucket, and then using the performance of these algorithms to determine which slow but accurate algorithm is most likely to do best (Bensusan and Giraud-Carrier (2000)).

Stacking

1. Introduction: Also called a stacking generalization, this involves training a learning algorithm to combine the predictions of several other training algorithms. First, all of the other algorithms are trained using the training set, then a combiner algorithm is used to make a final prediction using all the predictions of the other algorithms as inputs.

2. Stacking as an Ensemble Generalizer: If an arbitrary stacking combiner algorithm is used, then stacking can theoretically represent any of the ensemble techniques described above. In practice, a single-layer logistic regression model is often used as the combiner.
3. Stacking Performance: Stacking typically yields performance better than any single one of the individual trained models (Wolpert (1992)). It has been successfully used on both supervised learning tasks (e.g., regression, see Breiman (1996)) as well as unsupervised learning tasks (e.g., density estimation, Smyth and Wolpert (1999)). It has been used to outperform BMA (Clarke (2003)) and to estimate bagging's error rate (Wolpert and MacReady (1999), Rokach (2010)). The two top performers in the Netflix competition used BLENDING which may be considered to be a form of stacking (Sill, Takacs, Mackey, and Lin (2009)).

Ensemble Averaging vs. Basis Spline Representation

1. Parallel between Spline Representation and Ensemble Averaging Techniques: From a real-valued inference (i.e., regression + transformed classification) point-of-view, there are significant similarities between ensemble averaging techniques and basis spline representation techniques. While ensemble averaging attempts to aggregate over hypotheses set, multi-spline basis representation attempts to represent the splines over the set of the basis spline representation function set. Further, there are also parallels in the way in which the weights are inferred; for basis splines, this is achieved by a combination of best-fit and penalization, whereas for the ensemble aggregator this done via the variance-bias optimization mechanism.
2. Difference: Ensemble averaging may be viewed as a dual pass training exercise effectively – the first training pass trains the individual response basis functions themselves, and the second pass trains the weights across the basis function set.
3. Focus of the Training Passes: Focus of the first pass in ensemble averaging is simply bias reduction, i.e., enhancing the closeness of fit and reduction of Bayes' risk. The

second pass performs the ensemble averaging with a view to reducing the variance – this is comparable to the smoothening pass.

4. Ensemble Averaging vs. Multi-Pass State Inference: Boas reduction is comparable to the shape preservation pass, whereas the ensemble averaging/variance reduction is comparable to the smoothening pass.
4. Differences between Ensemble Averaging and Multi-Pass State Inference:
    a. The shape preserving pass ends up emitting a sequence of parameters that correspond to the stretches across the univariate predictor ranges to represent a SINGLE latent state.
    b. The calibration run during the shape preserving pass produces a set of calibrated parameters for the full set of basis splines for shape preservation (i.e., error minimization).
    c. Training each basis function separately druing the bias reduction phase of ensemble averaging does not really purport to "infer" any latent state.
    d. The second phase of ensemble aggregation simply re-works the basis weights across all the "low bias" basis functions, again no notion of "re-inferrring of states" involved.
    e. The smoothing phase of the multi-phase latent-state construction may work on a latent state quantification metric that is different from the shape preserving pass; while this does loosen the shape preservation impact, it is no more different to the equivalent step in ensemble averaging.

ANN Ensemble Averaging

Overview

1. Committee Machines: ANN Ensemble averaging one of the simplest types of committee machines. Along with boosting, it is one of the 2 types of static committee machines (Haykin (1999)). In contrast to standard network design techniques in which many networks are generated but only one is retained, ANN ensemble averaging holds on to the less satisfactory networks, but with lower weights (Hashem (1997)).
2. ANN Properties Manipulated by the Ensemble: This comes here from Naftaly, Intrator, and Horn (1997) via Wiki (Ensemble Averaging):
    a. In any ANN, the bias can be reduced at the cost of increased variance, AND
    b. In a group of networks, the variance can be reduced at no cost to bias.
3. ANN Variance-Bias Optimization: Ensemble averaging creates group of ANN each with low bias and high variance, then combines them to form an ANN that has low bias and low variance, thereby offering an optimal bias-variance resolution (Clemen (1989), Geman, Bienenstock, and Doursat (1992)).

Techniques and Results

1. Methodology: The bias-variance optimization idea above provides an obvious strategy: create a set of experts with low bias and high variance and average them. From an ANN PoV, what this means is the creation of a set of experts with varying parameters; often these are the initial synaptic weights, although other factors such as learning rate, momentum etc. may be varied as well (the drawbacks of varying weight decay and early stopping is discussed in Naftaly, Intrator, and Horn (1997)).
2. The Algorithm:

a. Generate N experts, each with their own initial values (the initial values are usually chosen randomly from a distribution).
b. Train each ANN separately.
c. Use the ensemble to combine the experts and average their values.

3. Classes of ANN Domain Experts: Alternatively, domain knowledge may be used to generate several classes of experts. An expert from each class is trained, and then combined.
4. Linear Combination of Experts: In place of averaging, one may also weight them using $\tilde{y}(\vec{x}, \alpha) = \sum_{j=1}^{p} \alpha_j y_j(\vec{x})$ where $y_j(\vec{x})$ correspond to the individual experts, and $\{\alpha\}$ to the set of weights. The optimizatioj problem of extracting $\{\alpha\}$ is readily solved through standard neural network approaches. Thus a meta-network where each each neuron is an ANN in itself can be trained, and the synaptic weights on the final network is the weight applied to each ANN expert (Hashem (1997)).
5. Negative Correlation Learning: A more recent ANN ensemble averaging method called negative correlation learning (Liu and Yao (1999)) has been widely used in evolutionary computing.
6. Benefits of ANN Esemble Averaging:
    a. The sreulting committee is less complex than the single machine that achieves the same level of performance (Pearlmutter and Rosenfeld (1990)).
    b. The resulting committee can be trained more easily on smaller input sets (Haykin (1999)).
    c. The resulting committee typically has improved performance over any single network (Hashem (1997)).
    d. The risk of over-fitting is lessened, since there are fewer parameters (weights) that need to be set (Haykin (1999)).

Boosting

Overview

1. Definition: Boosting is a machine learning meta-algorithm for reducing bias in supervised learning (Wiki (Boosting)). Boosting techniques were developed in response to the question (Kearns (1988)): Can a set of weak learners create a strong learner (Schapire (1990))?
2. Weak Learner vs. String Learner: A weak learner is defined to be a classifier that is only slightly correlated with the true classification (i.e., it can label examples better tha random guessing). In contrast, a strong learner is arbitrarily well-correlated with true classification.
3. Fast Hypotheses Boosters: Algorithms that achieve boosting quickly simply become known as "boosting algorithms". The ARC technique (Freund and Schapire (1997) – adaptive re-sampling and combination), as a general technique, quickly became more or less synonymous with boosting (Breiman (1998)).

Philosophy behind Boosting Algorithms

1. Principle: While boosting is not algorithmically constrained, most boosting algorithms consist of iteratively learning weak classifiers with respect to a distribution, and eventually combining them to make a strong classifier. The weights of addition are related to the weak learners' accuracy.
2. Sample Re-weighting: For purposes of re-weighting, the examples that are mis-classified gain weight, and those that are correctly classified loose weight (some boosting algorithms reduce the weights of repeatedly misclassified samples – e.g., boost by majority and BrownBoost). Thus future weak learners focus more on the examples that previous weak learners misclassified.

3. The Original Boosting Algorithms: The original ones, proposed by Schapire (1990) (a recursive majority gate formulation algorithm) and Freund (boos by majority – Llew, Baxter, Bartlett, and Frean (2000)) were not adaptive, and could not take full advantage of the weak learners.
4. Probably Approximately Correct Boosters: Only algorithms that are provable correct on the probably approximately correct framework are called boosting algorithms. Other algorithms that are similar in spirit to boosting are called "leveraging algorithms", although they are sometimes incorrectly referred to as boosting algorithms (Llew, Baxter, Bartlerr, and Frean (2000)).

Popular Boosting Algorithms and Drawbacks

1. Variations among Boosting Algorithms: The main variation among the many boosting algorithms is their method of weighting the training data points and the corresponding hypotheses.
2. Popular Boosters: The first one was AdaBoost; recent ones include LPBoost, TotalBoost, BrownBoost, MadaBoost, and LogitBoost. Many boosting algorithms fit into the AnyBoost framework (Llew, Baxter, Bartlett, and Frean (2000)), which demonstrates that boosting essentially performs a gradient descent slide in the function space using a convex cost function.
3. Application: Boosting algorithms are used in Computer Vision, where individual classifiers detecting contrast changes can be combined to identify facial features (OpenCV Cascade Classifier).
4. Boosting under Random Classification Noise: Long and Servedio (2010) suggest that many of the bossting algorithms listed above are probably flawed. They conclude that convex potential boosters cannot withstand random classification noise, thus rendering the applicability of such algorithms for noisy, real world data questionable.
5. Causes for the Boosting Flaw: Long and Servedio (2010) show that if any non-zero fraction of the training data is mislabeled, the boosting algorithm tries extremely hard to correctly classify the training examples, and fails to produce a model with accuracy

better than 0.5. This result does not apply to branching program based boosters, but does apply to AdaBoost, LogitBoost, and others.

Bootstrap Aggregating

Overview and Sample Generation

1. Definition: Bootstrap aggragating (bagging) is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression (Wiki (Bootstrap aggregating)). It also reduces variance and helps avoid over-fitting. Although it is usually applied to decision tree techniques, it can be used with any type of method. Bagging is a special case of BMA.
2. Bagging Bootstrap Samples: Given a training set D of size n, bagging generates m new training sets D_i each of size n' by sampling from D uniformly and with replacement. This is called a bootstrap sample.
3. Sampling with Replacement: By sampling with replacement, some observations may be repeated in D_i; for $n = n'$ and large n, D_i is expected to have $1 - \frac{1}{e}$ (i.e, ~ 63%) of the unique samples of D, the rest being duplicates (more generally, when drawing with replacement n' values from a set of n (different and equally likely), the expected number unique draws is $n\left(1 - e^{-\frac{-n'}{n}}\right)$ (Aslam, Popa, and Rivest (2003)).
4. Bagging Ensemble Combiners: The m models are fit using the above m bootstrap samples and combined by averaging the output (for regression) or voting (for classification).
5. Applications: Bagging leads to improvements for unstable procedures (Breiman (1996), Sahu, Runger, and Apley (2011)) which includes neural nets, classification and regression trees, and subset selection in linear regression.
6. Caveat with kNN: On the other hand, blind application of bagging can mildly degrade performance of stable methods such as kNN (Breiman (1996)).

Bagging with 1NN – Theoretical Treatment

1. 1NN Improvements with Bagging: It is well known that the error-rate of a 1NN classifier is at most twice that of the Bayes' classifier, but there are no guarantees that this classifier will be consistent. By careful choice of the re-samples, bagging can lead to substantial improvements on the performance of the 1NN classifier.
2. 1NN Enhancement Algorithm: By taking a large number of re-samples of the data of size n' each, the bagged nearest neighbor classifier will be consistent provided $n' \to \infty$ diverges, but $\frac{n'}{n} \to 0$ as the sample size $n \to \infty$.
3. Bagging 1NN Formulation: Under infinite simulation, the bagged nearest neighbor classifier may be treated as a weighted nearest neighbor classifier. Suppose that the feature space is d-dimensional. Denote by $C_{n,n'}^{bnn}$ the bagged nearest neighbor classifier based on a training set of size n, with resamples each of size n'. In the infinite sampling case, under certain regularity conditions on the class distributions, the excess risk (is this a proxy for the error rate?) has the following asymptotic expansion (Samworth (2012)): $R_R\left(C_{n,n'}^{bnn}\right) - R_R\left(C_{Bayes}\right) = \left[B_1 \frac{n'}{n} + B_2 \frac{1}{n'^{4/d}}\right]\{1 + O(1)\}$ for some constants B_1 and B_2. The optimal choice of n' that balances the two terms in the asymptotic expansion is given by $n' = Bn^{\frac{d}{d+4}}$ for some constant B.

Bias-Variance Dilemma

Overview and Motivation

1. Definition: Bias Variance Dilemma (or trade off) refers to the problem of simultaneously minimizing the bias (i.e., how accurate the model is across different training sets) and the variance of the model error (how sensitive is the model to small changes in the training set) (Bias-Variance Dilemma (Wiki)).
2. Applicability Suite: This trade off applies to all forms of supervised learning – classification, function fitting (Geman, Bienenstock, and Doursat (1992)), and structured output learning.
3. Motivation: Ideally one wants to choose a model that captures the irregularities in the training data, which at the same time generalizes well to unseen data.
4. High Bias Models: High-bias models are intuitively simple models, and imposed restrictions on the kinds of irregularities that can be learned (examples include linear classifiers). Problem is that they underfit, i.e., they do not learn the relationship between the predicted (i.e., target) variables and the features.
5. High Variance Models: These can learn many kinds of complex irregularities, which unfortunately includes the noise in the training data as well (i.e., overfitting).
6. Origin of the Tradeoff: A common model selection criterion is that decrease of bias with an increase in model complexity results in increase of variance. Other considerations such as error losses and complexity costs lead to alternate tradeoff criteria. The choice of model may also introduce biases that correspond to useful previous information, e.g., the output may need to be range bound.

Bias Variance Decomposition

1. Setup: We employ the terminology of Vijayakumar (2007). Let the data specified $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ be derived from the true model $y = f(x) + \epsilon$ where ϵ is a random error with $E(\epsilon) = 0$. Using this we train a model $g(x|D)$ to approximate f. We also set $g(x_i) = g_i$ and $f(x_i) = f_i$.
2. Mean Squared Error: $MSE = \frac{1}{N}\sum_{i=1}^{N}[y_i - g(x_i|D)]^2$. The quantity of interest is the expectation of the MSE across the different realizations of the data: $E[MSE] = \frac{1}{N}\sum_{i=1}^{N} E[y_i - g(x_i|D)]^2$
3. MSE Expansion: Re-writing $\frac{1}{N}\sum_{i=1}^{N}[y_i - g_i]^2$ as $\frac{1}{N}\sum_{i=1}^{N}[y_i - f_i + f_i - g_i]^2$, we recognize that $y_i - f_i$ is the noise term ϵ above, and $f_i - g_i$ is simply the proxying error. Thus, $E[y_i - g_i]^2 = E[y_i - f_i]^2 + E[f_i - g_i]^2 + 2E\{[y_i - f_i][f_i - g_i]\} = E[\epsilon^2] + E[f_i - g_i]^2 + 2\{E[y_i f_i] - E[f_i^2] - E[y_i g_i] + E[f_i g_i]\}$. Given that f is deterministic, $E[y_i f_i] - E[f_i^2] = 0$. Likewise, $E[y_i g_i] = E[g_i(f_i + \varepsilon)] = E[f_i g_i] + E[\varepsilon g_i] = E[f_i g_i]$ if we assume that $E[\varepsilon g_i] = 0$. Thus the last term also vanishes. Therefore $E[y_i - g_i]^2 = E[\epsilon^2] + E[f_i - g_i]^2$.
4. MSE Reduction: $E[f_i - g_i]^2 = E[f_i - E[g_i] + E[g_i] - g_i]^2 = E[f_i - E[g_i]]^2 + E[E[g_i] - g_i]^2 + 2\{E([f_i - E[g_i]][E[g_i] - g_i])\}$. Here $\{E([f_i - E[g_i]][E[g_i] - g_i])\} = E\{f_i E[g_i]\} - E\{E[g_i]E[g_i]\} - E\{f_i g_i\} + E\{g_i E[g_i]\} = f_i E[g_i] - \{E[g_i]\}^2 - f_i E[g_i] + \{E[g_i]\}^2 = 0$. Thus $E[f_i - g_i]^2 = E[f_i - E[g_i]]^2 + E[E[g_i] - g_i]^2$.
5. MSE Final Form: $E[y_i - g_i]^2 = E[\epsilon^2] + E[f_i - E[g_i]]^2 + E[E[g_i] - g_i]^2$. Here $E[\epsilon^2]$ is the irreducible sample error (assuming infinite sample), $E[f_i - E[g_i]]^2$ is the square of the bias term, and $E[E[g_i] - g_i]^2$ is the model variance term. Cast in another manner, $E[y_i - g_i]^2 = Mean\ Squared\ Error = Irreducible\ Error + Bias^2 + Model\ Variance$.

Bias Variance Optimization

1. Impact of Feature Addition: Feature selection/filtering and dimensionality reduction can decrease variance by simplifying the models. Adding features (predictors) tends to decrease bias at the expense of introducing extra variance.
2. Approaches for Bias Variance Tuning: Learning algorithms typically have some tunable parameters that control bias and variance. For example:
    a. Generalized linear models can be regularized to increase their bias.
    b. In neural nets, deeper models with more layers will have stronger variance, this regularization (as in GLM) is applied.
    c. In k-nearest neighbor models, a high value of k leads to low variance.
    d. In Instance based learning, regularization can be achieved by varying the mixture of prototypes and examplars (Gagliardi (2011)).
    e. In decision trees, the depth of the tree determines the variance. Decision trees are commonly pruned to control the variance (James, Witten, Hastie, and Tibshirani (2013)).
3. Mixture Models and Ensemble Learning: These provide another way to address the bias-variance tradeoff (Ting, Vijayakumar, and Schaal (2011), Fortmann-Roe (2012)). For example, as seen earlier, boosting combines many "weak" (high bias) models in an ensemble that has greater variance than the individual models, while bagging combines strong learners in a way that reduces their variance.

Computational Learning Theory

1. Purpose: Computational learning theiry is a mathematical field related to the analysis of machine learning algorithms. It seeks to address issues of:
    a. Learnability/Learn Feasibility/Learn Complexity
    b. Learning Speed/Performance
    c. Learn Formulation Criterion
    d. Learn Basis/Representation Choice
    e. Probabilistic Learning Framework and Viability
2. Learn Complexity: In addition to performance bounds computational learning theory studies the time complexity and flexibility of learning. In computational learning theory, a computation is considered feasible if it can be performed in polynomial time (Computational Learning Theory (Wiki)). There are two kinds of time complexity results:
    a. Positive Results => Showing that a certain class of functions is learnable in polynomial time.
    b. Negative Results => Showing that a certain class of cuntions cannot be learned in polynomial time.
3. Non Polynomial Time Results: Negative results often rely on commonly believed, but as yet unproven assumptions such as:
    a. Computational Complexity - $P \neq NP$
    b. Cryptographic – One-way functions exist
4. Types of Approaches: There are several different approaches to computational learning theory. These differences are based on making assumptions about inference principles used to generalize from limited data. These include different conceptions of probability (frequentist, Bayesian) and different assumptions on the generation of samples.
5. Computational Learning Theory Approaches:
    a. Probably Approximately Correct Learning (PAC) proposed by Valiant (1984)
    b. VC Theory proposed by Vapnik (1971)

c. Bayesian Inference
d. Algorithmic learning theory, from the work of Gold (1967)
e. Online machine learning algorithms, from the work of Nick Littlestone

6. Practical Algorithms from Computational Learning Theory: PAC theory inspired boosting, VC theory led to support vector machines, and Bayesian inference led to belief networks (by Judea Pearl).

Empirical Risk Minimization (ERM)

Overview

1. Definition: ERM is a principle in statistical learning which defines a family of learning algorithms constructed using a specific risk principle, and is used to provide performance bounds on these algorithms.
2. Background/Setting: Consider the following situation typical of supervised learning problems. We have 2 spaces of objects X and Y and would like to learn a function (called a hypothesis) $h_i: X \to Y$ which outpts an object $y \in Y$ given $x \in X$. We have a set of training examples $(x_1, y_1), \dots, (x_N, y_N)$ where $x_i \in X$ is the input and $y_i \in X$ is the corresponding response, and we wish to get the response predictor hypothesis $h(x_i)$.
3. Training Set Generation: Formally we assume that there is a joint probability distribution $P(x, y)$ over X and Y, and the training set consists of N instances drawn i.i.d. from $P(x, y)$. Note that the assumptions of joint probability allows us to model uncertainties in the predictors (e.g., noise from data) because y is not a deterministic function of x, but rather a random variable with a conditional distribution $P(y|x)$ for a fixed x.
4. ERM Loss Function: We also assume that we are given a non-negative real-valued loss-function $L(\hat{y}, y)$ that quantifies how different the prediction $\hat{y}$ of the hypothesis is from the true outcome y. The risk associated with the hypothesis is then defined as the expectation of the loss function: $R(h) = E\{L[h(x), y]\} = \int L[h(x), y] \mathrm{d}\, P(x, y)$. A commonly used loss function for classification is the $0 - 1$ loss function $L(\hat{y}, y) = I(\hat{y} \neq y)$ where $I(\dots)$ is the indicator notation.
    a. Expectation Maximization Insight => The least squares loss function corresponds to maximizing the joint normal probability $P(x, y)$. If $P(x, y)$ is not Gaussian, and/or if there is a non-uniform (i.e., Bayesian, not frequentist)

prior, then the corresponding for the loss function may also be derived (it will not be Gaussian in general).

5. The Goal of the Computational Learning Algorithm: To find the hypothesis h^* among the fixed class of hypotheses H that minimizes the risk $R(h)$: $h^* = \underset{h \in H}{\arg min}\, R(h)$

The Loss Function and the Empirical Risk Minimization Principles

1. Choice of Loss Function: The choice of loss function is critical in computing the hypothesis – this choice significantly impacts the algorithm convergence rate. Further it is also important for the loss function to be convex (Rosasco, Vito, Caponnetto, Fiana, and Verri (2004)).
2. ERM Definition: In general $R(h)$ cannot be computed since $P(x, y)$ is unknown to the learning algorithm (this situation is referred to as agnostic learning). Empirical risk is the approximation that results from averaging the loss function evenly across the training set (this simply corresponds to a uniform $P(x, y)$): $R_{Emp}(h) = \frac{1}{N}\sum_{i=1}^{N} L[h(x_i), y_i]$ (See Empirical Risk Minimization (Wiki)).
3. Problem Statement: Empirical Risk Minimization Principle states that the learning algorithm should choose a hypothesis h^* that minimizes the empirical risk: $h^* = \underset{h \in H}{\arg min}\, R_{Emp}(h)$. Depending upon the nature of the learner, this base formulation may end up as a combinatorial optimization problem for which globally optimal solutions are infeasible, thereby forcing a resort to meta-heristic techniques.

ERM Complexity

1. ERM Complexity Analysis: ERM for a classification problem with a $0-1$ loss function is known to be NP-hard even for relatively simple class of functions such as linear classifiers (owing to the supra-exponential growth of the combinatorial

optimization search space – Feldman, Guruswamy, Raghavendra, and Wu (2010)). However, when the minimial empirical risk is zero AND when the data is linearly separable, it can be solved efficiently.

2. ERM Complexity Handling Approaches: In practice, machine learning algorithms cope with NP-hard situations as above either by using a convex approximation to $0-1$ loss function (e.g., hinge loss for SVM – this alters the combinatorial search space onto a convex real search space) that is easier to work with/optimize, or by posing simplifying assumptions on $P(x,y)$ (thereby not being agnostic anymore).
    a. Loss Functions for Classification => The $0-1$ indicator loss function can be implemented via the Heaviside step function. To convert it to a convex function, a hing loss function is often used: $L[h(x),y] = [-yh(x)]_+$

Statistical Learning Theory

1. Motivation: Statistical Learning Theory provides the framework for machine learning by drawing from the fields of statistics and functional analysis (Statistical Learning Theory (Wiki), Mohri, Rostamizadeh, and Talwalkar (2012)), with a wide variety of applications (e.g., Sidhu and Caffo (2014)).
2. Statistical Regularization: Overfitting is symptotic of unstable solution, i.e., small training perturbations can cause large shifts in the learned functions. It can be shown that if the stability for the solution can be guaranteed, generalization and consistency are guaranteed as well (Vapnik, Chervonenkis (1971), Mukherjee, Niyogi, Poggio, and Rifkin (2006)). Regularization can address overfitting and provide problem stability.
3. Regularization by Hypothesis Restriction: Regularization can also occur by restricting the hypothesis space. A common example is the restriction of H to linear functions – thereby reducing the problem to that of linear regression. H may also be reduced to polynomials of degree p, exponentials, or bounded functions on L_1. Restrictions of the hypothesis space avoids over-fitting because the form of the possible functions is

limited, and therefore may avoid choosing the function that gives the empirical risk to be arbitrarily close to zero.

4. Tikhonov Regularization: This consists of minimizing $\frac{1}{n}\sum_{i=1}^{n} L[h(x_i), y_i] + \gamma \|f\|_H{}^2$ where γ is a fixed positive parameter referred to as the regularization parameter. Tikhonov regularization ensures the existence, uniqueness, and stability of the solution (Poggio, Rosasco, Ciliberto, Frogner, Evangelopoulus (2012)).

Vapnik-Chervonenkis Theory

Introduction

1. Purpose: A form of the computational learning theory, VC theory aims to explain the learning process from a statistical point of view by applying the empirical processes independent of the probability distribution (VC Theory (Wiki)).
2. Components of the VC Theory: As detailed in Vapnik (1989, 2000), the VC theory covers 4 main parts:
    a. Theory of consistency of the learning process => What are the necessary and sufficient conditions for the consistency of a learning process based on empirical risk minimization?
    b. Non-asymptotic theory of the rate of convergence of a learning process => How fast is the rate of convergence of the learning process?
    c. Theory of control of the generalization ability of a learning process => How can one control the rate of convergence (i.e., the generalization ability) of the learning process?
    d. Theory of construction learning machines => How can one construct the algorithms that control the generalization ability?

Empirical Processes

1. Background: Let $X_1, \dots, X_n$ be the random elements defined on a measurable space (X, A). Define the empirical measure $\mathbb{P} = \frac{1}{n}\sum_{i=1}^{n} \delta(X_i)$ where δ stands for the Dirac notation. Further, using the notation of van der Vaart and Wellner (2000), denote $\mathbb{Q}f = \int f d\mathbb{Q}$ for any probability measure $\mathbb{Q}$.
2. The Empirical Measure Map: Let $\Im$ be a class of measurable functions $f: X \to \mathbb{R}$. The empirical measure above induces a map from $\Im$ to $\mathbb{R}$ given by $f \to \mathbb{P}_n f$.

Notice the similarity between the success counting measure $\mathbb{P}_n$ and the $0-1$ empirical loss function.

3. Empirical Process Theory: Let $\|\mathbb{Q}\|_{\mathfrak{J}} = sup\{|\mathbb{Q}f|: f \epsilon \mathfrak{J}\}$ where sup is the supremum operator on the set, representing the least upper bound of $|\mathbb{Q}f|$. Empirical process theory aims at identifying the classes $\mathfrak{J}$ for which the following statements hold. In both cases we assume that the underlying distribution of the data $\mathbb{P}$ is unknown in practice.
    a. $\|\mathbb{P}_n - \mathbb{P}\|_{\mathfrak{J}} \to 0$, i.e., this is the uniform law of large numbers
    b. $\mathbb{G}_n = \sqrt{n}(\mathbb{P}_n - \mathbb{P}) \to \mathbb{G} \in l^{\infty}(\mathfrak{J})$, i.e., this is the generalized uniform central theorem limit.
4. Glivenko-Cantelli and $\mathbb{P}$-Donsker Classes: If the law of large numbers can be explicitly established across all elements of $\mathfrak{J}$, the class $\mathfrak{J}$ is called Glivenko-Cantelli. If the validity of the central limit theorem (under the assumption $\underset{f \in \mathfrak{J}}{sup} |f(x) - \mathbb{P}f| < \infty \; \forall \; x$) can be established, then the class $\mathfrak{J}$ is called $\mathbb{P}$-Donsker. Obviously a $\mathbb{P}$-Donsker class is Glivenko-Cantelli in probability by the application of the Slutsky's theorem.
5. Determination of $\mathfrak{J}$: The main challenge here is to determine $\mathfrak{J}$ that satisfies the LLN and the CLT criteria bove (under regularity conditions) for all $f \in \mathfrak{J}$. Intuitively, $\mathfrak{J}$ cannot be too large, and the geometry of $\mathfrak{J}$ will play an important role.
6. Size of $\mathfrak{J}$: One way of measuring how big $\mathfrak{J}$ is is by using covering numbers. The covering number $N(\varepsilon, \mathfrak{J}, \|.\|)$ is the minimum number of balls $\{g : \|g - f\| < \varepsilon\}$ needed to cover $\mathfrak{J}$ fully (obviously assuming there is an underlying norm on $\mathfrak{J}$). The entropy number is the logarithm of the covering number.
7. Sufficiency conditions for $\mathfrak{J}$ using Glivenko-Cantelli: A class $\mathfrak{J}$ is Glivenko-Cantelli if it is $\mathbb{P}$-measurable inside of an envelope $\mathbb{F}$ such that $\mathbb{P}^*\mathbb{F} < \infty$ and satisfies $\underset{\mathbb{Q}}{sup} \left[\varepsilon \|\mathbb{F}\|_{\mathbb{Q}}, \mathfrak{J}, L_1(\mathbb{Q})\right] < \infty$ for every $\varepsilon > 0$.
8. Sufficiency Conditions for $\mathfrak{J}$ being $\mathbb{P}$-Donsker: This criterion is a version of the Dudley's theorem. If $\mathfrak{J}$ belongs to the class of functions such that

$\int_0^\infty \sup_{\mathbb{Q}} \sqrt{logN\left[\varepsilon\sqrt{\int|\mathbb{F}|^2 d\mathbb{P}}, \Im, L_2(\mathbb{Q})\right]} d\varepsilon < \infty$, then $\Im$ is $\mathbb{P}$-Donsker for every probability measure $\mathbb{P}$ such that $\mathbb{P}^*\mathbb{F}^2 < \infty$.

Bounding the Empirical Loss Function

1. Symmetrization: The majority of treatments on how to bound empirical processes rely on symmetrization, application of the maximal concentration inequalities, and chaining. Symmetrization is usually the first step in these proffs, and since it is used in many machine learning proofs on bounding empirical loss functions (including the proof of VC inequality) we treat it in some detail.
2. The Empirical Process and its Symmetrized Counterpart: Consider the empirical process $(\mathbb{P}_n - \mathbb{P})f = \frac{1}{n}\sum_{i=1}^n [f(X_i) - \mathbb{P}f]$. Here we establish the connection between the empirical processes above and its symmetrized equivalent $\mathbb{P}_n^0 f = \frac{1}{n}\sum_{i=1}^n \varepsilon_i f(X_i)$ where ε_i is an independent random variable that can be ± 1 with a probability 0.5 each. This symmetrized process, therefore, is a Rademacher process, conditional on the data X_i. Therefore this process is a sub-Gaussian process by the Hoeffding's inequality.
3. The Symmetrization Inequality: The Symmetrization bounds the LLN criterion of the empirical process theory. Forevery non-decreasing convex $\Phi: \mathbb{R} \to \mathbb{R}$ and a class of measurable functions $\Im$, $E\Phi(\|\mathbb{P}_n - \mathbb{P}\|_\Im) \leq E\Phi(2\|\mathbb{P}_n^0\|_\Im)$.
4. Conceptual idea behind the proof by Symmetrization: The proof of the symmetrization lemma lies on introducing independent copies of the original variables X_i (sometimes referred to as the ghost sample) and replacing the innder expectation of $\Phi(\|\mathbb{P}_n - \mathbb{P}\|_\Im)$ with these copies. After an application of Jensen's inequality, different signs could be introduced (hence the name symmetrization) without changing the expectation.
5. Proof Steps:

a. Introduction of the Ghost Sample => The empirical measure $\mathbb{P}f$ is replaced by $Ef(Y_i)$ – thereby providing the motivation for introducing the ghost sample variables $Y_1, \ldots, Y_n$ as independent copies of $X_1, \ldots, X_n$. For fixed values of $X_1, \ldots, X_n$ one has $\|\mathbb{P}_n - \mathbb{P}\|_{\mathfrak{I}} = \sup_{f \in \mathfrak{I}} \frac{1}{n} |\sum_{i=1}^n [f(X_i) - Ef(Y_i)]| \leq E_Y \sup_{f \in \mathfrak{I}} \frac{1}{n} |\sum_{i=1}^n [f(X_i) - f(Y_i)]|$.

b. Apply Jensen's Inequality => Apply the convex operator Φ to both sides: $\Phi(\|\mathbb{P}_n - \mathbb{P}\|_{\mathfrak{I}}) \leq E_Y \Phi\left\{\frac{1}{n} \|\sum_{i=1}^n [f(X_i) - f(Y_i)]\|_{\mathfrak{I}}\right\}$. We are able to switch the locations of E_Y and Φ, because Φ is convex, and we have applied Jensen's inequality to convert this to an inequality.

c. Take expectation with respect to X: $E\Phi(\|\mathbb{P}_n - \mathbb{P}\|_{\mathfrak{I}}) \leq E_X E_Y \Phi\left\{\frac{1}{n} \|\sum_{i=1}^n [f(X_i) - f(Y_i)]\|_{\mathfrak{I}}\right\}$. Given that only X is the variate on the RHS, the expectation simply reduces to E_X on RHS (and E on LHS).

d. Sign Perturbation Step => Note that adding aminus sign ahead of $[f(X_i) - f(Y_i)]$ does not alter the RHS, because it is a symmetric function of and . Thus the RHS remains the same under sign perturbation: $E_X E_Y \Phi\left\{\frac{1}{n} \|\sum_{i=1}^n e_i [f(X_i) - f(Y_i)]\|_{\mathfrak{I}}\right\}$ for a random Rademacher sequence e_i given as $(e_1, e_2, \ldots, e_n) \in \{-1, +1\}$. Therefore $E\Phi(\|\mathbb{P}_n - \mathbb{P}\|_{\mathfrak{I}}) \leq E_\varepsilon E\Phi\left\{\frac{1}{n} \|\sum_{i=1}^n \varepsilon_i [f(X_i) - f(Y_i)]\|_{\mathfrak{I}}\right\}$.

e. Triangle Inequality and Convexity of Φ => $E\Phi(\|\mathbb{P}_n - \mathbb{P}\|_{\mathfrak{I}}) \leq \frac{1}{2} E_\varepsilon E\Phi\left\{2 \left\|\frac{1}{n} \sum_{i=1}^n \varepsilon_i f(X_i)\right\|_{\mathfrak{I}}\right\} + \frac{1}{2} E_\varepsilon E\Phi\left\{2 \left\|\frac{1}{n} \sum_{i=1}^n \varepsilon_i f(Y_i)\right\|_{\mathfrak{I}}\right\}$. Since the 2 right hand terms are equal, we get $E\Phi(\|\mathbb{P}_n - \mathbb{P}\|_{\mathfrak{I}}) \leq E_\varepsilon E\Phi\left\{2 \left\|\frac{1}{n} \sum_{i=1}^n \varepsilon_i f(X_i)\right\|_{\mathfrak{I}}\right\} \leq E\Phi(2\|\mathbb{P}_n^0\|_{\mathfrak{I}})$.

6. Proving Empirical CLT's Using Symmetrization: Proof for the CLT's proceed analogously to the LNN. First use symmetrization to pass the process to $\mathbb{P}_n^0$, and then argue conditionally on the data using the fact that Rademacher processes are simple processes with nice properties.

VC Index

1. Motivation and Setup: Consider a collection $\mathbb{C}$ of subsets of the sample space χ. The collection of sets $\mathbb{C}$ is said to pick out a certain subset of finite set $S = \{x_1, \ldots, x_n\} \in \chi$ if $S = S \cap C$ from $C \in \mathbb{C}$. $\mathbb{C}$ is said to shatter S if it picks out each of its 2^n subsets. The VC-index $V(\mathbb{C})$ of $\mathbb{C}$ is the smallest number n for which NO SET of size n is shattered by $\mathbb{C}$ (in fact, the VC-index is similar the $VC_{Dimension} + 1$ for an appropriately chosen classifier set).
2. Bounds on the VC Class Subset Number: Sauer's lemma then states that the number $\Delta_n(\mathbb{C}, x_1, \ldots, x_n)$ of subsets picked out by a VC class $\mathbb{C}$ satisfies $\max_{x_1, \ldots, x_n} \Delta_n(\mathbb{C}, x_1, \ldots, x_n) \le \sum_{j=0}^{V(\mathbb{C})-1} \binom{n}{j} \le \left(\frac{ne}{V(\mathbb{C})-1}\right)^{V(\mathbb{C})-1}$. This is polynomial in the number of subsets (i.e.,$n^{V(\mathbb{C})-1}$) rather than exponential. Intuitively, this means that finite VC index implies that $\mathbb{C}$ has an apparent simplistic structure.
3. VC Sub-graph Bounds: A similar bound can be shown (for a different constant, same polynomial order) for the so-called VC subgraph classes. For a function $f: X \to \mathbb{R}$, the sub-graph is a subset of $\chi \times \mathbb{R}$ such that $\{(x, t): t < f(x)\}$. A collection of is called a VC sub-graph class if all the sub-graphs form a VC-class.
4. Covering Number for the $0-1$ Loss Function: Consider a set of indicator functions $I_C = \{1_C: C \in \mathbb{C}\}$ in $L_1(\mathbb{Q})$ for the discrete empirical measure $\mathbb{Q}$ (or any type of probability measure $\mathbb{Q}$ - discrete or not). It can be shown that for any ≥ 1 $N\left(\varepsilon, I_C, L_r(\mathbb{Q})\right) \le k\, V(\mathbb{C})(4e)^{V(\mathbb{C})} \left(\frac{1}{\varepsilon}\right)^{r(V(\mathbb{C})-1)}$.
5. Entropy Number for the Symmetric Convex Hypotheses Hull: Next consider the symmetric convex hull of a set $\Im_{CONVEX}$, which is a collection of functions of the form $\sum_{i=1}^{m} \alpha_i f_i$ with $\sum_{i=1}^{m} |\alpha_i| \le 1$. Then, if $N\left[\varepsilon \sqrt{\int |\mathbb{F}|^2 d\mathbb{P}}, \Im_{CONVEX}, L_2(\mathbb{Q})\right] < C\left(\frac{1}{\varepsilon}\right)^{\frac{2V}{V+2}}$.
6. Convergence of the Convex Hull: The most important consequence of the above is that the power of $\frac{1}{\varepsilon}$ - $\frac{2V}{V+2}$ is strictly less than 2, which is just enough so that entropy

integral is guaranteed to converge, therefore the class $\mathfrak{I}_{CONVEX}$ is going to be $\mathbb{P}$-Donsker.

7. VC Index of a Sub-graph Class: Any finite dimensional vector space $\mathfrak{I}$ of measurable functions $f: X \to \mathbb{R}$ is a VC-subgraph of index smaller than or equal to $dim(\mathfrak{I}) + 2$.
8. Generalizations of VC Sub-graph: There are generalizations of the notion of VC subgraph class, e.g., there is the notion of pseudo-dimension (Bousquet, Boucheron, and Lugosi (2004)).

VC Classifier Framework

1. Introduction: Let χ be a feature space and $\Upsilon = \{0, 1\}$. The function $f: X \to \Upsilon$ is called the classifier. Similar to before, we define the shattering coefficient (also referred to as growth coefficient) as $S(\mathfrak{I}, \mathrm{n}) = \underset{x_1, \dots, x_n}{max} |\{f(x_1), f(x_2), \dots, f(x_n)\}, f \in \mathfrak{I}|$
2. Shattering Coefficient Relation: In terms of the previous section the shattering coefficient is precisely $\underset{x_1, \dots, x_n}{max} \Delta_n(\mathbb{C}, x_1, \dots, x_n)$ with $\mathbb{C}$ being a collection of all the sets as laid out above. Further, using Sauer's lemma, it can be shown that $S(\mathfrak{I}, \mathrm{n})$ is going to be a polynomial in n provided that class $\mathfrak{I}$ has a finite VC dimension, or equivalently, the collection $\mathbb{C}$ has a finite VC index.
3. VC Indicator Empirical Risk: Let $D_n = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$ be the data set. As always, we assume that P_{XY} is unknown, thus we work on bounding the $0 - 1$ indicator empirical risk: $\hat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} I[f(X_n) \neq Y_n]$.
4. VC Inequality Bounds: For binary classification and $0 - 1$ loss function, we have the following generalization bounds:
    a. $\mathbb{P}\left(\underset{f \in \mathfrak{I}}{sup} \left|\hat{R}_n(f) - R(f)\right| > \varepsilon\right) \leq 8\, S(\mathfrak{I}, \mathrm{n}) e^{-\frac{n\varepsilon^2}{32}}$
    b. $E\left(\underset{f \in \mathfrak{I}}{sup} \left|\hat{R}_n(f) - R(f)\right|\right) \leq 2\sqrt{\frac{logS(\mathfrak{I}, \mathrm{n}) + log2}{n}}$
5. Implication of the VC Inequality Bounds: The impact of the VC inequality relation is that, as the sample size increases, provided $\mathfrak{I}$ has a finite VC dimension, the

empirical $0-1$ risk becomes a good proxy for the expected $0-1$ risk. Note that both RHS of the two inequalities will converge to 0 provided $S(\mathfrak{F}, \mathrm{n})$ grows polynomially in n.

6. Connection between the Classifier Framework and the Empirical Process Framework: With the empirical classifier framework, on deals with the modified empirical process $\left|\hat{R}_n(f) - R(f)\right|_{\mathfrak{F}}$, but the other components are the same. As before the proof of the first VC inequality relies on symmetrization, and then argue conditionally using the data with the concentration inequalities – in particular the Hoeffding's inequality (Bousquet, Boucheron, and Lugosi (2004)).

Probably Approximately Correct (PAC) Learning

Introduction

1. PAC Learning Motivation: PAC learning is a framework for mathematical analysis of machine learning (Probably Approximately Correct Learning (Wiki), Valiant (1984)). In this framework, the learner selects a generalization hypothesis from a certain class of possible functions. The goal is that, with high probability (the "probably" part) the selected hypothesis function will have a low generalization error (The "approximately" part). The learner must be able to learn the concept given any arbitrary approximation ratio, probability of success, or distribution of samples. The original model has been exoanded later to treat noise (mis-classified samples).
    a. Probably/Approximately => This model uses $P(x, y)$, which is the "probably" part, and within that chooses the low error, the "approximately" part. Since $P(x, y)$ is explicit, the treatment in regards to empirical risk is different.
2. The PAC Approach and Goals: An important innovation of the PAC framework is the introduction of the computational complexity theory concepts to machine learning (for e.g., VC deals primarily with "hypothesis complexity"). In particular, the learner is expected to find efficient functions (i.e., the time and space requirements are bounded to a polynomial on the sample size), and the learner itself must implement an efficient procedure (requiring an example count to be bounded to a polynomial of the concept size, modified by the approximation and the likelihood bounds).

PAC Definitions and Terminology

1. Sample Background: To elaborate on the PAC learnability, we use the 2 samples employed in Natarajan (1991) and Kearns and Vazirani (1994). The first is the problem of character recognition given an array of bits encoding a binary image. The

other is the problem of locating the interval that correctly classifies points inside as positive and outside as negative.

2. The PAC Instance Space: Let X be a set called the instance space or encoding of all the samples, and each instance has a length assigned. In the character recognition problem the instance space is $\{0,1\}^n$. In the interval problem, this would be $X = \mathbb{R}$ where $\mathbb{R}$ denotes the set of all real numbers.
3. PAC Concept and the Concept Class: A concept is the subset $c \in X$. One concept is the set of all patterns in bits $X = \{0,1\}^n$ that encode the picture of the letter P. An example concept from the second example is the set of all numbers between $\frac{\pi}{2}$ and 10. A concept class C is a set of concepts over X. For instance, this could be the set of all the subsets of array of bits that are skeletonized 4-connected (the width of the font being 1).
4. The PAC Procedure: Let $EX(c, D)$ be a procedure that draws an example x using a probability distribution D and gives the correct label $c(x) = \begin{cases} 1 & x \in C \\ 0 & Else \end{cases}$.
5. PAC Learnability and the Learning Algorithm: Say that there is an algorithm A that, given access to $EX(c, D)$ alongwith inputs ε and δ that, with a probability of at least $1 - \delta$, A outputs a hypothesis h that has error less than or equal to ε with examples drawn from X using the distribution D. If there is an algorithm for every concept $c \in C$, for every distribution D over X, and for all $0 < \varepsilon < \frac{1}{2}$ and $0 < \delta < \frac{1}{2}$, then C is PAC learnable (or distribution-free PAC learnable). Further, we can also say that A is a PAC learning algorithm for C.
6. Efficient PAC Algorithm: An algorithm runs in time t if it draws at most t samples and requires at most t time steps. A concept class is efficiently PAC learnable if it is PAC learnable by an algorithm that runs in time polynomial in $\frac{1}{\varepsilon}$, $\frac{1}{\delta}$, and the instance length.

Bayesian Analysis

Framework Symbology

1. General Framework Formulation:
   a. $P(A \mid B) = \frac{P(B \mid A).P(A)}{P(B)}$
2. $P(A \mid B)$: This refers to a population sample restricted exclusively to those consisting of ***B*** – in this population what is the chance of ***A*** occurring? $P(B \mid A)$ refers to the opposite.
3. Relation between $P(A \mid B)$ and $P(B \mid A)$: $P(A \mid B) \propto P(B \mid A)$, and $P(A \mid B) \propto P(A)$
   1. $P(A \mid B)P(B) = P(B \mid A)P(A) = P(A \cap B)$
   2. Since hypothesis H connects to evidence set E, arity/valence typically stays at 2.
   3. $P(A \mid B) > P(B \mid A)$ implies that A cannot cause B – but B can cause A?
4. Hierarchical Bayesian Network: Represented as $P(\theta, \varphi \mid x) \Rightarrow P(x \mid \theta)P(\theta \mid \varphi)P(\varphi)$, etc: There can be multiple such φ, where each φ stands for a specific parameter of parameter.

Framework Glossary

1. Derivative Entity: The entity whose dynamics are determined by the evolution of a stochastic variate, and whose specific facets/measures are observable.
2. Prior Distribution: $P(A)$ is referred to as the prior – it is really the unconditional distribution.
3. Posterior Distribution: $P(A \mid B)$ is referred to as the posterior – it is really the conditional distribution.

4. Marginal distributions: Both $P(A)$ and $P(B)$ are also referred to as the marginal distributions.
5. Likelihood function: $P(B \mid A)$ is referred to as the likelihood function.
6. Hyper-parameter: In Framework Symbology, point #3, φ (the parameter of parameters) is called a hyper-parameter.
7. Improper Prior: In the Framework Symbology $P(\theta \mid \varphi)$ is referred to as the improper prior.

Purpose of the SKU

1. Basic Framework and Software SKU
2. Generation of fixed + methodological principle-based Prior SKU
3. Generation of Likelihood + Posterior Distributions
4. Generation of the Conjugate Priors/Posteriors
5. Incorporation of the specific Schemes – Bayes Network, Model Validation etc:

Applicability

1. Evidentiary Belief Analysis: Purpose of Bayes' analysis both stochastic prior/posterior processes, as well as evidentiary belief analysis – not just purely stochastic prior/posteriors.
2. Usage quiet general: Simply stems from the realization that most modern systems rely on the concept "past determines future".
3. Historical Usage of Bayesians: Celestial mechanics observations used to be parameterized using Bayesian approaches, but that was discarded in favor of using deterministic parameters with experimental uncertainties.
4. Generalized Decision Rules: MLE, MAP etc: are referred to as decision rules – fundamentally use inference techniques for parameter calibration.

5. Likelihood Estimates: Remember that likelihood $\Lambda \equiv \Lambda\left(\vec{X} \mid \vec{\alpha}\right)$, where $\vec{X}$ refers to the set of observations, and $\vec{\alpha}$ refers to the set of models and/or parameters. Thus likelihood estimates are, in a sense, the analogue of calibration – a methodology for assigning the probabilities of transforming observations from models/parameters.
    - Likelihood estimates estimate the likelihood of parameters' assuming a certain value given the observation set, not the other way.
    - What constitutes a "parameter" is simple. Every numerical model input that goes into specifying/characterizing a distribution is a parameter (e.g., μ, σ^2 for normal distribution, and p/q for binomial distribution, etc)
    - Likelihood dependence on models => Likelihood formulation is the only place where physics of all the inherent dynamics sows up. This is the reason why all he model validation activities use likelihood ratios, Bayes' factors etc:
6. Likelihood Estimation vs. Eigenization: The Bayesian formulation $Posterior = Likelihood \times \Pr ior$ is analogous to the Eigenization formulation $Y = \lambda X$. In fact, if the posterior and the prior are in the same family (conjugates), then the analogy with eigenization is even closer.
    - In eigenization, transformation of the specified vector set occurs using a mixing operator that is a linear combination of the inputs; in Bayesian transformation, the mixing occurs through the conditioning operator using a convex combination of the hyper-parameter distributions.
7. Product Probability Distribution Convolutions: This is what posteriors are composed of – convolutions of likelihood and prior – to produce convolved posteriors. This is applicable whenever multi-distribution products are convolved to produce a compound convolution. If the input distributions are conjugate, it makes the compounding convolution all the more easier.
    - Distributions that arise as a result of common factor conditional independence make themselves very amenable to such convolutions. If they are conjugates, that aids their closed-form tractability as well.

- Multi-feature classifier convolutions => Previous observations maybe applied for conditionally independent Bayesian classifier feature convolutions, i.e., $Posterior = P(C \mid F_1)P(C \mid F_2)...P(C \mid F_i)...P(C \mid F_n)$

8. Posterior vs. Likelihood Simplified:
   - Posterior $P(\theta \mid X)$ => Probability that the parameters are θ given the evidence (or observations) X .
   - Likelihood $P(X \mid \theta)$ => Probability that the observation is X given that the parameter is θ
9. Bayesian Decomposition of Technical Signals: In general, the signal core drivers are limited (like systemic/idiosyncratic factors), but the product specific manifestations are more varied. Bayesian frameworks well suited for these.
10. Bayesian Learning: Bayesian learning occurs because of "enhancing" the posterior – enhancement happens via time evolution – through the set of data sampled over time.
11. Bayesian Techniques in NLP: Application of Bayesian techniques (particularly as practiced in Spam filtering etc:) to natural language processing challenges appears to be a good area. Use NLP in conjunction with Bayesian techniques to work out probabilistic knowledge models.
12. Frequentist Predictions and Bayesian Prediction: Bayesian techniques are slanted more towards inference, whereas frequentist techniques inference. Where predictions are employed in a Bayesian setting, they are used primarily as an out-of-sample framework quality control check, and prediction in itself is not the goal.

Analysis of Bayesian Systems

1. Conditional Probability vs. Parameter Uncertainty: As a concept, conditional probability is more general applied than targeted handling of parameter uncertainty (which, in itself, is a variant of the belief process/system). However, both use "trimming of the input universe" coming out of the observation sub-set.

- Example of conditional probability application => Bayesian spam filtering techniques, naïve Bayesian classification techniques.
- Examples of handling parameter uncertainty => Parameter prior distributions, Approximate Bayesian Computation, Empirical Bayes' Method, Maximum a-posteriori estimate etc:

2. Stochastic formulation of Bayesian systems: All the typical stochastic formulation/analysis should be applicable in the case of Bayesian systems as well?

3. Usage of Maximum Expectation: If $\tilde{V}_{Obs}$ is the observed measure, then the calibration of the parameter P_a for model M_a is done from the solution to $\max\{\Phi_a(M_a, V, P_a)\} \Rightarrow \tilde{V}_{Obs}$, where Φ_a is the distribution corresponding to M_a. Likewise, the calibration of the parameter P_b for model M_b is done from the solution to $\max\{\Phi_b(M_b, V, P_b)\} \Rightarrow \tilde{V}_{Obs}$. Now, whichever probability is bigger represents the better model proxy.

4. Sufficient Statistics in Bayes' Classifier: Differently banded sufficient statistics may be used to indicate correspondence with different classes – the correspondence represented by a stochastic matrix is stochastic, i.e., $POSTERIOR(\varsigma_k) = \sum_j TM(j,k) PRIOR(\zeta_j)$. Here

   - ς_k represents the posterior parameters for the sufficient statistics set corresponding to the classifier k
   - ζ_j represents the prior parameters for the sufficient statistics set corresponding to the classifier j
   - $TM(j,k)$ represents the Bayesian Translation Matrix from the Prior to the Posterior. Such intra-family closed transition forms for $TM(j,k)$ may be possible only for conjugates.

Advantage of Bayesian over other systems

1. Parameter Uncertainty: Bayesian likelihood estimation and Bayes' factor calculation occur over the full parameter space distribution, thereby automatically accounting for parameter uncertainties.
2. Classical Maximum Likelihood Estimate: Occurs simply for a single parameter, so there is no parameter distribution and/or uncertainty estimate.
3. Frequentist approach: Even more horrible, as it is simply a confidence interval estimate based on sigma cutoffs.

Bayesian Networks

1. DAG: Bayesian DAG representations should be format able as sequentially dependent stochastic variate trees, each with their independent marginal distributions – joints maybe described using a copula.
2. SKU DAG Search Quality: When using Bayesian network to determine the SKU's DAG ancestry/causality, always be careful of the "weak" graph links. Use something like Bayes' K factor decibans to filter out weak DAGs.

Hypothesis Testing

1. Types of Model Validation: It is important to distinguish between validation across different types of models, versus parameter validation ranges within a single model. Of course, Bayesian techniques can be used for both.
2. Over-fitting avoidance in Bayesian Hypothesis Testing: Automatically happens through the incorporation of the model/parameter uncertainties.
3. NULL Hypothesis: Refers to the single CLASSICAL hypothesis invoked as the sole explainer of observed phenomenon.
4. Penalizing Loss Functions: Bayesian analysis accommodates this too.

5. Hypothesis Testing:
    - Type 1 Error => This is also called FALSE POSITIVE. NULL Hypothesis is TRUE, but the experimental result rejects the NULL hypothesis.
    - Type 2 Error => This is also called FALSE NEGATIVE. NULL Hypothesis is FALSE, but the experimental result accepts the NULL hypothesis.
6. Specificity: Specificity is the fraction that truly negates the NULL Hypothesis.
    - TN => Number of TRUE Negatives
    - TP => Number of TRUE Positives
    - FN => Number of FALSE Negatives
    - FP => Number of FALSE Positives
    - $Specificity = \frac{TN}{TN + FP}$
7. Sensitivity: Sensitivity is the fraction that is truly positive of all those tested.
    - $Sensitivity = \frac{TP}{TP + FN}$
8. Positive/Negative Predictive Value:
    - Positive Predictive Value (PPV) => If the test result is positive, how well does that predict an actual presence? $PPV = \frac{TP}{TP + FP}$.
    - Negative Predictive Value (NPV) => If the test result is negative, how well does that predict an actual absence? $NPV = \frac{TN}{TN + FN}$.

Bayesian Updating

1. Successive Conjugate Bayesian Updates: Successively updating a conjugate prior using conjugate likelihood's results in progressively narrowing of the variance for the posterior. This is so despite the mean in itself getting shifted.
2. Successive Convolutions: Above observations need not apply for conjugate convolutions alone – they apply for any convolution. Specifically, if the starting prior

is $N_0\left(\mu_0,{\sigma_0}^2\right)$, and you apply successive likelihood's $N_i\left(\mu_i,{\sigma_i}^2\right), i=1..n$, the final variance is given by $\frac{1}{{\sigma_{FINAL}}^2}=\sum_{i=0}^{n}\frac{1}{{\sigma^2}_i}$, and the final mean by $\mu_{FINAL}=\frac{\sum_{i=0}^{n}\left\{\mu_i/{\sigma_i}^2\right\}}{\sum_{i=0}^{n}\left\{1/{\sigma_i}^2\right\}}$.

Maximum Entropy Techniques

1. Discrete Entropy Maximization Distribution: The maximum entropy prior on a discrete space, given that the probability is normalized to unity, is the prior distribution that assigns equal probability to each state.
2. Entropy Maximization Continuous Distribution: The maximum entropy prior given that the density is normalized with a given mean and a variance, is a normal distribution!
3. MAXENT to Update Priors: Maximization of the posterior information (or MINIMIZATION of the posterior entropy) given a prior is equivalent to minimizing the information of posterior relative to a given prior.

Priors

1. Uninformed Priors: These are TRICKY. Make sure ALL possible uninformed prior states are factored in before you do this. Input group invariants (e.g., rotational, transformation, or other group transformation invariants) can be used to determine uninformed priors.
   - Prior proposition Order Unimportant => All that matters when constructing the prior distribution is how many of the given parametric propositions that are testable and successful exist, not their order.
2. Bayesian Principle of Indifference: If multiple mutually exclusive and collectively exhaustive variables $x_1,x_2,...,x_n$ span the distribution variate space, then the

incremental cumulative distribution is the same across any one, i.e., $\frac{\Phi_1(x_1)}{\mathrm{N}_1} dx_1 = \frac{\Phi_2(x_2)}{\mathrm{N}_2} dx_2 = \ldots = \frac{\Phi_i(x_i)}{\mathrm{N}_i} dx_i = \ldots = \frac{\Phi_n(x_n)}{\mathrm{N}_n} dx_n$. Here $\Phi_i(x_i)$ is the distribution of the i[th] variate, and N_i is the corresponding normalizer.

3. MLE-based Prior Estimator: As a starting prior, it may be worth using frequentist MLE to kick start the initial prior estimate – to some extent this is the same as "uninformed prior" above.
4. Prior in Finance: Both prior as well hyperprior (hyperhyperprior etc) are useful in finance. For example the hazard rate can be parametrically by other parameters, which will be the hyper-parameters.
    - Financial Stochastic Volatility => Given that "volatility" is really a parameter, stochastic volatility is just a "prior" on the volatility, i.e., all the Bayesian techniques for inferring distributions of the volatility should be used.

Predictive Posteriors and Priors

1. Posteriors are always Bayesian: Posterior – by definition – is the distribution in the parameter space given the observation set. Therefore, they are applicable only to Bayesian belief systems, not frequentist systems.
2. Predictive Posterior: Say the posterior is $p_{POST}\left(\theta \mid \vec{X}, \alpha\right)$. The predictive posterior is defined as the predictive distribution for the next value of the observation x, given the earlier $\vec{X}$ and α, i.e., $p_{POST}\left(x \mid \vec{X}, \alpha\right) = \int_{\theta} p(x \mid \theta) p\left(\theta \mid \vec{X}, \alpha\right) d\theta$.
    - It is imply an expectation of $p(x \mid \theta)$ over the posterior distribution $p\left(\theta \mid \vec{X}, \alpha\right)$.
3. Predictive Prior: $p_{PRIOR}(x \mid \alpha) = \int_{\theta} p(x \mid \theta) p(\theta \mid \alpha) d\theta$. Almost identical to predictive posterior, except for the absence of $\vec{X}$ - owing to the fact that there have been no observations thus far.

Approximate Bayesian Computation

1. Philosophy: Full mathematical formulation of the mathematical basis of the model (and therefore the explicit likelihood) is hard – much easier simply to just formulate the stage-by-stage model algorithm.
    - Some of the reasons why estimation of likelihood is hard: multivariate inputs resulting in input variable curse of dimensionality, poorly formulated output to latent input variate map, etc:
2. Frequentist ABC (AFC): This is equally as valid as Bayesian – in this case, it becomes good old-fashioned Monte-Carlo. You can still algorithmize the Monte-Carlo evolution/computation, and run MLE-type parameter estimation and model evaluation/validation.
3. Applications of ABC/AFC:
    - Prior Parameter Estimation
    - Model Choice/Evaluation
    - Inferring starting point/pre-divergence times computation
    - All the above in one go
4. Components of ABC:
    - Non-zero sufficient statistics Convergence ε
    - Insufficient/inaccurate summary statistics
    - Challenges with Model Selection before ABC
    - Too much sensitivity to priors and the parameter ranges
    - Curse of Dimensionality
    - Model Ranking off of summary statistics => Bayes' factor of summary statistics goes out of synch with Bayes' factor of the Model.
5. Non-zero Tolerance ε : Non-zero tolerance to the outcome measure results in lack of precision as well as bias.
    - Solution => Theoretical/practical studies of the sensitivity of the posterior distribution to the tolerance. "Noisy ABC" methods are a solution.

6. Inaccurate/Insufficient Summary Statistics: Sufficient statistics effectively reduces the dimensionality of the observations by trimming the full UNIVERSE of observations to a set of parsimonious parameters. Insufficient/inaccurate summary statistics cause information loss due to inflated credible intervals.
    - Solution => Automatic selection and semi-automatic identification of sufficient summary statistics; better model validation checks.
    - Sufficient Statistics vs. Point Acceptance => If acceptance/rejection of the target distribution is based on a target point metric (say, target Euclidean distance to within a tolerance), then a single sequence of point parameter sampling is enough.
    - ABC using point value parameter input vs. ABC using the parameter prior => The simulation should run using the entire parameter prior suite as opposed to the point value parameter set, since only parameter prior enables wholesale acceptance/rejection based on sufficient statistics.
7. Insufficient or Misspecified Models: This happens when the models chosen for investigation are not representative, or lack predictive power.
    - Solution => Careful selection of models, and evaluation of their predictive power.
8. Priors and Parameter Ranges: This happens when the conclusions are too sensitive to the choice of the priors, thereby rendering the model choices meaningless.
    - Solution => Check the sensitivity of the Bayes' factor to the choice of prior. Use he theoretical results available regarding the choice of priors, as well as several varied alternative methods for model validation.
9. Curse of Dimensionality: This causes low parameter acceptance rates and/or results in over-fitting. This makes it hard to distinguish model error from insufficient exploration of the parameter space.
    - Solution => Use applicable and appropriate parsimonious methods for model fitting, as well as penalized information criteria (AIC, BIC etc;) to detect over-fitting. Investigate methods to speed up the process of parameter space exploration.

10. Model Ranking with Summary Statistics: The computation of Bayes' factor with summary statistics may be totally unrelated to the Bayes factors on the original data, which may render the results/conclusions meaningless.
    - Solution => Only use the summary statistics that fulfill the necessary and sufficient condition to produce a consistent Bayesian "Likelihood Model" choice.
11. Implementation: This is caused by low safe-guards/protection to the common assumptions in the simulation and inference processes.
    - Solution => Standard Software Methods, plus "model hedge ratios" and model predicted variances, as well as choice of a divergent set of models/model parameters/"secure" processes (e.g., model VaR, etc:).
12. Inverse ABC: Typically forward ABC (as in prior -> likelihood -> posterior) is used in conjunction with sufficient statistics approach to determine the appropriate starting prior for the analysis. How about "inverting" the forward ABC to compute the starting point?
    - Reverse ABC used along with adjoint algorithmic/automatic differentiation techniques might enable computation of the path-wise sensitivity, as well as simulation start/appropriate prior calibration.
13. Application to Financial Monte Carlo: In finance, ABC may be used for MC path-wise Greeks/measures, thereby probabilistically inferring the simulation state variate start.
    - Typical financial MC evolution would rules/algorithm would indicate that the explicit computation of the likelihood is hard/infeasible, therefore ABC is appropriate.
14. Target Check Tolerance Bias: Given that most probability distributions (including exponential distribution, in particular normal distributions) are asymmetric around an arbitrary variate, an estimation bias is introduced whenever the target match is done using a tolerance parameter.
15. Sufficient Statistics Tolerance Bias: Checks on target outcomes within tolerance introduces bias, so moment distributions on outcomes as proxy for the check on target outcomes – this is called sufficient statistics. However, bear in mind that finite

number of moments may still not be enough to completely capture the sufficient statistics, e.g., normal distribution needs two moments (μ and σ^2), other exponential distributions need more than two (but finite), and non-exponential may need literally infinite set of moments to be fully characterized.

16. ABC + MAP/Bayes' Estimator: The ABC algorithm using parametric priors is:
    - Generate a distribution of θ based on the prior distribution.
    - Generate the posterior distribution using the model mnemonics, and the prior.
    - Choose the prior distribution that best matches the measured posterior distribution – by default, this automatically corresponds to using MAP.
17. Re-use of the Posterior Samplings: For a given set of input-to-output samplings, different posteriors may be generated from different priors that are assembled by assigning appropriate population weights to the input-to-output samplings.

Measurement and Parametric Calibration

1. Measurement vs. Calibration: Unbiased outcome of a set of measurements for the same set space of inputs – if it exactly the same set of inputs, the measurement outcome will just be the average. Calibration, however, has inference built into it.
    - Cliché – measurement is "discovery", whereas calibration is "inference".
2. Sources of Output Stochasticity:
    - Measurement Stochasticity => Stochasticity introduced by random measurement error, despite deterministic inputs
        - Handled using the frequentist MLE framework
    - Stochasticity from Parameter Uncertainty => Deterministic input and stochastic output, where the output determinism is eliminated purely due to a diffuse belief system
        - Handled using Bayesian techniques such as MAP, etc:
    - Model stochasticity => Deterministic input result in stochastic output owing to model introduced stochasticity

    - Handled using stochastic calibration inside a regression analyzer
  - Stochastic Input => Stochastic output occurs purely due to the transformation of the input; further the input should be characterisable
  - Combination of any of the above still may result in output stochasticity. Thus, the corresponding analysis techniques are a combination of the above too.
3. Curve/Parameter Fitting with MLE/MAP: Curve/parameter can only be justified in a logical inference framework. The framework may use MLE (in a frequentist set-up), or MAP (along with Bayes' estimator inside of a cost function) in a Bayesian set-up.
4. Measurement Stochasticity vs. Model Stochasticity: Typical frequentist MLE cost-function based regression only deals with measurement stochasticity. However, distribution of the posterior may also result from model stochasticity. Therefore de-convolution techniques that separate the measurement stochasticity from model stochasticity are needed.

Regression Analysis

1. Machine Learning as Regression: Learning is construed as a calibration process, where the output of calibration (viz., the parameter) is really produced as a consequence of regression.
2. Causality of Response on Predictor: Often, the predictor and the response variables result from one or more common factor moves. If the common factors are orthogonalizable into precisely that one factor that solely drives predictor and response (and none else, including no idiosyncratic drivers), the dependence shows symptoms of causality, i.e., 100% (spontaneous or delayed) correlation between predictor and response.
    - However, causality still maintains stochasticity. In addition to the driving factor, the measurement errors also contribute to the stochasticity of causal observations set.

3. Regression Analysis Focus: It focuses on the conditional realization of the measured outputs given the observed inputs (rather than the joint realization of the inputs and the outputs), although these two statements are completely equivalent.
4. Frequentist Regression Axiom Set:
   - All the measurements for y_i correspond only to the input vector set $\{x_j\}$.
   - The only source of error is the instrumentation/measurement error, which is random and characterizable.
   - The measurement error bias is zero (error $\mu \Rightarrow 0$, but error $\sigma^2 \neq 0$).
5. Frequentist Regression Setup: Given the model $y = f(x) \Rightarrow y_i = \beta x_i + \varepsilon$ where $i = 1,...,n$ are the observations $\varepsilon_i \approx N(0,\sigma^2)$, i.e., no bias, the frequentist multi-observation likelihood is becomes $P(y_1 \mid x_1,\beta,\sigma^2)..P(y_n \mid x_n,\beta,\sigma^2) = \prod_{i=1}^{n} e^{\left\{-\frac{[y_i-\beta x_i]^2}{2\sigma^2}\right\}}$.
5. Frequentist Measurement Log-likelihood: $\Lambda = \frac{-\sum_{i=1}^{n}(y_i - \beta x_i)^2}{2\sigma^2}$. Thus, maximum likelihood estimate corresponds to minimization of the log-likelihood, which in turn works out to least squares minimization (LSM) of the log-likelihood.
6. Log likelihood Least Squares Minimization:

$$\frac{\partial \Lambda}{\partial \beta} = 0 \Rightarrow -\frac{1}{2\sigma^2}\frac{\partial}{\partial \beta}\sum_{i=1}^{n}(y_i - \beta x_i)^2 = 0 \Rightarrow \sum_{i=1}^{n} x_i y_i - \beta \sum_{i=1}^{n} x_i^{\,2} = 0 \Rightarrow \beta = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^{\,2}}.$$

7. Matrix re-formulation of Likelihood:

$$\prod_{i=1}^{n} P(y_i \mid x_i,\beta,\sigma^2) \Rightarrow P\left(\vec{Y} \mid \vec{X},\beta,\sigma^2\right) \propto \exp\left[-\frac{1}{2\sigma^2}\left(\vec{Y} - \beta\vec{X}\right)^T\left(\vec{Y} - \beta\vec{X}\right)\right]$$

8. Matrix re-formulation of the Least Squares Regressor: $\beta = \left[\vec{X}^T\vec{X}\right]^{-1}\vec{X}^T\vec{Y}$. This is also referred to as the Penrose-Moore pseudo-inverse methodology.
9. Multi-factor Frequentist EM Based Regression – Practice Steps:

- Step #1: Given a model (i.e., model + parameter), find the likelihood of the given outcome $P\left(y_i \mid \{x_j\}, \{\beta_j\}, \vec{\theta} \right)$, i.e., all the measurements corresponding to a specific set of inputs $\{x_j\}$ where
    - y_i is the observation i , given $i : 1 \to n$.
    - $\{x_j\}$ is the input vector j , given $j : 1 \to m$.
    - $\{\beta_j\}$ is the parameter co-efficient vector j , given $j : 1 \to m$.
    - $\vec{\theta}$ is the set of parameters characterizing the error term ε_i in the input distribution, i.e., $y_i = \sum_{j=1}^{m} x_j \beta_j + \varepsilon_i$.
- Step #2: Convolve the measurement set across the same input set, i.e., extract the joint likelihood $P\left(\vec{Y} \mid \vec{X}, \beta, \vec{\theta} \right) \Rightarrow \prod_{i=1}^{n} P\left(y_i \mid \{x_j\}, \{\beta_j\}, \vec{\theta} \right)$
- Step #3: Maximize the joint likelihood $P\left(\vec{Y} \mid \vec{X}, \beta, \vec{\theta} \right)$
    - Alternatively, maximize the log of joint likelihood.
    - For exponential conjugates, this typically works out to the maximization of some closed form, tractable cost function. Identify that cost function.
- Step #4: Minimize the Cost Function $\frac{\partial CostFunction}{\partial \vec{\beta}} = 0$ gives the corresponding $=1 \hat{\beta}$. Verify that $\left\| \frac{\partial^2 CostFunction}{\partial \vec{\beta}^2} \right\|_{\vec{\beta} = \hat{\beta}} > 0$, i.e., the cost function at $\hat{\beta}$ is a true minimum.
- Step #5: There will need to be m equations, one each for each β_j. Set these up to be solved using a linear/non-linear framework.

10. <u>Multi-node, Multi-factor Frequentist EM Based Regression – Practice Steps:</u>

- $y_i = \sum_{j=1}^{m} \beta_j x_{ij} + \varepsilon_i$, where $\varepsilon_i \approx \mathrm{N}(0, \mu^2)$, $i : 1 \to n$, and $j : 1 \to m$. The corresponding calibrated β_k is $\beta_k = \dfrac{\sum_{i=1}^{n} x_{ik} \left[y_i - \sum_{j=1, j \neq k}^{m} \beta_j x_{ij} \right]}{\sum_{i=1}^{n} x_{ik}^{\ 2}}$.

- Re-cast the above to form a frameable equation set as $\sum_{j=1}^{m} \beta_j \sum_{i=1}^{n} x_{ij} x_{ik} = \sum_{i=1}^{n} y_i x_{ik}$, or as $\sum_{j=1}^{m} \beta_j \alpha_{jk} = \gamma_k$, where $\alpha_{jk} = \sum_{i=1}^{n} x_{ij} x_{ik}$ and $\gamma_k = \sum_{i=1}^{n} y_i x_{ik}$.

- Matrix form of the above: $\vec{\gamma}_k = \vec{X}_K^{\ T} \vec{Y}$, and $\vec{\alpha}_{JK} = \vec{X}_K^{\ T} \vec{X}_J$. Thus, $\vec{\beta} \vec{X}_K^{\ T} \vec{X}_J = \vec{X}_K^{\ T} \vec{Y} \Rightarrow \vec{\beta} \vec{X}^T \vec{X} = \vec{X}^T \vec{Y}$. Thus, $\vec{\beta} = \left[\vec{X}^T \vec{X} \right]^{-1} \vec{X}^T \vec{Y}$.

Bayesian Regression Analysis

1. Non-Parametric Frequentist/Bayesian Regression: Answers the following question: Which among the input models (plus parameters) provides the most likelihood and/or MAP?
2. Parametric Frequentist/Bayesian Regression: Answers the following question: Assuming that the observation set satisfies MLE/MAP, what should the model parameters be?
3. Terminology fix – Regression Likelihood vs. Typical Bayesian Likelihood: While in typical frequentist/Bayesian treatments the likelihood is written as $L(x_i \mid \theta)$, in Bayesian likelihood it is written as $L(y_i \mid x_i, \theta)$ or $L(y_i \mid \theta)$. Thus, the y_i in regression likelihood is actually the x_i in regular; further, the corresponding x_i in regression is latent.

4. Problem with MLE-based Calibration: MLE dictates that only the model (due to the presence of instrumentation uncertainty and/or the inherent model transform stochasticity) causes stochasticity in the output. Bayesian techniques, however, allow for the stochasticity to arise out of imprecision resulting from model specification too.
5. Bayesian Basis for Observations Analysis: Treating the observations $\{y_i\}_{i=1}^{m}$ as the fixed axiomatic starting points, both the frequentist and the Bayesian techniques try to say something about the world that is limited to these observations – Bayesian techniques, however, also try to say something about the parameter set that result in these observations.
6. Parameter vs. Model Confusion: Wherever a Bayesian treatise says $P(y \mid \theta)$ it means $P(y \mid Model, \theta)$. Further, both $P(y \mid \theta)$ or $P(y \mid Model, \theta)$ simply refer to the probabilistic belief of the truth ness of the model (the prior is the model belief).
   - Probability of y given θ (represented as $P(y \mid \theta)$) is to be read as: ***What is the probability that*** y ***is valid in that particular world where*** θ ***is valid***? (See Figure 1). Specifically,

     $$\mathrm{P}(y \mid \theta) = \mathrm{P}(y = VALID \mid \theta = VALID) = \frac{\mathrm{P}(y = VALID, \theta = VALID)}{\mathrm{P}(y = INVALID, \theta = VALID) + \mathrm{P}(y = VALID, \theta = VALID)}$$

     , or $\mathrm{P}(y \mid \theta) = \dfrac{\mathrm{P}(y, \theta)}{\mathrm{P}(\neg y, \theta) + \mathrm{P}(y, \theta)}$.
   - Likewise, probability of θ given y (represented as $P(\theta \mid y)$) is to be read as: ***What is the probability that*** θ ***is valid in that particular world where*** y ***is valid***? (Again, see Figure 1). Specifically,

     $$\mathrm{P}(\theta \mid y) = \mathrm{P}(y = VALID \mid \theta = VALID) = \frac{\mathrm{P}(y = VALID, \theta = VALID)}{\mathrm{P}(y = VALID, \theta = INVALID) + \mathrm{P}(y = VALID, \theta = VALID)}$$

     , or $\mathrm{P}(\theta \mid y) = \dfrac{\mathrm{P}(y, \theta)}{\mathrm{P}(y, \neg\theta) + \mathrm{P}(y, \theta)}$.
7. Bayes' Theorem Derivation:
   - $$\mathrm{P}(\theta = VALID \mid y = VALID) = \frac{\mathrm{P}(y = VALID, \theta = VALID)}{\mathrm{P}(y = VALID, \theta = VALID) + \mathrm{P}(y = VALID, \theta = INVALID)}$$

- $\mathrm{P}(\theta = VALID \mid y = VALID) = \dfrac{\mathrm{P}(y = VALID, \theta = VALID)}{\mathrm{P}(y = VALID)} = \dfrac{\mathrm{P}(y = VALID, \theta = VALID)}{\mathrm{P}(y = VALID)} \dfrac{\mathrm{P}(\theta = VALID)}{\mathrm{P}(\theta = VALID)}$

- $\mathrm{P}(\theta = VALID \mid y = VALID) = \dfrac{\mathrm{P}(\theta = VALID)}{\mathrm{P}(y = VALID)} \dfrac{\mathrm{P}(y = VALID, \theta = VALID)}{\mathrm{P}(\theta = VALID)}$

- $\mathrm{P}(\theta = VALID \mid y = VALID) = \dfrac{\mathrm{P}(\theta = VALID)}{\mathrm{P}(y = VALID)} \dfrac{\mathrm{P}(y = VALID, \theta = VALID)}{\mathrm{P}(y = INVALID, \theta = VALID) + \mathrm{P}(y = VALID, \theta = VALID)}$

- $\mathrm{P}(\theta = VALID \mid y = VALID) = \dfrac{\mathrm{P}(\theta = VALID)}{\mathrm{P}(y = VALID)} \mathrm{P}(y = VALID \mid \theta = VALID)$, where we have used

  $\mathrm{P}(y = VALID \mid \theta = VALID) = \dfrac{\mathrm{P}(y = VALID, \theta = VALID)}{\mathrm{P}(y = INVALID, \theta = VALID) + \mathrm{P}(y = VALID, \theta = VALID)}$

8. Bayes' Theorem Derivation: Circling back the Definitions:
   - $\mathrm{P}(y = VALID \mid \theta = VALID) = \mathrm{P}(y \mid \theta)$
   - $\mathrm{P}(\theta = VALID \mid y = VALID) = \mathrm{P}(\theta \mid y)$
   - Thus, from above, $P(\theta \mid y) = P(y \mid \theta) \dfrac{P(\theta)}{P(y)}$
   - Finally, $Likelihood = \dfrac{P(y \mid \theta)}{P(y)} = \dfrac{\mathrm{P}(y, \theta)}{(\mathrm{P}(y, \neg\theta) + \mathrm{P}(y, \theta))(\mathrm{P}(\neg y, \theta) + \mathrm{P}(y, \theta))}$
9. Posterior Maximum vs. Maximum of $\mathrm{P}(y, \theta)$: We are really interested in maximizing $\mathrm{P}(y, \theta) = \mathrm{P}(y \mid \theta)\mathrm{P}(\theta) = \mathrm{P}(\theta \mid y)\mathrm{P}(y)$. Since $\mathrm{P}(y)$ is the same, no matter what θ is, the above, in practice, reduces to maximizing $\mathrm{P}(\theta \mid y) = \dfrac{\mathrm{P}(y \mid \theta)}{\mathrm{P}(y)} \mathrm{P}(\theta)$, i.e., it corresponds to maximizing the posterior.
10. Incorporating Parameter Belief Uncertainty: Assuming that the parameter belief process is modeled as $\beta_k \sim \mathrm{N}(\mu_k, {\sigma_k}^2)$

$$\beta_{k,MAP} = \frac{{\sigma_k}^2 \sum_{i=1}^{n} x_{ik} \left[y_i - \sum_{j=1, j \neq k}^{m} \beta_j x_{ij} \right] + \sigma^2 \mu_k}{{\sigma_k}^2 \sum_{i=1}^{n} {x_{ik}}^2 + \sigma^2 \mu_k} = \frac{{\sigma_k}^2 \beta_{k,MLE} \sum_{i=1}^{n} {x_{ik}}^2 + \sigma^2 \mu_k}{{\sigma_k}^2 \sum_{i=1}^{n} {x_{ik}}^2 + \sigma^2 \mu_k}.$$

11. Matrix Form of the above: $\beta_{MAP} = \left[\vec{\sigma_\beta}^{\,2} \vec{X}^{\,T} \vec{X} + \vec{\sigma}^{\,2} \vec{\mu}_\beta \right]^{-1} \left[\vec{\sigma_\beta}^{\,2} \vec{\beta}_{MLE} \vec{X}^{\,T} \vec{X} + \vec{\sigma}^{\,2} \vec{\mu}_\beta \right]$

12. Multi-factor Bayesian EM Based Regression – Practice Steps: The extra steps required to enhance MLE based regression for Bayesian analysis are listed here:
    - Choose a parameter prior
    - Convolve the likelihood with the parametric distribution
    - Work out the prior parameters that calibrate to the maximization of the log MAP
    - Re-express $\vec{\beta}_{MAP}$ in terms of $\vec{\beta}_{MLE}$

13. Inferring the Prior from the Observed Posterior: In place of parametrically specifying the prior, its moments may also be inferred from the observed posterior and the likelihood.

14. Alternate Parameter Calibration using Cost Functions: In addition to MAP extensions to MLE, calibration may also be performed by optimizing (minimizing typically) a cost function from the given posterior distribution $\Pi(\theta)$, i.e.,

    $E_\Pi\left[\hat{\theta}(x)\right] = \int C\left(\theta, \hat{\theta}(x)\right)\Pi(\theta)d\theta$. Here $C\left(\theta, \hat{\theta}(x)\right)$ is the Cost Function, $\Pi(\theta)$ is the posterior distribution, and $\hat{\theta}(x)$ is referred to as the Bayes' estimator.

    - The Bayes' estimator may be looked at as a pivot $\hat{\theta}(x)$ in the parameter space from which the departure cost is measured, i.e., $C\left(\theta, \hat{\theta}(x)\right) = f\left(\theta - \hat{\theta}(x)\right)$..
    - Practical meaning of the Bayes' estimator => It provides an alternate "central cost pivot" anchor (this pivot, in general, will not be a traditional central measure such as mean/median/mode), and depending upon the model/prior combination, the prior may simply end up being the mean/median/mode.
    - MAP viewed in the Cost Function/Bayes' Estimator Framework => MAP is essentially a zero-order cost function, so there is not "estimator" corresponding to it – really!

15. Typical Cost Function/Bayes' Estimator Combination:

- Mean Squared Error (MSE) Minimization as the Cost Function => Minimization of MSE $E\left[\left\{\theta-\hat{\theta}(x)\right\}^2\right]$ results in $\hat{\theta}(x) \Rightarrow E[\theta]$, the mean.
- MSE as Cost Function, $x \sim \mathrm{N}(0,\sigma^2)$, and prior $\theta \sim \mathrm{N}(\mu,\tau^2)$ => In this case, the posterior is also normal, and the Bayes' estimator is given as $\hat{\theta}(x)=\frac{\sigma^2}{\sigma^2+\tau^2}\mu+\frac{\sigma^2}{\sigma^2+\tau^2}x$.
- MSE as Cost Function, x is Poisson i.i.d, and prior $\theta \sim Gamma(a,b)$ => In this case, the posterior is also Gamma, and the Bayes' estimator is given as $\hat{\theta}(x)=\frac{nx+a}{n+\frac{1}{b}}$.
- MSE as Cost Function, x is Uniform i.i.d ($x \sim U(0,\theta)$), and prior $\theta \sim Pareto(\theta_0,a)$ => In this case, the posterior is also Pareto, and the Bayes' estimator is given as $\hat{\theta}(x)=\frac{(n+a)\max[\theta_0,x_1,\ldots,x_n]}{a+n-1}$.
- Quintile Cost Function => $C\left(\theta,\hat{\theta}(x)\right)=a\left|\theta-\hat{\theta}(x)\right|$ for $\theta-\hat{\theta}(x)\geq 0$, and $C\left(\theta,\hat{\theta}(x)\right)=b\left|\theta-\hat{\theta}(x)\right|$ for $\theta-\hat{\theta}(x)<0$. In this case, $\hat{\theta}(x)=\frac{a}{a+b}$.

Extensions to Regression Analysis

1. Penalizing Smootheners: Penalizing smootheners are the consequence of Bayes' estimation applied on the Quadratic Penalties with Gaussian Priors (also referred to with maxim "The Penalty is the Prior").
2. Non-Gaussian Priors: In this case, the smoothing estimation process is called the Generalized Linear Model.
3. Inference Schemes:
    - Help probabilistically infer what happened in the past.

- Set up/identify a probabilistic causal framework.
- Inference schemes are less dependent on the precision of the models/physics than prediction schemes (this may be a CONTROVERSIAL STATEMENT).

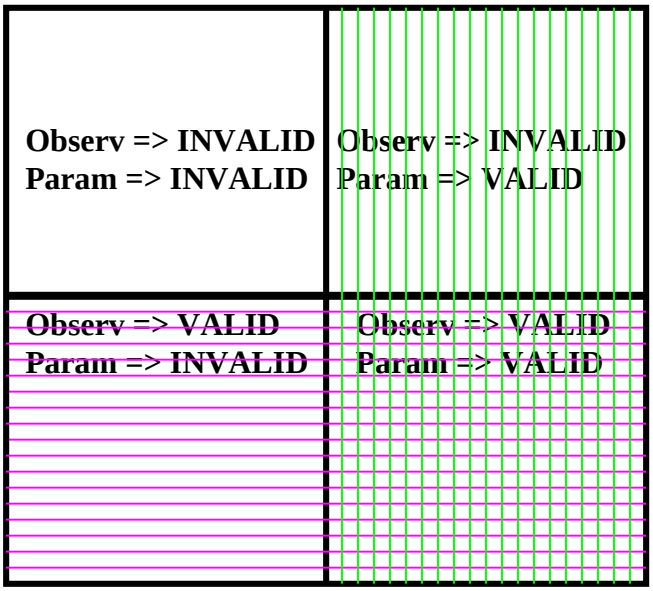
4. Prediction Schemes:
    - Help predict probabilistic future outcomes.
    - Significantly dependent upon the precision of the models.
    - Automatically built for outcome variance handling/hedging.
5. Inference/Prediction vs. "Smoothness": "Smoothness" schemes are simply just good mathematical/cognition citizen schemes. Clearly this is different from inference/prediction schemes as laid out.
6. Inference Based Curve Fitting: Here, the target function is treated as a realization of a stochastic process. Hence the model hypothesis estimation, credible interval estimate, etc: are all automatically available as part of the Bayesian Inference Framework.
7. Bayesian Optimizing Inference Schemes: Are all optimizing inference schemes castable as Bayesian? Clearly, given that regression schemes are part of a calibration framework, they can be reduced to Bayesian formulation, with specific priors having been extracted for smoothing/optimizing regressors. How about other inference schemes?

Spline Analysis of Bayesian Systems

1. Prior Spline to Posterior Spline: Given that any function may be represented by a suitable choice of splines, the prior and posterior may also be represented by them, and might potentially simply some of the analysis/formulation.
    - However, since probabilities can extend in the variate ranges $[-\infty,+\infty]$, perhaps specialized basis representations should be chosen; e.g., from $[-\infty,a]$ for the left extrapolator spline, and $[b,+\infty]$ for the right extrapolator spline. $[a,b]$ should be the range of workability.
    - Prior to posterior spline formulation =>

- Choose segment i.
- Prior Spline => $\lfloor R_{jk} \rfloor_i$.
- Posterior Spline => $\lfloor O_{jk} \rfloor_i$.
- Stating that $Prior_i \times \Lambda_{ij} = Posterior_j$ where Λ_{ij} is the Likelihood tensor, we can see that $\Lambda_{ij} = [R_{kl}]_i \times \lfloor R_{pq} \rfloor_j$.
- More specifically, $\Lambda_i = \lfloor R_{jk} \rfloor_i [I]_{jkpq} \times \lfloor O_{pq} \rfloor_i [I]_{pqjk}$, where ${}_i[I]$ is the identity matrix.

Figure #1
Observation/Parameter Quadrant



Optimizer

Constrained Optimization using Lagrangian

1. Base Set up: Use the Lagrangian objective function to optimize a multi-variate function $L(x,y)$ to incorporate the constraint $g(x,y)=c$ as: $\Lambda(x,y,z)=L(x,y)+\lambda[g(x,y)-c]$. Here λ is called the Lagrange multiplier, and use one Lagrange multiplier per constraint.
2. Optimize (Maximize/Minimize) the Lagrangian: Optimize (i.e., maximize/minimize) the Lagrangian with respect to x, y, and λ - thus $\frac{\partial\Lambda(x,y,z)}{\partial x}=0$, $\frac{\partial\Lambda(x,y,z)}{\partial y}=0$, and $\frac{\partial\Lambda(x,y,z)}{\partial\lambda}=0$. Notice that $\frac{\partial\Lambda(x,y,z)}{\partial\lambda}=0$ automatically implies the validity of the constraint $g(x,y)=c$, thereby accommodating it in a natural way.
   - Further, $\frac{\partial^2\Lambda(x,y,z)}{\partial\lambda^2}=0$ always, since $\Lambda(x,y,z)$ is linear in λ. Further, since the constraint is true by the optimizer construction, there should be no explicit dependence on λ.
3. Comparison with unconstrained optimization:
   - Unconstrained Optimization results in $\frac{\partial L(x,y)}{\partial x}=0$ and $\frac{\partial L(x,y)}{\partial y}=0$.
   - Constrained Optimization results in $\frac{\partial\Lambda(x,y,z)}{\partial x}=\frac{\partial L(x,y)}{\partial x}+\lambda\frac{\partial g(x,y)}{\partial x}$ and $\frac{\partial\Lambda(x,y,z)}{\partial y}=\frac{\partial L(x,y)}{\partial y}+\lambda\frac{\partial g(x,y)}{\partial y}$. $\lambda\Rightarrow 0$ automatically reduces the constrained case to the unconstrained.
   - Advantage of the Lagrange Multiplier Incorporation into Optimization => This ends up converting the constrained formulation space over on to the unconstrained, thereby providing with as many equations as the number of optimization unknowns, and one equation each per every constraint.

- Drawback of the Lagrange Multiplier Optimization Incorporation => While it lets you passively move to the unknowns' space from the constrained formulation space, this may end up impacting the application of certain boundary conditions, such as financial boundary, natural boundary conditions, Not-A-Knot boundary condition, etc:

4. Constraints of inequality $g(x,y) < c, g(x,y) > c$: Solutions to these constraints exist either inside the unconstrained set, in which case the unconstrained equations can be solved, or they don't, in which case convert the inequality to an equality and give it the Lagrange multiplier treatment.
5. Comparison with Convex Optimization: Convex optimization is predicated on the presence of at least one minimum/maximum in the zone of interest. One example is variance/covariance constrained optimization, where both the variance/covariance and the constraints are both quadratic. Eigenization (just another type of constrained variance/covariance optimization) is another.
6. Penalizing Optimizer as a Constrained Optimization Setup: Naively put, $\Lambda = Good - Bad$, where, from a calibration point of view, "Good" refers to closeness of fit, and "Bad" to the curvature/smoothness penalty. Thus, constrained optimization here corresponds to "maximize the Good, and minimize the Bad".
    - De-coupling "Good/Bad" from closeness of fit and curvature/smoothness penalty, respectively, can lead to alternate optimization framework formulations in finance, along with its insights.

Least Squares Optimizer

1. Least Squares Optimization Formulation:
    - $\mu_i(x_i) = y_i$
    - $\hat{\mu}_i(x_i) = \sum_{j=1}^{n} a_j B_{ij}(x_i)$

- $S = \sum_{i=1}^{m} \left\{ \mu_i(x_i) - \hat{\mu}_i(x_i) \right\}^2 = \sum_{i=1}^{m} \left\{ y_i - \sum_{j=1}^{n} a_j B_{ij}(x_i) \right\}^2 = \sum_{i=1}^{m} \left\{ {y_i}^2 - 2 y_i \sum_{j=1}^{n} a_j B_{ij}(x_i) + \left[\sum_{j=1}^{n} a_j B_{ij}(x_i) \right]^2 \right\}$

- $\frac{\partial S}{\partial a_j} = 2 \left(\sum_{i=1}^{m} a_j B_{ij}(x_i) - y_j \right) \left[\sum_{i=1}^{m} a_j B_{ij}(x_i) \right]$

2. <u>Least Squares Matrix Formulation</u>:

   - $Y = [y_1, ..., y_m]^T$; $B_i\left(\vec{x}\right) = \left[B_{i,1}\left(\vec{x}\right), ..., B_{i,m}\left(\vec{x}\right) \right]$; $a = [a_1, ..., a_m]^T$

   - Then, $\left[\frac{\partial \vec{S}}{\partial A} \right] = 2AB\left(\vec{x}\right) B^T\left(\vec{x}\right) A - 2YB\left(\vec{x}\right)$

   - As expected, $y_i = \sum_{j=1}^{n} a_j B_{ij}(x_i)$ is the optimized least squares tight fit.

Multi-variate Distribution

1. Mean/Variance Location Dependence: The mean is sensitive to both translation and rotation, unless the distribution is mean centered. The variance along a given fixed direction, however, is not sensitive to rotation of the basis.
   - This also implies that the maximal/minimal variances are invariant to representational basis changes. They are, however, sensitive to scaling, though, as PCA itself is.
2. Dimensional Independence vs. Dimensional Realization Independence: Dimensions are distinct (pressure, temperature etc.), but the realizations in those dimensions need not be. Therefore correlated unit vectors only apply to actual realizations (and they are NOT scale invariant).
3. Orthogonal Data Set in the Native Basis Representation: If a data set is orthogonal under a given basis, then $\langle x_i x_j \rangle = 0$ (See Figure 1).
4. Non-orthogonal Data Set in the Native Basis Representation: In this case, the native representation basis results in $\langle x_i x_j \rangle \neq 0$ (See Figure 2). Thus, an orthogonalization operation needs to be performed such that under the new basis $\langle x_i' x_j' \rangle = 0$ (See Figure 3).
5. Orthogonalization as Principal Components Extraction: As can be seen from figures 2 and 3, under the new schematic axes $\langle x_i' x_j' \rangle = 0$. Further these correspond to the principal components, i.e., orthogonal components where the variance is an extremum.
6. Orthogonalized Representation: Upshot of all these is that, if the representation basis is structured such that $\langle x_i x_j \rangle = 0$ for all $i \neq j$, then these representations automatically correspond to principal components as well.
7. Full Rank Matrix: This is simply an alternate term for multi-collinear matrix.

Parallels between Vector Calculus and Statistical Distribution Analysis

1. DOT PRODUCT => The notion of dot product is analogous to covariance operation in statistical analysis, i.e., DOT PRODUCT => $\vec{x}_i \vec{x}_j = 0$ if $\vec{x}_i \perp \vec{x}_j$, and covariance => $\langle x_i x_j \rangle = 0$ if x_i and x_j are orthogonal to each other.

2. Distance Metric => Vector Euclidean distance is $\sum\left[(x_1 - x_2)^2 + (y_1 - y_2)^2 + \ldots\right]$ is equivalent to the variance $\sum\left[(x_i - \mu_i)^2 + \ldots\right]$. Further, extremizing the Euclidean/Frobenius distance is analogous to variance minimization/maximization techniques.

Linear Systems Analysis and Transformation

Matrix Transforms

1. Co-ordinate Rotation and Translation: $\begin{bmatrix} x' \\ y' \end{bmatrix} = A\begin{bmatrix} x \\ y \end{bmatrix} + B$, where A is the rotating transformer, and B is the translator.
2. Scaling vs. Rotation: Say $A = \begin{bmatrix} a_{11} a_{21} \\ a_{21} a_{22} \end{bmatrix}$. If
   - $a_{11} = a_{22} \neq 1$, and $a_{12} = a_{21} = 0$, it becomes pure scaling.
   - $a_{11} \neq a_{22}$, and $a_{12} = a_{21} = 0$, it becomes differential elongation/compression.
   - $a_{12} \neq 0$ or $a_{21} \neq 0$, it becomes rotation.
3. Uses of Gaussian Elimination:
   - Linearization
   - Orthogonalization
   - Inversion
   - Diagonalization
   - Lower/Upper Triangle Decomposition (LU Decomposition)
   - Independent Component Extraction
   - Principal Component Analysis
4. Diagonal Identity Matrix Conception: Given a matrix A, what matrix M should it be transformed by to get a diagonal identity matrix, i.e., $M.A = I$? Answer is $M = A^{-1}$. Thus, diagonalization is also an inversion operation.
   - Diagonalization == Orthogonalization == Inversion == ICA? Diagonalization, of course, is unique only to a diagonal entry, whereas inversion corresponds to a specific choice of the diagonal entry.

Systems of Linear Equations

1. Importance of Diagonal Dominance in Gauss-Seidel: Is diagonal dominance important because the dominant diagonal's contribution to the RHS drives the given equation's value, and therefore the iterative accuracy?
2. Eigenization Square Matrix Inversion Conceptualization: Given a source matrix $F_{SOURCE,0} = \left\{f_{ij}\right\}_{i,j=0}^{n-1}$ and an initialized target inverse identity matrix $F_{INV,0} = \left\{I\right\}_{n \times n}$, achieve a suitable set of p transformations simultaneously on F_{SOURCE} and F_{INV} to eventually make $F_{SOURCE,p} = \left\{I\right\}_{n \times n}$, so that the corresponding $F_{SOURCE,p} = \left\{f_{INV,i,j}\right\}_{i,j=0}^{n-1}$ becomes the inverse.
3. Valid Inversion Rules: These are fairly straight forward application of the Gaussian elimination scheme:
   - Scale a single F_{SOURCE} row by a constant => scale the corresponding F_{INV} row by the same constant.
   - Add/subtract any pair of F_{SOURCE} rows from each other => add/subtract the same pair of F_{INV} rows from each other.
4. Matrix Inversion using Gaussian Elimination:
   - Scan across the diagonal entries.
   - Scan through the rows corresponding to each diagonal entry.
   - If a given row entry call value is zero, or its index corresponds to that of a diagonal, skip.
   - Calculate the $WorkColFactor$ as $WorkColFactor = \dfrac{DiagonalEntry}{CallValueEntry}$.
   - Scan all the cells in the column of the current cell.
   - Apply the $WorkColFactor$ product to each entry in the current working column of the source matrix.
   - Do the same as above to the inverse matrix.
   - Subtract the entries corresponding to the designated diagonal column from the working column to make the current entry zero.

- Do the same as above to the inverse matrix as well.
- After the completion of all such diagonal scans, row scans, and working column scans, re-scan the diagonal again.
- Scale down each entry of the source matrix by itself, so that the source matrix entries now constitute an identity ($\{I\}_{n\times n}$) matrix.
- Do the same as above to the inverse matrix as well.

5. Non-invertible Coefficient Matrix, but Solution exists: For unprocessed coefficient matrices, certain conditions (such as zero diagonal entries) may cause the coefficient matrix to be technically non-invertible, but that does not automatically mean that the system is unsolvable – a simply re-casting of the basic linear system set may be all that is required.
6. Linear Basis Re-arrangement: Sometimes, a singular coefficient matrix (with zero determinant, therefore non-invertible) may be re-arranged to create an invertible coefficient matrix. After inversion, it can be re-structured again to extract the inverse (which is just a coefficient Jacobian).
7. Rows/Columns as "Preferred Linear Basis Sequence" for Matrix Manipulation: Consider the solution to $AX = Z$, where X and Z are columns. In this case, the notion of constraint linear representation is maintained exclusively in rows. Therefore, all elimination/scaling basis operations need to be applied on that basis. In considering the solution to $AX = Z$, where X and Z are rows, columns now become the preferred linear basis sequence.
8. Regularization before Gauss Elimination: Before the Gauss Elimination can process, we need to diagonalize the matrix. Row swapping is a more robust way to diagonalize than row accumulation due to a couple of reasons:
    - Matrix Row Swap vs. Row Cumulate => In all cases, row swap can be transformed into row accumulation. When inverting, however, the row swapping AND accumulation should both be done on both the SOURCE and the TARGET matrices.
    - From a core linear operation set point-of-view, row accumulation is the inverse of the eventual of Gauss Elimination, so the danger is that the

diagonalization gains of row swap may be undone during the intermediate stages of Gauss Elimination.

- Even more important is that row swapping simply retains the original information, by just re-arranging the row set.

9. Row Swapping Caveats:
   - Always swap rows by retaining the directionality of the scan AND by retaining the scan initial node preceding the swap (to choose the pivot). One way to do this is by starting the scan at $row+1$ (or from $row=0$ if the edge has been reached), AND always keeping the scan sequence forward/backward.
   - Also ensure that the target swapped row is "valid", i.e., it's post-swapped diagonal entry should be non-zero. If this cannot be achieved through the scan, then that is an error condition.
10. Diagonalization/Inversion Algorithm: Work in terms of an intermediate transform variate Z produced by re-arranging the original coefficient matrix and Y such that the A in $AX = Z$ is now invertible. The following would be the steps:
    - First, re-arrange the equation system set to identify a suitable pair A and Z such that A is invertible, and Z is estimated from $AX = Z$.
    - The re-arranging linear operation set will produce A such that

      $BY = Z \Rightarrow AX = BY \Rightarrow X = A^{-1}BY$.
11. More General Inverse Transformation re-formulation: Given the coefficient matrix $\{a_{qj}\}_{j,q=0}^{n-1}$, the set of unknown variables $\{x_j\}_{j=0}^{n-1}$, and the RHS $\{y_q\}_{q=0}^{n-1}$, y_p, y_q, and the corresponding z_p and z_q are given as $\sum_{j=0}^{n-1} a_{qj}x_j = y_q; \sum_{j=0}^{n-1} a_{pj}x_j = y_p \Rightarrow \{z_j = y_j\}_{j=0}^{n-1}$.

    On transformation (i.e., adding row p to row q), you get

    $\sum_{j=0}^{n-1}(a_{pj} + a_{qj})x_j = y_p + y_q \Rightarrow \{z_j = y_j\}_{j=0, j\neq q}^{n-1}; z_q = y_p + y_q$. This clearly implies that

    $\frac{\partial z_q}{\partial y_p} = 1$, or $B_{qp} = B_{qp} + 1$.

Orthogonalization

1. 2-D Orthogonalization: In 2D, you need to fix one 1D orthogonal axis to be able to orthogonalize the other.
2. 2D Equation System: $x_1 = a_{11}s_1 + a_{12}s_2$ and $x_2 = a_{21}s_1 + a_{22}s_2$. This has 4 unknowns, so one solution for this is as follows: Fix $a_{11} = 1$ and $a_{12} = 0$. This results in 2 unknowns. Setting $\langle x_1^{\,2} \rangle = 1$, $\langle x_2^{\,2} \rangle = 1$, and $\langle x_1 x_2 \rangle = \rho$, you get $a_{11}^{\,2} + a_{12}^{\,2} = 1$, and $a_{21}^{\,2} + a_{22}^{\,2} = 1$ => $a_{21} = \rho$, and $a_{22} = \sqrt{1-\rho^2}$. Thus all unknowns are determined.
3. n-D Orthogonalization:
   - $$\begin{bmatrix} x_0 \\ . \\ . \\ . \\ x_n \end{bmatrix} = \begin{bmatrix} a_{0,0} \ldots a_{0,n-1} \\ \ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots \\ a_{n-1,0} \ldots a_{n-1,n-1} \end{bmatrix} \begin{bmatrix} s_0 \\ . \\ . \\ . \\ s_n \end{bmatrix}$$
   - Number of diagonal entries => n.
   - Number of non-diagonal entries => $n^2 - n$.
   - Net Number of equations => Number of diagonal entries + Number of non-diagonal entries => $n + \frac{n(n-1)}{2} = \frac{n(n+1)}{2}$.
4. n-D Unknowns Analysis:
   - Number of Unknowns => n^2.
   - Fix the first row, and the number of unknown becomes => $n^2 - n$.
   - Fixing the row takes off one equation, so the number of equations => $\frac{n(n+1)}{2} - 1$.
   - Number of equations = Number of Unknowns => $\frac{n(n+1)}{2} - 1 = n^2 - n \Rightarrow (n-1)(n-2) = 0$.

Gaussian Elimination

1. n-D Gaussian Elimination: What does it work? If you fix a row, you can rotate the other rows to eliminate dependence on an ordinate of the fixed row.
2. Row Fixation as a Basis Choice: Row elimination does not automatically make the matrix diagonal or orthogonalize it – all it does is to eliminate dependence on a stochastic variate.
3. Number of Elimination Rotations: The first row fixation results in $n-1$ rotations, the second row fixation results in $n-2$ rotations, and so on. Thus, the total number of rotating transformations is $\frac{(n-1)(n-2)}{2}$ (i.e., the same as the number of unknowns seen earlier). The result of these transformations is a lower/upper triangular matrix.
4. Final Reversing Sweep: An additional reverse sweep would eliminate similar column dependencies as well – resulting in another $\frac{(n-1)(n-2)}{2}$ rotation choices.
5. Number of Sweep Operations: Total result of all the rotations and their corresponding choices => $2.\frac{(n-1)(n-2)}{2} = (n-1)(n-2).$

Regression Analysis

Linear Regression

1. Regression vs. Calibration: Regression implies over-determination, with the extra measurements providing the additional degrees of freedom to help extract statistics. Calibration, however, implies right-sized number of inputs and outputs.
2. Calibrator Estimator Problem Set: Techniques presented here are useful for the following set of problems:
   - Regression Analysis
   - Density Estimation
   - Multi-variate (Polychotomous) Logistic Regression
   - Survival Analysis
   - Spectral Density Estimation
3. Regression Terminology: $\vec{Y} = \left[\vec{\beta}\right]\vec{X} + \vec{\varepsilon}$
   - Names for $\vec{Y}$ => Regressand, endogenous variables, response variables, measured variables, dependent variables (note that none of these terms explicitly invoke causality).
   - Names for $\vec{X}$ => Regressor, exogenous variable, predictor variable, covariates, explanatory variables, input variables, independent variables, design matrix.
   - Names for $\left[\vec{\beta}\right]$ => Parameter matrix, effects matrix, matrix of regression coefficients.
   - Names for $\vec{\varepsilon}$ => Error term, disturbance, noise.
4. Generalized Linear Regression (GLR): GLR optimizes the log likelihood function in place of least squares used in linear regression.
   - The fit equation can be re-cast in a very generalized form, thus the objective function can be of the form $\eta_i = g(\mu_i) = \sum_{j=1}^{n} a_j B_{ij}(x_i)$, not just $y_i = \sum_{j=1}^{n} a_j B_{ij}(x_i)$.
5. Some GLR re-casting examples:

- Poisson Regression => $g(\mu_i) = \log(\mu_i)$
- Logistic Regression => $g(\mu_i) = \log\left(\frac{\mu_i}{1-\mu_i}\right)$

6. Generalized Additive Models: Here the representative space functionals are a set/mixture of the additive state functionals.
   - Further, the smoothing formulation can be re-cast as a confidence band estimation/optimization problem with the 2 extraneously specified credible band limits C_1 and C_2, where $\sum_{i=1}^{n}\left[\hat{\mu}(x_i) - \sum_{j=1}^{m} B_j(x_i)\right]^2 \le C_1$, and $B^T D^T D B \le C_2$.
7. Regression Hat Matrix: The Regression Hat Matrix can be decomposed or represented in terms of the well-known Demmler-Reinsch basis functionals, and therefore derive from all its spectral properties and insights.
8. Non-Gaussian Priors: In this case, the smoothing estimation process is called the Generalized Linear Model.

Assumptions underlying Basic Linear Regression

1. Weak Exogeneity of Predictor Variables: This assumption indicates that $\vec{X}$ is fixed, thereby the sub-set being limited to measurement, not observation. This assumption may be lightened by assuming an errors-in-variables approach to linear regression formulation.
2. Linearity of the Predictor-Response Relationship: Given that the there is a choice in the basis functions, this is not too restrictive. However, over-fitting should be regularized out (e.g., by using ridge/lasso/Bayesian regression techniques).
3. Constant Response Variable Variance (Homoscedasticity): This means that different response variables have the same variance in their errors, regardless of the predictors. Sometimes, this impact may be reduced by using a transformation of the response variable (e.g., using a log normal link function).

4. Independence of Errors: It is also assumed that the errors of the response variables are uncorrelated with each other.
5. Lack of multi-collinearity: This assumption states that the design matrix $\vec{X}$ must be of full rank. Under special circumstances, multi-collinearity may be handled using an adjustment to the formulation (e.g., if $\vec{X}$ has effect sparsity). In general, more computationally intensive iterated algorithms for parameter estimation (such as those used in generalized linear models) do not suffer from this, as it is typical when handling categorical predictors to introduce a separate indicator variable predictor for each category (which inevitably introduces multi-collinearity).

Multi-variate Regression Analysis

1. Multi-variate Regression Analysis vs. Errors-in-Variables - Setup: Say $\{x_0,...,x_j,...,x_{m-1},y_0,...,y_i,...,x_{n-1}\}$ contains the measurement set, where $\{x_j\}_{j=0}^{m-1}$ is the set of predictor variables, and $\{y_i\}_{i=0}^{n-1}$ is the set of response variables. We propose $\hat{y}_i = \sum_{j=0}^{m-1} \beta_{i,j} x_j$ as the sequence set of regressors, and attempt to minimize $\langle S_i \rangle$, where $S_i = [\hat{y}_i - y_i]^2$.
2. Multi-variate Regression Analysis vs. Errors-in-Variables - Formulation:
   - $S_i = [\hat{y}_i - y_i]^2 = \left[\sum_{j=0}^{m-1} \beta_{i,j} x_j - y_i\right]^2 = \left[\beta_{i,k} x_k + \sum_{j=0, j\neq k}^{m-1} \beta_{i,j} x_j\right]$
   - $= {\beta_{i,k}}^2 {x_k}^2 + \left[\sum_{j=0, j\neq k}^{m-1} \beta_{i,j} x_j - y_i\right]^2 + 2\beta_{i,k} x_k \left[\sum_{j=0, j\neq k}^{m-1} \beta_{i,j} x_j - y_i\right]$
   - $\langle S_i \rangle = {\beta_{i,k}}^2 \langle {x_k}^2 \rangle + \left\langle \left[\sum_{j=0, j\neq k}^{m-1} \beta_{i,j} x_j - y_i\right]^2 \right\rangle + 2\beta_{i,k} \left[\sum_{j=0, j\neq k}^{m-1} \beta_{i,j} \langle x_k x_j \rangle - \langle x_k y_i \rangle\right]$
   - $\frac{\partial \langle S_i \rangle}{\partial x_k} = 2\beta_{i,k} \langle {x_k}^2 \rangle + 2\left[\sum_{j=0, j\neq k}^{m-1} \beta_{i,j} \langle x_k x_j \rangle - \langle x_k y_i \rangle\right] = 0 \Rightarrow \sum_{j=0}^{m-1} \beta_{i,j} \langle x_k x_j \rangle = \langle x_k y_i \rangle$

- Further, $\frac{\partial \langle S_i \rangle}{\partial x_k} = 2\beta_{i,k} \left\langle x_k{}^2 \right\rangle > 0$, thereby indicating that the optimization is a minimum.
- Compact Formulation => $\beta X X^T = X^T Y \Rightarrow \beta = \frac{X^T Y}{X X^T}$.

3. Multi-variate Regression Analysis – Explicit Measurement Error plus Errors-in-Variables:
   - We now set $\hat{y}_i = \sum_{j=0}^{m-1} \beta_{i,j} x_j + \varepsilon_i \Rightarrow \beta = \frac{X^T Y - X^T \varepsilon}{X X^T}$, where ε_i is the corresponding component measurement error. If $X^T \varepsilon = 0$, i.e., if ε is uncorrelated with X, we recover the compact formulation as before.

Multi-variate Predictor/Response Regression

1. Nomenclature:
   - $i = 0,...,n-1$ Response Variables $\{y_i\}_{i=0}^{n-1}$
   - $j = 0,...,m-1$ Predictor Variables $\{x_i\}_{i=0}^{m-1}$
   - $p = 0,...,q-1$ Observations
2. Linear Regression – Set up:
   - $\hat{y}_i = \sum_{j=0}^{m-1} \beta_{i,j} x_j$
   - $\varepsilon_i = y_i - \hat{y}_i = y_i - \sum_{j=0}^{m-1} \beta_{i,j} x_j$
   - $S_i = \varepsilon_i{}^2 = \left[y_i - \sum_{j=0}^{m-1} \beta_{i,j} x_j \right]^2$
3. k-Term Separation:
   - $S_I = \beta_{i,k}{}^2 \sum_{p=0}^{q-1} x_{k,p}{}^2$

- $S_{II} = 2\beta_{i,k}\sum_{p=0}^{q-1} x_{k,p}\left[\sum_{j=0, j\neq k}^{m-1}\beta_{i,j}x_{j,p} - y_{i,p}\right]$
- $S_{III} = \sum_{p=0}^{q-1}\left[\sum_{j=0, j\neq k}^{m-1}\beta_{i,j}x_{j,p} - y_{i,p}\right]$
- $S = S_I + S_{II} + S_{III}$

4. Minimizing Optimizer Set up:
   - $\frac{\partial S}{\partial \beta_{i,k}} = \frac{\partial S_I}{\partial \beta_{i,k}} + \frac{\partial S_{II}}{\partial \beta_{i,k}} + \frac{\partial S_{III}}{\partial \beta_{i,k}} = \sum_{p=0}^{q-1}\left[\left(\sum_{j=0}^{m-1}\beta_{i,j}x_{k,p}x_{j,p}\right) - x_{k,p}y_{i,p}\right] = 0$
   - Thus $\sum_{j=0}^{m-1}\beta_{i,j}\sum_{p=0}^{q-1}x_{k,p}x_{j,p} = \sum_{p=0}^{q-1}x_{k,p}y_{i,p}$
   - Set $\sum_{p=0}^{q-1}x_{j,p}x_{k,p} = \alpha_{j,k}$, the predictor-predictor covariance.
   - Set $\sum_{p=0}^{q-1}y_{i,p}x_{k,p} = \gamma_{i,k}$, the predictor-response covariance.
   - The above becomes $\sum_{j=0}^{m-1}\beta_{i,j}\alpha_{j,k} = \gamma_{i,k}$.

5. Second derivative – minimizing optimizer verification: $\frac{\partial^2 S}{\partial \beta_{i,k}^{\ 2}} = \sum_{p=0}^{q-1} x_{k,p}^{\ 2}$ => Given that the predictor variance is greater than zero, the $\beta_{i,k}$ that results from $\frac{\partial S}{\partial \beta_{i,k}} = 0$ corresponds to a minima.

6. Expanded Formulation: Considering just the regression co-efficient set $\{\beta_{i,j}\}_{j=0}^{m-1}$, we get

$$\begin{bmatrix} \alpha_{0,0} & \dots \alpha_{0,m-1} \\ . & \dots . \\ . & \dots . \\ . & \dots . \\ \alpha_{m-1,0} & \dots \alpha_{m-1,m-1} \end{bmatrix} \begin{bmatrix} \beta_{i,0} \\ . \\ . \\ . \\ \beta_{i,m-1} \end{bmatrix} = \begin{bmatrix} \gamma_{i,0} \\ . \\ . \\ . \\ \gamma_{i,m-1} \end{bmatrix}$$

7. The α Matrix: The α matrix does not depend on i, i.e., it is dependent only on j and k, therefore it can be pre-computed across the observation set $\{x_j\}_{j=0}^{m-1}$ and re-used across all β 's and γ 's.
   - Given that $\sum_{p=0}^{q-1} x_{j,p} x_{k,p} = \alpha_{j,k}$, the α matrix can be re-written as $X^T X$. This way the observations are directly accommodated, without the need to extract variances and co-variances.
8. The β Matrix: Recall that the β matrix is solved a single y set at a time, i.e., $\beta_{i,k}$ across all k is specified in a single linear system (corresponding to the γ matrix).
9. The γ Matrix: The γ matrix does not depend on j, i.e., it is dependent only on i and k. Thus, the pre-computation of γ is not as automatic as for the α matrix.
   - Given that $\sum_{p=0}^{q-1} y_{i,p} x_{k,p} = \gamma_{i,k}$, the γ matrix can be re-written as $X^T Y$. This way the observations are directly accommodated, again without the need to parametrically extract variances and co-variances.
10. Bringing it all together: Finally, $\sum_{j=0}^{m-1} \beta_{i,j} \alpha_{j,k} = \gamma_{i,k}$ may be re-written as $X^T X \beta = X^T Y$. Thus, $\beta = \left(X^T X\right)^{-1} X^T Y$.
11. Variance/Covariance Form in OLS Formulation: The coefficient of β in the least squares expansion term is twice the predictor-response covariance, i.e., $Coefficient of \beta = 2\langle xy \rangle$. The coefficient of β^2 in the least squares expansion term is the response-response covariance, i.e., $Coefficient of \beta^2 = 2\langle x_i x_j \rangle$. Therefore, $\beta_{OLS} \Rightarrow \frac{\langle xy \rangle}{\langle x_i x_j \rangle}$. Also remember that the usage of the squared term in OLS causes the formulation occur with the second moment terms at most – thereby resulting in a nice workable formulation with Gaussian distribution.

OLS on Basis Spline Representation

1. Base Formulation: We start with the point squared-error term:

   $S_p = \left[y_p - \sum_{i=0}^{n-1} \beta_i f_i(x_p) \right]^2$. After expanding and k-separating, this results in:

   - $S = \sum_{p=0}^{q-1} S_p = \sum_{p=0}^{q-1} {y_p}^2 + {\beta_k}^2 \sum_{p=0}^{q-1} {f_k}^2(x_p) + 2\beta_k \left\{ \sum_{\substack{i=0 \\ i \neq k}}^{n-1} \beta_i \sum_{p=0}^{q-1} f_i(x_p) f_k(x_p) - \sum_{p=0}^{q-1} f_k(x_p) y_p \right\} - 2 \sum_{\substack{i=0 \\ i \neq k}}^{n-1} \beta_i \sum_{p=0}^{q-1} f_i(x_p) y_p + \sum_{\substack{i=0 \\ i \neq k}}^{n-1} \sum_{\substack{j=0 \\ j \neq k}}^{n-1} \beta_i \beta_j \sum_{p=0}^{q-1} f_i(x_p) f_j(x_p)$

   - Turning the point summands into expectations:

     $\langle S \rangle = \langle y^2 \rangle + {\beta_k}^2 \langle {f_k}^2(x) \rangle + 2\beta_k \left\{ \sum_{\substack{i=0 \\ i \neq k}}^{n-1} \beta_i \langle f_i(x) f_k(x) \rangle - \langle f_k(x) y \rangle \right\} - 2 \sum_{\substack{i=0 \\ i \neq k}}^{q-1} \beta_i \langle f_i(x) y \rangle + \sum_{\substack{i=0 \\ i \neq k}}^{q-1} \sum_{\substack{j=0 \\ j \neq k}}^{q-1} \beta_i \beta_j \langle f_i(x) f_j(x) \rangle$

2. Basis Spline OLS Optimizer:

   - $\frac{\partial \langle S \rangle}{\partial \beta_k} = 2\beta_k \langle {f_k}^2(x) \rangle + 2 \left\{ \sum_{\substack{i=0 \\ i \neq k}}^{n-1} \beta_i \langle f_i(x) f_k(x) \rangle - \langle f_k(x) y \rangle \right\}$

   - Further $\frac{\partial^2 \langle S \rangle}{\partial {\beta_k}^2} = 2 \langle {f_k}^2(x) \rangle > 0,$ so there is always a minimum.

   - $\frac{\partial \langle S \rangle}{\partial \beta_k} = 0 \Rightarrow \sum_{i=0}^{n-1} \beta_i \langle f_i(x) f_k(x) \rangle - \langle f_k(x) y \rangle \Rightarrow \beta = \left[F^T F \right]^{-1} \left[F^T Y \right]$. This is the basis spline equivalent of linear regression off of linear basis function, i.e., $\beta = \left[X^T X \right]^{-1} \left[X^T Y \right]$.

OLS on Basis Spline Representation with Roughness Penalty

1. Base Formulation: We start with the penalized point squared-error term:

   $S_p = \left[y_p - \sum_{i=0}^{n-1} \beta_i f_i(x_p) \right]^2 + \lambda \int_{x_{Left}}^{x_{Right}} \left[\frac{\partial^m \hat{\mu}(x)}{\partial x^m} \right]^2 dx$. Note that the roughness penalizer term on the right is independent of x_p, so should have no dependence on the point expectations.

2. Penalizing Optimizer Formulation:

$$\int_{x_{Left}}^{x_{Right}}\left[\frac{\partial^m \hat{\mu}(x)}{\partial x^m}\right]^2 dx = {\beta_k}^2\int_{x_{Left}}^{x_{Right}}\left[\frac{\partial^m f_k}{\partial x^m}\right]^2 dx + 2\beta_k\sum_{\substack{i=0\\ i\neq k}}^{n-1}\beta_i\int_{x_{Left}}^{x_{Right}}\left[\frac{\partial^m f_i}{\partial x^m}\right]\left[\frac{\partial^m f_k}{\partial x^m}\right]dx + \sum_{\substack{i=0\\ i\neq k}}^{n-1}\sum_{\substack{j=0\\ j\neq k}}^{n-1}\beta_i\beta_j\int_{x_{Left}}^{x_{Right}}\left[\frac{\partial^m f_i}{\partial x^m}\right]\left[\frac{\partial^m f_j}{\partial x^m}\right]dx$$

3. Full Basis Spline Penalizing Optimizer:

$$\langle S\rangle = {\beta_k}^2\left[\left\langle {f_k}^2\right\rangle + \lambda\int_{x_{Left}}^{x_{Right}}\left[\frac{\partial^m f_k}{\partial x^m}\right]^2 dx\right] + 2\beta_k\left\{\sum_{\substack{i=0\\ i\neq k}}^{n-1}\left[\beta_i\langle f_i f_k\rangle + \beta_i\int_{x_{Left}}^{x_{Right}}\left(\frac{\partial^m f_i}{\partial x^m}\right)\left(\frac{\partial^m f_k}{\partial x^m}\right)dx\right] - \langle f_k y\rangle\right\} - 2\sum_{\substack{i=0\\ i\neq k}}^{n-1}\beta_i\langle f_i y\rangle + \sum_{\substack{i=0\\ i\neq k}}^{n-1}\sum_{\substack{j=0\\ j\neq k}}^{n-1}\beta_i\beta_j\left[\left\langle f_i f_j\right\rangle + \int_{x_{Left}}^{x_{Right}}\left[\frac{\partial^m f_i}{\partial x^m}\right]\right[$$

. Here we substitute f_i for $f_i(x)$, for ease of parsing.

4. Full Basis Spline Penalizing Optimizer – Second Derivative:

$\frac{\partial^2\langle S\rangle}{\partial {\beta_k}^2} = 2\left[\left\langle {f_k}^2\right\rangle + \lambda\int_{x_{Left}}^{x_{Right}}\left[\frac{\partial^m f_k}{\partial x^m}\right]^2 dx\right] > 0$ for $\lambda > 0$ and $x_{Left} > 0$. Thus, under these situations, a minimum is possible in S.

5. Full Basis Spline Penalizing Optimizer – Solution: $\beta = [Q]^{-1}[F^T Y]$, where

$$Q_{ij} = \left\langle f_i(x) f_j(x)\right\rangle + \lambda\int_{x_{Left}}^{x_{Right}}\left[\frac{\partial^m f_i(x)}{\partial x^m}\right]\left[\frac{\partial^m f_j(x)}{\partial x^m}\right]dx .$$

Extensions to Linear Regression Methodology

1. Generalized Linear Models (GLM): Here the response variables are bounded and/or discrete, thereby necessitating the usage of a link function g that relates the mean of the response variables to the predictors, i.e., $\left\langle \vec{Y}\right\rangle = g\left(\beta' \vec{X}\right)$ (e.g., transformation between $(-\infty,+\infty)$ - the range of the linear predictor and the range of the linear response variables).
2. Common Examples of Usage under a GLM Setting:
    - Poisson Regression for Count Data
    - Logistic/Probit Regression for Binary Data
    - Multinomial logistic/probit regression for ordered data
    - Ordered probit regression for ordinal data

3. Hierarchical Linear Models: Also referred to as multi-level regression, this technique organizes the data into a hierarchy of regressions. The final response is hierarchical on the intermediate/bottom-most predictors.
4. Error-in-variables: This allows the predictor variables to modeled under error. This may cause $\left[\vec{\beta}\right]$ to be biased, but that is typically in the form of attenuation (meaning that the effects tend to go to zero).

Linear Regression Estimator Extensions

1. Objective of the Estimation Methods:
   - Computational Simplicity
   - Preferably result in a closed form solution
   - Robust with regards to heavy tailed distributions
   - Produce desirable statistical properties such as consistency and asymptotic efficiency with minimal theoretical assumptions/restrictions.
2. Ordinary Least Squares (OLS): As seen earlier, $\beta = \left(X^T X\right)^{-1} X^T Y$. OLS estimator is unbiased and consistent if $\left\langle x_i \varepsilon_i \right\rangle = 0$, and if $\left\langle \varepsilon_i^2 \right\rangle$ is independent of i (the homoscedastic assumption).
3. Generalized Least Squares (GLS): GLS handles heteroscedasticity using a covariance matrix of the errors Ω. GLS minimizes a weighted sum of squared residuals $\{w_i\}$ of OLS, where $w_i \propto \frac{1}{\mathrm{var}iance(\varepsilon_i)}$. GLS can be viewed as applying a linear transform to the data so that the OLS assumptions are met. GLS analysis finally produces $\beta = \left(X^T \Omega^{-1} X\right)^{-1} X^T \Omega^{-1} Y$.
4. Percentage Least Squares: Here OLS is performed on the percentage transforms of the predictors. Thus, as opposed to typical OLS where the errors are additive, here the errors become multiplicative.

5. Iteratively Re-weighted Least Squares: This technique is used where both heteroscedasticity and correlated errors exist, but their structures are unknown. The following are the typical steps:
   - First, a GLS/OLS is performed on a provisional covariance structure, and the residuals are obtained from a fit.
   - Using these residuals, an improved covariance structure is estimated.
   - A subsequent GLS is then performed to estimate the weights until convergence.
   - 1-2 iterations of the above should be enough to obtain β estimates usually.
6. Instrumental Variables (IV): If the regressors and the error variables are correlated, then regression is performed on a set of instrumental variables $\{z_i\}$ that have the property $\langle z_i \varepsilon_i \rangle = 0$. In this case β is given as

   $\beta = \left(X^T Z \left(Z^T Z \right)^{-1} Z^T X \right)^{-1} X^T Z \left(Z^T Z \right)^{-1} Z^T Y$.
7. Total Least Squares: This approach handles the predictor variables and the response variables more evenly, thereby automatically accommodating errors-in-variables.
8. MLE: This method is used when the errors belong to the distribution family $f(\theta)$ that is not Gaussian – if $f(\theta)$ is Gaussian, the method becomes OLS.
   - Least Absolute Deviation => Here $f(\theta)$ corresponds to the Laplace error distribution. Least absolute deviation is more robust and less sensitive to errors than OLS, but is less efficient.
9. Ridge/Lasso Regression: These are penalizing estimators that deliberately introduce bias into the estimation procedure. They generally produce lower mean-squared error than OLS, as well as handle multi-collinearity at times.
10. Adaptive Estimation: The adaptive estimation is used to estimate the ε distribution. Assuming that ε is independent of $\vec{X}$, it estimates ε in the first step, and then uses MLE in the second step to estimate β.
11. Quantile Regression: Quantile regression techniques focus on the conditional quantiles of $\langle Y \mid X \rangle$ rather than the mean of $\langle Y \mid X \rangle$ as a linear function $\beta' \vec{X}$ of the quantiles.

12. Mixed Model Estimators: Mixed model estimators handle the case where the dependencies between the different y_i's have a known structure, so the mixed models regress across them.
13. Principal Component Regression (PCR): PCR is used when the number of predictors is huge, or when there are strong correlations across the predictors. However, there is no a priori cause for the principal components to serve as the target predictors. To address this, the partial least squares regression can be used.
14. Least angle Regression: This has been developed to handle high dimensional covariate vectors, or for situations where there are more covariates than observations.
15. Theil-Sen Estimator: IN this estimation, the regression line is chosen to have the slope that is the mean of the slopes of the lines through the sample point, thus it is less sensitive to outliers. It is also a simple and robust technique.

Bayesian Approach to Regression Analysis

11. Penalizing Smootheners: Penalizing smootheners are the consequence of Bayes' estimation applied on the Quadratic Penalties with Gaussian Priors (also referred to with maxim "The Penalty is the Prior").
12. Inference Based Curve Fitting: Here, the target function is treated as a realization of a stochastic process. Hence the model hypothesis estimation, credible interval estimate, etc: are all automatically available as part of the Bayesian Inference Framework.
13. Bayesian Optimizing Inference Schemes: Are all optimizing inference schemes castable as Bayesian? Clearly, given that regression schemes are part of a calibration framework, they can be reduced to Bayesian formulation, with specific priors having been extracted for smoothing/optimizing regressors. How about other inference schemes?

Component Analysis

Independent Component Analysis (ICA) - Specification

1. Definition: Give me the independent source signals that cause the sequence of observations that I observe. This is a plain orthogonalization challenge.

2. Nomenclature: $x_i : i = 0,...,n-1 \Rightarrow$ n random variables, and the vector $\left[\vec{X}\right]_k$ is the kth snapshot of all x_i's. After many observations $m >> n$, say that you have measured $\left\langle x_p x_q \right\rangle$ and $\left\langle x_p{}^2 \right\rangle$ for all $p,q \in \{0,...,n-1\}$.

3. Problem Statement: Assume, without loss of generality, that x_p and x_q are mean-centered. We look for the orthogonal factor set $\{s_j\}_{j=0}^{n-1}$ such that $\left\langle s_k s_l \right\rangle = 0$ for $k \neq l$, and $\left\langle s_k{}^2 \right\rangle \neq 0$ otherwise, for all $k,l \in \{0,...,n-1\}$.

4. Formulation: Let $x_p = \sum_{j=0}^{n-1} a_{p,j} s_j$, and $x_q = \sum_{j=0}^{n-1} a_{q,j} s_j$. Now

$$x_p x_q = \sum_{j=0}^{n-1} a_{p,j} a_{q,j} s_j{}^2 \Rightarrow \left\langle x_p x_q \right\rangle = \sum_{j=0}^{n-1} a_{p,j} a_{q,j} \left\langle s_j{}^2 \right\rangle, \text{ and}$$

$$x_p{}^2 = \sum_{j=0}^{n-1} a_{p,j}{}^2 s_j{}^2 \Rightarrow \left\langle x_p{}^2 \right\rangle = \sum_{j=0}^{n-1} a_{p,j}{}^2 \left\langle s_j{}^2 \right\rangle.$$

- If you work in the correlation space of x_p and x_q instead of the covariance space, we work on the corresponding covariates ε_p and ε_q, with $\left\langle \varepsilon_p{}^2 \right\rangle = 1$, and $\left\langle \varepsilon_p \varepsilon_q \right\rangle = \rho_{pq}$ (recall $\varepsilon_p \approx \frac{x_p}{\sqrt{\left\langle x_p{}^2 \right\rangle}}$). In this case, the corresponding orthogonal factor equations become $\sum_{j=0}^{n-1} a_{p,j} a_{q,j} \left\langle s_j{}^2 \right\rangle = \rho_{pq}$, and $\sum_{j=0}^{n-1} a_{p,j}{}^2 \left\langle s_j{}^2 \right\rangle = 1$.

5. Under-determined Uniqueness of Orthogonality: From $\langle \varepsilon_p \varepsilon_q \rangle = \rho_{pq}$, you have n^2 RHS values, and therefore that many equations. But there are $n^2 + n$ unknowns - n^2 from $a_{p,q}$, and n from $\langle s_j{}^2 \rangle$.

- Implicit $\langle s_j \rangle = 0$ assumption => Bear in mind that, in addition to $\langle x_p x_q \rangle$ and $\langle x_p{}^2 \rangle$, we also know $\langle x_p \rangle$ (which, by our axioms, is zero). Thus we have n equations for each $x_p = \sum_{j=0}^{n-1} a_{p,j} s_j$, and n unknowns $\langle s_j \rangle$, which we, arbitrarily for now, satisfy using $\langle s_j \rangle = 0$.
- One way of right determining the orthogonal equation unknowns => If we impose $\langle s_j{}^2 \rangle = 1$ for all j, then you have n^2 equations and n^2 unknowns, thereby it is right-determined.
- n-D Basis Coefficients Calibration => Remember $\langle x_p x_q \rangle = \sum_{j=0}^{n-1} a_{p,j} a_{q,j}$, and $\langle x_q x_p \rangle = \sum_{j=0}^{n-1} a_{q,j} a_{p,j}$. Thus, these 2 are essentially the same set – this equivalence reduces/takes off $\frac{(n-1)(n-2)}{2}$ equations from the system, thereby providing that much degrees of freedom for orthogonalization.
- However, given the non-linearity inherent in the product terms, solving this system is still not trivial.

6. Alternate Formulation: Instead of specifying x_p and x_q in terms of s_j, we do the reverse.

- $s_p = \sum_{j=0}^{n-1} a_{p,j} x_j$ and $s_q = \sum_{j=0}^{n-1} a_{q,j} x_j$.
- $s_p s_q = \left[\sum_{i=0}^{n-1} a_{p,i} x_i \right] \left[\sum_{j=0}^{n-1} a_{q,j} x_j \right] = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{p,i} a_{q,i} x_i x_j$.
- $\langle s_p s_q \rangle = 0$ for $p \neq q \Rightarrow \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{p,i} a_{q,i} \langle x_i x_j \rangle = 0$.

- $\langle s_p s_p \rangle = \langle s_p{}^2 \rangle$ for $p = q \Rightarrow \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{p,i} a_{p,i} \langle x_i x_j \rangle = \langle s_p{}^2 \rangle$.
- Thus, this is no easier – the complexity is only more than with the original formulation!

7. Non-linearity Reduction in an Optimizing Framework: The above ICA is non-linear in the coefficients, but if you find a way to introduce optimization into this, you may be able to reduce it to a sequence of linear equations (given that it is quadratic non-linear).

Independent Component Analysis (ICA) - Formulation

1. Nomenclature:
   - Signal Components: $0,\ldots,n-1$ => $\{S_i\}_{i=0}^{n-1}$.
   - Data Points: $0,\ldots,q-1$ => $\{x_{j,p}\}_{p=0}^{q-1}$.
   - The correspondingly observed x_j is $x_j = \sum_{i=0}^{n-1} a_{i,j} S_i$ where $a_{i,j}$ is the mixing matrix.
   - Goal: For the m observations given by $\{x_{j,p}\}_{p=0}^{q-1}$, estimate $a_{i,j}$ as well as S_i.
2. Estimation Technique #1 - Minimization of Mutual Information (MMI): MMI family of algorithms use metrics such as Kullback-Leibler divergence, maximum entropy etc: and minimize them.
3. Estimation Technique #2 - Maximization of non-Gaussianity: This family uses kurtosis/negentropy as the metric, and maximizes them using the calculated signal term S_i.
4. Common Estimation Steps across all Algorithms:
   - Mean-centering
   - Whitening (using Eigenization)
   - Dimensionality reduction

5. ICA Formulation Extensions:
   - Linear Noisy ICA: $x_j = \sum_{i=0}^{n-1} a_{i,j} S_i + n_j$, where $n_j \sim N(0, \rho_{ij})$.
   - Non-linear ICA: $\vec{x} = f(\vec{S} \mid \theta) + \vec{n}$, where $f(\vec{S} \mid \theta)$ is a non-linear mixer.
6. Binary ICA: In binary ICA, both the source and the monitors are both in the binary form $x_i = \vee_{j=0}^{n-1}(g_{ij} \wedge S_j)$, where $\wedge$ is the Boolean AND, $\vee$ is the Boolean OR, and $g_{ij} = 1$ indicates that the i[th] source is observed by the j[th] monitor. Noise is not explicitly modeled – it may be treated as another source.
7. Binary ICA Solution – Continuous Heuristics: The simple, heuristic solution is to assume that the predictor and the response variables are continuous, extract a real-valued g_{ij}, then round it to imply the individual g_{ij}. This has, of course, been shown to be inaccurate in many cases.
8. Binary ICA Solution - Dynamic Programming: Here the observation matrix X is broken down into sub-matrices, and targeted inferences are run individually on them. This has been shown to be accurate under moderate levels of noise.

Principal Component Analysis

1. Definition: The orthogonal transformation that transforms the data to a new co-ordinate system such that the greatest variance by any projection of the data comes to lie along the first co-ordinate (the first principal component), the second greatest variance along the second co-ordinate, and so on.
2. Alternate Phenomenon Nomenclature: The following are the names PCA is known as under the individual sub-fields:
   - Quality Control => Karhunen-Loeve Transform (KLT)
   - Linear Algebra => Proper Orthogonal Decomposition (POD), Singular Value Decomposition (SVD) of the design matrix $\vec{X}$, Eigen-value Decomposition (EVD) of $X^T X$, Factor Analysis
   - Psychometrics => Eckert-Young theorem

- Meteorological Science => Schmidt-Mirsky theorem, Empirical Orthogonal Functions (EOF)
- Noise and Vibration => Empirical Eigen-function Decomposition, Empirical Component Analysis, Quasi-harmonic Modes, Spectral Decomposition
- Structural Dynamics => Empirical Modal Analysis

3. PCA Computation Results Representation: The result of the PCA computation typical contain the following:
   - Component Scores/Factor Scores => The transformed variable values corresponding to a particular point.
   - Loadings => The weight by which each standardized original variable should be multiplied to get the component score.
4. Canonical Correlation Analysis (CCA): CCA identifies the transformed co-ordinate systems that optimally describe the cross-variance between two data-sets.
5. PCA Nomenclature and Set up:
   - Observation set: $p = 0,...,q-1$
   - Dimensions: $i = 0,...,n-1$
   - For a target point $\vec{P}$, the co-ordinates are $\left\{\vec{x}_{i,p}\right\}_{i=0}^{n-1}$. Thus, point $\vec{P}$ has a unit vector $\hat{P} = \frac{x_{0,p}\hat{x}_0 + x_{1,p}\hat{x}_1 + ... + x_{i,p}\hat{x}_i + ... + x_{n-1,p}\hat{x}_{n-1}}{\sqrt{{x_{0,p}}^2 + {x_{1,p}}^2 + ... + {x_{i,p}}^2 + ... + {x_{n-1,p}}^2}}$.
   - The Target PCA vector is $\hat{W} = w_0\hat{x}_0 + w_1\hat{x}_1 + ... + w_i\hat{x}_i + ... + w_{n-1}\hat{x}_{n-1}$ => $\left\{w_i\right\}_{i=0}^{n-1}$, under the constraint $\sum_{i=0}^{n-1} {w_i}^2 = 1$.
6. Projection of $\vec{P}$ on $\hat{W}$:
   - $\vec{P}.\hat{W} = \sum_{i=0}^{n-1} w_i x_{i,p}$
   - Note $\left\langle \vec{P}.\hat{W} \right\rangle = \sum_{i=0}^{n-1} w_i \left\langle x_{i,p} \right\rangle = 0$, since $\left\langle x_{i,p} \right\rangle = 0$. Thus, the projection is also mean-centered.
7. Square Error of the Projection of $\vec{P}$ on $\hat{W}$:

- $S_p = \left[\sum_{i=0}^{n-1} w_i x_{i,p}\right]^2 = \left[\sum_{i=0}^{n-1} w_i x_{i,p}\right]\left[\sum_{j=0}^{n-1} w_j x_{j,p}\right]$
- k-separating this projection yields $S_p = \left[w_k x_{k,p} + \sum_{i=0, i\neq k}^{n-1} w_i x_{i,p}\right]\left[w_k x_{k,p} + \sum_{j=0, j\neq k}^{n-1} w_j x_{j,p}\right]$
- Thus $S_p = {w_k}^2 {x_{k,p}}^2 + 2w_k x_{k,p} \sum_{i=0, i\neq k}^{n-1} w_i x_{i,p} + \left[\sum_{i=0, i\neq k}^{n-1} w_i x_{i,p}\right]^2$
- The True Variance (referred from now on as the unconstrained variance U) is the cumulated k-separated point errors across all the sample points:

  $$U = \sum_{p=0}^{q-1} S_p = {w_k}^2 \sum_{p=0}^{q-1} {x_{k,p}}^2 + 2w_k \sum_{i=0, i\neq k}^{n-1} w_i \sum_{p=0}^{q-1} x_{i,p} x_{k,p} + \left[\sum_{i=0, i\neq k}^{n-1} w_i \sum_{p=0}^{q-1} x_{i,p}\right]^2$$

- Re-cast in another way:

  $$U = \sum_{p=0}^{q-1} S_p = {w_k}^2 \sum_{p=0}^{q-1} {x_{k,p}}^2 + 2w_k \sum_{i=0, i\neq k}^{n-1} w_i \sum_{p=0}^{q-1} x_{i,p} x_{k,p} + \sum_{i=0, i\neq k}^{n-1} \sum_{j=0, j\neq k}^{n-1} w_i w_j \sum_{p=0}^{q-1} x_{i,p} x_{j,p}$$

8. Recast the Unconstrained Variance into Variance/Covariance Grouping: By applying the corresponding expectation as opposed to sample accumulation, we get
   - $U = {w_k}^2 \left\langle {x_k}^2 \right\rangle + 2w_k \sum_{i=0, i\neq k}^{n-1} w_i \left\langle x_i x_k \right\rangle + \sum_{i=0, i\neq k}^{n-1} \sum_{j=0, j\neq k}^{n-1} w_i w_j \left\langle x_i x_j \right\rangle$
9. Two methods to work out the details of PCA Solution:
   - The first (and the best) is to use the formulated unconstrained projected variance, and to incorporate the constraint using a Lagrange multiplier.
   - The second is to use the constraint explicitly right during the formulation, thereby doing away with the external constraint.

Principal Component Analysis – Constrained Formulation

1. The Constraint N: The U above is subject to the constraint $N = \sum_{i=0}^{n-1} {w_i}^2 = 1$. This forces the eventual solution to lie on the n-D sphere with ordinates $\{w_i\}_{i=0}^{n-1}$.

2. Reconstitute the Constrained Optimizer: We optimize U subject to the constraint $N = \sum_{i=0}^{n-1} w_i^{\,2} = 1$. To put it differently, we optimize V, where $V = \frac{U}{N}$, thereby the scaling with N automatically ensures that the constraint is applied.

3. Formulate $\frac{\partial V}{\partial w_k}$: $\frac{\partial V}{\partial w_k} = \frac{1}{N^2}\left[N\frac{\partial U}{\partial w_k} - U\frac{\partial N}{\partial w_k} \right]$. The w_k corresponding to $\frac{\partial V}{\partial w_k} = 0$ and $\frac{\partial^2 V}{\partial w_k^{\,2}} < 0$ results in the maximization of the constrained variance.

4. Extract $\frac{\partial U}{\partial w_k}$: $\frac{\partial U}{\partial w_k} = 2w_k\left\langle x_k^{\,2}\right\rangle + 2\sum_{i=0, i\neq k}^{n-1} w_i\left\langle x_i x_k\right\rangle$. Note that this reduces to $\sum_{i=0}^{n-1} w_i\left\langle x_i x_k\right\rangle = 0$.

5. k-separation of N and the extraction of $\frac{\partial N}{\partial w_k}$: $N = w_k^{\,2} + \sum_{i=0, i\neq k}^{n-1} w_i^{\,2}$, and so $\frac{\partial N}{\partial w_k} = 2w_k$.

6. The Optimizer Simplification: Given that $N \neq 0$, we simply seek to solve $N^2\frac{\partial V}{\partial w_k} = N\frac{\partial U}{\partial w_k} - U\frac{\partial N}{\partial w_k} = 0$.

7. k-separated Optimizer Series: $N\frac{\partial U}{\partial w_k} - U\frac{\partial N}{\partial w_k} =$

$$-2w_k^{\,2}\sum_{i=0, i\neq k}^{n-1} w_i\left\langle x_i x_k\right\rangle + 2w_k\left[w_k^{\,2}\left\langle x_k^{\,2}\right\rangle - \sum_{i=0, i\neq k}^{n-1}\sum_{j=0, j\neq k}^{n-1} w_i w_j\left\langle x_i x_j\right\rangle \right] + 2\left[\sum_{i=0, i\neq k}^{n-1} w_i^{\,2}\right]\left[\sum_{j=0, j\neq k}^{n-1} w_j\left\langle x_j x_k\right\rangle\right]$$

8. $\frac{\partial^2 V}{\partial w_k^{\,2}}$: $\frac{\partial^2 V}{\partial w_k^{\,2}} = \frac{1}{N^3}\left[2U\left(\frac{\partial N}{\partial w_k}\right)^2 - N\left(2\frac{\partial U}{\partial w_k}\frac{\partial N}{\partial w_k} + U\frac{\partial^2 N}{\partial w_k^{\,2}}\right) + N^2\frac{\partial^2 N}{\partial w_k^{\,2}}\right]$ => this is so fucking intractable!

9. Problems with Explicit Constraint Incorporation:
   - $\frac{\partial V}{\partial w_k}$ is out of control, non-intuitive, irreducible, and devoid of formulaic meaning
   - $\frac{\partial^2 V}{\partial w_k^{\,2}}$ even more so

- These terms are so derivationally challenging even in 2D PCA, as will soon be seen.

2D Principal Component Analysis – Constrained Formulation

1. Base Setup:
   - Target Vector: $\hat{W} = \frac{w_x\hat{x} + w_y\hat{y}}{\sqrt{w_x^{\,2} + w_y^{\,2}}} = \frac{w_x\hat{x} + w_y\hat{y}}{\sqrt{N}}$ where $N = w_x^{\,2} + w_y^{\,2}$, consistent with earlier terminology.
   - Point Vector Under Consideration: $\hat{P} = \frac{x_p\hat{x} + y_p\hat{y}}{\sqrt{x_p^{\,2} + y_p^{\,2}}}$.
2. Point-Target Projection:
   - $\hat{P}.\hat{W} = \frac{w_x x_p + w_y y_p}{\sqrt{w_x^{\,2} + w_y^{\,2}}\sqrt{x_p^{\,2} + y_p^{\,2}}}$.
   - Projection of $\vec{P}$ on $\hat{W} = \hat{P}.\hat{W} * \left|\vec{P}\right| = \frac{w_x x_p + w_y y_p}{\sqrt{w_x^{\,2} + w_y^{\,2}}}$.
3. Point Projection Squared Error and Sample Variance:
   - Point Error Squared $V_p = \frac{U_p}{N} = \frac{\left[w_x x_p + w_y y_p\right]^2}{N}$
   - $U = \sum_{p=0}^{q-1} U_p = w_x^{\,2}\sum_{p=0}^{q-1} x_p^{\,2} + w_y^{\,2}\sum_{p=0}^{q-1} y_p^{\,2} + 2w_x w_y \sum_{p=0}^{q-1} x_p y_p$
   - $V = \frac{U}{N} = \frac{w_x^{\,2}\left\langle x^2\right\rangle + w_y^{\,2}\left\langle y^2\right\rangle + 2w_x w_y\left\langle xy\right\rangle}{N}$ after migrating to the expectations framework.
4. Unconstrained Variance and Constraint Derivatives:
   - $\frac{\partial U}{\partial w_x} = 2w_x\left\langle x^2\right\rangle + 2w_y\left\langle xy\right\rangle$
   - $\frac{\partial^2 U}{\partial w_x^{\,2}} = 2\left\langle x^2\right\rangle$

- $\frac{\partial N}{\partial w_x} = 2w_x$
- $\frac{\partial^2 N}{\partial w_x^{\,2}} = 2$

5. Constrained Variance Maximizer Set up: From before, $\frac{\partial V}{\partial w_k} = \frac{1}{N^2}\left[N\frac{\partial U}{\partial w_k} - U\frac{\partial N}{\partial w_k} \right]$.

   Thus, $\frac{\partial V}{\partial w_k} = \frac{1}{N^2}\left[-2w_x^{\,2}w_y\langle xy\rangle + 2w_x w_y^{\,2}\left(\langle x^2\rangle - \langle y^2\rangle\right) + 2w_y^{\,3}\langle xy\rangle \right] = 0$.

6. Note on PCA for uncorrelated variates: If x and y are perfectly uncorrelated, that implies that the data should be all over, with no preferential principal component. This means that the data lies with uniform density over the n-D spherical shell.
   - The fact that no solution is possible in such a case is clear from the fact that the coefficients of $w_x^{\,2}$ in $\frac{\partial V}{\partial w_k}$ is $\langle xy\rangle$, which for mean-centered data is zero, as $\langle xy\rangle = \langle x\rangle\langle y\rangle = 0$ (i.e., the solution requires both w_x and w_y to be zero, which is impossible). It is obvious from intuition that this should be true for orthogonal higher dimensions as well – we will see a proof of this soon.

7. Variance/Covariance Coefficient Representation: Setting $\langle x^2\rangle \to \sigma_X$, $\langle y^2\rangle \to \sigma_y$, and $\langle xy\rangle \to \rho_{xy}\sigma_X\sigma_Y$, we get $\frac{\partial V}{\partial w_k} = \frac{1}{N^2}\left[-2w_x^{\,2}w_y\rho_{xy}\sigma_X\sigma_Y + 2w_x w_y^{\,2}\left(\sigma_X^{\,2} - \sigma_Y^{\,2}\right) + 2w_y^{\,3}\rho_{xy}\sigma_X\sigma_Y \right] = 0$. Eliminating $w_y = 0$ from consideration as a possible solution, and setting $\alpha = \frac{w_y}{w_x}$, the above equation becomes $\alpha^2\rho_{xy}\sigma_X\sigma_Y + \alpha\left(\sigma_X^{\,2} - \sigma_Y^{\,2}\right) - \rho_{xy}\sigma_X\sigma_Y = 0$ (or, more succinctly, $\alpha^2\langle xy\rangle + \alpha\left(\langle x^2\rangle - \langle y^2\rangle\right) - \langle xy\rangle = 0$).

8. Solution to α: $\alpha = \frac{\sigma_Y^{\,2} - \sigma_X^{\,2} \pm \sqrt{\sigma_X^{\,4} + \sigma_Y^{\,4} + 2\left(2\rho_{XY}^{\,2} - 1\right)\sigma_X^{\,2}\sigma_Y^{\,2}}}{2\rho_{xy}\sigma_X\sigma_Y}$
   - Solution for $\rho_{xy} = 1$, the perfectly correlated case => In this case, the solution should intuitively correspond to the diagonals $\sigma_X = \pm\sigma_Y$ - the corresponding

variance axis, depending on which corresponds to the major/minor component. This may be seen by plugging $\rho_{xy} = 1$ in the solution for α:

$$\alpha = \frac{\sigma_Y{}^2 - \sigma_X{}^2 \pm \sqrt{\sigma_X{}^4 + \sigma_Y{}^4 + 2\sigma_X{}^2\sigma_Y{}^2}}{2\rho_{xy}\sigma_X\sigma_Y} = \frac{\sigma_X}{\sigma_Y}, \frac{\sigma_Y}{\sigma_X}$$

- Solution for $\rho_{xy} = 0$, the perfectly uncorrelated case => Here, $\alpha = \frac{0}{0}$, or ∞, either of which can be a solution. Thus, there can be no solution for the $\rho_{xy} = 0$ case.

2D Principal Component Analysis – Lagrange Multiplier Based Constrained Optimization

1. Base Setup: We consider the projected variance as a constrained optimization exercise:
   - $\Lambda = w_x{}^2\langle x^2\rangle + w_y{}^2\langle y^2\rangle + 2w_x w_y\langle xy\rangle - \lambda\left(w_x{}^2 + w_y{}^2 - 1\right)$
   - $\Lambda = \left(\langle x^2\rangle - \lambda\right)w_x{}^2 + \left(\langle y^2\rangle - \lambda\right)w_y{}^2 + 2w_x w_y\langle xy\rangle + \lambda$
2. Optimization of the above with respect to w_x and w_y:
   - $\frac{\partial\Lambda}{\partial w_x} = 2\left(\langle x^2\rangle - \lambda\right)w_x + 2w_y\langle xy\rangle = 0$
   - $\frac{\partial\Lambda}{\partial w_y} = 2w_x\langle xy\rangle + 2\left(\langle x^2\rangle - \lambda\right)w_y = 0$
   - To ensure consistency of the linear system above, you need the determinant across the equation set to vanish, i.e., $\left(\langle x^2\rangle - \lambda\right)\left(\langle y^2\rangle - \lambda\right) - \langle xy\rangle = 0$. This should be true independent of the constraint.
3. Constraint Optimizer Formulation: Second derivative:
   - $\frac{\partial^2\Lambda}{\partial w_x{}^2} = 2\left(\langle x^2\rangle - \lambda\right) < 0$ ONLY if $\lambda > \langle x^2\rangle$.

- $\frac{\partial^2 \Lambda}{\partial w_y^{\,2}} = 2\left(\left\langle y^2 \right\rangle - \lambda\right) < 0$ ONLY if $\lambda > \left\langle y^2 \right\rangle$.
- Of course, if $\lambda > \left\langle x^2 \right\rangle$ and $\lambda > \left\langle y^2 \right\rangle$, that satisfies both of the above.
- $\frac{\partial \Lambda}{\partial \lambda} = 0$ => $w_x^{\,2} + w_y^{\,2} = 1$. Thus, this recovers the constraint as expected.

4. Solution to λ: From the linear system collinearity, we get

   $\lambda^2 - \left(\left\langle x^2 \right\rangle - \left\langle y^2 \right\rangle\right)\lambda + \left\langle x^2 \right\rangle\left\langle y^2 \right\rangle - \left\langle xy \right\rangle^2 = 0$. Thus,

   $$\lambda_{1,2} = \frac{\left\langle x^2 \right\rangle + \left\langle y^2 \right\rangle \pm \sqrt{\left(\left\langle x^2 \right\rangle - \left\langle y^2 \right\rangle\right)^2 + 4\left\langle xy \right\rangle^2}}{2}.$$

5. Residuals Analysis: Set $\Re = \frac{\sqrt{\left(\left\langle x^2 \right\rangle - \left\langle y^2 \right\rangle\right)^2 + 4\left\langle xy \right\rangle^2} - \left(\left\langle x^2 \right\rangle - \left\langle y^2 \right\rangle\right)}{2}$. Thus, $\Re > 0$.

   Further, $\lambda_1 = \left\langle x^2 \right\rangle + \Re$, and $\lambda_2 = \left\langle y^2 \right\rangle - \Re$. Thus, $\lambda_1 > \left\langle x^2 \right\rangle$, and $\lambda_2 < \left\langle y^2 \right\rangle$.
   - $\left\langle y^2 \right\rangle - \lambda_1 = \left\langle y^2 \right\rangle - \left\langle x^2 \right\rangle - \Re$, which may or may not satisfy the original criterion.
   - $\left\langle y^2 \right\rangle - \lambda_2 = \Re$, which eliminates this from the first PCA.
   - Thus, the best candidate for the first PCA is λ_1.

6. w_x / w_y Solution: Remember $w_y = -\frac{\lambda - \left\langle x^2 \right\rangle}{\left\langle xy \right\rangle} w_x$, and that $w_x^{\,2} + w_y^{\,2} = 1$. Thus,

   $$w_x = \frac{\pm \left\langle xy \right\rangle}{\sqrt{\left(\lambda - \left\langle x^2 \right\rangle\right)^2 + \left\langle xy \right\rangle^2}} \text{ and } w_y = \frac{\mp \left(\lambda - \left\langle x^2 \right\rangle\right)}{\sqrt{\left(\lambda - \left\langle x^2 \right\rangle\right)^2 + \left\langle xy \right\rangle^2}}.$$

7. Principal Components:
   - $w_{1x} = \frac{\left\langle xy \right\rangle}{\sqrt{\left(\lambda_1 - \left\langle x^2 \right\rangle\right)^2 + \left\langle xy \right\rangle^2}}$; $w_{1y} = \frac{\left(\lambda_1 - \left\langle x^2 \right\rangle\right)}{\sqrt{\left(\lambda_1 - \left\langle x^2 \right\rangle\right)^2 + \left\langle xy \right\rangle^2}}$
   - $w_{2x} = \frac{\left\langle xy \right\rangle}{\sqrt{\left(\lambda_2 - \left\langle x^2 \right\rangle\right)^2 + \left\langle xy \right\rangle^2}}$; $w_{2y} = \frac{\left(\lambda_2 - \left\langle x^2 \right\rangle\right)}{\sqrt{\left(\lambda_2 - \left\langle x^2 \right\rangle\right)^2 + \left\langle xy \right\rangle^2}}$

8. Orthogonality of the Principal Components:

- $\vec{W}_1 = w_{1x}\hat{x} + w_{1y}\hat{y}$
- $\vec{W}_2 = w_{2x}\hat{x} + w_{2y}\hat{y}$
- $\vec{W}_1.\vec{W}_2 = w_{1x}w_{2x} + w_{1y}w_{2y} = \dfrac{\langle xy\rangle^2 + \left(\lambda_1 - \langle x^2\rangle\right)\left(\lambda_2 - \langle x^2\rangle\right)}{\sqrt{\left(\lambda_1 - \langle x^2\rangle\right)^2 + \langle xy\rangle^2}\sqrt{\left(\lambda_2 - \langle x^2\rangle\right)^2 + \langle xy\rangle^2}}$
- $\lambda_1 - \langle x^2\rangle = \dfrac{-\langle x^2\rangle + \langle y^2\rangle + \sqrt{\left(\langle x^2\rangle - \langle y^2\rangle\right)^2 + 4\langle xy\rangle^2}}{2}$
- $\lambda_2 - \langle x^2\rangle = \dfrac{-\langle x^2\rangle + \langle y^2\rangle - \sqrt{\left(\langle x^2\rangle - \langle y^2\rangle\right)^2 + 4\langle xy\rangle^2}}{2}$
- Thus, $\left(\lambda_1 - \langle x^2\rangle\right)\left(\lambda_2 - \langle x^2\rangle\right) = -\langle xy\rangle^2$ => $\vec{W}_1.\vec{W}_2 = 0$.

9. <u>Sample Cross-Variance and Joint Expectation under the new Principal Components</u>:

   As always, $\hat{P} = \dfrac{x_p\hat{x} + y_p\hat{y}}{\sqrt{x_p{}^2 + y_p{}^2}}$.

   - Projection of $\hat{P}$ on $\vec{W}_1$ => $P_1 = w_{1x}x_p + w_{1y}y_p$
   - Projection of $\hat{P}$ on $\vec{W}_2$ => $P_2 = w_{2x}x_p + w_{2y}y_p$
   - $P_1P_2 = \left(w_{1x}x_p + w_{1y}y_p\right)\left(w_{2x}x_p + w_{2y}y_p\right) = w_{1x}w_{2x}x_p{}^2 + w_{1y}w_{2y}y_p{}^2 + \left(w_{1x}w_{2y} + w_{1y}w_{2x}\right)x_py_p$
   - $\langle P_1P_2\rangle = w_{1x}w_{2x}\langle x^2\rangle + w_{1y}w_{2y}\langle y^2\rangle + \left(w_{1x}w_{2y} + w_{1y}w_{2x}\right)\langle xy\rangle$
   - $w_{1x}w_{2x}\langle x^2\rangle = \dfrac{\langle xy\rangle^2\langle x^2\rangle}{\sqrt{\left(\lambda_1 - \langle x^2\rangle\right)^2 + \langle xy\rangle^2}\sqrt{\left(\lambda_2 - \langle x^2\rangle\right)^2 + \langle xy\rangle^2}}$
   - $w_{1y}w_{2y}\langle y^2\rangle = \dfrac{\left(\lambda_1 - \langle x^2\rangle\right)\left(\lambda_2 - \langle x^2\rangle\right)\langle y^2\rangle}{\sqrt{\left(\lambda_1 - \langle x^2\rangle\right)^2 + \langle xy\rangle^2}\sqrt{\left(\lambda_2 - \langle x^2\rangle\right)^2 + \langle xy\rangle^2}}$
   - Thus, $w_{1x}w_{2x}\langle x^2\rangle + w_{1y}w_{2y}\langle y^2\rangle = \dfrac{\langle xy\rangle^2\left(\langle x^2\rangle - \langle y^2\rangle\right)}{\sqrt{\left(\lambda_1 - \langle x^2\rangle\right)^2 + \langle xy\rangle^2}\sqrt{\left(\lambda_2 - \langle x^2\rangle\right)^2 + \langle xy\rangle^2}}$, since $\left(\lambda_1 - \langle x^2\rangle\right)\left(\lambda_2 - \langle x^2\rangle\right) = -\langle xy\rangle^2$.

- $\left(w_{1x}w_{2y} + w_{1y}w_{2x}\right)\langle xy\rangle = \dfrac{\langle xy\rangle^2\left(\left\langle x^2\right\rangle - \lambda_1\right) + \langle xy\rangle^2\left(\left\langle x^2\right\rangle - \lambda_2\right)}{\sqrt{\left(\lambda_1 - \left\langle x^2\right\rangle\right)^2 + \langle xy\rangle^2}\sqrt{\left(\lambda_2 - \left\langle x^2\right\rangle\right)^2 + \langle xy\rangle^2}}$
- Given that $\left\langle x^2\right\rangle - \lambda_1 + \left\langle x^2\right\rangle - \lambda_2 = \left\langle y^2\right\rangle - \left\langle x^2\right\rangle$,

  $\left(w_{1x}w_{2y} + w_{1y}w_{2x}\right)\langle xy\rangle = \dfrac{\langle xy\rangle^2\left(\left\langle y^2\right\rangle - \left\langle x^2\right\rangle\right)}{\sqrt{\left(\lambda_1 - \left\langle x^2\right\rangle\right)^2 + \langle xy\rangle^2}\sqrt{\left(\lambda_2 - \left\langle x^2\right\rangle\right)^2 + \langle xy\rangle^2}}$
- Combining all the above, we see that $\left\langle P_1P_2\right\rangle = 0$, showing that extracting principal components is identically the same as diagonalizing the data set covariance.

10. PCA as an Eigenization Operation: The diagonal entries of the 2D Matrix get converted to $\lambda_1 - \left\langle x^2\right\rangle$ and $\lambda_2 - \left\langle x^2\right\rangle$ by virtue of the constraint incorporation. Thus, eigenization corresponds to norm-preserving constraint variance maximization.
11. PCA Variance Range: If $\left\langle x^2\right\rangle > \left\langle y^2\right\rangle$, the range of variance resides between $\left\langle x^2\right\rangle + \varepsilon_1$ and $\left\langle y^2\right\rangle - \varepsilon_2$. If the data set is already orthogonally represented in its native basis, then $\varepsilon_1 = \varepsilon_2 = 0$.

n-D Principal Component Analysis – Lagrange Multiplier Based Constrained Optimization

1. The Lagrangian Set up: The main step is to separate the U term into variance and the co-variance sub-components.
2. Lagrangian – Point Projected Variance Partitioning:

   $$U_p = \left[\sum_{i=0}^{n-1} w_i x_{i,p}\right]\left[\sum_{j=0}^{n-1} w_j x_{j,p}\right] = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} w_i w_j x_{i,p} x_{j,p} = \sum_{i=0}^{n-1} {w_i}^2 {x_{i,p}}^2 + \sum_{i=0}^{n-1}\sum_{j=0, j\neq i}^{n-1} w_i w_j x_{i,p} x_{j,p}$$

3. Lagrangian – Partitioning Variance: $U = \sum_{i=0}^{n-1} {w_i}^2 \left\langle {x_i}^2\right\rangle + \sum_{i=0}^{n-1}\sum_{j=0, j\neq i}^{n-1} w_i w_j \left\langle x_i x_j\right\rangle$

4. Formulation of the Lagrangian: Setting the constraint as $N = \sum_{i=0}^{n-1} w_i^{\,2} - 1$, you get

$$\Lambda = U - \lambda N = \sum_{i=0}^{n-1} w_i^{\,2} \left\langle x_i^{\,2} \right\rangle + \sum_{i=0}^{n-1} \sum_{j=0,\, j\neq i}^{n-1} w_i w_j \left\langle x_i x_j \right\rangle - \lambda \left(\sum_{i=0}^{n-1} w_i^{\,2} - 1 \right)$$

- $\Lambda = \sum_{i=0}^{n-1} w_i^{\,2} \left(\left\langle x_i^{\,2} \right\rangle - \lambda \right) + \sum_{i=0}^{n-1} \sum_{j=0,\, j\neq i}^{n-1} w_i w_j \left\langle x_i x_j \right\rangle + \lambda$

5. Decompose the Lagrangian into variance/covariance sections: $\Lambda = \upsilon + \xi$, where

$\upsilon = \sum_{i=0}^{n-1} w_i^{\,2} \left(\left\langle x_i^{\,2} \right\rangle - \lambda \right)$, and $\xi = \sum_{i=0}^{n-1} \sum_{j=0,\, j\neq i}^{n-1} w_i w_j \left\langle x_i x_j \right\rangle$.

6. k Separate υ and ξ: $\upsilon = w_k^{\,2} \left(\left\langle x_k^{\,2} \right\rangle - \lambda \right) + \sum_{i=0, i\neq k}^{n-1} w_i^{\,2} \left(\left\langle x_i^{\,2} \right\rangle - \lambda \right)$, and

$$\xi = 2 w_k \sum_{i=0, i\neq k}^{n-1} w_i \left\langle x_i x_k \right\rangle + \sum_{i=0, i\neq k}^{n-1} \sum_{j=0,\, j\neq i,k}^{n-1} w_i w_j \left\langle x_i x_j \right\rangle$$

7. Derivatives of Λ: $\frac{\partial \Lambda}{\partial w_k} = 2 w_k \left(\left\langle x_k^{\,2} \right\rangle - \lambda \right) + 2 \sum_{i=0, i\neq k}^{n-1} w_i \left\langle x_i x_k \right\rangle = 0$. Also,

$\frac{\partial^2 \Lambda}{\partial w_k^{\,2}} = 2 \left(\left\langle x_k^{\,2} \right\rangle - \lambda \right)$, thus all the earlier observations made with regards to 2D PCA apply here too.

8. Constraint on the Linear System from $\frac{\partial \Lambda}{\partial w_k} = 0$: Again, the above linear system should have zero determinant $L = 0$. This will, in turn, produce n eigen-values and eigen-components. $L = \begin{bmatrix} \langle x_0^2 \rangle - \lambda & \langle x_0 x_1 \rangle & \cdots & \langle x_0 x_j \rangle & \cdots & \langle x_0 x_{n-1} \rangle \\ \langle x_1 x_0 \rangle & \langle x_1^2 \rangle - \lambda & \cdots & \cdot & \cdots & \\ \cdot & \cdot & \cdots & \cdot & \cdots & \\ \cdot & \cdot & \cdots & \cdot & \cdots & \\ \cdot & \cdot & \cdots & \cdot & \cdots & \\ \langle x_i x_0 \rangle & \langle x_i x_1 \rangle & \cdots & \langle x_i x_j \rangle & \cdots & \langle x_i x_{n-1} \rangle \\ \cdot & \cdot & \cdots & \cdot & \cdots & \\ \cdot & \cdot & \cdots & \langle x_j^2 \rangle - \lambda & \cdots & \\ \cdot & \cdot & \cdots & \cdot & \cdots & \\ \langle x_{n-1} x_0 \rangle & \langle x_{n-1} x_1 \rangle & \cdots & \langle x_{n-1} x_j \rangle & \cdots & \langle x_{n-1}^2 \rangle - \lambda \end{bmatrix}$.

9. Diagonalized L: If L is already diagonal, then are 3 implications.
   - λ_i corresponds to the i[th] variance.
   - $\max(\lambda_i)$ is the maximum variance, i.e., the principal component.
   - All the components are orthogonal, i.e., the order of $abs(\lambda_i)$ determines the order of the principal component.
10. PCA Determination Steps:
    - Diagonalize the basis to create the new ones in which the data set is orthogonal (i.e., $\langle x_i x_j \rangle = 0$).
    - Each diagonal entry corresponds to the eigen-value, and to the corresponding principal component.
11. PCA Variance Range: The 2D variance range analysis is broadly true for this case as well. Consider the partitioned variance $U = \sum_{i=0}^{n-1} w_i^2 \langle x_i^2 \rangle + \sum_{i=0}^{n-1} \sum_{j=0, j \neq i}^{n-1} w_i w_j \langle x_i x_j \rangle$. Given that (without loss of generality) $w_i w_j \langle x_i x_j \rangle \geq 0$ is always true - because of the following:
    - If $\langle x_i x_j \rangle < 0$, then $w_i w_j < 0$, and

- If $\langle x_i x_j \rangle > 0$, then $w_i w_j > 0$.

Thus, the diagonalized basis corresponds to the maximal/minimal variance range, with maximum corresponding to the most significant PC, and the minimum corresponding to the least significant PC.

Information Theoretic Analysis of PCA

1. Information Preservation Property of PCA: If $\vec{X} = \vec{S} + \vec{n}$ where $\vec{X}$ is the data set, $\vec{S}$ is the signal, and $\vec{n}$ is the noise, then it can be shown that if both $\vec{S}$ and $\vec{n}$ are Gaussian, and $\vec{n}$ is an orthogonal matrix (i.e., the noise components are orthogonal to each other), then PCA maximizes the mutual information $I(\vec{y} \mid \vec{S})$ between $\vec{S}$ and $\vec{y}$, where $\vec{y} = W_L^T \vec{X}$, the L PCA components.
   - If the noise is still Gaussian and orthogonal i.i.d., but $\vec{S}$ is not, PCA ends up minimizing the net information loss $I(\vec{X} \mid \vec{S}) - I(\vec{y} \mid \vec{S})$.
   - If the noise is orthogonal and i.i.d., and is more Gaussian than $\vec{S}$ in the Kullback-Leibler sense, then PCA is still optimal.
   - If the noise is co-dependent, then PCA is not optimal anymore (in terms of information loss, information being measured by Shannon entropy) irrespective of where the signal is Gaussian or not.

Empirical PCA Estimation from the Data Set

1. Calculate the central measures: Calculate the sample mean, the deviations from the mean, and the covariance matrix C.
2. Eigenize the covariance Matrix: Find the eigen-values and eigen-vectors of C from $V^{-1}CV = D$, where V is the matrix of eigen-vectors, and D is a diagonal matrix.

3. Eigen-vector Ordering: Re-arrange the eigen-vectors and eigen-values in the order of decreasing eigen-values.
4. Cumulative Eigen-Energy: Compute the cumulative energy content for each eigen-vector from $g[m] = \sum_{q=1}^{m} D(q,q)$, where m is the total number of eigen-vectors, and the summation runs over the ordered eigen-vector set.
    - Select the sub-set of eigen-vectors as the PCA proxy until the desired energy level has been achieved, e.g., $\frac{g(L)}{g(M)} \geq 0.9$.
5. Conversion to Z-Scores:
    - Convert the source data to Z-scores ($\vec{Z}$) by scaling it element-by-element by the variance.
    - Project the Z-scores of the data onto the new basis as: $Y = W^{*}\vec{Z}$, where W^{*} is the matrix of the eigen-vectors.

Kriging

1. Definition: Kriging is an estimation technique that infers the value of a random field at an unobserved location from the nearby samples. Simply put, kriging is a stochastic spline interpolation technique – the nodal samples themselves are only stochastically observed, i.e., $x_0 = \alpha x_1 + (1-\alpha)x_2$ implies $\langle x_0 \rangle = \alpha \langle x_1 \rangle + (1-\alpha)\langle x_2 \rangle$, and $\langle {x_0}^2 \rangle = \alpha \langle {x_1}^2 \rangle + (1-\alpha)\langle {x_2}^2 \rangle$.
2. Moment Stationarity:
   - Stationarity in the first moment => This means that $x_i = x_j$ for all $i \neq j$. The second moment may vary, though.
   - Stationarity in the second moment => Here, the correlation between x_1 and x_2 is posited to depend solely on $h = |x_1 - x_2|$ (or, more generally, $\vec{h}$).
     - Variance $\gamma(x_1, x_2) = \gamma(|x_1 - x_2|) = \gamma(h)$.
     - Co-variance $\varsigma(x_1, x_2) = \varsigma(|x_1 - x_2|) = \varsigma(h)$.
     - From these, the variance and the covariance may also be computed – based on N samples:
       - $\gamma(\vec{h}) = \frac{1}{2N(\vec{h})} \sum_{p=1}^{N(\vec{h})} \left[x_{i,p} - x_{i+h,p} \right]^2$
       - $\varsigma(\vec{h}) = \frac{1}{N(\vec{h})} \sum_{p=1}^{N(\vec{h})} \left[x_{i,p} x_{i+h,p} - m(x_i) m(x_i + \vec{h}) \right]$
       - $m(x_i)$ is the sample mean, i.e., $m(x_i) = \frac{1}{N(\vec{h})} \sum_{p=1}^{N(\vec{h})} x_{i,p}$.
3. Linear Estimation for a Location:

- $\hat{x}_0 = \begin{bmatrix} w_0 & w_1 & w_2 & \dots & w_{n-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ . \\ . \\ . \\ x_{n-1} \end{bmatrix} = W^T X = \sum_{i=0}^{n-1} w_i(x_0) * x_i$

  - Objectives when calculating the weights => a) Unbias: this implies that the estimation mean is the same as the sample mean. b) Minimal variance of estimation: Somehow, $(\hat{x}_0 - x_0)$ should be minimal.

4. Ordinary Kriging: Error is $(\hat{x}_0 - x_0)$. Since the random field is assumed to be stationary, i.e., $\langle x_i \rangle = \langle x_j \rangle$, $\sum_{i=0}^{n-1} w_i(x_0) = 1$.

   - $\begin{bmatrix} W^T & -1 \end{bmatrix} \begin{bmatrix} Var_i & Co\,\mathrm{var}\langle x_i x_0 \rangle \\ Co\,\mathrm{var}\langle x_i x_0 \rangle & Var_i \end{bmatrix} \begin{bmatrix} W^T \\ -1 \end{bmatrix} = Var(x_0) + \sum_{i,j} w_i w_i \langle x_i x_j \rangle - 2\sum_i w_i \langle x_i x_0 \rangle$

   - This shows the dependence of the error variance on the individual γ and ς operators.

   - Kriging System => This is created by the minimization of $Var[\varepsilon(x_0)] = \begin{bmatrix} W^T & -1 \end{bmatrix} \begin{bmatrix} Var_i & Co\,\mathrm{var}\langle x_i x_0 \rangle \\ Co\,\mathrm{var}\langle x_i x_0 \rangle & Var_i \end{bmatrix} \begin{bmatrix} W^T \\ -1 \end{bmatrix}$, subject to the $\sum_{i=0}^{n-1} w_i(x_0) = 1$ constraint, which is handled by using the Lagrange multipliers λ.

5. Simple Kriging: Both simple and ordinary kriging assume stationarity on the first moment, but in simple kriging, it is further assumed that the first moment (i.e., the mean) is known.

6. Universal Kriging: Universal kriging assumes a polynomial trend model for the location estimator, i.e., $\hat{x}_0 = \sum_{i=0}^{n-1} \beta_i f_i(\vec{x})$.

7. IRFk Kriging: Same as universal kriging, but assumes an unknown polynomial in $\vec{x}$.

8. Disjunctive kriging: Disjunctive kriging is a non-linear generalization of kriging. Log-normal kriging is an example.

9. Kriging in Computer Simulation Outputs: Kriging is used in the interpolation of the data coming out as response variables from deterministic computer simulations. Typical steps involved are:
    - Generate several location specific parameters and responses.
    - Use kriging to interpolate between them.

1.

Figure #1
Data Set Orthogonal in its Native Representative Basis

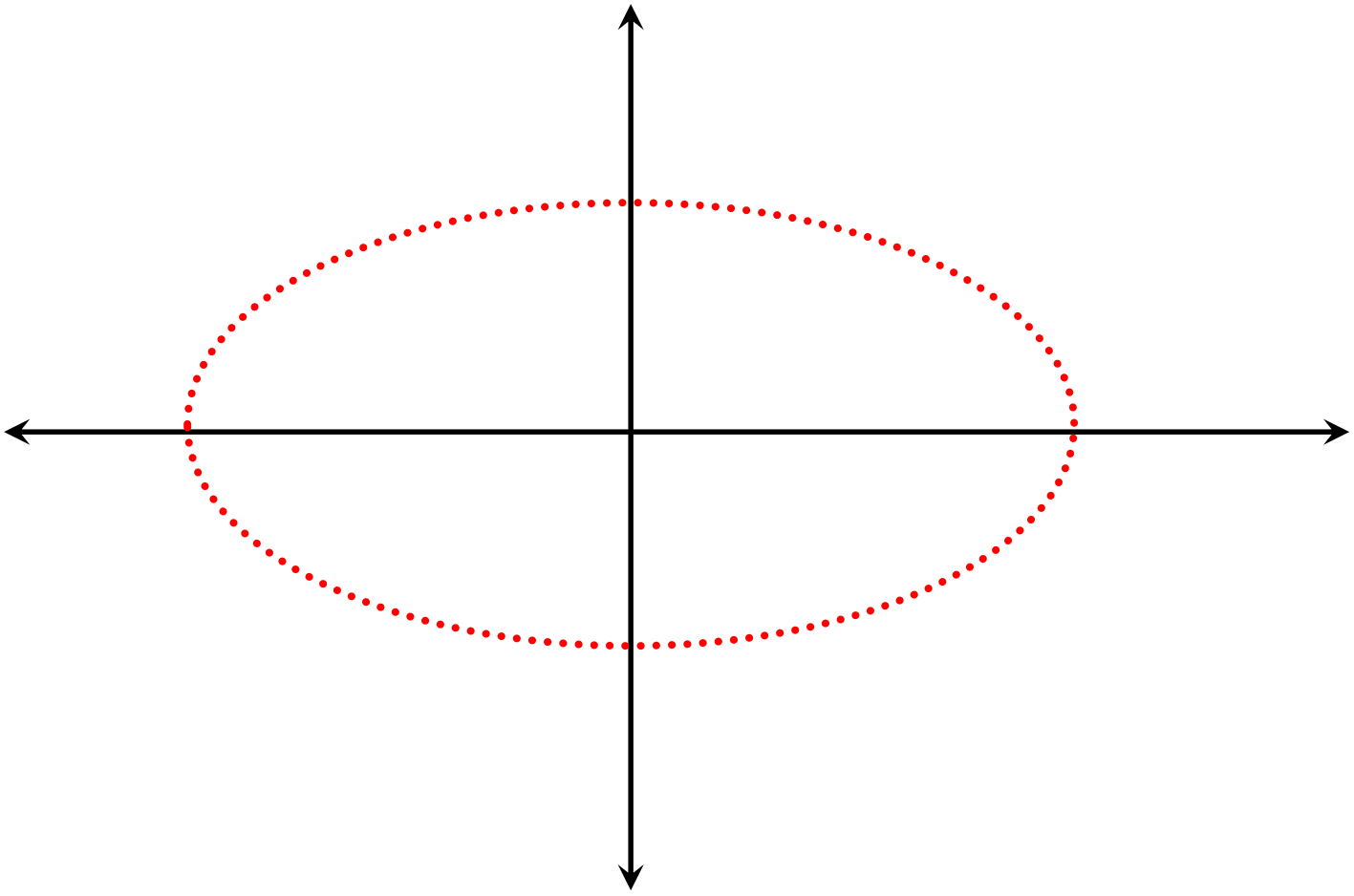


Figure #2
Data Set NOT Orthogonal in its Native Representative Basis

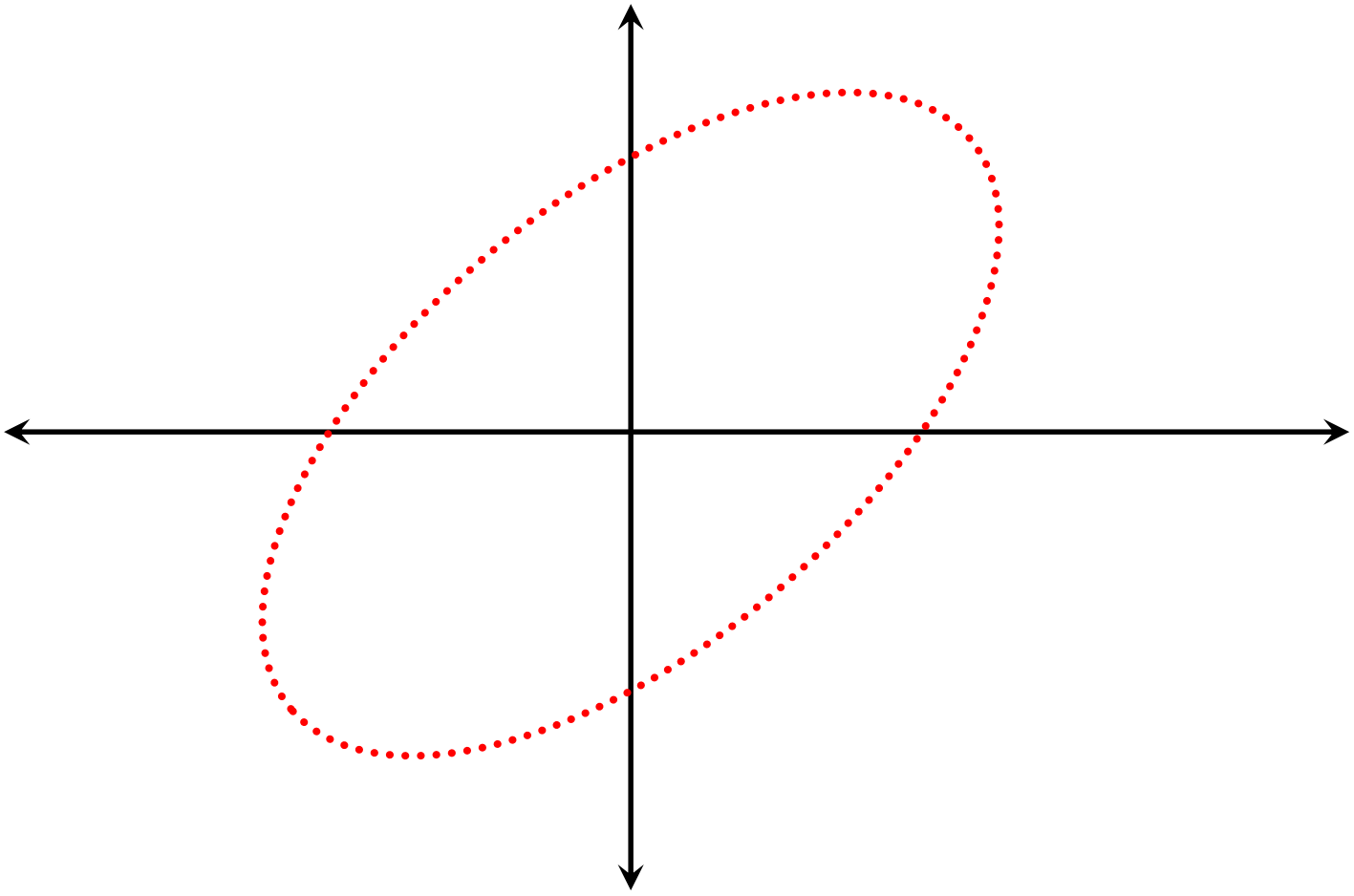
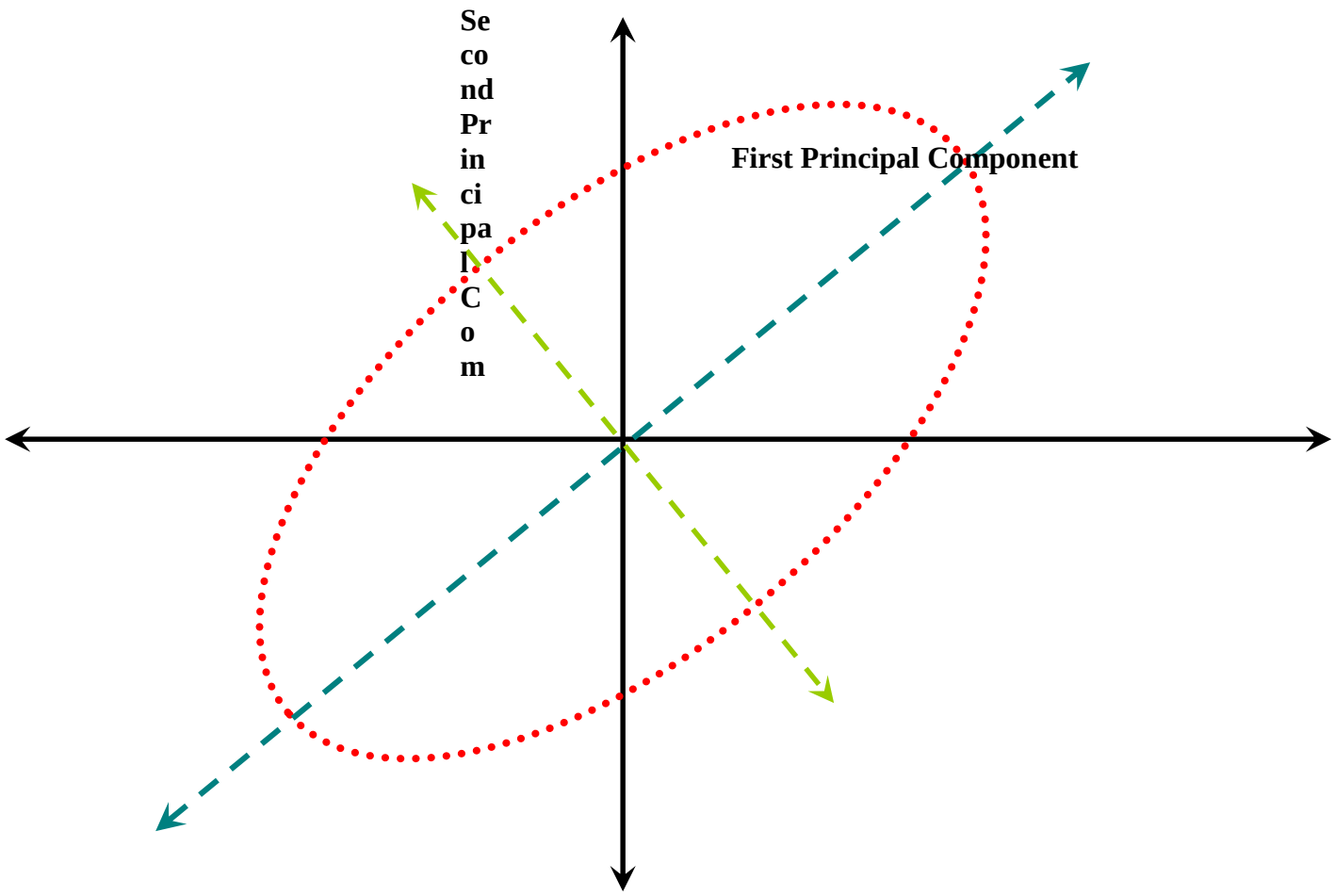


Figure #3
Principal Components that Diagonalize the Data Set



References

- Abdi, H. (2007): Discriminant Corespondence Analysis, in *Encyclopedia of Measurement and Statistic (N. J. Salkind (ed.))* **Sage**, Thousand Oaks, CA.
- Abraham, R., J. E. Marsden, and T. S. Ratiu (1988): *Chapter 5 => Manifolds, Tensor Analysis, Applications,* in *Applied Mathematical Sciences* **Springer-Verlag**, New York.
- Acharyya, R. (2008): *A New Approach for Blind Source Separation for Convolutive Sources* **VDM Verlag Dr. Mueller e K**.
- Achtert, E., C. Bohm, and P. Kroger (2006): DeLiClu: Boosting Robustness, Completeness, Usability, and Efficiency of Hierarchical Clustering by a Closest Pair Ranking *LNCS: Advances in Knowledge Discovery and Data Mining* **3918** 119-128.
- Achtert, E., C. Bohm, P. Kroger, and A. Zimek (2006): Mining Hierarchies of Correlation Clusters *Proceedings of the 18th International Conference on Scientific and Statistical Database Management (SSDBM)* 119-128.
- Achtert, E., C. Bohm, H. P. Kriegel, P. Kroger, I. Muller-Gorman, and A. Zimek (2006): Finding Hierarchies of Subspace Clusters *LNCS: Advances in Knowledge Discovery in Databases* **4213** 446-453.
- Achtert, E., C. Bohm, H. P. Kriegel, P. Kroger, I. Muller-Gorman, and A. Zimek (2007): Detection and Visualization of Subspace Cluster Hierarchies *LNCS: Advances in Databases: Concepts, Systems, and Applications* **4443** 152-163.
- Achtert, E., C. Bohm, H. P. Kriegel, P. Kroger, and A. Zimek (2007): On Exploring Complex Relationships of Correlation Clusters *19th International Conference on Scientific and Statistical Database Management (SSDBM 2007)* ***7***.
- Agrawal, R., J. Gehrke, D. Gunopulos, and P. Raghavan (2003): Automatic Subspace Clustering of High-Dimensional Data *Data Mining and Knowledge Discovery* **11** 5.
- Aharon, M., M. Elad, and A. Bruckstein (2006): k-SVD: An Algorithm for Designing Over-complete Dictionaries for Sparse Representation *IEEE Transactions on Signal Processing* **54 (11)** 4311-4322.

- Ahdesmaki, J., and K. Strimmer (2010): Feature Selection in omics Prediction Problems using CAT Scores and False non-discovery Rate Control *Annals of Applied Statistics* **4 (1)** 503-519.
- Aho, A. V., R. Sethi, and J. D. Ullman (1986): *Compilers: Principles, Techniques, and Tools*, **Addison-Wesley Longman**, Boston.
- Ailon, N., M. Charikar, and A. Newman (2005): Aggregating inconsistent Information: Ranking and Clustering *STOC '05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing* 684-693.
- Aizerman, M. A., E. M. Braverman, and L. I. Rozonoer (1964): Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning *Automation and Remote Control* **25** 821-837.
- Aloise, D., A. Deshpande, P. Hanson, and P. Popat (2009): NP-Hardness of Euclidean Sum-of-Squares Clustering *Machine Learning* **75** 245-249.
- Altman, N. S. (1992): An Introduction to Kernel and Nearest-Neighbor Non-parameteric Regression *The American Statistician* **46 (3)**: 175-185.
- Amorim, R. C., and B. Mirkin (2012): Minkowski, Metric, Feature Weighting, And Anamalous Cluster Initializing using K-Means Clustering *Pattern Recognition* **45 (3)** 1061-1075.
- Anlauf, J. K., and M. Biehl (1989): The AdaTron: An Adaptive Perceptron Algorithm *Europhysics Letters* **10** 687-692.
- Anderson, B. D. O., and J. B. Moore (1979): *Optimal Filtering*, **Prentice Hall**, New York.
- Anderson, T. W. (1958): *An Introduction to Multivariate Analysis* **Wiley**.
- Andreasen, M. M. (2008): *Non-linear DSGE Models, The Central Difference Kalman Filter, and The Mean Shifted Particle Filter*.
- Ankerst, M., M. Breuning, H. P. Kriegel, and J. Sander (1999): OPTICS: Ordering Points to to Identify the Clustering Structure *ACM SIGMOD International Conference on Management of Data* 49-60.
- Artheu, A. Bhowmick (2009): A Theoretical Analysis of the Lloyd's Algorithm for k-Means Clustering.

- Arthur, D., B. Manthey, and H. Roeglin (2009): k-Means has Polynomial Smoothed Complexity *Proceedings of the 50th Cymposium of the Foundations of Computer Science (FOCS).*
- Aslam, J. A., R. A. Popa, and R. L. Rivest (2007): On Estimating the Size and Confidence of a Statistical Audit *Proceedings of the Electroniv Voting Technology (EVT '07)* **Boston MA**.
- Auffarth, B. (2010): Clustering a Genetic Algorithm with Biased Mutation Operator *WCCI CEC*.
- *Autoregressive Least Squares*.
- Aycinena, M. A. (2005): *Probabilistic Geometric Grammars for Object Recognition*, PhD Thesis, **Massachussetts Institute of Technology**.
- Bagon, S., and M. Galun (2011): Large Scale Correlation Clustering Optimization **arXiV**.
- Bailey, K. (1994): *Numerical Taxonomy and Cluster Analysis* Typologies and Taxonomies.
- Baker, J. (1975): The DRAGON System – An Overview, *IEEE Transactions on Acoustics, Speech, and Signal Processing* **23**: 24-29.
- Bansal, N., A. Bloom, and S. Chawla (2004): Correlation Clustering *Machine Learning Journal (Special Issue on Theoretical Advances in Data Clustering)* 86-113.
- Bar-Shalom, Y., X. R. Li., and T. Kirubarajan (2001): *Estimation with Applications to Tracking and Navigation*, **John Wiley & Sons**, New York.
- Basak, S. C., V. R. Magnuson, C. J. Niemi, and R. R. Regal (1988): Determining Structural Similarity of Chemicals using Graph Theoretic Indices *Discrete Applied Math* **19** 17-44.
- Baum, L. E., and T. Petrie (1966): Statistical Inference for Probabilistic Functions of Finite State Markov Chains *The Annals of Mathematical Statistics* **37 (6)**: 1554-1563.
- Baum, L. E., and J. A. Eagon (1967): An Inequality with Applications to Statistical Estimation for Probabilistic Functions of Markov Processes and to a Model for Ecology, *Bulletin of the American Mathematical Society* **73 (3)**: 360.
- Baum, L. E., and G. R. Sell (1968): Growth Transformations for Functions on Manifolds, *Pacific Journal of Mathematics* **27 (2)**: 211-227.

- Baum, L. E., T. Petrie, G. Soules, and N. Weiss (1970): A Maximization Technique Occuring in the Statistical Analysis of Probabilistic Functions of Markov Chains, *The Annals of Mathematical Statistics* **41**: 164.
- Baum, L. E. (1972): An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of a Markov Process, *Inequalities* **3**: 1-8.
- Becker, H. (2005): A Survey of Correlation Clustering.
- Behnke, S. (2003): Hierarchical Neural Networks for Image Interpretation *Lecture Notes in Computer Science* **2766 Springer**.
- Bengio, Y. (2009): Learning Deep Architectures for AI *Foundations and Trends in Machine Learning* **2 (1)** 1-127.
- Bengio, Y., N. Boulanger-Lewandowski, and R. Pascanu (2013): *Advances in Optimizing Recurrent Networks* **arXiV**.
- Bengio, Y., A. Courville, and P. Vincent (2013): Representation Learning: A Review and new Perspectives *IEE Transactions PAMI, Special Issue Learning Deep Architectures*.
- Bensusan, H., and C. G. Giraud-Carrier (2000): Discovering Task Neighborhoods through Landmark Learning Performances *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery* 325-330.
- Berger, A. L., V. J. D. Pietra, and S. A. D. Pietra (1996): Maximum Entropy Approach in Natural Language Processing *Computational Linguistics* **22 (1)**: 39-71.
- Bewley, A., R. Shekhar, S. Leonard, B. Upcroft, and B. Lever (2011): Real-time Volume Estimation of a Dragline Payload *IEEE International Conference on Robotics and Automation (ICRA '11)* 1571-1576.
- Beyer, K., J. Goldstein, R. Ramakrishnan, and U. Shaft (1999): When is Nearest Neighbor Meaningful? *Database Theory – ICRT '99* 217-235.
- Bhandari, M., and A. Joenssen (2008): *Clinical Research for Surgeons* **Thieme**.
- Bias-variance dilemma (Wiki): *Wikipedia Entry for Bias-variance dilemma*.
- Bierman, G. J. (1977): *Factorization Methods for Discrete Sequential Estimation*, **Academic Press**.
- Binder, D. A. (1978): Bayesian Cluster Analysis *Biometrika* **65** 31-38.

- Binder, D. A. (1981): Approximations to Bayesian Clustering Rules *Biometrika* **68** 275-285.
- Bingham, E., and H. Mannila (1996): Maximum Entropy Approach in Matural Language Processing *Computational Linguistics* **22 (1)**: 39-71.
- Bishop, C. M. (2006): *Pattern Recognition and Machine Learning* **Springer** New York.
- Bishop, M. and E. Thompson (2001): Random Projection in Dimensionality Reduction: Applications to Image and Text Data *Proceedings of the 26th Annual International Conference on Knowledge Discovery and Data Mining, ACM 2001*.
- Blakeslee, S. (1995): In Brain's Early Growth, Time Table may be Critical *The New York Times Science Section* B5-B6.
- Blanco, J. L., J. Gonzalez, and J. A. Fernandez-Madrigal (2008): An Optimal Filtering Algorithm for Non-parametric Observation Models in Robot Localization, *IEEE International Conference on Robotics and Automation (ICRA '08)* 461-466.
- Blanco, J. L., J. Gonzalez, and J. A. Fernandez-Madrigal (2010): An Optimal Filtering Algorithm for Non-parametric Observation Models: Applications to Localization and SLAM *The International Journal of Robotics Research (IJRR)* **29 (14)**: 1726-1742.
- Bohm, C., K. Kailing, P. Kroger, and A. Zimek (2004): Computing Clusters of Correlation Connected Objects *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data – SIGMOD '04* 455-467.
- Bolstad, W. M. (2010): *Understanding Computational Bayesian Statistics* **Wiley**.
- Boosting (Wiki): *Wikipedia Entry for Boosting*
- Bootstrap Aggregating (Wiki): *Wikipedia Entry for Bootstrap Aggregating*
- Boser, B. E. I. M. Guyon, and V. N. Vapnik (1992): A Training Algorithm for Optimal Margin Classifiers *5th Annual ACM Workshop on COLT* **ACM Press** Pittsburgh, PA.
- Bottou, L. (1991): *Une Approche Theorique de l'Apprentissage Connexionniste: Applications a la Reconnaissance de la Parole* **Universite de Paris XI**.

- Boudaren, M. Y., E. Monfrini, W. Pieczynski, and A. Aissani (2012): Dempster-Shafer Fusion of Multisensor Signals in Non-stationary Makovian Context, *EURASIP Journal on Advances in Signal Processing* **134**.
- Boudaren, M. Y., E. Monfrini, and W. Pieczynski (2012): Unsupervised Segmentation of Random Discrete Data Hidden with Switching Noise Distributions, *IEEE Signal Processing Letters* **19 (10)** 619-622.
- Bousquet, O., S. Boucheron, and G. Lugosi (2004): Introduction to Statistical Learning Theory *Advanced Lectures in Machine Learning Lecture Notes in Artificial Intelligence - Bousquet, von Luxburg, and Ratsch (editors)* **3176** 169-207.
- Box, G. E. P. (1957): Evolutionary Operation: A Method for Increasing Industrial Productivity *Applied Statistics* **6** 81-101.
- Boyd, C. R., M. A. Tolson, and W. S. Copes (1987): Evaluating Trauma Care: The TRISS Method - Trauma Score and the Injury Severity Score *Journal of Trauma* **27 (4)** 370-378.
- Breiman, L. (1996): Bagging Predictors *Machine Learning* **24 (2)** 123-140.
- Breiman, L. (1998): Arcing Classifier *Annals of Statistics* **26 (3)** 801-849.
- Bremner,D., E. Demaine, J. Erickson, J. Iacono, S. Langerman, P. Morin, and G. Touissant (2005): Output Sensitivie Algorithms for Computing Nearest Neighbor Decision Boundaries *Discrete and Computational Geometry* **33 (4)** 593-604.
- Brodely, C. E., and M. A. Friedl (1999): Identifying and Eliminating Mislabeled Training Instances *Journal of Artificial Intelligence Research* **11** 131-167.
- Brown, G., J. Wyatt, R. Harris, and X. Yao (2005): Diversity Creation Methods: A Survey and Categorization *Information Fusion* **6 (1)** 5-20.
- Brunelli, R. (2009): *Template Matching Techniques in Computer Vision: Theory and Practice* **Wiley**.
- Bucy, R. S. and P. D. Joseph (2005): *Filtering for Stochastic Processes with Applications to Guidance (2nd edition),* **AMS Chelsea Publ**.
- Buetler, R., T. Kaufman, and B. Pfister (2005): Integrating a non-probabilistic Grammar into large Vocabulary Continuous Speech Recognition, *IEEE Workshop on Automated Speech Recognition and Understanding* 104-109.

- Bufill, E. J. Agusti, and R. Blesa (2011): Human Neoteny Revisited: The Case of Synaptic Plasticity *American Journal of Human Biology* **23 (6)** 729-739.
- Can, F., and E. A. Ozkarahan (1990): Concepts and Effectiveness of the Cover-Coefficient-Based Clustering Methodology for Text Databases *ACM Transactions on Database Systems* **15 (4)** 483.
- Canton-Ferrer, C., J. R. Casas, and M. Pardas (2011): Human Motion Capture using Scalable Body Models, *Computer Vision and Image Understanding* **115 (10)**: 1363-1374.
- Carmi, A., P. Gurfil, and D. Kanevsky (2010): Methods for Spase Signal Recovery using Kalman Filtering with embedded pseudo-measurement norms and quasi-norms, *IEEE Transactions on Signal Processing* **58 (4)**: 2405-2409.
- Carpenter, G. A., and S. Grossberg (1988): The ART of adaptive pattern recognition by a self-organizing neural network *Computer* **21** 77-88.
- Carroll, J. D., and J. Chang (1970): Analysis of Individual Differences in Multi-dimensional Scaling via an n-way Generalization of the Eckert-Young Decomposition *Psychometrika* **35** 283-319.
- Cattell, R. B. (1943): The Description of Personality: Basic Traits resolved into Clusters *Journal of Abnormal and Social Psychology* **38** 476-506.
- Chomsky, N. (1957): *Syntactic Structures*, **Mouton and Co. Publishers** Den Haag, Netherlands.
- Chomsky, N. (1980): *Rules and Representations*, **Oxford: Basil Blackwell.**
- Ciresan, D. C., A. Giusti, L. M. Gambardella, and J. Schmidhuber (2012): Deep Neural Networks Segment Neuronal Membranes in Electron Microsocy Images, in: *Advances in Neural Information Processing Systems (NIPS 2012)* **Lake Tahoe**.
- Ciresan, D. C., U. Meier, L. M. Gambardella, and J. Schmidhuber (2010): Deep Big Simple Neural Nets for Handwritten Digit Recognition *Neural Computation* **22** 3207-3220.
- Ciresan, D. C., U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber (2011): Flexible, High Performance Convolutional Neural Networks for Image Classification *International Joint Conference on Artificial Intelligence (IJCAI-2011 Barcelona).*

- Ciresan, D. C., U. Meier, J. Masci, and J. Schmidhuber (2012): Multi-column Deep Neural Network for Image Classification *IEEE Conference on Computer Vision and Pattern Recognition CVPR 2012*.
- Ciresan, D. C., U. Meier, and J. Schmidhuber (2012): Multi-column Deep Neural Network for Traffic Sign Classification *Neural Networks*.
- Ciresan, D. C., U. Meier, and J. Schmidhuber (2012): Multi-column Deep Neural Network for Image Classification *Technical Report No. IDSIA-04-12'*.
- Clark, A. (2001): *Unsupervised Language Acquisition: Theory and Practice,* PhD Thesis, **University of Sussex**.
- Clarke, B. (2003): Bayes Model Averaging and Stacking when Model Approximation Error cannot be ignored *Journal of Machine Learning Research* 683-712.
- Clemen, R. T. (1989): Combining Forecasts: Review and Annotated Bibliography *International Journal of Forecasting* **5 (4)** 559-583.
- Clopinet, I. G., and A. Elisseeff (2003): An Introduction to Variabel and Feature Selection *The Journal of Machine Learning Research* **3** 1157-1182.
- CLUSTER (2009): The CLUSTER Procedure: Clustering Methods *SAS/STAT 9.2 Users Guide* **SAS Institute**.
- Cluster Analysis (Wiki): *Wikipedia Entry for Cluster Analysis*.
- Coates, A., H. Lee, and A. Y. Ng (2011): An Analysis of Single-Layer Networks in Unsupervised Fesature Learning *International Conference on Artificial Intelligence and Statics (AISTATS).*
- Coates, A., and A. Y. Ng (2012): Learning Feature Representations with k-Means, in: Montavon, Orr, and Muller (eds.) *Neural Networks: Tricks of the Trade* **Springer**.
- Cohen, J., P. Cohen, S. G. West, and L. S. Aiken (2002): *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences 3rd Edition* **Routledge**.
- Collins, M. (2002): Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with the Perceptron Algorithm *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '02)*.
- Computational Learning Theory (Wiki): *Wikipedia Entry for Computational Learning Theory*.
- Conditional Random Field (Wiki): *Wikipedia Entry for Conditional Random Field*.

- Coomans, D., and D. L. Massart (1982): Alternative k-Nearest Neighbor Rules in Supervised Pattern Recognition: Part I: k-Nearest Neighbor Classification using Alternative Voting Rules *Analytica Chimica Acta* **136** 15-27.
- Correlation Clustering (Wiki): *Wikipedia Entry for Correlation Clustering*.
- Cortes, C., and V. N. Vapnik (1995): Support Vector Networks *Machine Learning* **20** 273-297.
- Cover, T. M. (1965): Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition *IEEE Transactions on Electronic Computers* **EC-14 (3)** 326-334.
- Cover, T. M., and P. E. Hart (1967): Nearest Neighbor Pattern Classification *IEEE Transactions on Information Theory* **13 (1)** 21-27.
- Crammer, K., and Y. Singer (2001): On the Algorithmic Implementation of Multiclass Kernel Based Vector Machines *Journal of Machine Learning Research* **2** 265-292.
- Csurka, G., C. C. Dance, L. Fan, J. Willamowski, and C. Bray (2004): Visual Categorizations with Bags of Keypoints *ECCV Workshop on Statistical Learning in Computer Vision*.
- Curbastro, R. (1892): Resume de quelques travaux sur lessystemes variables defonctions associes a une forme differentielle quadratique *Bulletin des Sciences Mathematiques* **2 (16)**: 167-189.
- Dahl, G., T. N. Sainath, and G. E. Hinton (2013): Improving Deep Neural Networks for LVCSR using Rectified Linear Units and Dropout *ICASSP '13*.
- Darroch, J. N. and D. Ratcliff (1972): Generalized Iterative Scaling for Log-Linear Models *The Annals Mathematical Studies* **43 (5)**: 1470-1480.
- Dasgupta, S., and Y. Freund (2009): Random Projection Trees for Vector Quantization *IEEE Transactions On* **55** 3229-3242.
- Day, N. E. (1969): Estimating the Components of the Mixture of a Normal Distribution *Biometrika* **56 (3)** 463-474.
- Decision List (Wiki): *Wikipedia Entry for Decision List*.
- Decision Tree (Wiki): *Wikipedia Entry for Decision Tree*.
- Deep Learning (Wiki): *Wikipedia Entry for Deep Learning*.

- Defays, D. (1977): An Efficient Algorithm for a Complete-Link Method *The Computer Journal* **20 (4)** 364-366.
- Dempster, A. P., N. M. Laird, and D. B. Rubin (1977): Maximum Likelihood from Incomplete Data via the EM Algorithm *Journal of the Royal Statistical Society Series B* **39 (1)** 1-38.
- Deng, H., G. Runger, and E. Tuv (2011): Bias of Importance Measures for Multi-Valued Attributes and Solutions *Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN)*.
- Dhillon, I. S., and D. M. Modha (2001): Concept Decomposition for Large Sparse Text Data using Clustering *Machine Learning* **42 (1)** 143-175.
- Dietterich, T. G. and G. B. Bakiri (1995): Solving Multiclass Learning Problems via Error-Correcting Output Codes *The Journal of Artificial Intelligence Research* **2 (2)** 263-286.
- Ding, C., and X. He (2004): k-Means Clustering via Principal Component Analysis *Proceedings on the International Conference on Machine Learning (ICML 2004)* 225-232.
- Dinov, I. D. (2008): Expectation Maximization and Mixture Modeling Tutorial *California Digital Library Statistics Online Computational Resource*.
- DISTANCE (2009): The DISTANCE Procedure: Proximity Measures *SAS/STAT 9.2 Users Guide* **SAS Institute**.
- Domingos, P. (2000): Bayesian Averaging of the Clasifiers and the Over-fitting Problem *Proceedings of the 17th International Conference on Machine Learning (ICML)* 223-230.
- Doucet, A., N. De Freitas, K. Murphy, and S. Russell (2000): Rao-Blackwellised particle filter for dynamic Bayesian Networks, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence* 176-183.
- Doucet, A., N. De Freitas, and N. J. Gordon (2001): *Sequential Monte-Carlo Methods in Practice,* **Springer**.
- Drineas, P., A. Frieze, R. Kannan, S. Vempala, and V. Vinay (2004): Clustering Large Graphs via Singular Value Decomposition *Machine Learning* **56** 9-33.

- Drucker, H., C. J. C. Burges, L. Kaufman, A. J. Smola, and V. N. Vapnik (1997): Support Vector Regression Machines, in: *Advances in Neural Information Processing Systems 9 (NIPS 1996)* **MIT Press**.
- Duan, K. B. and S. S. Keerthi (2005): Which is the Best Multi-class SVM? An Empirical Study *Proceedings of the 6th International Workshop on Multiple Classifier Systems* **3541** 278.
- Duda, R. O., P. E. Hart, and D. G. Stork (2001): *Pattern Classification 2nd Edition* **Wiley**, New York.
- Dunn, J. (1974): Well Separated Clusters and Optimal Fuzzy Partitions *Journal of Cybernetics* **4** 95-104.
- Durbin, R., S. R. Eddy, A. Krogh, and G. Mitchison (1999): *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids* **Cambridge University Press**.
- Eddy, S. R., and R. Durbin (1994): RNA Sequence Analysis using Covariance Models, *Nuclei Acids Res.* **22 (11)**: 2079-2088.
- Egmont-Petersen, M., D. de Ridder, and H. Handels (2002): Image Processing with Neural Networks – A Review *Pattern Recognition* **35 (10)** 2279-2301.
- Einicke, G. A. (2006): Optimal and Robust Non-causal Filter Formulations, *IEEE Trans. Signal Processing* **54 (3)**: 1069-1077.
- Einicke, G. A. (2007): Asymptotic Optimality of the Minimum Variance Fixed Interval Smoother, *IEEE Trans. Signal Processing* **55 (4)**: 1543-1547.
- Einicke, G. A., J. C. Ralston, C. O. Hargrave, D. C. Reid, and D. W. Hainsworth (2008): Longwall Mining Automation: An Application of Minimum Variance Smoothing, *IEEE Control Systems Magazine* **28 (6)**: 1543-1547.
- Einicke, G. A. (2009): Asymptotic Optimality of the Minimum Variance Fixed Interval Smoother, *IEEE Trans. Signal Processing* **54 (12)**: 2904-2908.
- Elkan, C. (2003): Using the Triangle Inequality to Accelerate k-Means *Proceedings of the 20th International Conference on Machine Learning (ICML)*.
- Elman, J. L., E. A. Bates, M. H. Johnson, A. Karmiloff-Smith, D. Parisi, and K. Plunkett (1996): *Rethinking Innateness* **MIT Press**.

- Empirical Risk Minimization (Wiki): *Wikipedia Entry for Empirical Risk Minimization*.
- Ensemble Averaging (Wiki): *Wikipedia Entry for Ensemble Averaging*.
- Ensemble Learning (Wiki): *Wikipedia Entry for Ensemble Learning*.
- Ester, M., H. P. Kriegel, J. Sander, and X. Xu (1996): A Density-Based Algorithm for Creating Clusters in Large Spatial Databases with Noise *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)* 226-231.
- Estivill-Castro, V. (2002): Why so many Clustering Algorithms – A Position Paper *ACM SIGKDD Explorations Newsletter* **4 (1)** 65-75.
- Everitt, B. S., S. Landau, M. Leese, and D. Stahl (2011): Miscellaneous Clustering Methods, in *Cluster Analysis, 5th Edition* **John Wilet and Sons**, Chichester, UK.
- Farber, I., S. Gunnemann, H. P. Kriegel, P. Kroger, E. Muller, E. Schubert, T. Seidl, and A. Zimek (2010): On Using Class Labels on Evaluation of Clusterings *MultiClust: Discovering, Summarizing, and Using Multiple Clusterings ACM SIGKIDD*.
- Feldman, V., V. Guruswamy, P. Raghavendra, and Y. Wu (2010): *Agnostic Learning of Monomials by Halfspaces is Hard* **arXiV**.
- Ferreira, C. (2001): Gene Expression Programming: A New Adaptive Algorithm for Solving Problems *Complex Systems* **13 (2)** 87-129.
- Ferreira, C. (2002a): *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence* **Angra do Heroismo** Portugal.
- Ferreira, C. (2002b): Mutation, Transposition, and Recombination: An Analysis of the Evolutionary Dynamics *Proceedings of the 6th Joint Conference on Information Sciences, 4th International Workshop on Frontiers in Evolutionary Algorithms* Research Traingle Park, NC.
- Ferreira, C. (2002c): Combinatorial Optimization by Gene Expression Programming: Inversion Revisited *Proceedings of the Argentine Symposium on Artificial Intelligence* Santa Fe, Argentina.

- Ferreira, C. (2006a): Automatically Defined Functions in Gene Expression Programming, in: *Genetic Systems Programming: Theory and Experiences, Studies in Computational Intelligence* (**13) Springer Verlag**.
- Ferreira, C. (2006b): Designing Neural Networks Using Gene Expression Programming, in: *Applied SoftComputing Technologies: The Challenge of Complexity* **Springer Verlag**.
- Ferreira, C. (2006c): *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence* **Springer Verlag**.
- Ferris, M. C., and T. S. Munson (2002): Interior-Point Methods for Massive Support Vector Machines *SIAM Journal on Optimization* **13 (3)** 783-804.
- Figueiredo, M. A. T., and A. K. Jain (2006): Unsupervised Learning of Finite Mixture Models **24 (3)** 381-396.
- Fisher, R. A. (1936): The Use of Multiple Measurements in Taxonomic Problems *Annals of Eugenics* **7** 179-188.
- Fisher, R. A. (1938): The Statistical Utilization of Multiple Measurements *Annals of Eugenics* **8** 376-386.
- Flury, R., and N.Shepard (2008): Bayesian Inference based only on Simulated Likelihood: Particle Filter Analysis of Dynamic Economic Models, *OFRC Working Papers Series 2008fe32* **Oxford Financial Research Centre**.Forgy, E. W. (1965): Cluster Analysis of Multi-variate Data: Efficiency versus Interpretability of Classifications *Biometrics* **21** 768-769.
- Foroutan, I. and J. Sklansky. (1987): Feature Selction for Automatic Classification of non-Gaussian Data *IEEE Transactions on Systems, Man, and Cybernetics* **17 (2)** 187-198.
- Fortmann-Roe, S. (2012): *Understanding the Bias-Variance Tradeoff*
- Fowlkes, E. B., and C. L. Mallows (1983): A Method for Comapring Two Hierarchical Clusterings **78** 553-569.
- Frahling, G., and C. Sohler (2006): A Fast k-Means Implementation using Coresets *Proceedings of the 22nd Annual Symposium on Computational Geometry (SoCG)*.

- Freund, Y., and R. E. Schapire (1997): A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting *Journal of Computer and System Sciences* **55 (1)** 119-139.
- Freund, Y., and R. E. Schapire (1998): Large Margin Classification using the Perceptron Algorithm *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT'98)* **ACM Press**.
- Freund, Y., and R. E. Schapire (1999): Large Margin Classification using the Perceptron Algorithm *Machine Learning* **37 (3)** 277-296.
- Frey, B. J. and D. Dueck. (2007): Clustering by Passing Messages between Data Points *Science* **315 (5814)** 972-976.
- Friedman, G. J. (1959): Digital Simulation of an Evolutionary Process *General Systems Yearbook* **4** 171-184.
- Friedman, J. H. (1989): Regularized Discriminant Analysis *Journal of the American Statistical Association* **84 (405)** 165-175.
- Frost, R., R. Hafiz, and P. Callaghan (2007): Modular and Efficient Top-down Parsing for Ambiguous Left-Recursive Grammars, *10th International Workshop on Parsing Technologies (IWPT), ACL-SIGPARSE* 109-120.
- Frost, R., R. Hafiz, and P. Callaghan (2008): Parsers Combinatiors for Ambiguous Left-Recursive Grammars, *10th International Conference on Practical Aspects of Declarative Languages (PADL), ACL-SIGPLAN* **4092** 167-181.
- Fruhwirth, R. (1987): Reversible-jump Markov Chain Monte-Carlo Computation and Bayesian Model Determination, *Nucl. Instrum. Methods* **A262**: 444-450.
- Fukushima, K. (1980): Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by a Shift in Position *Biol. Cybern.* **36** 193-202.
- Gagliardi, F. (2011): Instance-based Classifiers applied to Medical Databases *Artificial Intelligence in Medicine* **52 (3)** 123-139.
- Gallant, S. I. (1990): Perceptron-Based Learning Algorithms *IEEE Transactions on Neural Networks* **1 (2)** 179-191.
- Garcia Adeva, J. J., U. Cervino, and R. Calvo (2005): Accuracy and Diversity in Ensembles of Text Categorisers *CLEI Journal* **8 (2)** 1-12.

- Gashler, M., C. Giraud-Carrier, and T. Martinez (2008): Decision Tree Ensemble – Small Heterogenous is better than large Homogenous *The 7th International Conference on Machine Learning and Applications* 900-905.
- Geman, S., E. Bienenstock, and R. Doursat (1992): Neural Networks and the Bias/Variance Dilemma *Neural Computation* **4** 1-58.
- Ghahramani, Z. and M. I. Jordan (1997): Factorial Hidden Markov Models, *Machine Learning* **29 (2/3)**: 245-273.
- Gill, J. (2008): *Bayesian Methods: A Social and Behavioral Sciences Approach (2nd edition)*, **Chapman and Hall/CRC**, London.
- Glick, N. (1973): Sample Based Multinomial Classification *Biometrics* **29 (2)** 241-256.
- Gnanadesikan, R. (1977): *Methods for Statistical Data Analysis of Multivariate Observations* **Wiley**.
- Goodstein, J. R. (1982): The Italian Mathematics of Relativity *Centaurus* **26 (3)**: 241-261.
- Gold, E. (1967): Language Identification in the Limit, *Information and Control* **10 (5)**: 447-474.
- Golub, G. H., and C. F. Van Loan (1996): *Matrix Computations*, **Johns Hopkins Studies in Mathematical Sciences**, Baltimore.
- Gordon, N. J., D. J. Salmond, and A. F. M. Smith (1993): Novel Approach to Non-linear/Non-Gaussian State Space Estimation, *IEEE Proceeding F on Radar and Signal Processing* **140 (2)**: 107-113.
- Graves, A., and J. Schmidhuber (2009): Offline Handwriting Recognition in Multidimenstional Recurrent Neural Networks, in: *Advances in Neural Information Procesing Systems 22 (NIPS '22): Neural Information Processing Systems (NIPS) Foundation: Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta (eds.)*
- Graves, A., M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber (2009): A Novel Connectionist System for Improved Unconstrained Handwriting Recognition *IEEE Transactions on Pattern Analysis and Machine Recognition* **31 (5)**.

- Green, P. J. (1995): Reversible-jump Markov Chain Monte-Carlo Computation and Bayesian Model Determination, *Biometrika* **82 (4)**: 711-732.
- Greene, W. H. (1993): *Econometric Analysis 5th Edition* **Prentice Hall**.
- Griffiths, D. J. (1999): *Introduction to Electrodynamics 3rd Edition* **Saunders College Publishing**.
- Hale, J. (2006): Uncertainty about the rest of the Sentence, *Cognitive Science* **30 (4)**: 643-672.
- Hall, P., B. U. Park, and R. J. Samworth (2008): Choice of Neighbor Order in Nearest-Neighbor Classification *Annals of Statistics* **36 (5)** 2135-2152.
- Hamerley, G., and C. Elkan (2002): Alternatives to the k_means Algorithms that find better Clusterings *Proceedings of the 11th International Conference on Information and Knowledge Management*.
- Hamilton, J. (1994): *Time Series Analysis*, **Princeton University Press**.
- Hamilton, W. R. (1854-1855): On some Extensions of Quaternions *Philosophical Magazine* **7-9**: 492-499, 125-137, 261-269, 46-51, 280-290.
- Harrell, F. (2001): *Regression Modeling Strategies* **Springer-Verlag**.
- Harshman, R. A. (1970): Foundations of the PARAFAC Procedure: Models and Conditions for an Explanatory Multi-Model Factor Analysis *UCLA Working Papers in Phonetics* **16** 1-84.
- Har-Peled, S., D. Roth, and D. Zimak (2003): Constraint Classification for Multi-class Classification and Ranking, in: *Advances in Neural Information Procesing Systems 15: Proceedings of the 2002 Conference: B. Becker, S. Thrun, and K. Obermayer (eds.)*, **MIT Press**.
- Hart, P. E. (1968): The Condensed Nearest Neighbor Rule *IEEE Transactions on Information Theory* **18** 515-516.
- Hartigan, J. A. (1975): *Clustering Algorithms* **John Wiley & Sons**.
- Hartigan, J. A., and M. A. Wong (1979): Algorithm AS 136: A k-Means Clustering Algorithm *Journal of the Royal Statistical Society* **C28 (1)** 100-108.
- Hartigan, J. A. (1975): *Clustering Algorithms* **John Wiley & Sons**.

- Hashem, S. (1997): Optimal Linear Combnations of Neural Networks *Neural Computation* **10 (4)** 599-614.
- Haussler, D., M. Kearns, and R. E. Schapire (1994): Bounds on the Sample Complexity of Bayesian Learning using Information Theory and the VC Dimension *Machine Learning* **14** 83-113.
- Haykin, S. (1999): *Neural Networks – A Comprehensive Foundation 2nd Edition* **Prentice Hall** Upper Saddle River, NJ.
- Hazewinkel, M. (2001b): Tensor Density *Encyclopedia of Mathematics* **Springer**.
- He, X., R. S. Zemel, and M. A. Carreira-Perpinnan (2004): Multi-scale Condition Random Fields for Image Labeling, *IEEE Computer Society*.
- He, X., D. Cai, and P. Niyogi (2005): Tensor Subspace Analysis *Advances in Neural Information Procesing Systems 18 (NIPS).*
- Hernandez, D. (2013): The Man Behind the Google Brain: Andrew Ng and the Quest for the New AI *Wired 10 May 2013*.
- Hidden Markov Model (Wiki): *Wikipedia Entry for Hidden Markov Model*.
- Hierarchical Clustering (Wiki): *Wikipedia Entry for Hierarchical Clustering*.
- Higham, N. J. (2002): *Accuracy and Stability of Numerical Algorithms (2nd Ed),* **Society for Industrial And Applied Mathematics**, Philadelphia.
- Hinton, G., E. (2002): Training Products of Experts by Minimizing Contrastive Divergence *Neural Computation* **14** 1771-1800.
- Hinton, G., E., S. Osindero, and Y. Teh (2006): A Fast Learning Algorithm for Deep Belief Nets *Neural Computation* **18 (7)** 1527-1554.
- Hinton, G., E. (2007): Learning Multiple Layers of Representation *Trends in Cognitive Sciences* **11** 428-434.
- Hinton, G., E. (2009): Deep Belief Networks *Scholarpedia* **4 (5)** 5947.
- Hinton, G., E. (2010): A Practical Guide to Training Restricted Boltzmann Machines *Technical Report UTML TR 2010-003* **Department of Computer Science, University of Toronto**.
- Hinton, G., E., D. Leng, D. Yu, G. Dahl, A. Mohamed, M. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury (2012): Deep Neural Networks

for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups *IEEE Signal Processing Magazine* **11** 82-97.

- Hitchcock, F., L. (1927): The Expression of a Tensor or a Polyadic as a Sum of Products *Journal of Mathematics and Physics* **6** 164-189.
- Ho, T. (1995): Random Decision Forests *Proceedings of the 3rd International Conference on Document Analysis and Recognition* 278-282.
- Hochreiter, S. (1991): *Untersuchungen zu Dynamischen Neurolanen Netzen* Diploma Thesis **Institute fur Informatik, Technische Universitat Munchen**.
- Hochreiter, S., and J. Schmidhuber (1997): Long Short-Term Memory *Neural Computation* **9 (8)** 1735-1780.
- Hochreiter, S., Y. Bengio, P. Frasconi, and J. Schmidhuber (2001): Gradient Flow in Recurrent Nets: The Difficulty od Learning Long-Term Dependencies, in: *A Field Guide to Dynamic Recurrent Neural Networks: S. C. Kremer, and J. F. Kolen (eds.)* **IEEE Press**.
- Hoeting, J. A., D. Madigan, A. E. Raftery, and C. T. Volinsky (1999): Bayesian Model Averaging: A Tutorial *Statistical Science* **14 (4)** 382-401.
- Hoffmann, H. (2007): Kernel PCA for Novelty Detection *Pattern Recognition* **40** 863-874.
- Honarkhah, M., and J. Caers (2010): Stochastic Simulation of Patterns Using Distance-Based Pattern Modeling *Mathematical Geosciences* **42** 487-517.
- Horning, J. J. (1969): *A Study of Grammatical Inference*, PhD Thesis, **Stanford University**.
- Hosmer, D. W. and S. Lemeshow (2000): *Applied Logistic Regression 2nd Edition* **Wiley**.
- Hsu, C. W. and C. J. Lin (2002): A Comparison of Methods for Multiclass Support Vector Machines *IEEE Transactions on Neural Networks*.
- Hsu, C. W., C. C. Chang, and C. J. Lin (2003): *A Practical Guide to Support Vector Classification*.
- Huang, Z (1998): Extensions to the k-Means for Clustering Large Data Sets with Categorical Values *Data Mining and Knowledge Discovery* **2** 283-304.

- Huang, X., M. Jack, and Y. Ariki (1990): *Hidden Markov Models for Speech Recognition*, **Edinburgh University Press**.
- Huang, X., A. Acero, and H. W. Hon (2001): *Spoken Language Processing* **Prentice Hall**.
- Hubert, L., and P. Arabie (1985): Comparing Partitions *Journal of Classification* **2 (1)**.
- Huth, R., C. Beck, A. Philipp, M. Demuzere, Z. Ustrnul, M. Cahynova, J. Kysely, and O. E. Tveito (2008): Classification of Atmospheric Circulation Patterns – Recent Advances and Applications *Annals of NY Academy of Sciences* **1146** 105-152.
- Inaba, N., M. Katoh, and J. Imai (1994): Applications of Weighted Voronoi Diagrams and Randomization to Variance-Based k-Clustering *Proceedings of the 10th ACM Symposium on Computational Geometry* 332-339.
- Inoue, K., K. Hara, and K. Urahama (2009): Robust Principal Component Analysis *Proceedings of IEEE Conference on Computer Vision* 591-597.
- James, G (2003): Variance and Bias for General Loss Functions *Machine Learning* **51** 115-135.
- James, G., D. Witten, T. Hastie, and R. Tibshirani (2013): *An Introduction to Statistical Learning*
- Jazwinski, A. H. (1970): *Stochastic Processes and Filtering Theory*, **Academic Press**, New York.
- Jelinek, F., L. Bahl, and R. Mercer (1975): Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech, *IEEE Transactions on Information Theory* **21 (3)**: 250.
- Joachims, T. (1999): Transductive Inference for Text Classification using Supprt Vector Machines *Proceedings of the 1999 International Conference on Machine Learning (ICML 1999)* 200-209.
- Jordan, M. I., and C. M. Bishop (2004): *Neural Networks* **Chapman and Hall, CRC Press** Boca Raton FL.
- Julier, S. J., and J. K. Uhlmann (1997): A new Extension of the Kalman Filter to Nonlinear Systems, *Int. Symp.Aerospace/Defense Sensing, Simulation, and Controls* **3**.

- Kaelbling, L. P., M. Littman, and A. Cassandra (1998): Planning and Acting in Partially Observable Stochastic Domains, *Artificial Intelligence* **101**: 99-134.
- Kailath, T. (1968): An Innovation Approach to Least-Squares Estimation Part I: Linear Filtering in Additive White Noise, *IEEE Transactions on Automatic Control* **13 (6)**: 646-655.
- Kailing, K., H. P. Kriegel, and P. Kroger (2004): Density-Connected Subspace Clustering in High-Dimensional Data *SIAM International Conference on Data Mining (SDM '04)* 246-257.
- Kalman Filter (Wiki): *Wikipedia Entry for Kalman Filter*.
- Kanungo, T., D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu (2002): An Efficient k-Means Clustering Algorithm: Analysis and Implementation *IEEE Transaction on Pattern Recognition and Machine Intelligence* **24** 881-892.
- Kearns, M. (1988): *Thoughts on Hypethesis Boosting*
- Kearns, M., and U. Vazirani (1994): *An Introduction to Computational Learning* **MIT Press**.
- Kim, T. K., and R. Cipolla (2009): Canonical Correlation Analysis for Video Volume Tensors for Action Categorization and Detection *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31 (8)** 1415-1428.
- Kindermann, R., and J. L. Snell (1980): *Markov Random Fields and Their Applications* **American Mathematical Society**.
- Kitagawa, G. (1996): Monte-Carlo Filter and Smoother for Non-linear and Non-Gaussian State Space Models, *Journal of Computational and Graphical Statistics* **5 (1)**: 1-25.
- Klein, D., and C. Manning (2003): Accurate Unlexicalized Parsing, *Proceedings of the 41st Meeting of the Association for Computational Linguistics* 423-430.
- Klein, M. (1972): *Mathematical Thought from Ancient to Modern Times* **3** 1122-1127.
- Kleinberg, J. (2002): An Impossibility Theorem for Clustering *Proceedings of the Neural Information Processing Systems Conference*.

- Klivans, A. R., and R. A. Servedio (2006): Toward Attribute Efficient Learning of Decision Lists and Parities *Journal of Machne Learning Research* **7 (12)** 587-602.
- Kolda, T. G., and B. W. Bader (2009): Tensor Decompositions and Applications *SIAM Review* **51 (3)** 455-500.
- Kraskov, A., H. Stogbauer, R. G. Andrzejak, and P. Grassberger (2003): *Hierarchical Clustering based on Mutual Information* **arXiV**.
- Krauth, W., and M. Mezard (1987): Learning Algorithms with Optimal Stability in Neural Networks *J. of Physics A: Math. Gen.* **10** L745-L752.
- Kriegel, H. P., P. Kroger, and A. Zimek (2009): Clustering High-Dimensional Data: A Survey on Subspace Clustering, Pattern-based Clustering, and Correlation-Clustering *ACM Transactions on Knowledge Discovery from Data (TKDD)* **1 (3)** 1-58.
- Kriegel, H. P., P. Kroger, J. Sander, and A. Zimek (2011): Density-Based Clustering *WIREs Data Mining and Knowledge Discovery* **1 (3)** 231-240.
- Kroonenberg, P. M., and J. de Leeuw (1980): Principal Component Analysis of Three-Mode Data by means of Alternating Least Squares Algorithms *Psychometrika* **45** 69-97.
- Kuncheva, C., and L. Whitaker (2003): Measures of Diversity in Classifier Ensembles *Machine Learning* **51** 181-207.
- Kudo, M. and J. Sklansky. (2000): Comparison of Algorithms that select Features for Pattern Classifiers *Pattern Recognition* **33 (1)** 25-41.
- K-Means Clustering (Wiki): *Wikipedia Entry for K-Means Clustering*.
- K-Nearest Neighbors Algorithm (Wiki): *Wikipedia Entry for K-Nearest Neighbors Algorithm*.
- Lafferty, J., A. McCallum, and F. Pereira (2001): Conditional Random Fields: Probabilistic Models for segmenting and labeling Sequence Data, *Proc. 18th International Conf. On Machine Learning* **Morgan Kaufmann** 282-289.
- Lanchantin, P. and W. Pieczynski (2005): Unsupervised Restoration of Hidden Nonstationary Markov Chain using Evidential Priors, *IEE Transactions on Signal Processing* **53 (8)**: 3091-3098.

- Lang, S. (1972): *Differential Manifolds* **Addison-Wesley Publisher Co** Reading, Massachussetts.
- La Rochelle, H., D. Erhan, A. Courville, J. Bergstra, and Y Bengio (2007): An Empirical Evaluation of Deep Architectures on Problems with many Factors of Variation *Proceedings of the 24th International Conference on Machine Learning* 473-480.
- Lathauwer, L. D., B. D. Moor, and J. van de Walle (2000a): A Multilinear Singular Value Decomposition *SIAM Journal of Matrix Analysis and Applications* 2**1 (4)** 1253-1278.
- Lathauwer, L. D., B. D. Moor, and J. van de Walle (2000b): On the best Rank-1 and Rank-(R1, R2, …, RN) Approximation of Higher-order Tensors *SIAM Journal of Matrix Analysis and Applications* 2**1 (4)** 1324-1342.
- Lauritzen, S. L. (1981): Time Series Analysis in 1880 – A Discussion of the Contributions made by T. N. Thiele, *International Statistical Review* **49**: 319-333.
- Lauritzen, S. L. (2002): *Thiele: Pioneer in Statistics*, **Oxford University Press**.
- Lavergne, T., O. Cappe, and F. Yvon (2010): Practical Very Large Scale CRFs *Proceedings of the 48th Annual Meeting of the ACL* 504-513.
- LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1989): Back-propagation Applied to Handwritten Zip Code Recognition *Neural Computation* **1 (4)** 541-551.
- LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998): Gradient-based Learning applied to Document Reconition *Proceedings of the IEEE* **86 (11)** 2278-2324.
- Lee, Y., Y. Lin, and G. Wahba (2001): Multi-Category Support Vector Machines *Computing Science and Statistics* **33**.
- Lee, Y., Y. Lin, and G. Wahba (2004): Multi-Category Support Vector Machines – Theory and Application to the Classification of Microarray Data and Satellite Radiance Data *Journal of the American Statistical Association* **99 (465)** 67-81.
- Li, S. Z. (2009): *Markov Random Field Modeling in Image Analysis* **Springer**.Linear Discriminant Analysis (Wiki): *Wikipedia Entry for Linear Discriminant Analysis*.
- Lin, D., and X. Wu (2009): Phrase Clustering for Discriminative Learning *Annual Meeting of the ACL and IJCNLP* 1030-1038.

- Liou, D. R., J. W. Liou, and C. Y. Liou (2013): *Learning Behaviors of Perceprion* **iConcept Press**.
- Liu, Y., and X. Yao (1999): Ensemble Learning via Negative Correlation *Neural Networks* **12 (20)** 1399-1404.
- Liu, J., W. Wang, and F. Ma (2011): A Regularized Auxiliary Particle Filtering Approach for System State Estimation and Battery Life Prediction, *Smart Materials and Structures* **20 (7)**: 1-9.
- Little, M. A., and N. S. Jones (2011): Generalized Methods and Solvers for Piece-wise Constant Signals: Part I *Proceedings of the Royal Society A.*
- Llew, M., J. Baxter, P. Bartlett, and M. Frean (2000): Boosting Algorithms as Gradient Descent *Advances in Neural Information Processing Systems* (S. A. Solla, T. K. Leen, and K. R. Muller, editors) **12** 512-518.
- Lloyd, S. (1982): Least Squares Quantization in PCM *IEEE Transactions in Information Theory* **28 (2)** 129-137.
- Logistic Regression (Wiki): *Wikipedia Entry for Logistic Regression*.
- Long, P. M., and R. A. Servedio (2010): Random Classification Noise Defeats all Convex Potential Boosters *Machine Learning* **78 (3)** 287-304.
- Lu, H. (2013): Learning Canonical Correlations of Paired Tensor Sets vis Tensor-to-Vector Projection *Proceedings of the 23rd International Joint Conference on Artificial Intelligence* **Beijing**, China.
- Lu, H., H. L. Eng, M. Thida, and K. N. Plataniotis (2010): Visualization and Clustering of Crowd Video Content in MPCA Subspace *Proceedings of the 19th ACM Conference on Information and Knowledge Management* **Toronto**, ON, Canada.
- Lu, H., K. N. Plataniotis, and A. N. Venetsanopoulus (2008): MPCA: Multilinear Principal Component Analysis of Tensor Objects *IEEE Transactions on Neural Networks* **19 (1)** 18-39.
- Lu, H., K. N. Plataniotis, and A. N. Venetsanopoulus (2009a): Uncorrelated Multilinear DiscriminantAnalysis with Regularization and Aggregation for Tensor Object Recognition *IEEE Transactions on Neural Networks* **20 (1)** 103-123.

- Lu, H., K. N. Plataniotis, and A. N. Venetsanopoulus (2009b): Uncorrelated Multilinear Principal Component Analysis for Unsupervised Multilinear Subspace Learning *IEEE Transactions on Neural Networks* **20 (11)** 1820-1836.
- Lu, H., K. N. Plataniotis, and A. N. Venetsanopoulus (2009c): Boosting Discriminant Learners for Gait Recognition using MPCA Features *EURASIP Journal on Image and Video Processing*.
- Lu, H., K. N. Plataniotis, and A. N. Venetsanopoulus (2011): A Survey of Multilinear Subspace Learning for Tensor Data *Pattern Recognition* **44 (7)** 1540-1551.
- Ma, Y., H. Derksen, W. Hong, and J. Wright (2007): Segmentation of Multivariate Mixed Data via Lossy Data Coding and Compression *IEEE Transactions on Pattern Analysis and Machine Intelligence* **29 (9)** 1546-1562.
- MacKay, C. (2003): *Information Theory, Inference, and Learning Algorithms* **Cambridge University Press**.
- MacQueen, J. B. (1967): Some Methods for Classification and Analysis of Multi-variate Observations *Proceedings of the 5th Berkeley Symposium of Mathematical Sciences and Probability* **University of California Press** 281-297.
- Mahajan, M., P. Nimbhorkar, and K. Varadarajan (2009): The Planar k-Means Problem is NP-Hard *Lecture Notes in Computer Science* **5431** 274-285.
- Manning, C. D., P. Raghavan, and H. Schutze (2008): *Introduction to Information Retrieval* **Cambridge University Press**.
- Marion, J. B., and S. T. Thornton (1995): *Classical Dynamics of Particles and Systems 4th Edition* **Saunders College Publishing**.
- Mark, J., and M. A. Goldberg (2001): Multiple Regression Analysis and Mass Assessment: A Review of the Issues *The Appraisal Journal* 89-109.
- Markoff, J. (2012): *How Many Computers to Identify a Cat? 16,000*.
- Markov Chain Monte-Carlo (Wiki): *Wikipedia Entry for Markov Chain Monte-Carlo*.
- Markov Model (Wiki): *Wikipedia Entry for Markov Model*.
- Markov Random Field (Wiki): *Wikipedia Entry for Markov Random Field*.
- Martines, H., Y. Bengio, and G. N. Yannakakis (2013): Learning Deep Psychological Models of the Affect *IEEE Computational Intelligence* **8 (2)** 20.

- Martinez, A. M., and A. C. Kak (2001): PCA *versus* LDA *IEEE Transactions on Pattern Intelligence and Machine Learning* **23 (2)** 228-233.
- Masraliez, C. J., and R. D. Martin (1977): Robust Bayesian Estimation for the Linear Model and Robustifying the Kalman Filter, *IEE Transactions Automatic Control*.
- Matisko, P., and V. Havlena (2012): Optimality Tests and Adaptive Kalman Filter, *Proceedings of the 16th IFAC System Identification Symposium* **Brussels**.
- Maximum Entropy Markov Model (Wiki): *Wikipedia Entry for Maximum Entropy Markov Model*.
- McCallum, A., D. Freitag, and F. Pereira (2000): Maximum Entropy Markov Models for Information Extraction and Segmentation, *Proc. ICML 2000* 591-598.
- McDonald, R., K. Hall, and G. Mann (2010): Distributed Training Strategies for the Structured Perceptron *Association for Computational Linguistics* 456-464.
- McLachlan, G. J. (2004): *Discriminant Analysis and Statistical Pattern Recognition* **Wiley Series in Probability and Statistics**, New Jersey.
- McLaughlan, G. J. (1988): *Mixture Models: Inference and Applications to Clustering* **Dekker**.
- McLaughlan, G. J. (2000): *Finite Micture Models* **Wiley**.
- McWilliam, N., and K. Loh (2008): *Incorporating Multi-Dimensional Tail Dependencies in the Valuation of Credit Derivatives (Working paper)*.
- Mehryar, M., A. Rostamizadeh, and A. Talwalkar (2012): *Foundations of Machine Learning* **MIT Press**.
- Meila, M. (2003): Comparing Clusterings by the Variation of Information *Learning Theory and Kernel Machines* **2777** 173-187.
- Menard, S. W. (2002): *Applied Logistic Regression 2nd Edition* **SAGE**.
- Metaheuristic (Wiki): *Wikipedia Entry for Metaheuristic*.
- Metz, C. (2013): Facebook's "Deep Learning" Guru Reveals the Future of AI *Wired 12 December 2013*.
- Meyer, D., F. Leisch, and K. Hornik (2003): The Support Vector Machine Under Test *Neurocomputing* **55 (1-2)** 169-186.

- Mika, S., B. Scholkopf, A. Smola, K. R. Muller, M. Scholz, and G. Ratsch (1999): Kernel PCA and De-noising in Feature Spaces *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II* 536-542.
- Mikolov, T., M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur (2010): Recurrent Neural Network Based Language Model *Interspeech*.
- Milewski, R. and V. Govindaraju (2008): Binarization and Cleanup of Handwritten Text from Carbon Copy Medical Images *Pattern Recognition* **41 (4)** 1308-1315.
- Mills, P. (2011): Efficient Statistical Classification of Satellite Measurements, *International Journal of Remote Sensing* **32 (21)**.
- Minka, T. (2002): *Bayesian model averaging is not model combination*
- Minsky, M. L., and S. A. Papert (1969): *Perceptrons* **MIT Press** Cambridge, MA.
- Mirkes, E. M. (2011): kNN and Potential Energy **University of Leicester**.
- Mirkes, E. M. (2011): k-Means K-Medoids **University of Leicester**.
- Mitchell, M. (1996): *An Introduction to Genetic Algorithms* **MIT Press** Cambridge MA.
- Mitchell, T. (1997): *Machine Learning*
- Mixture Model (Wiki): *Wikipedia Entry for Mixture Model*.
- MNIST (2013): *THE MNIST DATABASE of handwritten digits*.
- Mohri, M., A. Rostamizadeh, and A. Talwalkar (2012): *Foundations of Machine Learning* **The MIT Press**.
- Mohri, M., and A. Rostamizadeh (2013): *Perceptron Mistake Bounds*.
- Montieth, K., J. Carroll, K. Seppi, and T. Martinez (2011): Turning Bayesian Model Averaging into Bayesian Model Combination *Proceedings of the International Joint Conference on Neural Networks IJCNN '11* 2657-2663.
- Moussouris, J. (1974): Gibbs' and Markov Random Systems with Constraints *Journal of Statistical Physics* **10 (1)** 11-33.
- Multi-linear Subspace Learning (Wiki): *Wikipedia Entry for Multi-linear Subspace Learning*.
- Multi-linear Principal Component Analysis (Wiki): *Wikipedia Entry for Multi-linear Principal Component Analysis*.

- Multinomial Logistic Regression (Wiki): *Wikipedia Entry for Multinomial Logistic Regression*.
- Mukherjee, S., P. Niyogi, T. Poggio, and R. Rifkin (2006): Learning Theory: Stability is sufficient for Generalization, and necessary and sufficient for Consistency of Empirical Risk Minimization *Advances in Computational Mathematics* **25** 161-193.
- Myers, J. H. and E. W. Forgy (1963): The Development of Numerical Credit Evaluation Systems *Journal of American Statistical Association* **58 (303)** 799-806.
- Naftaly, U., N. Intrator, and D. Horn (1997): Optimal Ensemble Averaging of Neural Networks *Networks: Computation in Neural Systems* **8 (3)** 283-296.
- Natarajan, B. K. (1991): *Machine Learning – A Theoretical Approach* **Morgan Kaufmann Publishers**.
- Newberg, L. (2009): Error Statistics of Hidden Markov Model and Hidden Boltzmann Model Results, *BMC Bioinformatics* **10**: 212.
- Ng, R., and J. Han (1994): Efficient and Effective Clustering Method for Spatial Data Mining *Proceedings of the 20th VLDB Conference* 144-155.
- Ng, A., and J. Dean (2012): *Building High-level Features Using Large Scale Unsupervised Learning* **arXiV**.
- Nigsch, F., A. Bender, B. van Buuren, J. Tissen, E. Nigsch, and J. B. Mitchell (2006): Melting-Point Prediction employing k-Nearest Neighbor Algorithms and Geneti Parameter Optimization *Journal of Chemical Information and Modeling* **46 (6)** 2412-2422.
- Novikoff, A. B. (1962): On Convergence Proofs of Perceptrons *Symposium on the Mathematical Theory of Automata* **12** 615-622.
- Oltean, M., and C. Grosan (2003): A Comparison of Several Genetic Programming Techniques *Complex Systems* **14 (4)** 285-314.
- *OpenCV Cascade Classifier*
- Opitz, D., and R. Maclin (1999): Popular Ensemble Methods: An Empirical Study *Journal of Artificial Intelligence Research* **11** 169-198.
- Pais, A. (2005): *Subtle is the Lord: The Science and the Life of Albert Einstein* **Oxford University Press**.

- Palei, S. K., and S. K. Das (2009): Logistic Regression Model for Prediction of Roof Fall Risks in Bord and Pillar Workings in Coal Mines: An Approach *Safety Science* **47** 88.
- Panagakis, W., C. Kotropoulus, G. R. Arce (2010): Non-negative multilinear Principal Component Analysis of Auditory Temporal Modulations for Music Genre Classification *IEEE Transactions on Audio, Speech, and Language Processing* **18 (3)** 576-588.
- Pardo, B. and W. Birmingham (2005): *Modeling Form for Online following of Musical Performances*, **AIAA-05 Proceedings**.
- Parsing (Wiki): *Wikipedia Entry for Parsing*.
- Particle Filter (Wiki): *Wikipedia Entry for Particle Filter*.
- Pattern Recognition (Wiki): *Wikipedia Entry for Pattern Recognition*.
- Pearlmutter, B. A., and R. Rosenfeld (1990): Chaitkin-Kolmogorov Complexity and Generalization in Neural Networks *Proceedings of the 1990 Conference in Advances in Neural Information Processing Systems* **3** 931.
- Peduzzi, P., J. Concato, E. Kemper, T. R. Holford, and A. R. Feinstein (1996): A Simulation Study of the Number of Events per Variable in Logistic Regression Analysis *Journal of Clinical Epidemiology* **49 (12)** 1373-1379.
- Penrose, R. (2007): *The Road to Reality* **Vintage Books**.
- Perceptron (Wiki): *Wikipedia Entry for Perceptron*.
- Perriere, G. and J. Thioulouse (2003): Use of Correspondence Discriminant Analysis to Predict the Sub-cellular Location of Bacterial Proteins *Computer Methods and Programs in Bio-medicine* **70** 99-105.
- Pieczynski, W. (2002): Triplet Markov Chains (Chaines de Markov Triplet) *Comptes Rendus de l'Academie des Sciences – Mathematique, Series I* **335 (3)**: 275-278.
- Pieczynski, W. (2007): Multi-sensor Triplet Markov Chains and Theory of Evidence, *International Journal of Approximate Reasoning* **45 (1)**: 1-16.
- Pitt, O., and N. Shepard (1999): Filtering via Simulation: Auxiliary Particle Filters, *Journal of American Statistical Association* **94 (446)**: 590-591.

- Platt, J., N. Christianini, and J. Shawe-Taylor (2000): Large Margin DAGs for Multiclass Classification *Advances in Neural Information Processing Systems* **MIT Press**.
- Poggio, T., L. Rosasco, C. Ciliberto, C. Frogner, and G. Evangelopoulus (2012): *Statistical Learning Theory and Applications*
- Pollard, D. (1990): *Empirical Processes: Theory and Applications* **NSF-CBMS Regional Conference Series in Probability and Statistics Volume 2**.
- Polikar, R. (2006): Ensemble based Systems in Decision Making *IEEE Circuits and Systems Magazine* **6 (3)** 21-45
- Press, A., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (2007): *Numerical Recipes – The Art of Scientific Computing 3rd Edition* **Cambridge University Press**, New York.
- Probably Approximately Correct Learning (Wiki): *Wikipedia Entry for Probably Approximately Correct Learning*.
- Quadratic Classifier (Wiki): *Wikipedia Entry for Quadratic Classifier*.
- Quartz, S. R., and T. J. Sejnowski (1997): The Neural Basis for Cognitive Development: A Constructivist Manifesto *Behavioral and Brain Sciences* **20 (4)** 537-556.
- Rabiner, L. R. (1989): A Tutorial on Hidden Markov Models and selected Applications in Speech Recognition, *Proceedings of the IEEE* **77 (2)**: 257-286.
- Raina, R., A. Madhavan, and A. Ng (2009): Large Scale Deep Unsupervised Learning Using Graphics Processors *Proceedings of the 26th International Conference on Machine Learning*.
- Rajamani, M. R. (2007): *Data-based Techniques to improve State Estimation in Model Predictive Control*, PhD Thesis, **University of Wisconsin-Madison**.
- Rajamani, M. R., and J. B. Rawlings (2009): Estimation of the Disturbance Structure from Data using Semi-definite Programming and Optimal Weighting, **45**: 142-148.
- Rand, W. M. (1971): Objective Criteria for the Evaluation of Clustering Models *Journal of the American Statistical Association* **66 (336)** 846-850.

- Rao, C. R. (1948): The Utilization of Multiple Measurements in Problems of Biological Classification *Journal of the Royal Statistical Society, Series B* **10 (2)** 159-203.
- Rao, C. R. (1952): *Advanced Statistical Methods in Multivariate Analysis* **Wiley**.
- Rauch, H. E., F. Tung, and C. T. Striebel (1965): Maximum Likelihood Estimates of Linear Dynamic Systems, *Automatica AIAA* **3 (8)**: 1445-1450.
- Rechenberg, I. (1973): *Evolutionsstrategie* **Holzman-Froboog** Stuttgart.
- Reich, K. (1994): *Die Entwicklung des Tensorkalkuls, Science Networks Historical Studies, vol. 11* **Birkhauser**.
- Ridella, S., S. Rovetta, and R. Zunino (1997): Circular Back-propagation Networks for Classification *IEEE Transactions on Neural Networks* **8 (1)** 84-97.
- Riesenhuber, M., and T. Poggio (1999): Hierarchical Models of Object Recognition in Cortex *Narute Neuroscience* **11** 1019-1025.
- Rivest, L. R. (1987): Learning Decision Lists *Machine Learning* **2 (3)** 229-246.
- Robert, C. P. and G. Casella (2004): *Monte-Carlo Statistical Methods (2nd edition)*, **Springer**, New York.
- Rokach, L. (2010): Ensemble-based Classifers *Artificial Intelligene Review* **33 (1-2)** 1-39.
- Rosasco, L., E. D. Vito, Rosenblatt, F. (1957): The Perceptron – A Perceiving and Recognizing Automaton *Report 85-460-1* **Cornell Aeronautical Laboratory**.
- Rosenblatt, F. (1958): The Perceptron – A Probabilistic Model for Information Storage and Organization in the Brain *Psychological Review* **65 (6)** 386-408.
- Rosenblatt, F. (1962): *Principles of Neurodynamics* **Spartan Books** Washington DC.
- Roweis, S., and Z. Ghahramani (1999): A unified Review of Linear Gaussian Models, *Neural Comput.* **11 (2)**: 305-345.
- Roy, S., and D. K. Bhattacharyya (2005): An Approach to Finding Embedded Clusters Using Density Based Techniques *LNCS* **3816** 523-535.
- Rue, H., and L. Held (2005): *Gaussian Markov Random Fields: Theory and Applications* **CRC Press**.

- Sahu, A., G. Runger, and D.Apley (2011): Image denoising with multi-phase kernel principal component approach and an Ensemble Version *IEEE Applied Imagery Pattern Recognition Workshop* 1-7.
- Sainath, T. N., A. Mohamed, B. Kingsbury, and B. Ramabhadran (2013): Deep Convolutional Networks for LVCSR *Proceedings ICASSP*.
- Samworth, R. J. (2012): Optimal Weighted Nearest Neighbor Classifiers *Annals of Statistics* **40 (5)** 2733-2763.
- Sarawagi, S. and W. W. Cohen (2005): Semi-Markov Conditional Random Fields for Information Extraction, in *Advances in Neural Information Processing Systems 17 (L. K. Saul, Y. Weiss, and L. Boutto (eds.))* **MIT Press: Cambridge, MA** 1185-1192.
- Satish, L. and B. I. Gururaj (2003): *Use of Hidden Markov Models for Partial Discharge Pattern Classification* *IEEE Transactions on Dielectrics and Electrical Insulation*.
- Sculley, D. (2010): Web-scale k-Means Clustering *Proceedings of the 19th International Conference on the World Wide Web*.
- Schapire, R. E. (1990): The Strength of Weak Learnability *Machine Learning* **5 (2)** 197-227.
- Schmidhuber, J. (1992): Learning Complex, Extended Sequences Using the Principle of History Compression *Neural Computation* **4** 234-242.
- Schmidhuber, J. (2013): *My First Deep Learning System of 1991 + Deep Learning Timeline 1962-2013* **arXiV**.
- Schwenker, F., H. A. Kestler, and G. Palm (2001): Three Learning Phases for Radial-Basis-Function Networks *Neural Networks* **14** 439-458.
- Segal, I. E. (1956): Tensor Algebras over Hilbert Spaces I *Transactions of the American Mathematical Society* **81 (1)**: 106-134.
- Sequence Labeling (Wiki): *Wikipedia Entry for Sequence Labeling*.
- Scholkopf, B., A. Smola, and K. R. Muller (1996): Nonlinear Component Analysis as a Kernel Eigenvalue Problem.
- Settles, B. (2004): Bio-medical Named Entity Recognition using Conditional Random Fields and Rich Feature Sets, *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications* 104-107.

- Sha, F. and F. Pereira (2003): *Shallow Parsing with Conditional Random Fields*.
- Sharpe, R. W. (1997): *Differential Geometry: Cartan's Generalization of Klein's Erlangen Problem* **Springer-Verlag**, Berlin, New York.
- Shasha, D. (2004): *High-Performance Discovery in Time Series* **Springer** Berlin.
- Shaw, B., and T. Jebara (2009): Structure Preserving Embedding *Proceedings of 26th Annual International Conference on Machine Learning, ACM 2009*.
- Shen, J. (2006): A Stochastic Variational Model for Soft Mumford-Shah Segmentation *International Journal of Biomedical Imaging* **2006** 2-16.
- Shrager, J., and M. H. Johnson (1995): Timing in the Development of the Cortical Function: A Computational Approach, in: *B. Julesz and I. Kovacs (eds.): Maturational Windows and Adult Cortical Plasticity*.
- Shrager, J., and M. H. Johnson (1996): Dynamica Plasticity influences the Emergence of Function in a Simple Cortical Array *Neural Networks* **9 (7)** 1119-1129.
- Sibson, R. (1973): SLINK: An Optimally Efficient Algorithm for the Single-Link Cluster Method *The Computer Journal* **16 (1)** 30-34.
- Sidhu, G., and B. Caffo (2014): Exploiting Pitcher Decision-making using Reinforcement Learning *Annals of Applied Statistics* **8 (2)** 926-955.
- Sill, J., G. Takacs, L. Mackey, and D. Lin (2009): *Feature-Weighted Linear Stacking* **arXiV**
- Smith, M. R., and T. Martinez (2011): Improving Classification Accuracy by Identifying and Removing Instances that should be Misclassified *Proceedings of International Joint Conference on Neural Networks* 2690-2697.
- Smolensky, P. (1986): Information Processing in Dynamical System: Foundations of Harmony Theory, in: *Explorations of the Microstructure of Cognition: D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, Parallel Distributed Processing* **1** 194-281.
- Smyth, P., and D. H. Wolpert (1999): Linearly Combining Density Estimators via Stacking *Machine Learning Journal* **36** 59-83.
- Sollich, P., and A. Krogh (1996): Learning with Ensembles: How Over-fitting can be Useful *Advances in Neural Information Processing Systems* **8** 190-196.

- Starner, T. and A. Pentland (1995): Real Time American Sign Language Visual Recognition from Video using Hidden Markov Models, *Master's Thesis*, **MIT**.
- Statistical Classification (Wiki): *Wikipedia Entry for Statistical Classification*.
- Statistical Learning Theory (Wiki): *Wikipedia Entry for Statistical Learning Theory*.
- Steinhaus, H. (1957): Sur le Division des Corps Materiels en Parties *Bull. Acad. Polon. Sci.* **4 (12)** 801-804.
- Stigler, J., F. Ziegler, A. Gieseke, J. C. M. Gebhardt, and M. Rief (2011): The Complex Folding Network of Single Calmodulin Molecules, *Science* **334 (6055)**: 512-516.
- Stochastic Context Free Grammar (Wiki): *Wikipedia Entry for Stochastic Context Free Grammar*.
- Stramer, O., and R. Tweedie (1999): Langevin-Type Models II: Self-targeting Candidates for MCMC Algorithms, *Methodology and Computing in Applied Probability* **1 (3)**: 307-328.
- Strano, M., and B. M. Colosimo (2006): Logistic Regression Analysis for Experimental Determination of Forming Limit Diagrams *International Journal of Machine Tool and Manufacture* **46 (6)**.
- Stratonovich, R. L. (1959a): Optimum non-linear Systems which bring about a separation of a signal with constant Parameters from Noise, *Radiofizika* **2 (6)**: 892-901.
- Stratonovich, R. L. (1959b): On the theory of Optimal non-linear Filtering of Random Functions, *Theory of Probability and its Applications* **4**: 223-225.
- Stratonovich, R. L. (1960a): Application of the Markov Processes to Optimal Filtering, *Radio Engineering and Electronic Physics* **5 (11)**: 1-19.
- Stratonovich, R. L. (1960b): Conditional Markov Processes, *Theory of Probability and its Applications* **5**: 156-178.
- Strid, I., and K. Walentin (2009): Block Kalman Filtering for large-scale DSGE Models, *Computational Economics* **33 (3)**: 277-304.
- Supervised Learning (Wiki): *Wikipedia Entry for Supervised Learning*.
- Support Vector Machine (Wiki): *Wikipedia Entry for Support Vector Machine*.

- Sutton, C. and A. McCallum (2010): *Introduction to Conditional Random Fields*.
- Suykens, J. A. K., and J. P. L. Vandewalle (1999): Least Squares Support Vector Machine Classifiers *Neural Processing Letters* **9 (3)** 293-300.
- Szekely, G. J., and M. L.Rizzo (2005): Hierarchical Clustering vis Joint Between-Within Distances: Extending Ward's Minimum Variance Method *Journal of Classification* **22** 151-183.
- Tao, D., X. Li, X. Wu, and S. J. Maybank (2007): General Tensor Discriminant Analysis and Gabor Features for Gait Recognition *IEEE Transactions on Pattern Analysis and Machine Intelligence* **29 (10)** 1700-1715.
- Tarter, M. E. (1993): *Model Free Curve Estimation* **Chapman & Hall**.
- Taylor, C. (1997): Classification and Kernel Density Estimation *Vistas in Astronomy* **41 (3)** 417-441.
- Tensor (Wiki): *Wikipedia Entry for Tensor*.
- Terrell, D. G., and D. W. Scott (1992): Variable Kernel Density Estimation *Annals of Statistics* **20** 1236-1265.
- Thornton, C. L. (1976): Triangular Covariance Factorizations for Kalman Filtering, *NASA Technical Memorandum.* **33-798**.
- TIMIT (1993): Acoustic-Phonetic Continuous Speech Corpus *Linguistic Data Consortium* Philadelphia.
- Ting, J. A., S. Vijayakumar, and S. Schaal (2011): Locally Weighted Regression for Control in *Encyclopedia of Machine Learning* (eds. C. Sammut and G. I. Webb) **Springer** 615.
- Titterington, D. M., A. F. M. Smith, and U. E. Makov (1986): *Statistical Analysis of Finite Mixture Distributions* **Wiley**.
- Touissant, G. T. (2005): Geometric Proximity Graphs for improving Nearest Neighbor Methods in Instance-based Learning and Data-Mining *International Journal of Computational Geometry and Applications* **15 (2)** 101-150.
- Toutanova, K., and C. D. Manning (2000): Enriching the Knowledge Sources used in a Maximum-Entropy Part-of-Speech Trigger, *Proc. J. DIGDAT Conf. On Empirical Methods in NLP and Very Large Corpora (EMNLP/VLC-2000)* 63-70.
- Tree Structure (Wiki): *Wikipedia Entry for Tree Structure*.

- Tryon, R. C. (1939): *Cluster Analysis: Correlation Factor and Orthometric (Factor) Analysis for the Isolation of Unities in Mind and personality* **Edwards Brothers**.
- Tucker, L. D. (1966): Some Mathematical Notes on Three-Mode Factor Analysis *Psychometrika* **31 (3)** 279-311.
- Unsupervised Learning (Wiki): *Wikipedia Entry for Unsupervised Learning*.
- Utgoff, P. E., D. A. Stracuzzi (2002): Many Layered Learning *Neural Computation* **14** 2497-2529.
- Valiant, L. (1984): A Theory of the Learnable *Communications of the ACM* **27 (11)** 1134-1142.
- Van der Vaart, A. W., and J. A. Wellner (2000): *Weak Convergence and Empirical Processes with Applications to Statistics 2nd Edition* **Springer**.
- Vapnik, V. and A. Chervonenkis (1971): On the Uniform Convergence of relative Frequencies of Events to their Probabilities *Theory of Probability and its Applications* **16 (2)** 264-280.
- Vapnik, V. N. (1989): *Statistical Learning Theory* **Wiley Interscience**.
- Vapnik, V. N. (2000): *The Nature of Statistical Learning Theory, 2nd Edition* **Springer Verlag**.
- Vapnik Chervonenkis Theory (Wiki): *Wikipedia Entry for Vapnik Chervonenkis Theory*.
- Variable Kernel Density Estimation (Wiki): *Wikipedia Entry for Variable Kernel Density Estimation*.
- Vasilescu, M. A. O., and D. Terzopoulus (2007): Multilinear Projection for Appearance-based Recognition in the Tensor Framework *IEEE 11th International Conference on Computer Vision* 1-8.
- Vaswani, N. (2008): Kalman Filtered Compressed Sensing, *15th International Conference on Image Processing*.
- Vattani, A. (2011): k-Means requires exponentially many Iterations even in the Plane *Discrete and Computational Geometry* **45 (4)** 596-616.
- Venables, W. N., and B. D. Ripley (2002): *Modern Applied Statistics with S 4th Edition* **Springer Verlag**.

- Vijayakumar, S. (2007): *The Bias-Variance Tradeoff*.
- Voigt, W. (1898): *Die Fundamentalen Physikalicshen Eigenschaften der Krystalle in Elementarer Derstellung* **Von Veit**, Leipzig.
- Wallace, D. L. (1983): Comment *Journal of the American Statistical Association* **78** 569-579.
- Wan, E. A., and R. van der Merwe (2000): *The Unscented Kalman Filter for Nonlinear Estimation*.
- Wang, J. (2001): Generating Daily Changes in Market Variables Using a Multivariate Mixrure of Normal Distributions *Proceedings of the 33rd Winter Conferencve on Simulation, IEEE Computer Society* 283-289.
- Ward, J. H. (1963): Hierarchical Grouping to Optimize and Objective Function *Journal of the American Statistical Association* **58 (301)** 236-244.
- Werbos, P. (1974): *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences* PhD Thesis **Harvard University**.
- Wetcher-Hendricks, D. (2011): *Analyzing Quantitative Data: An Introduction for Social Researchers* **Wiley**.
- Wheeler, J. A., C. Misner, and K. S. Thorne (1973): *Gravitation* **W. H. Freeman & Co**.
- Wolpert, D. (1992): Stacked Generalization *Neural Networks* **5 (2)** 241-259.
- Wolpert, D. H., and RW. G. Macready (1999): An Efficient Method to Estimate Bagging's Generalization Error *Machine Learning Journal* **35** 41-55.
- Wolpert, D. H. (2001): *The Supervised Learning No Free Lunch Theorems*.
- Wolpert, D. M. (1996): Forward Models for Physiological Motor Control *Neural Networks* **9 (8)**: 1265-1279.
- Wong, W. and M. Stamp (2006): Hunting for Metamorphic Engines, *Journal in Computer Virology* **2 (3)**: 211-229.
- Xu, L., and M. I. Jordan (1996): On the Convergence Properties of the EM Algorithm for Gaussian Mixtures *Neural Computation* **8 (1)** 129-151.

- Yan, S., D. Xu, Q. Yang, L. Zhang, X. Tang, and H. J. Zhang (2005): Discriminant Analysis with Tensor Representation *IEEE Conference on Computer Vision and Pattern Recognition* **I** 526-532.
- Yu, G. (2012): Solving Inverse Problems with Piecewise Linear Estimators: From Gaussian Mixture Models to Structured Sparsity *IEEE Transactions on Image Processing* **21 (5)** 2481-2499.
- Yu, H., and J. Yang (2001): A Direct LDA Algorithm for High-Dimensional Data - with Application to Face Recognition *Pattern Recognition* **34 (10)** 2067-2069.
- Yuan, Y., and M. J. Shaw (1995): Induction of Fuzzy Decision Trees *Fuzzy Sets and Systems* **69** 125-139.
- Zadeh, R. B., and S. Ben-David (2009): A Uniqueness Theorem for Clustering *Proceedings of the Conference of Uncertainty in Artificial Intelligence*.
- Zenko, B. (2004): Is Combining Classifiers better than selecting the best One *Machine Learning* 255-273.
- Zha, H., C. Ding, M. Gu, X. He, and H. D. Simon (2001): Spectral Relaxation for K-Means Clustering *Neural Information Processing Systems (NIPS 2001)* **14** 1057-1064.
- Zhang, T., R. Ramakrishnan, and M. Livny (1996): An Efficient Data Clustering Method for Very Large Databases *Proceedings of International Conference on Management of Data* 103-114.
- Zhang, W., X. Wang, D. Zhao, and X. Tang (2012): Graph Degree Linkage: Agglomerative Clustering on a Directed Graph **arXiV**.
- Zhang, W., D. Zhao, and X. Tang (2013): Agglomerative clustering via maximum incremental path integral *Pattern Recognition* **46 (11)** 3056-3065.
- Zhao, D., and X. Tang (2008): Cyclizing Clusters viz Zeta Function of a Graph *Advances in Neural Information Processing Systems*.
- Zimek, A. (2008): Correlation Clustering.