Machine Learning & Predictive Analytics

Class 2

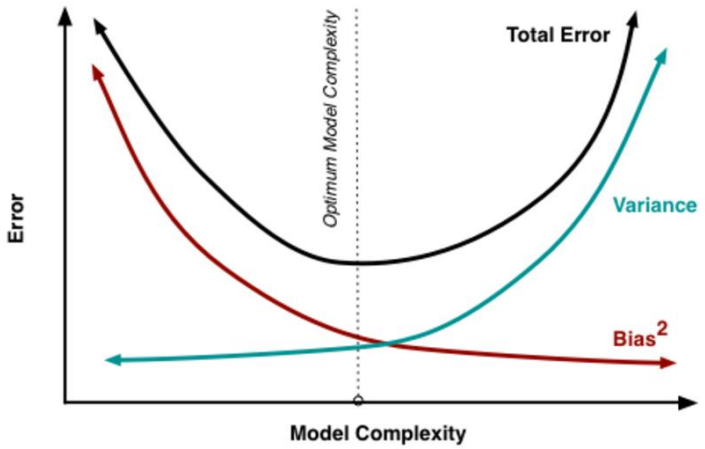
Arnab Bose, Ph.D.

MSc Analytics

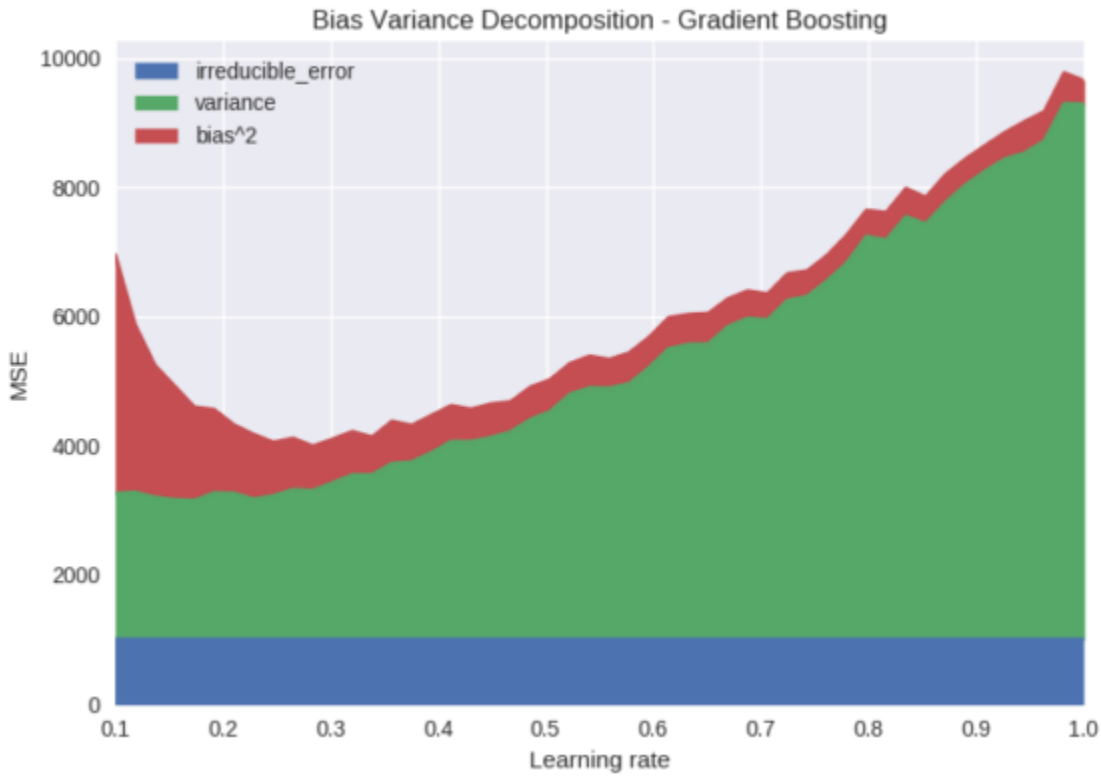
University of Chicago

Bias Variance Tradeoff

1. Bias – low capacity model that does not fit the training data well and has high training error – underfit
2. Variance – high capacity model that "learns" the training data too well and has high generalization error – overfit
3. Increasing (decreasing) a model's capacity / complexity / degrees-of-freedom reduces (increases) bias + increases (decreases) variance
4. NOTE – Mean Squared Error cost function incorporates both



Bias-Variance Tradeoff



https://devblogs.nvidia.com/bias-variance-decompositions-using-xgboost/

Bayes Classifier and Bayes Error

1. Bayes Error is the lowest possible error rate.

2. Bayes Classifier produces the lowest possible error rate, aka Bayes Error.

3. Bayes Classifier assumes knowledge of the conditional distribution of response given predictors (*posterior probability*) – i.e. we know everything these is to know, which does not happen in reality.

4. Bayes Classifier is the gold standard that classifiers try to approximate.

5. Bayes Error – another way to think about it – assume utopia where an oracle knows the true probability distribution that generates the data; there is still some noise in the distribution that causes test error = Bayes Error.

Bayesian Networks – Bayes Law

Based on evidence, estimate a "degree of belief" for possible outcomes

$$P(A|B) = \frac{P(B\ |A)\ * P(A)}{P(B)}$$

Prior, Posterior and Likelihood

$$P(A|B) = \frac{P(B\ |A)\ * P(A)}{P(B)}$$

Element	Terminology
$P(A)$	Prior
$\frac{P(B\ \|A)}{P(B)}$	Support of B for A
$P(A\|B)$	Posterior

$$Posterior = \frac{P(B\ |A)}{P(B)} * Prior$$

Prior, Posterior and Likelihood

$$P(A|B) = \frac{P(B\ |A)\ * P(A)}{P(B)}$$

Replace A with Hypothesis (H) and B with Observed data (O), $P(O) = 1$

Element	Terminology
$P(H)$	Prior
$P(O\|H)$	Likelihood
$P(H\|O)$	Posterior

$$Posterior = Likelihood * Prior$$

Bayesian Networks Characteristics

1. Bayesian networks aka directed graphical model aka belief network
2. Most variables influence each other
3. Most variables do not influence each other directly
4. Describe influence with a graph
5. Edges represent direct influence
6. Paths represent indirect influence
7. Computational and statistical savings come from omissions of edges

Bayesian Networks Example

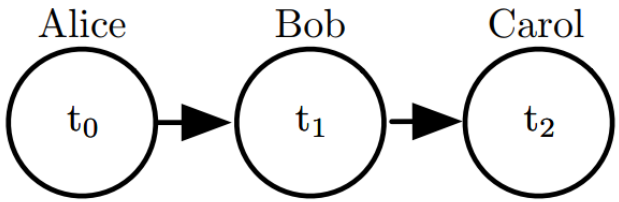


Figure 16.2

$$p(\mathbf{x}) = \Pi_i p(\mathrm{x}_i \mid Pa_{\mathcal{G}}(\mathrm{x}_i)). \tag{16.1}$$

$$p(\mathrm{t}_0, \mathrm{t}_1, \mathrm{t}_2) = p(\mathrm{t}_0)p(\mathrm{t}_1 \mid \mathrm{t}_0)p(\mathrm{t}_2 \mid \mathrm{t}_1). \tag{16.2}$$

Directed models work best when influence clearly flows in one direction

(Goodfellow 201

https://keras.io/examples/keras_recipes/bayesian_neural_networks/

Linear Algebra - Eigendecomposition

- Eigenvector and eigenvalue:

$$\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v}. \tag{2.39}$$

- Eigendecomposition of a diagonalizable matrix:

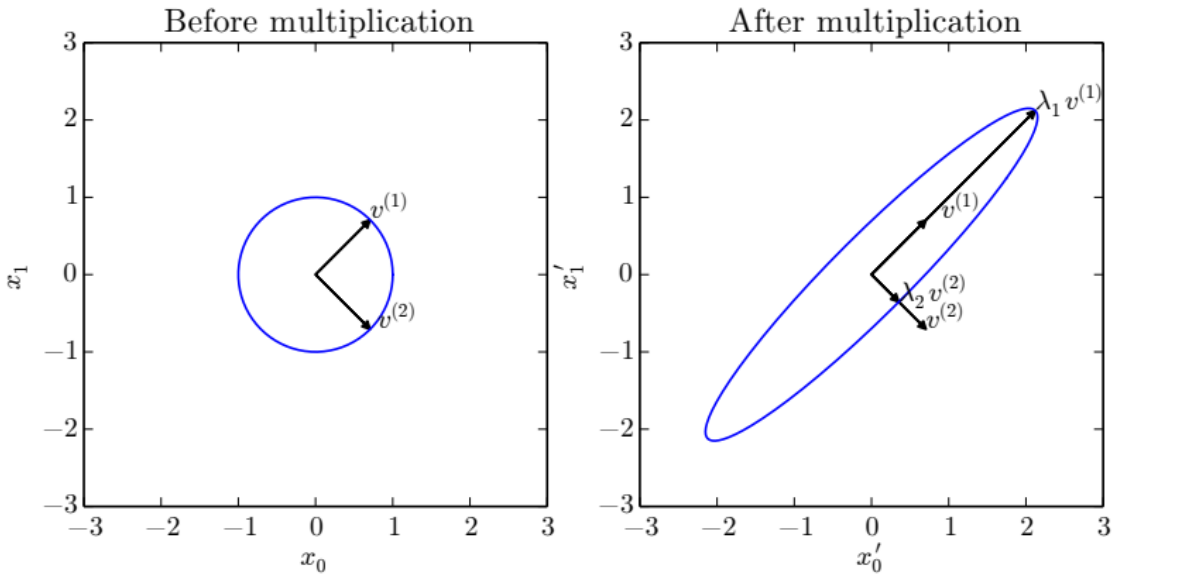
$$\boldsymbol{A} = \boldsymbol{V}\text{diag}(\boldsymbol{\lambda})\boldsymbol{V}^{-1}. \tag{2.40}$$

- Every real symmetric matrix has a real, orthogonal eigendecomposition:

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{\top} \tag{2.41}$$

(Goodfellow 2016)

Linear Algebra – Effect of Eigenvalues



(Goodfellow 2016)

Linear Algebra – Singular Value Decomposition (SVD)

- Similar to eigendecomposition
- More general; matrix need not be square

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\top}. \tag{2.43}$$

(Goodfellow 2016)

Tensors

- A tensor is an array of numbers, that may have
  - zero dimensions, and be a scalar
  - one dimension, and be a vector
  - two dimensions, and be a matrix
  - or more dimensions.

(Goodfellow 2016)

The Gradient

Take a function $f: \mathbb{R}^m \rightarrow \mathbb{R}$ that inputs matrix $A \in \mathbb{R}^m$ of size m and returns a real value.

The gradient of f is a matrix of partial derivatives, defined as:

Note that the size of $\nabla_A f(A)$ is always the same as the size of A. So if, in particular, A is just a vector $x \in \mathbb{R}^n$,

$$\nabla_x f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}.$$

It is very important to remember that the gradient of a function is *only* defined if the function is real-valued, that is, if it returns a scalar value. We can not, for example, take the gradient of $Ax, A \in \mathbb{R}^{n \times n}$ with respect to x, since this quantity is vector-valued.

The Hessian

Take a function $f: \mathbb{R}^m \rightarrow \mathbb{R}$ that inputs a vector x $\in \mathbb{R}^n$ and returns a real value.

The Hessian matrix (size $n \times n$) with respect to *x*:

$$\nabla_x^2 f(x) \in \mathbb{R}^{n \times n} = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}$$

In other words, $\nabla_x^2 f(x) \in \mathbb{R}^{n \times n}$, with

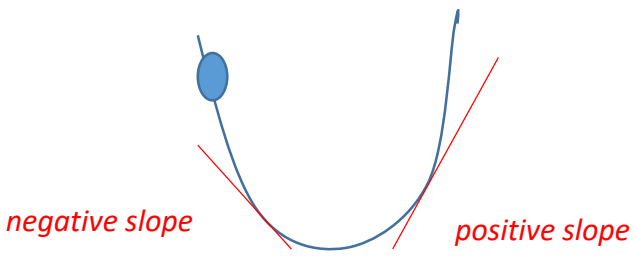
$$(\nabla_x^2 f(x))_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}.$$

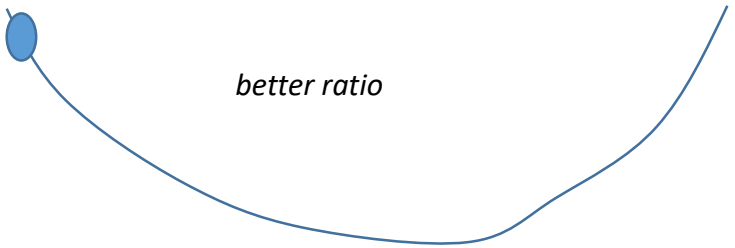
Note that the Hessian is always symmetric, since

$$\frac{\partial^2 f(x)}{\partial x_i \partial x_j} = \frac{\partial^2 f(x)}{\partial x_j \partial x_i}.$$

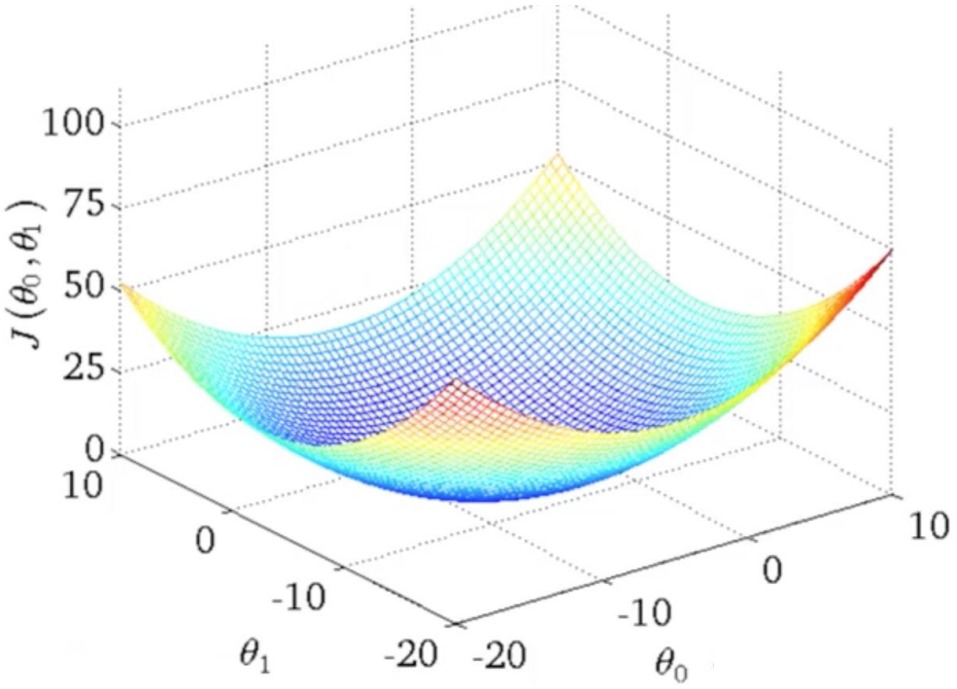
Gradient Descent and Learning Rate

1. Calculate the gradient using the first derivative of the cost function evaluated with tensor of the model parameters.
2. Advantage of gradient descent is not calculate Hessian: $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$. α is the learning rate
3. The reason for the "-" negative sign in the calculation is that you want to move in the opposite direction to the slope: for a positive (negative) slope, the function moves towards the minima if you move in the opposite to the direction of the slope, i.e. move right (left).
4. Disadvantage – can suffer from slow convergence, key is how to set the learning rate
5. Note – the maximum error reduction depends on the ratio of the gradient to the curvature.

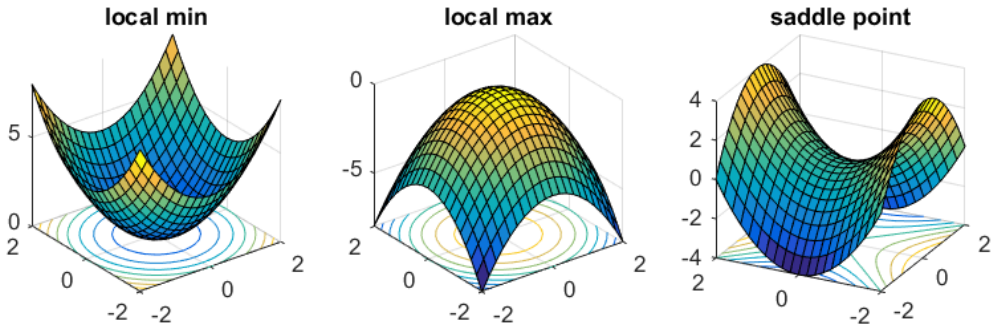




Convex Cost Surface



Second order optimization, Hessian and Eigendecomposition

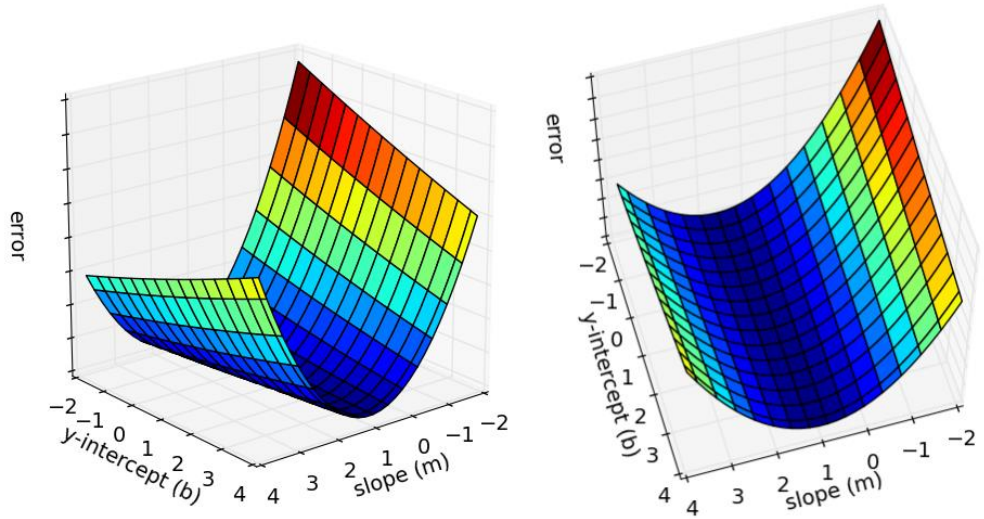


1. If Hessian is calculated, then use Newton's method that takes care of convergence using a learning rate determined from Hessian.
2. Assuming cost function is well approximated by a quadratic, the sensible learning rate is $\frac{1}{\lambda_{max}}$, where λ_{max} is the maximum eigenvalue of the Hessian.
3. Since Hessian is real and symmetric, can be decomposed into real eigenvalues and orthogonal basis of eigenvectors. They describe the directions of the principal curvature and the amount of curvature in each direction.
4. Second derivative of the cost function is in a specific direction of unit vector d given by $d^T Hd$ that measures the curvature.
5. When d is an eigenvector then second derivative is given by the corresponding eigenvalue.
6. Max(min) eigenvalue determine max(min) second derivative.
7. When d is not an eigenvector then directional second derivative is a weighted average of all eigenvalues.

Linear Regression with Gradient Descent

Generic optimization algorithm to minimize a cost function

1. Measures the local gradient of the cost function with regards to parameter vector and moves in the direction of descending gradient.
2. Rate of movement is determined by the learning rate hyperparameter
    1. If too small, takes a long time to converge.
    2. If too large, algorithm jumps around without finding a good solution.
3. The MSE for Linear Regression is a convex function – i.e. one global minimum.



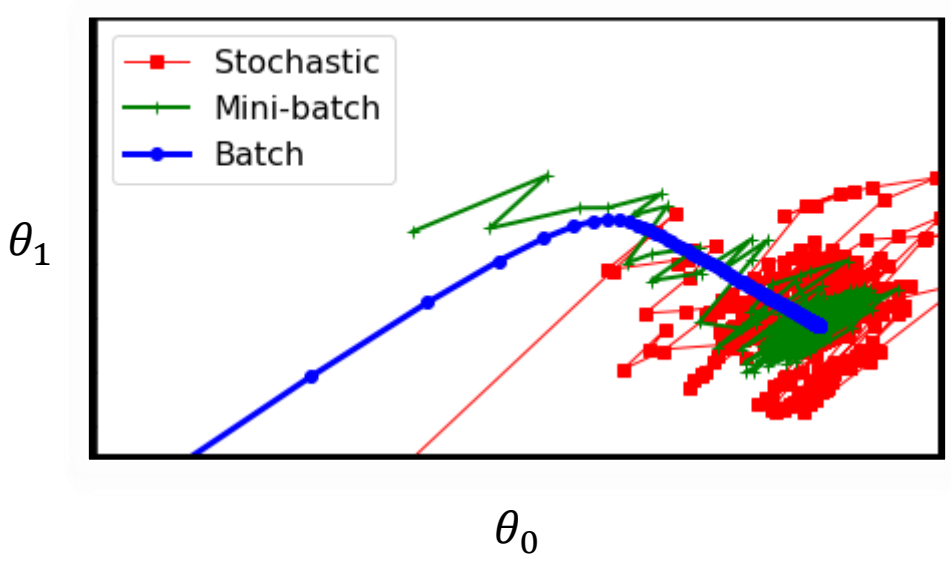
ML Terminologies

1. Batch – set of examples used in one iteration of model training.
2. Mini-batch – A small, randomly selected subset of the entire batch of examples run together in a single iteration of training or inference.
3. Iteration – A single update of a model's weights during training.
4. Epoch – A full training pass over the entire data set such that each example has been seen once.

https://developers.google.com/machine-learning/glossary/

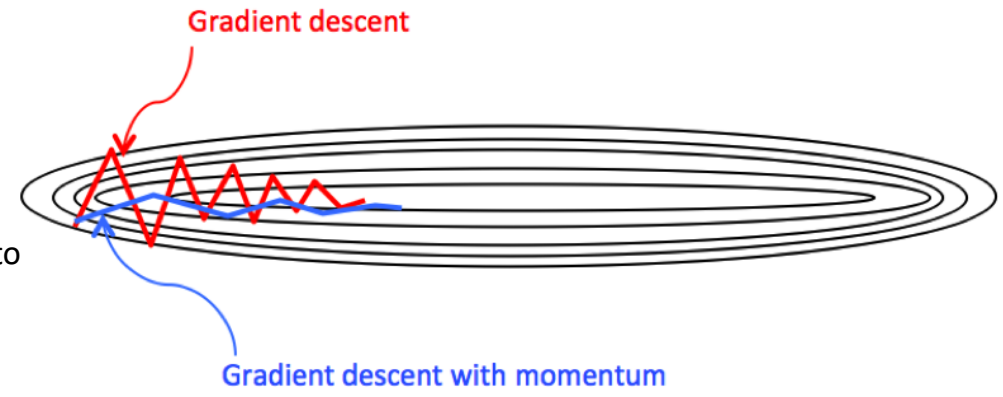
Gradient Descent Variants

1. Batch – compute gradient in one-shot with all training data
2. Stochastic – compute gradient using one random training data instance at a time
3. Mini-batch – compute gradient using small random sets of training data instances called mini-batches
4. Stochastic + Momentum – use weighted moving average of gradients
   (https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d)



Gradient Descent Optimizations

1. Momentum – use exponentially weighted average of past gradient to update for future gradient
   https://distill.pub/2017/momentum/
2. Nesterov Accelerated Gradient – similar to momentum but calculating gradient not at local position but slightly ahead
3. Adagrad – decays the learning rate as a function of contour steepness, but tends to slow down and stop early
4. Adadelta – reduces adagrad decreasing rate by using a fixed window for past gradients
5. RMSProp – fixes adagrad slow down by adding exponential weighted average of past gradients
6. Adam – adaptive moment estimation that combines the best of momentum and rmsprop to get an adaptive learning rate algorithm



https://medium.com/machine-learning-bites/deeplearning-series-deep-neural-networks-tuning-and-optimization-39250ff7786d

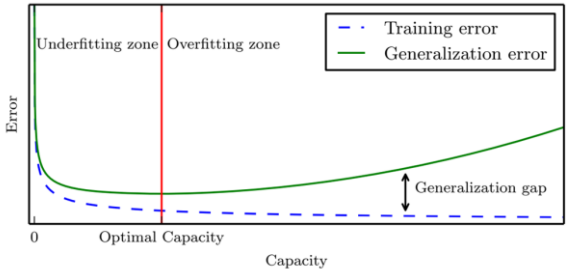
Training Sparse Models

1. Optimizations produce dense models where all parameters are non-zero.
2. For faster time performance, use sparse models with insignificant parameters set to zero.
3. One option is to use L_1 Lasso regularization.

Linear Model Selection & Regularization

Motivation

- Prediction Accuracy – if linear relation between dependent & independent then low bias; if *n* >> *p* then also low variance.
- Model Interpretability – irrelevant variables complicate models.



Methodologies

1. Subset Selection – identify a subset of *p* predictors.
2. Shrinkage aka Regularization – a modification to a learning algorithm that is intended to reduce its generalization error but not its training error.
3. Dimension Reduction – project *p* predictors into a *M*-dim space where *M* << *p*.

Subset Selection

1. Forward stepwise
    1. At each step gives the greatest additional improvement.
    2. Only viable option when *p* is very large.
    3. Not guaranteed to find best possible model – it is a computational tradeoff.

2. Backward stepwise
    1. At each step gives the greatest improvement removing a predictor.
    2. Requires that *n* is larger than *p* (to enable a full model fit to start).
    3. Not guaranteed to find best possible model – it is a computational tradeoff.

3. Hybrid
    1. Add a new variable, then remove any variables not improving model.
    2. Attempts to closely mimic best subset selection and retain computational advantage.

Shrinkage

$$RSS = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

1. Ridge regression

$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

2. Lasso regression

$$RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

2. Elastic Net (parameterized above using $\alpha \in [0, 1]$)

$$RSS + \alpha\lambda \sum_{j=1}^{p} \beta_j^2 + (1 - \alpha)\lambda \sum_{j=1}^{p} |\beta_j|$$

.

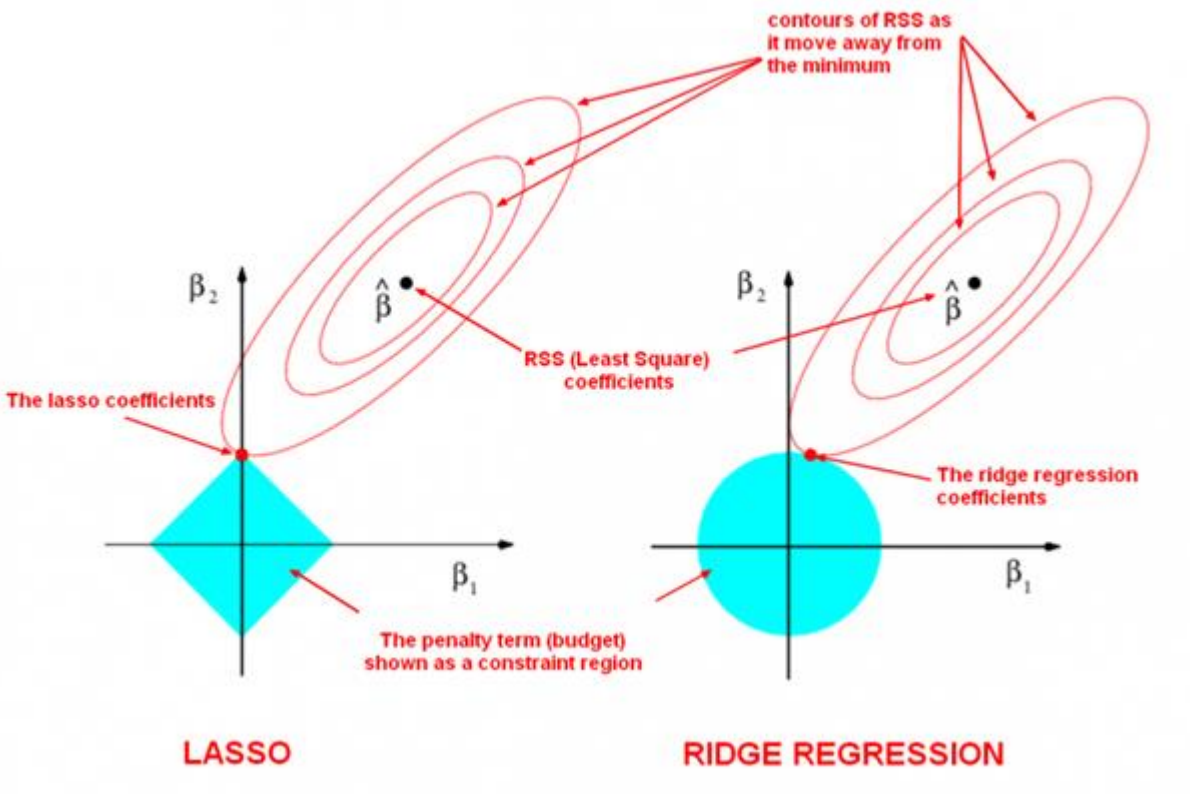
Shrinkage

1. Ridge regression
    1. Advantage – over least squares due to bias-variance trade-off; as λ increases, variance decreases, bias increases; works best in situations where least squares have high variance that happens due to correlated variables.
    2. Disadvantage – no variable set to 0 so model interpretation is a challenge when number of predictors are large.

2. Lasso regression
    1. Advantage – produces simpler and more interpretable models with subset of predictors, helps in feature selection.
    2. Disadvantage – arbitrarily choses which variable to remove.

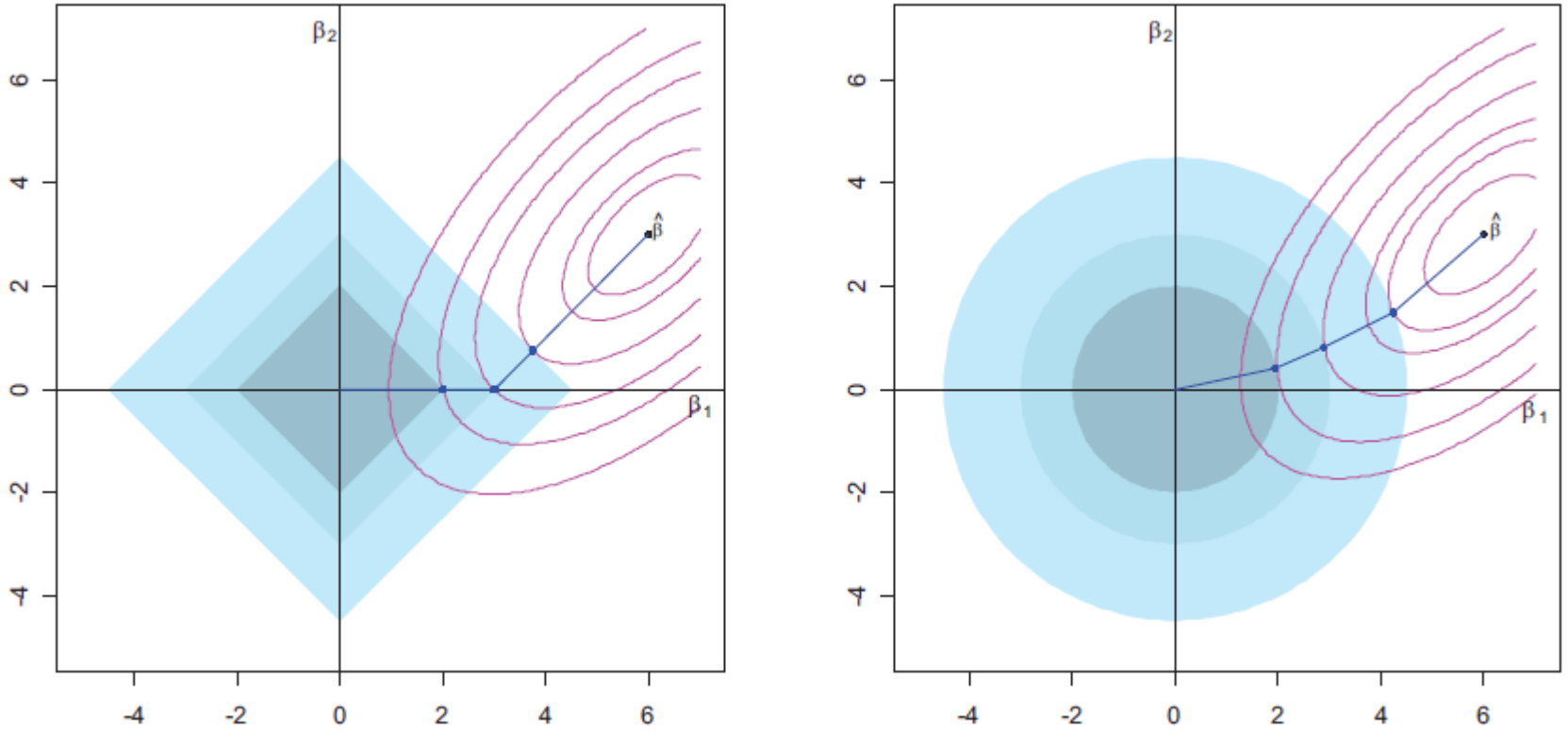
3. Elastic net – combines the advantages of Ridge and Lasso
    1. The Ridge part generates a sparse model.
    2. The Lasso part removes unwanted variables and stabilizes the Ridge part.

Shrinkage Visualization



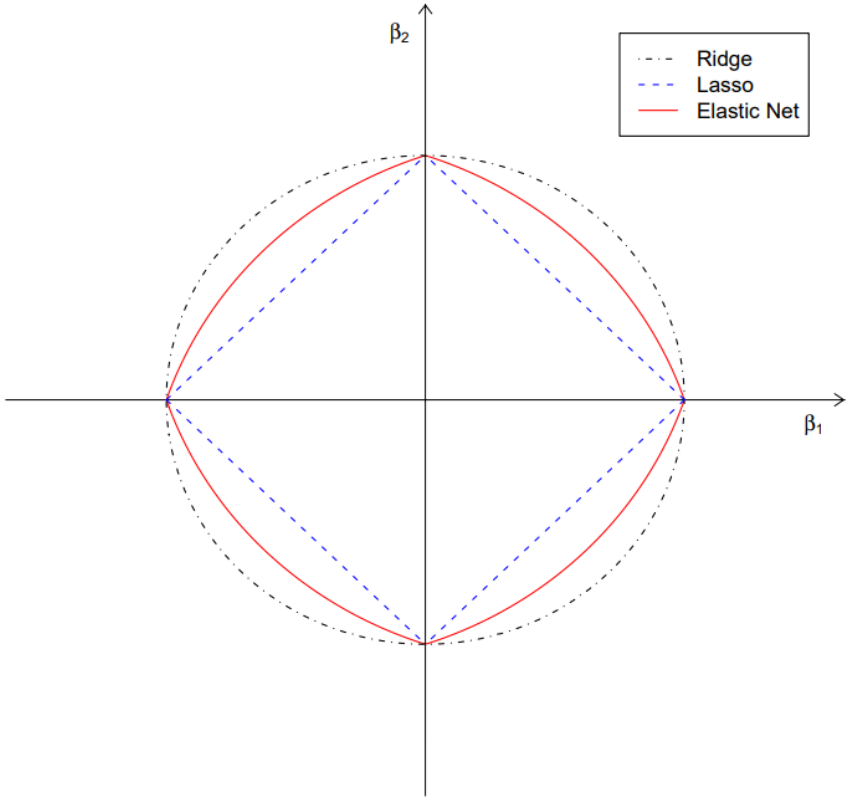
https://www.quora.com/How-would-you-describe-the-difference-between-linear-regression-lasso-regression-and-ridge-regression

Shrinkage Visualization



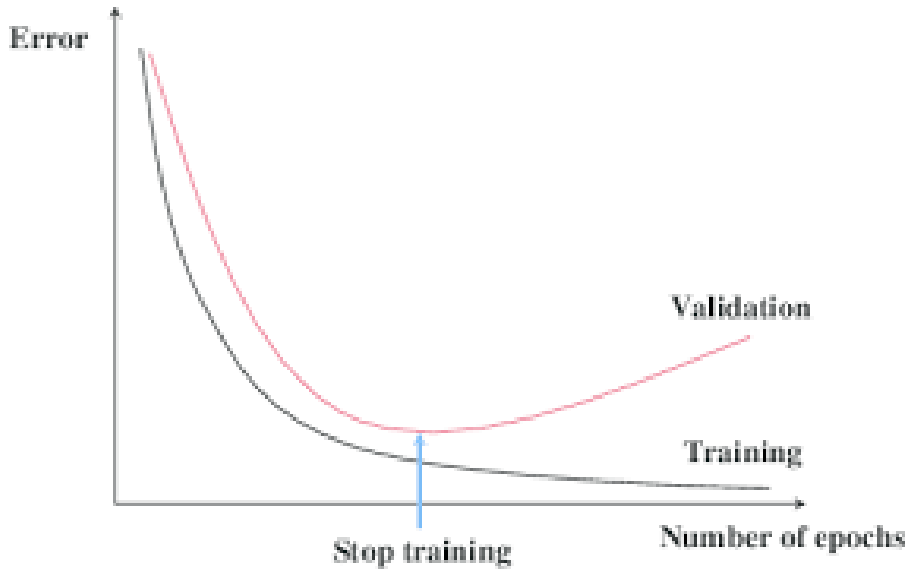
https://i.stack.imgur.com/UaoPh.png

Shrinkage Visualization



Early Stopping

A method for **regularization** that involves ending model training *before* training loss finishes decreasing. In early stopping, you end model training when the loss on a **validation data set** starts to increase, that is, when **generalization** performance worsens.



Dimension Reductions

1. Principal Components Analysis (PCA) – project *p* variables onto *M* dimensions to construct the first *M* principal components.
2. Principal Components Regression (PCR) – use the *M* principal components as predictors in a linear regression model fit using OLS.
3. Works well when first *M* principal components capture most of the variation in the predictors as well as the relation with the dependent variable.
4. NOTE – not a feature selection since each principal component is a linear combination of all *p* original predictors – consider ridge regression as a continuous version of PCR.
5. Partial Least Squares (PLS) – finds directions that help explain both the response and the predictors
6. T-distributed Stochastic Neighbor Embedding (t-SNE) – reduces dimensionality keeping similar points close and dissimilar points apart using non-linear approach with probability distributions

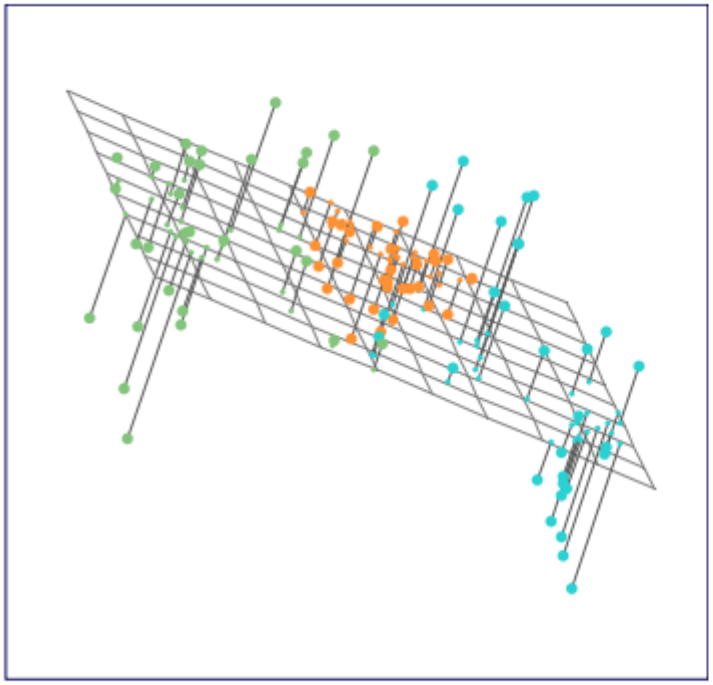
PCA Interpretation

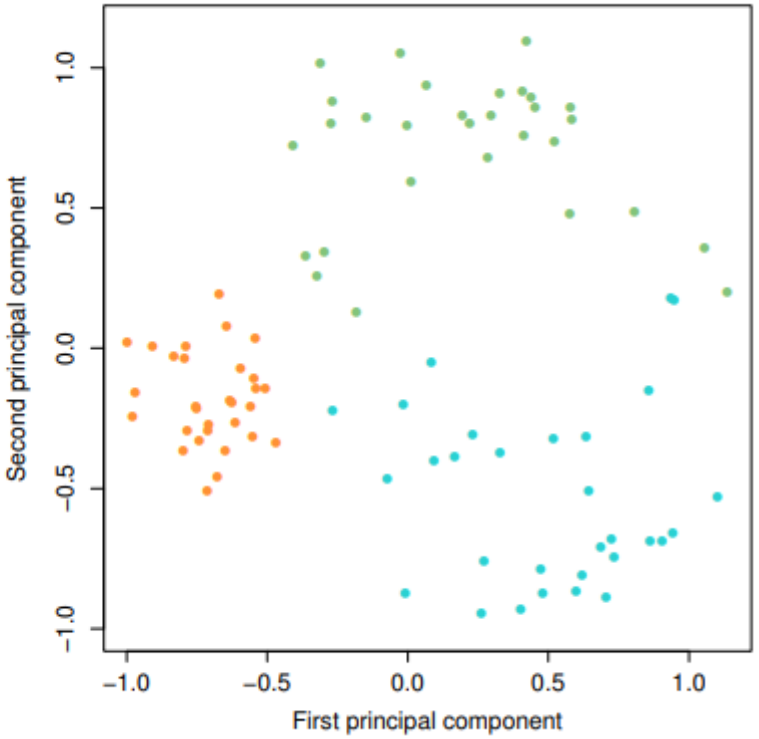
1. The first principal component loading vector has a very special property: it defines the line in p-dimensional space that is closest to the n observations (using average squared Euclidean distance as a measure of closeness).

2. The notion of principal components as the dimensions that are closest to the n observations extends beyond just the first principal component.

3. For instance, the first two principal components of a data set span the plane that is closest to the n observations, in terms of average squared Euclidean distance.

PCA Interpretation

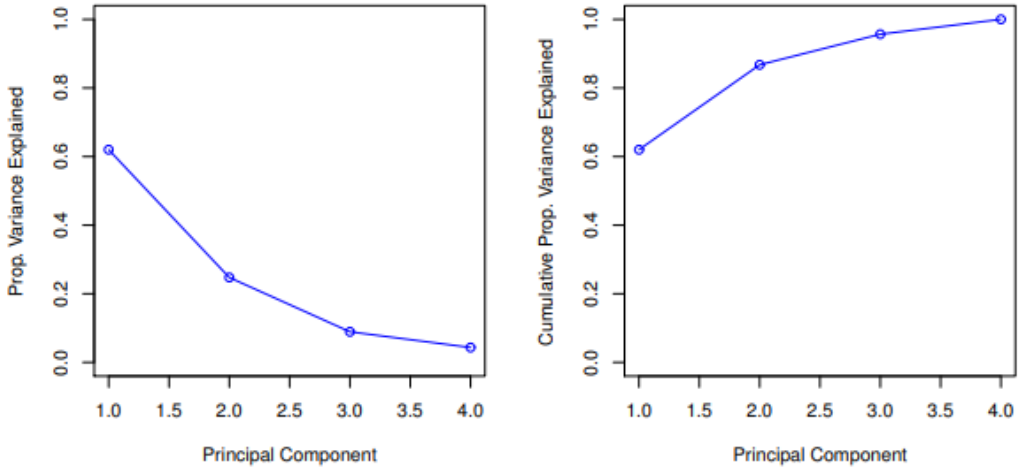




http://auapps.american.edu/alberto/www/analytics/ISLRLectures.html

More on PCA

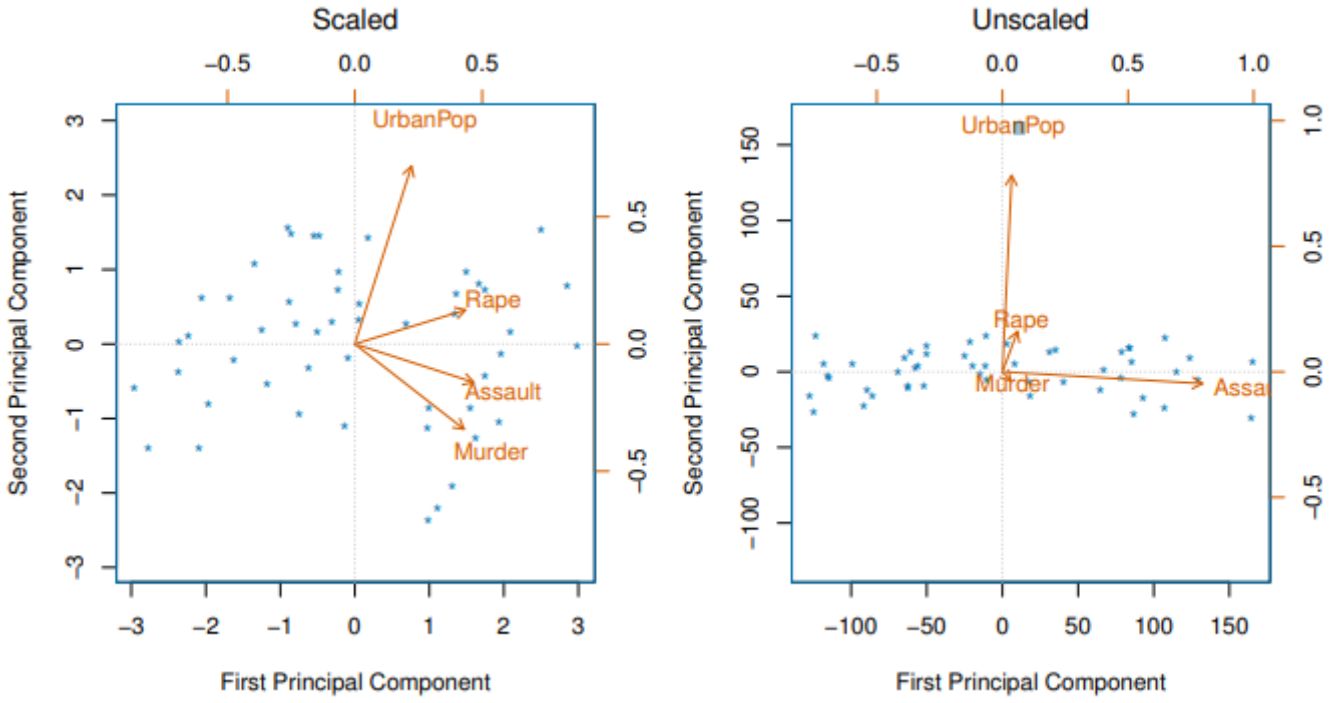
1. Useful information is the explained variance ratio of each principal component.
2. Need to select the right number of dimensions. https://stats.stackexchange.com/questions/119746/what-is-the-proper-association-measure-of-a-variable-with-a-pca-component-on-a/119758#119758
3. Incremental PCA (IPCA) – split training set into mini-batches and do IPCA on each mini-batch; good for large training sets.
4. Kernel PCA – projects instances into a very high dimensional feature space; good for preserving clusters of instances after projection.



http://auapps.american.edu/alberto/www/analytics/ISLRLectures.html

PCA scaling variables

1. If the variables are in different units, scaling each to have standard deviation equal to one is recommended.
2. If they are in the same units, you might or might not scale the variables.



http://auapps.american.edu/alberto/www/analytics/ISLRLectures.html

PCA to t-SNE

1. PCA is a linear algorithm that is not be able to interpret complex polynomial relationship between features.

2. t-SNE is based on probability distributions with random walk on neighborhood graphs to find the structure within the data. https://distill.pub/2016/misread-tsne/

Partial Least Squares

Linear combination of predictors for dimension reduction

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j$$

1. After standardizing the p predictors, PLS computes the first direction Z_1 by setting each ϕ_{1m} equal to the coefficient from the simple linear regression of Y onto X_j - ensures highest weight given to variables that are most strongly correlated to the response.

2. Subsequent directions are found by taking residuals when each of the above X_j are regressed on Z_1 and then repeating the above prescription to obtain Z_2.

Textbook Chapters

- Materials covered available in book:
  - DL: Chapters 5, 7, 16
  - HML: Chapters 4, 11
  - ISL: Chapters 4, 6
  - DLP: Chapter 3

- Gradient descent with momentum: https://distill.pub/2017/momentum/

- Code:
  - https://github.com/ageron/handson-ml2
  - https://pomegranate.readthedocs.io/en/latest/BayesianNetwork.html