An Introduction to Machine Learning

TRIPODS Summer Boorcamp: Topology and Machine Learning

August 6, 2018

General Set-up

Set-up and Goal

Suppose we have $X_1, X_2, \ldots, X_n$ data samples. Can we predict properites about any given $X_{n+1}, X_{n+2}, \ldots, X_N$?

Machine learning systems attempt to predict properties of unknown data based on the attributes or features of the data.

Supervised learning

- Data comes with attributes that we want our algorithm to predict
- Machine learning algorithm is given data attributes and desired outputs and the goal is to learn a way to map inputs to outputs in a general way
- 2 main types of learning:
  1. **Classification**: Data belongs to different classes/groups and we want to be able to predict which class/group unlabeled data belongs to
  2. **Regression:** Data labeled with one or more continuous variables (parameters) and the task is to predict the value of these variables for unknown data

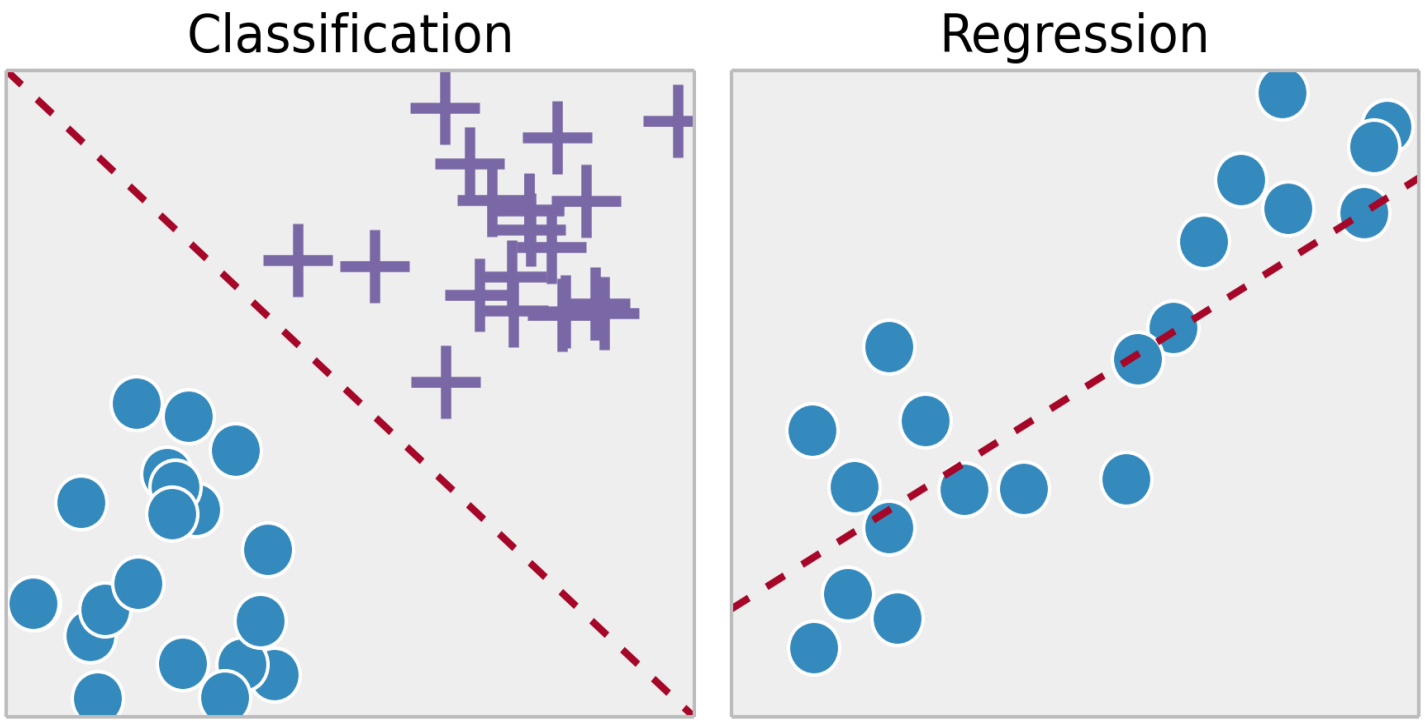
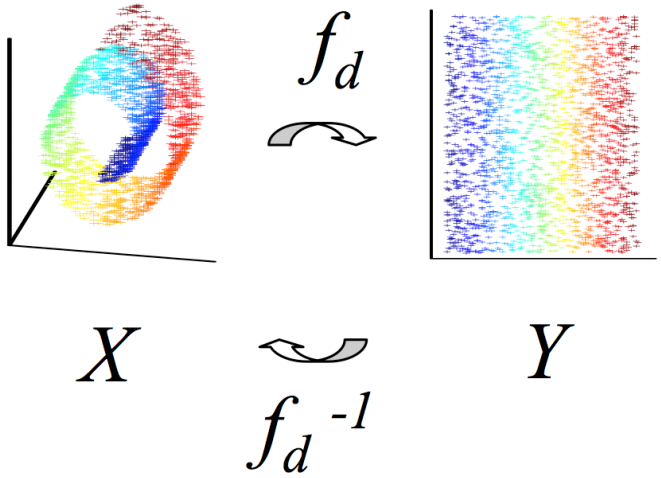
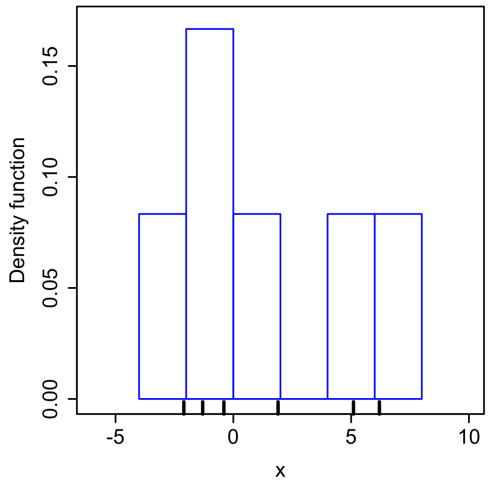


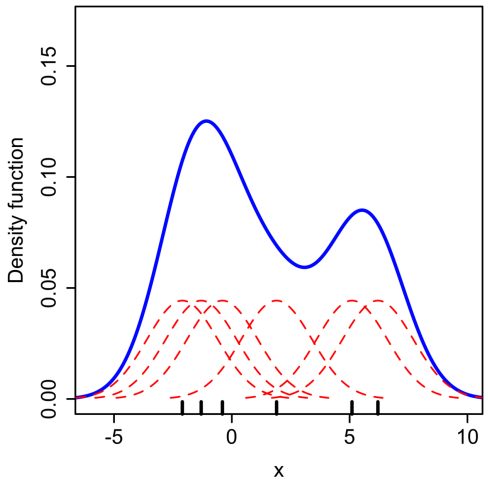
Image source: `http://ipython-books.github.io/featured-04/`

Unsupervised Learning

- No labels are given to the algorithm
- Training data consists of a set of input vectors $\{X_1, X_2, \ldots, X_n\}$ with no target outputs
- 3 Types of goals in this setting:
  1. **Clustering**: discover groups with similar features within the data
  2. **Density Estimation:** determine distribution of data within the input space
  3. **Dimensionality Reduction:** project data into a lower dimensional space than input space







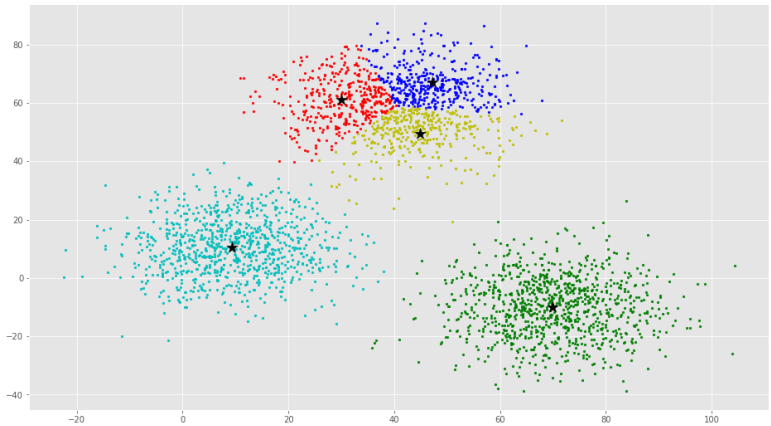


Image sources: `http://www.cs.otago.ac.nz/homepages/smartin/pub_long_images/Inversion.png`, `https://en.wikipedia.org/wiki/Kernel_density_estimation`, `https://mubaris.com/2017/10/01/kmeans-clustering-in-python/`

Applications of Machine Learning

- Spam email detection
- Improving weather prediction
- Targeted advertising and web searches
- Predicting emergency room wait times using staffing levels, patient data, charts, and layout of ER
- Identifying heart failure from physician's notes
- Predicting hospital readmissions
- Learning dynamical systems models directly from high-dimensional sensor data (Byron Boots, Georgia Tech)

Goals For Today

- Learn a commonly used supervised learning algorithm (support vector machines)
- Learn a commonly used unsupervised learning algorithm (k-means clustering)
- Understand challenges in machine learning
- Apply algorithms to real data

Support Vector Machines: Setup

A SVM constructs a hyperplane (or set of hyperplanes) in a high or infinite dimensional space, which can be used for classification, regression or other tasks.

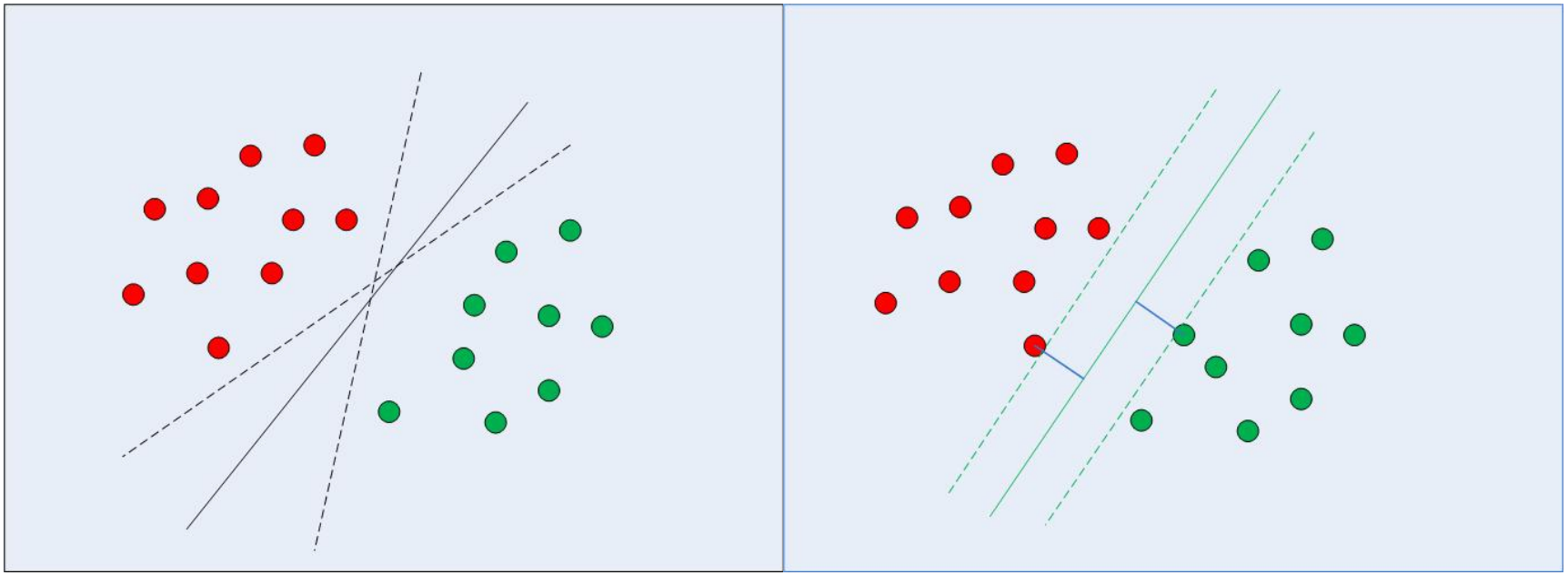


image source: `http://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/`

SVM: Linear, separable case

We want to find the hyperplane that maximizes the margin as follows:

Mathematical formulation

Given $X_1, X_2, \ldots X_n \in \mathbb{R}^p$, with labels $Y_i \in [-1, 1]$, let

$$H(x) = (\alpha + \beta^T x) \begin{cases} > 0 & \text{if } Y_i = 1 \\ < 0 & \text{if } Y_i = -1 \end{cases}$$

where $\alpha \in \mathbb{R}$, $||\beta|| = 1$, and $H(x) = 0$ is the decision boundary. $\beta = \sum_i w_i Y_i X_i$ yields the maximum margin, and we use convex optimixation to get α and weight vector w_i.

Decision function:

$$f(x) = \text{sign}(H(x))$$

SVM: Linear, non-separable case

In most cases there will be outliers so we need to allow for softer margins.

Mathematical formulation

Given $X_1, X_2, \ldots X_n \in \mathbb{R}^p$, with labels $Y_i \in [-1, 1]$. Now we choose α, β that maximimze the margin of separation (M) such that

$$Y_i(\alpha + \beta^T X_i) \geq M(1 - \delta_i) \forall i$$

where $\delta_i \geq 0$ for all i and $\sum_i \delta \leq C$ gives you a constraint of margin violation.

As before, $\beta = \sum_i w_i Y_i X_i$ and the decision function is given as

$$f(x) = \text{sign}(\alpha + \sum_i w_i Y_i X_i^T x)$$

SVM: non-linear case

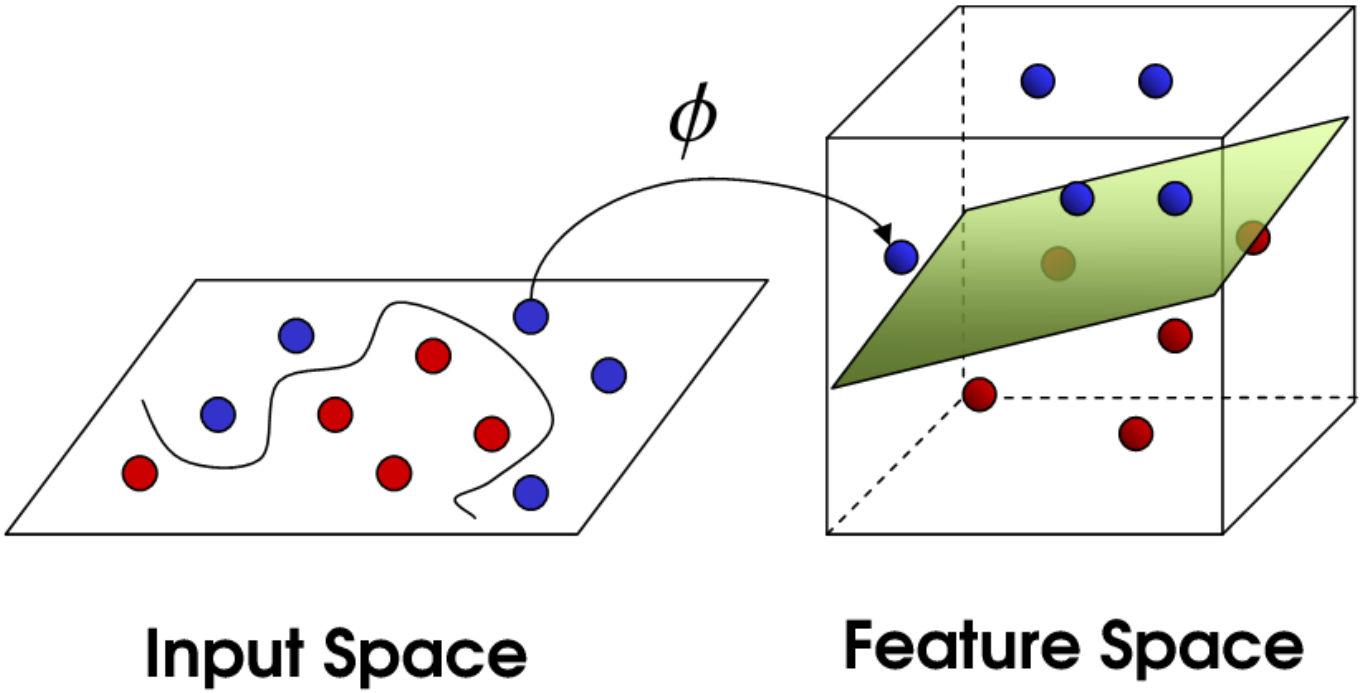


image source: `https://datascience.stackexchange.com/questions/17536/kernel-trick-explanation`

SVM: Kernel Trick

In most cases we do not want a linear separation boundary. Instead, we use a kernel trick.

1. Consider a non-linear transformation $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$.
2. Fit a linear decision boundary in $\mathbb{R}^{d'}$ using $(\phi(X_1), Y_1), (\phi(X_2), Y_2), \ldots)$ as the training data.
3. Define the kernel to be

$$K(x, x') = \phi(x)^T \phi(x')$$

4. Since the classifier only relies on $X_i^T x$, we get the decision function:

$$f(x) = \text{sign}(\alpha + \sum_i w_i Y_i K(X_i, x))$$

5. Using the kernel, we never actually have to transform the data!

Common kernel examples

- Polynomial-SVM:

$$K(X, X') = (X \cdot X')^d$$

When d is large, the kernel still only requires n computations, whereas explicit representation may not fit in memory

- Radial basis function:

$$K(X, X') = \exp(-\gamma||X - X'||^2)$$

Pros and Cons of SVM

Advantages

- Effective in high dimensional spaces
- Can be used for classification or regression
- Memory efficient since it only uses a subset of training points in the decision function
- Versatile since we can use different kernel functions for the decision function

Disadvantages

- Non-Probabilistic: SVMs do not directly provide probability estimates
- Method will likely not do well if the number of features is significantly greater than the number of samples

Clustering: K-means

Goal: cluster a data set $\{X_1, \ldots, X_N\}$ into K clusters $C_1 \ldots C_K$.

K-means clustering algorithm

1. Pick K random points $\mu_1, \ldots, \mu_K$ to be centroids of the K clusters $C_1, \ldots, C_K$
2. Assign each X_i to cluster j if $\min_{k \in [0,K]} d(X_i, \mu_k) = d(X_i, \mu_j)$
3. Update the centroids $\mu_1, \ldots, \mu_K$ by setting $\mu_j = \frac{1}{|C_j|} \sum_{X_i \in C_j} |X_i|$
4. Repeat 2 and 3 until none of the cluster assignments change

K-means clustering

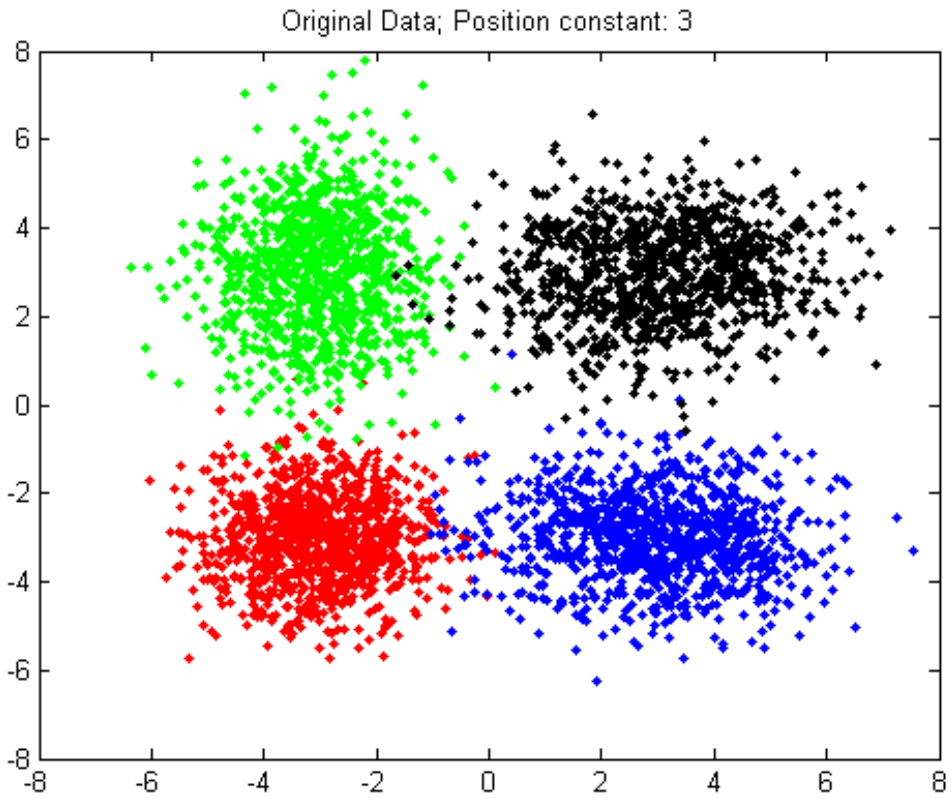


Image source: `http://webstaff.itn.liu.se/~reile/edu/TNM025/Matlab/html/KMeansDemo.html`

How do you choose K?

Assume you know the truth so you can compute the sum of squared errors. Use the Elbow Method:

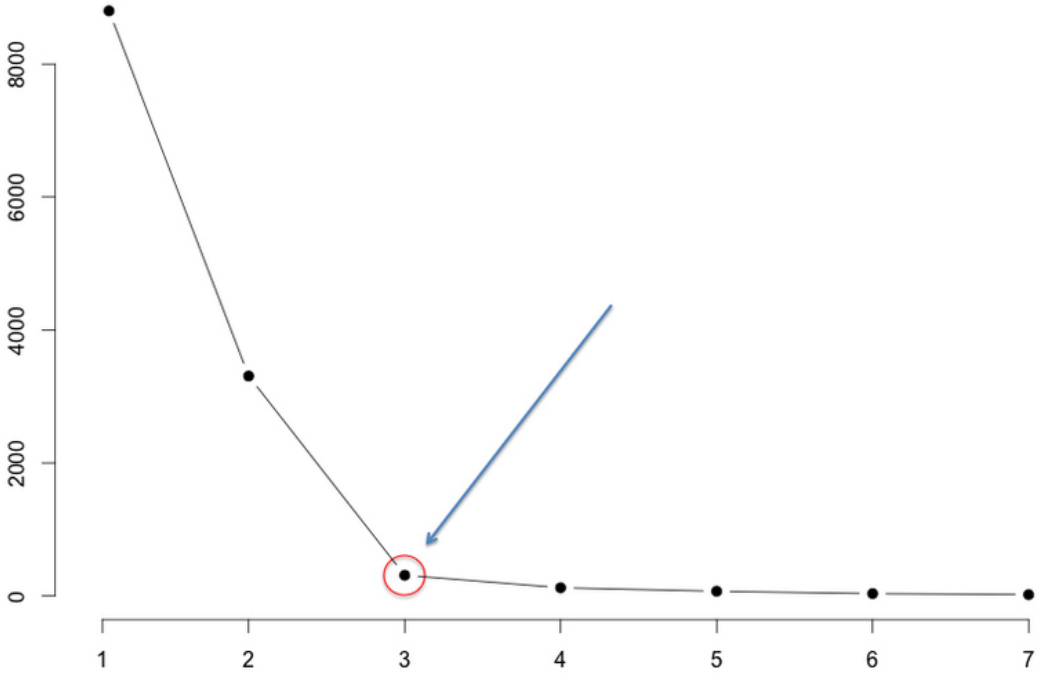


Image source: `https://mubaris.com/2017/10/01/kmeans-clustering-in-python/`

Pros and Cons K-means Clustering

Advantages

- Fast and easy to understand
- Easy to implement
- Works well for convex clusters

Disadvantages

- Will not work well if clusters are not convex
- Difficult to choose K
- Randomness in initial step makes results difficult to reproduce

When does K-means clustering fail?

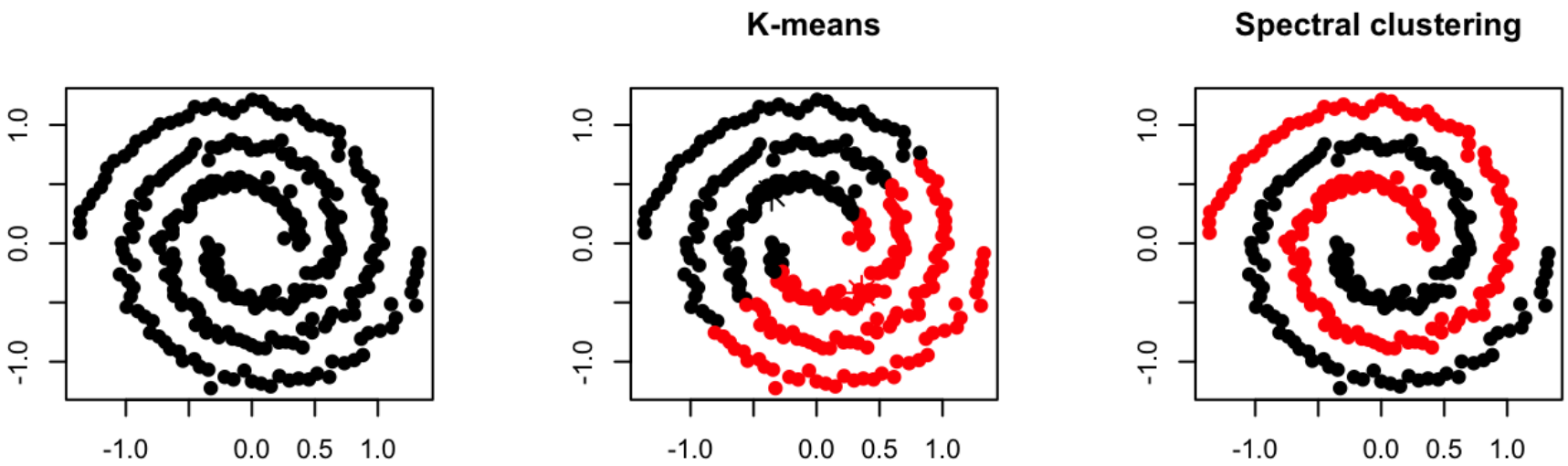


Image source: `http://scalefreegan.github.io/Teaching/DataIntegration/practicals/p2.html`

Overfitting

- “Hallucinating a classifier” occurs when data are not sufficient to determine the correct classifier
- Classifiers will encode random features in data which are not grounded in reality
- Overfitting can be decomposed into bias and variance:
  - Bias is the learner’s tendency to consistently learn the same (wrong) thing
  - Variance is the tendency to learn random things irrespective of real signal
- A more powerful algorithm is not necessarily better

Bias and Variance

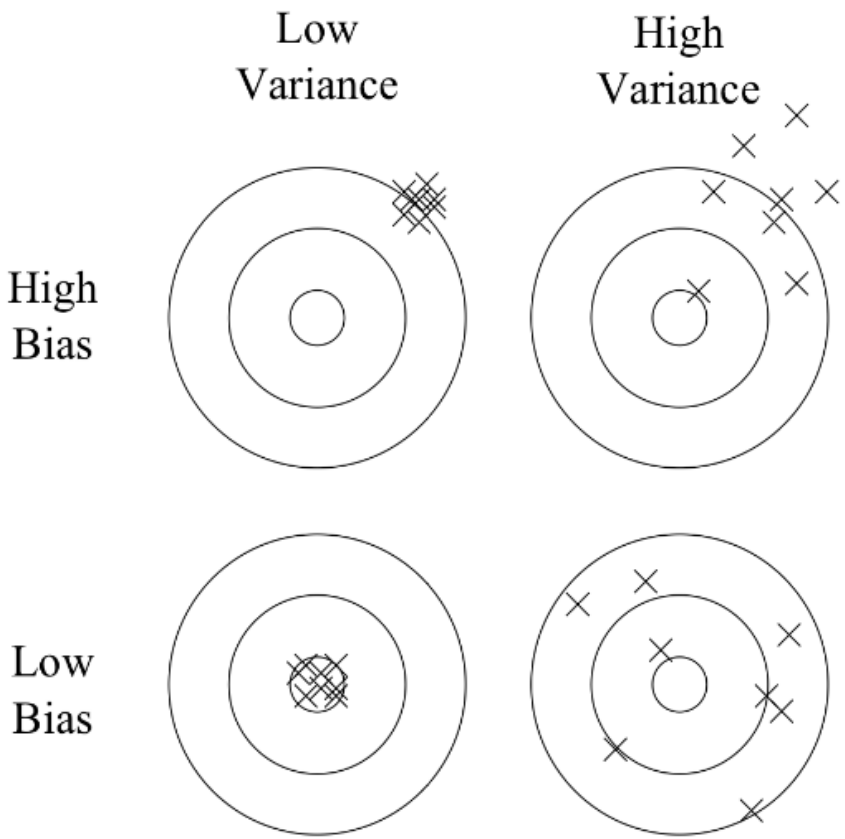


Image source: [2]

Curse of Dimensionality

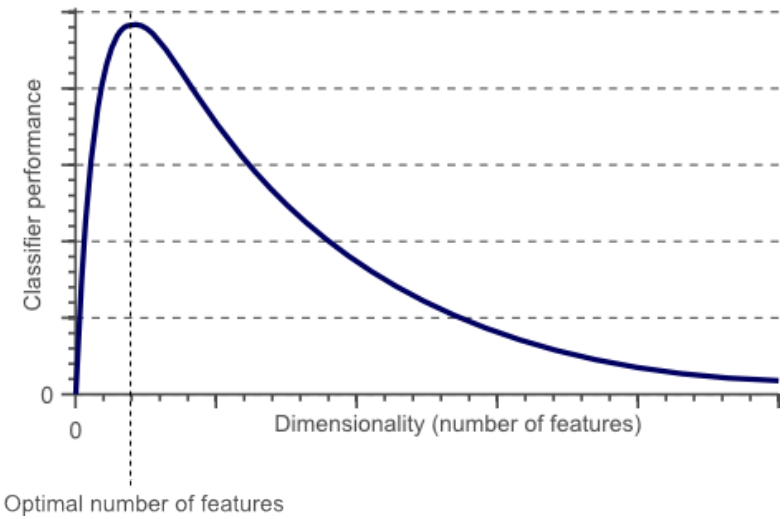


image source: `http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/`

- Second biggest problem in machine learning
- Generalizing correctly is exponentially more difficult as the dimensionality, or number of features, increases
- Good news: usually high dimensional data are concentrated on/near a lower dimensional manifold so we can use dimension reduction techniques to avoid this curse

Contamination of classifier

- Goal: General classifier
- Problem: Illusion of success
- Use cross-validation to avoid this:
  - Randomly divide training data into multiple subsets
  - Only use one subset for training at a time
  - Test each classifier on data not used for training
  - Average results to see how well the classifiers do

Useful Software Packages

- PyML, pyMC, scikit-learn in Python
- TensorFlow (Google)
- MATLAB's Statistics and Machine Learning toolbox
- Spider (MATLAB)
- Shogun
- mlpack in C++
- Torch (Lua) and PyTorch (Python)
- Weka (Java)
- Orange (Open source machine learning and data visualization)

References

[1] Bishop, Christopher. Pattern Recognition and Machine Learning. *Springer*. (2009)
[2] Domingos, Pedro. A Few Useful Things to Know about Machine Learning.
[3] Rudin, Cynthia and Kiri L. Wagstaff. Machine Learning for Science and Society. Mach Lean, *Springer*. (2013)
[4] Wagstaff, Kiri L. Machine Learning that Matters. *Proceedings of the 29th International Conferences on Machine Learning,* Edinburgh, Scotland. (2012)
[5] `scikit-learn.org/stable/user_guide.html`
[6] `https://www.quantstart.com/articles/Support-Vector-Machines-A-Guide-for-Beginners`
[7] `www.cs.columbia/edu/~kathy/cs4701/documents/jason_svm_tutorial.pdf`
[8] `https://mubaris.com/2017/10/01/`