Announcements

- Homework 4 due.
- Everyone else has to present tonight.
- Look into free trials for Google Cloud Platform or Amazon Web Services to host your app. For now build something which works locally. Work on an MVP (Minimal Viable Product), ie. get something working before making it fancy!
- **Working prototype is due in two weeks. These must be submitted via Github! Make a public repo on Github (free) with your code.**

Lecture 7 - Unsupervised Learning

Unlabeled machine learning problems

Lecture Outline

- What is unsupervised learning?
- K means, K Nearest Neighbors.
- Support Vector Machines and Kernels - The Kernel Trick.
- Principal Component Analysis.
- Mixture Models (if time permits)
- Topic Models and LDA (if time permits)

What is unsupervised learning?

- Any problem where we don’t have a fixed label (ie. no y variable)
- Examples:
  - K means
  - K Nearest Neighbors (which things are the ‘closest’ in some metric).
  - Gaussian MIxture Models
  - Topic Models, Latent Dirichlet Allocation (LDA).
  - Support Vector Clustering (SVC)/ SVM (labeled)

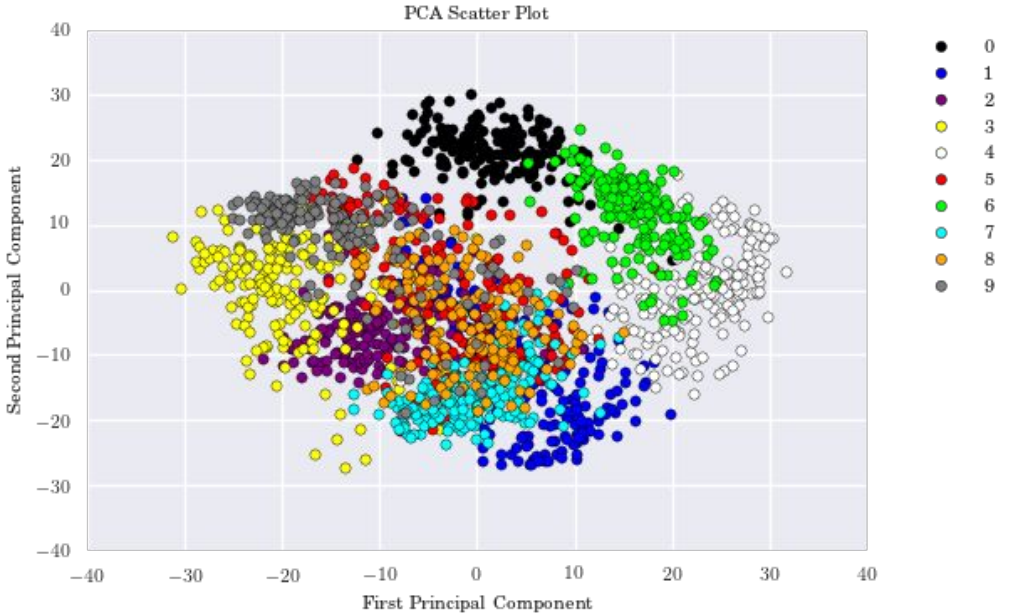
K Means Clustering

K means clustering

$$\operatorname{argmin}_S \sum_{k=1}^{K} \sum_{\mathbf{x} \in S_k} \|\mathbf{x} - \mu_i\|^2$$

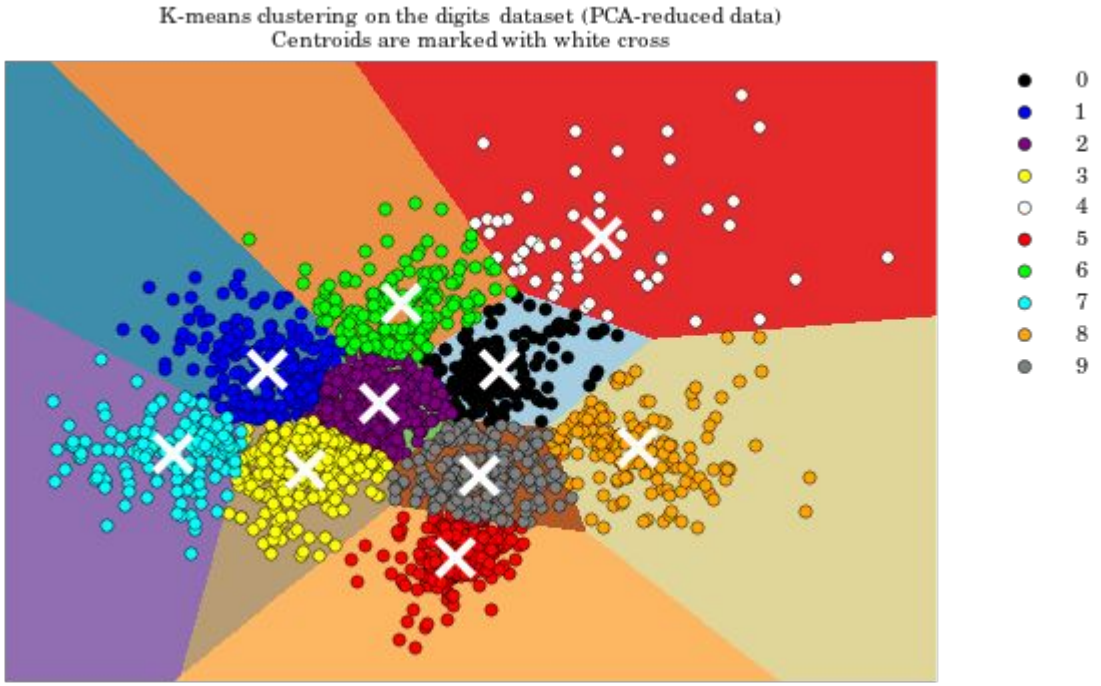
$$S = \{S_1, S_2, \cdots, S_K\}$$

- S_k sub region of space.
- μ_i centroid of the set S_k



Break your space into K fixed regions such that the distance (squared) from points to the centroid is of the region is minimized (**you choose K**).

K means clustering



An example of how the final clustering of the digits looks after K means.

The algorithm

1. Define initial points for the centroids

2. At time step t, assign a point to a cluster if it has the smallest squared distance to the centroid associated with it:

$$S_i^{(t)} = \left\{ x_p : \left\| x_p - m_i^{(t)} \right\|^2 \leq \left\| x_p - m_j^{(t)} \right\|^2 \ \forall j, 1 \leq j \leq k \right\},$$

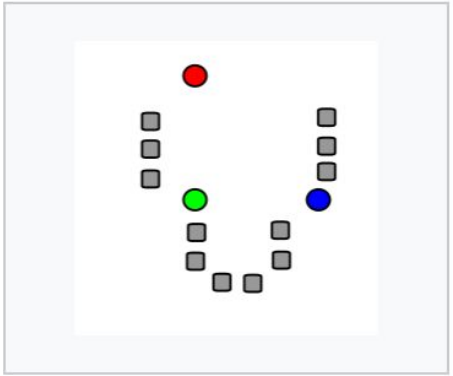
3. Recompute the new centroid based on the updated sets.

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

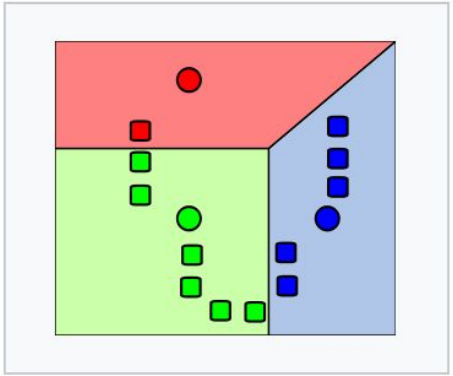
4. Continue 2-4 until regions don't change anymore.

The algorithm

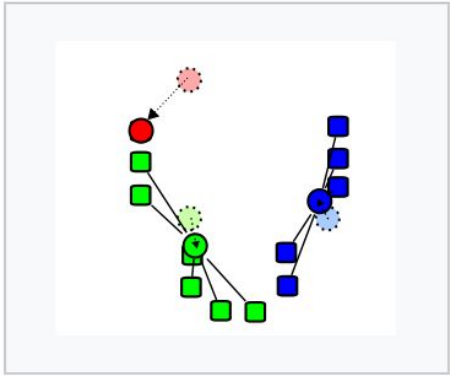
Demonstration of the standard algorithm



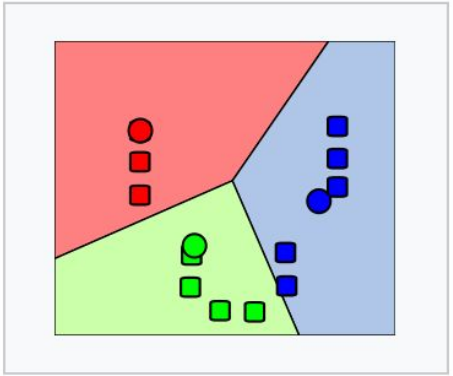
1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).



2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

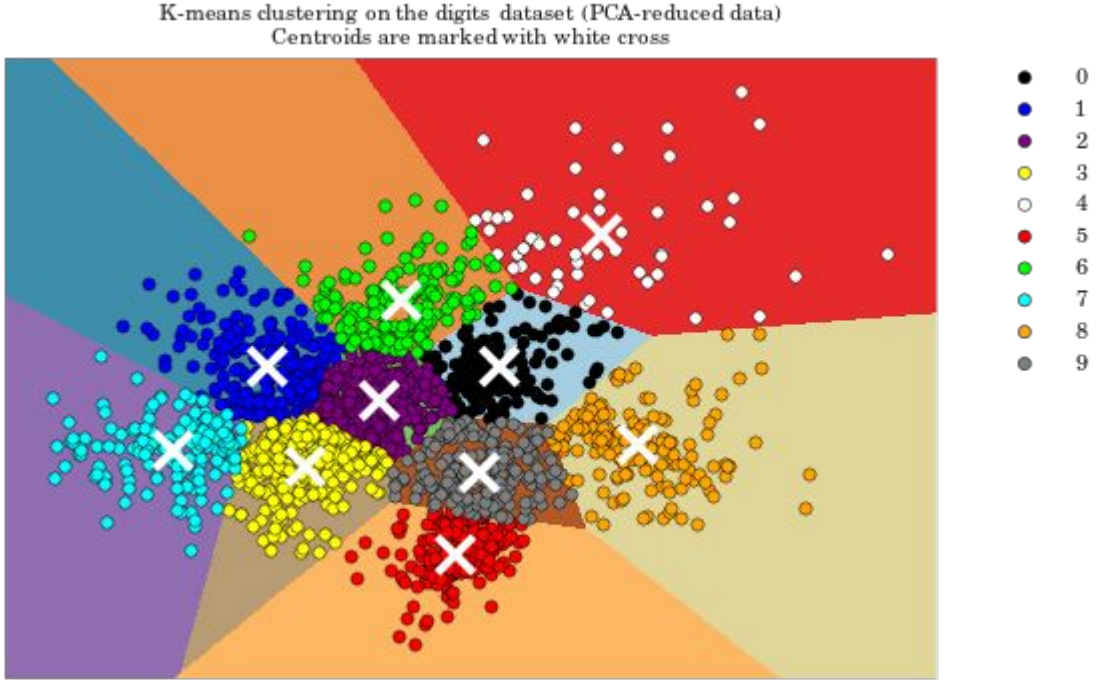


3. The centroid of each of the *k* clusters becomes the new mean.



4. Steps 2 and 3 are repeated until convergence has been reached.

Example 1 - Digit Recognition



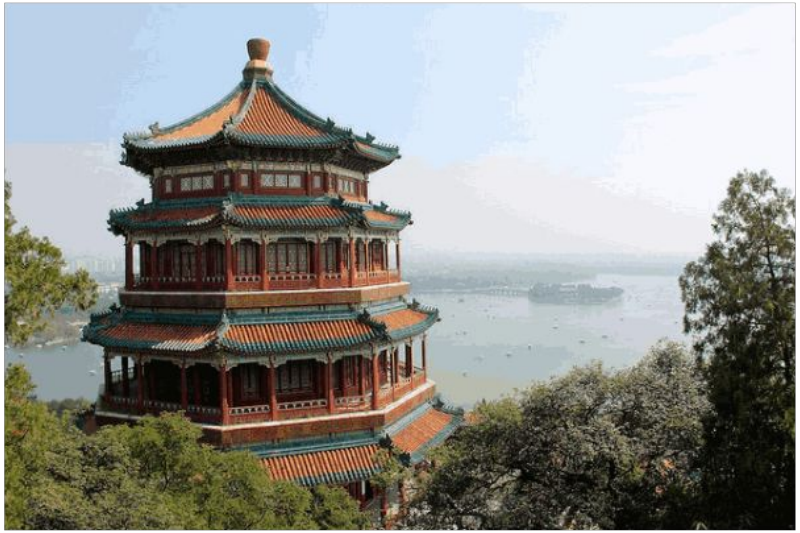
K means after applying PCA to plot in 2d

Example 2 - Image Compression

Original image (96,615 colors)

96, 615 colors

Quantized image (64 colors, K-Means)

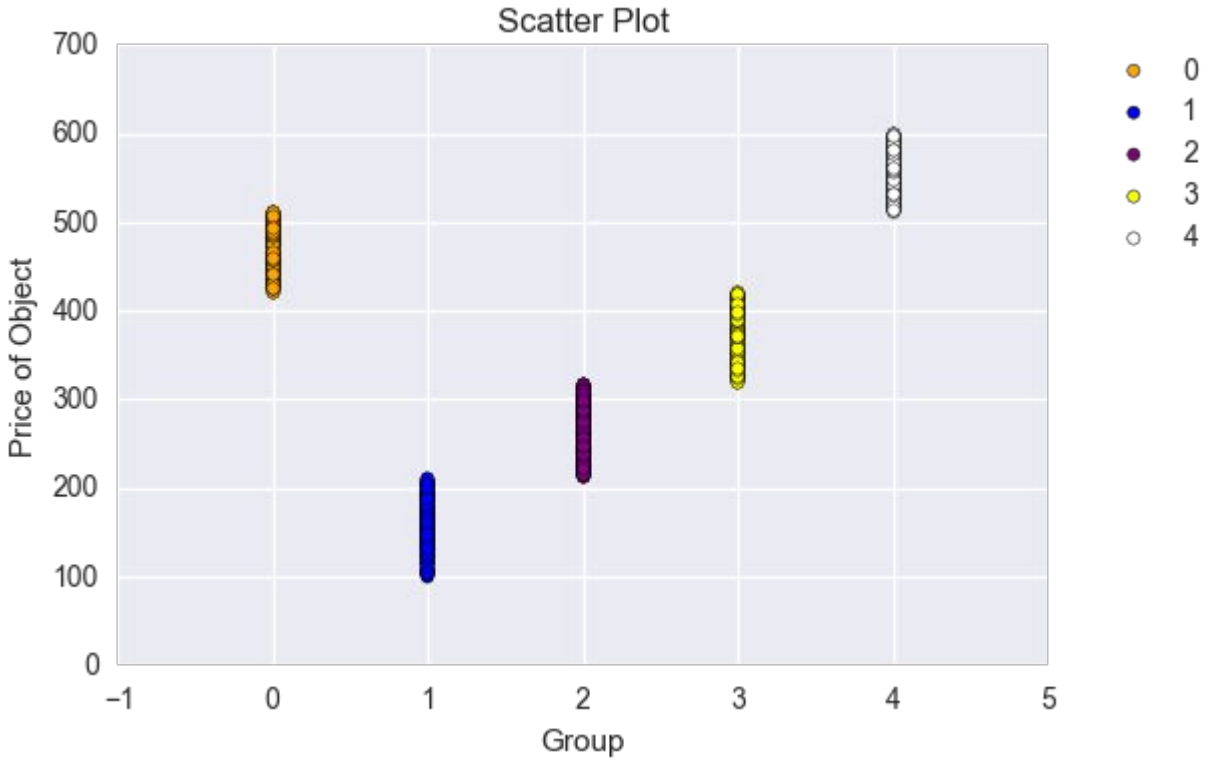


64 colors

Example 3

Problem: You have 100 items at a yard sale, and you know the exact price of each item. However you want to simplify matters and have only 5 boxes, with the price of each item being the same in any particular box. How do you choose the prices to put on the boxes?

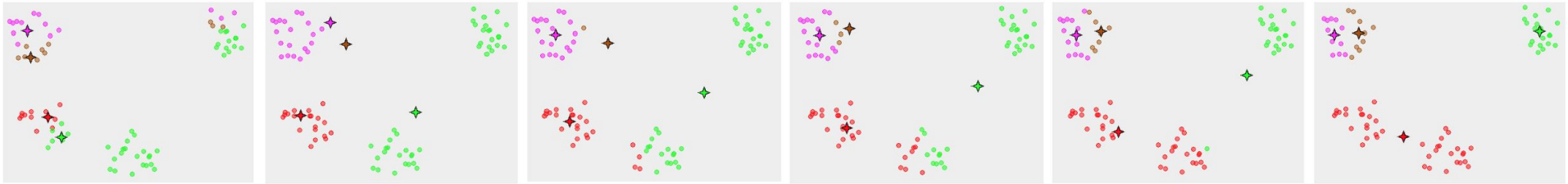
Pricing Groups



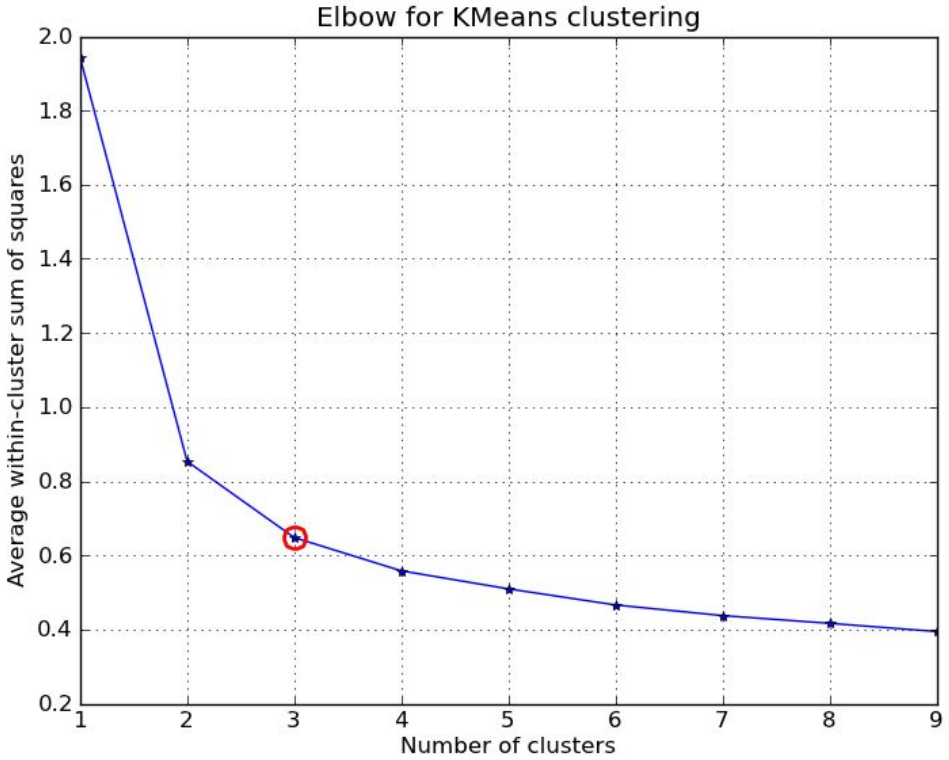
Solution: Break prices into 5 groups. Then for each group, the items that you're selling from that box will have the smallest distance to the 'centroid' or average price.

Properties

- Euclidean distance is used as a metric and variance is used as a measure of cluster scatter.
- The number of clusters *k* is an input parameter: an inappropriate choice of *k* may yield poor results. That is why, when performing k-means, it is important to run diagnostic checks for determining the number of clusters in the data set.
- Convergence to a local minimum may produce counterintuitive ("wrong") results (see example in Fig.).

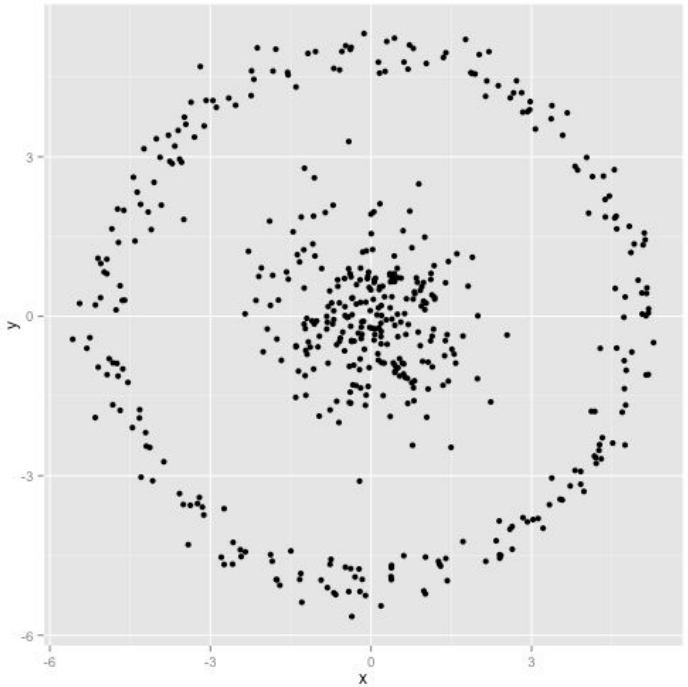


Elbow method for number of clusters

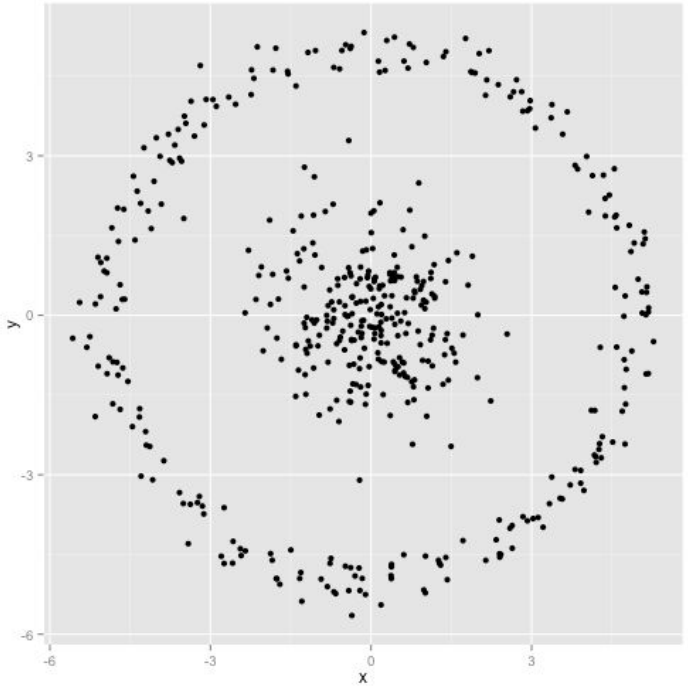


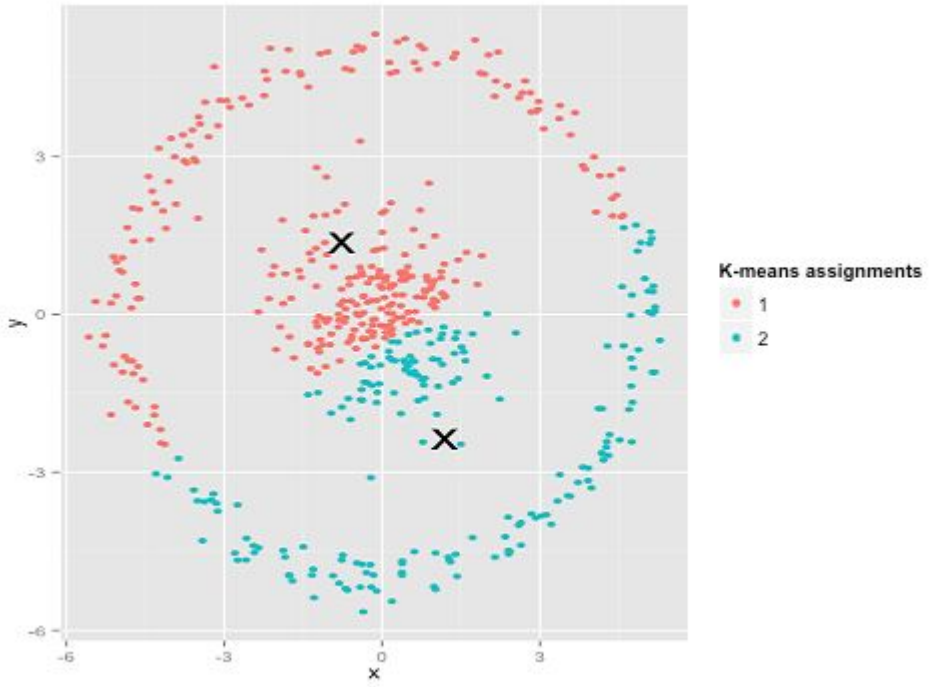
To choose the ‘optimal’ number of clusters, a typical technique is to choose the ‘elblow point’ on the graph, where the decrease in variance ‘levels off’.

K means with spherical data



K means with spherical data

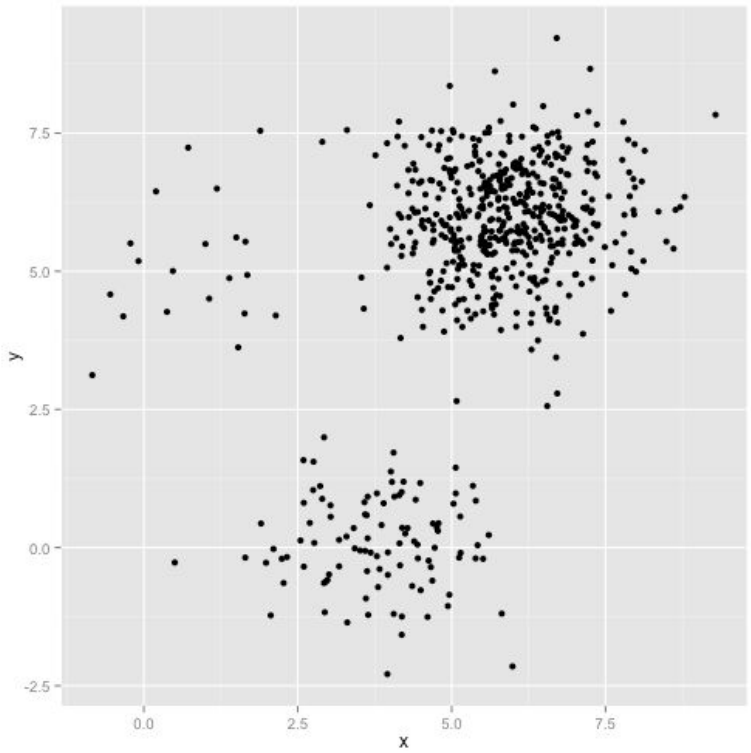




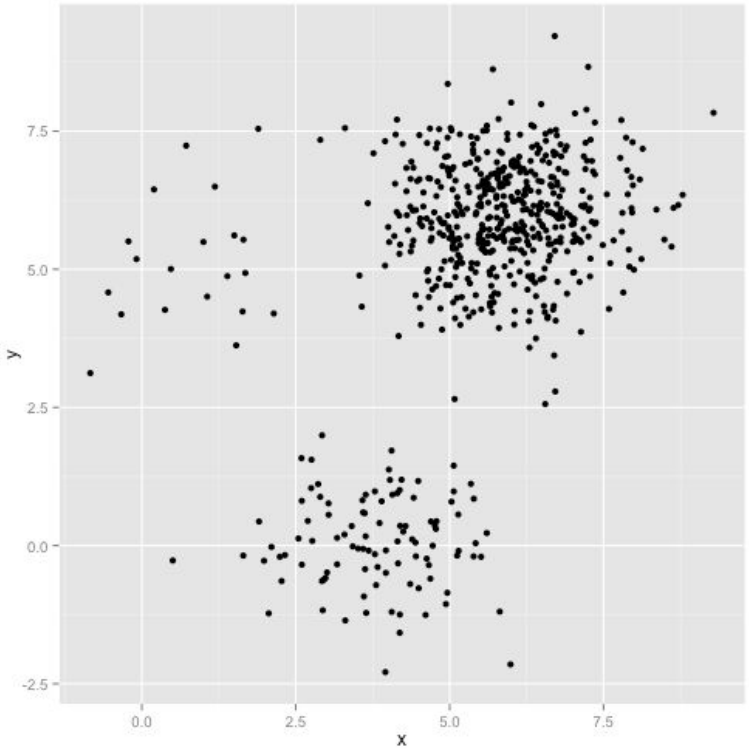
What happened?

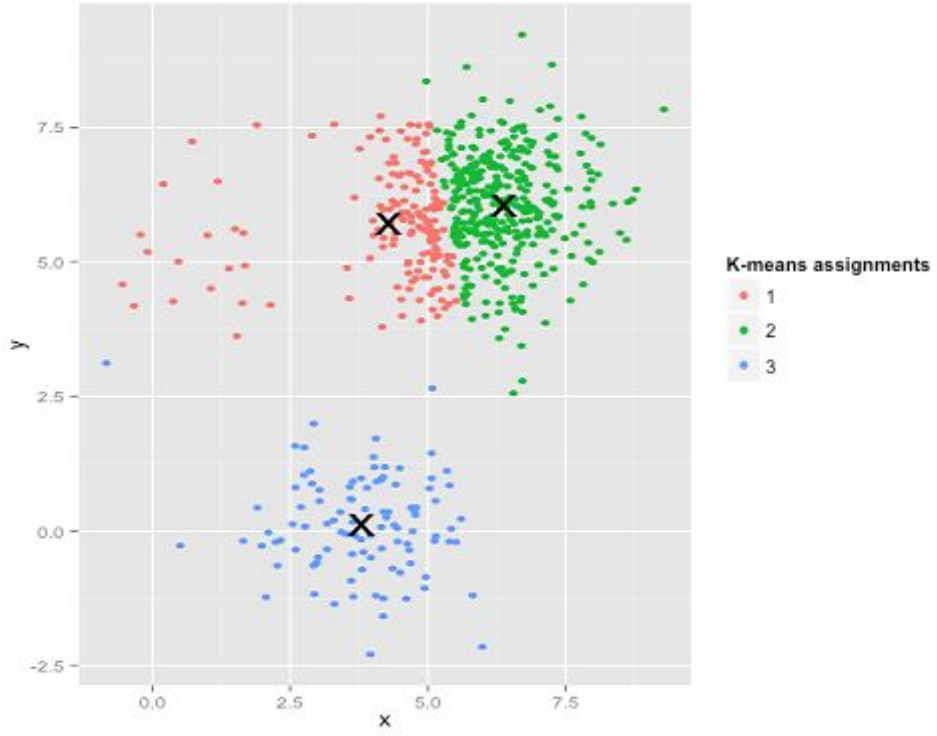
What happened here is a bit subtler. In its quest to minimize the within-cluster sum of squares, the k-means algorithm gives more "weight" to larger clusters. In practice, that means it's happy to let that small cluster end up far away from any center, while it uses those centers to "split up" a much larger cluster.

K means bias for groups with more points K=3



K means bias for groups with more points K=3

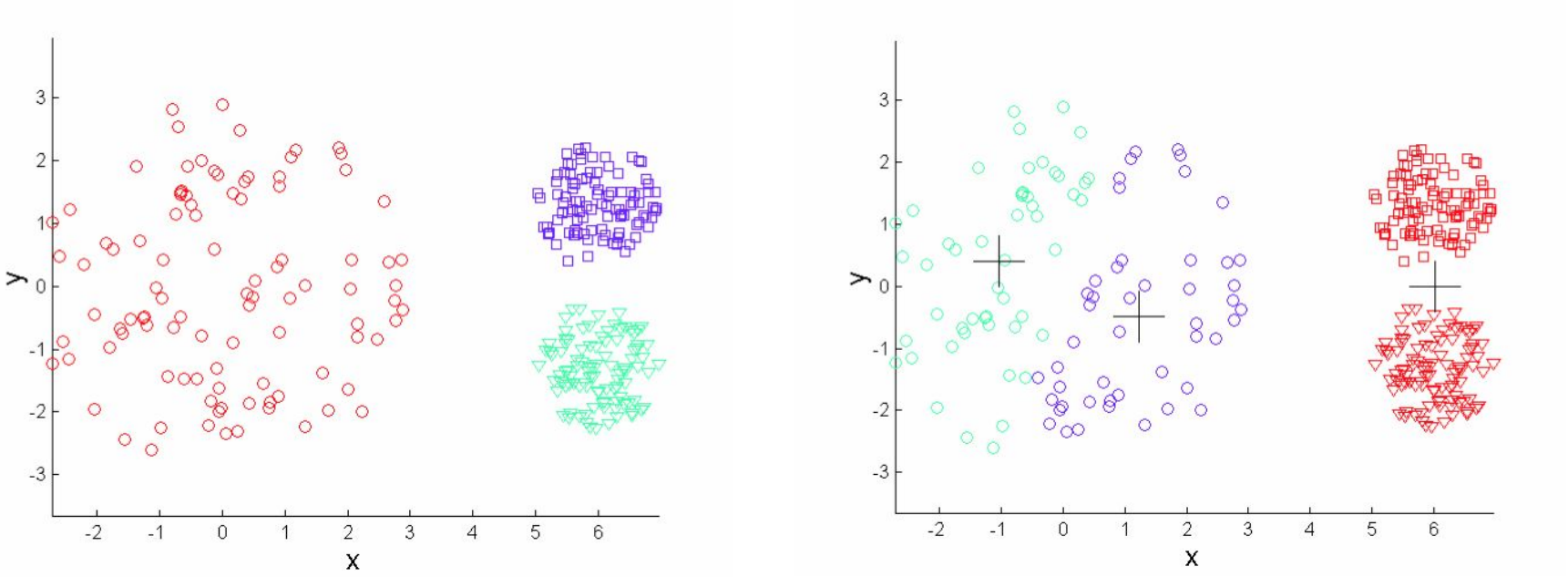




What happened?

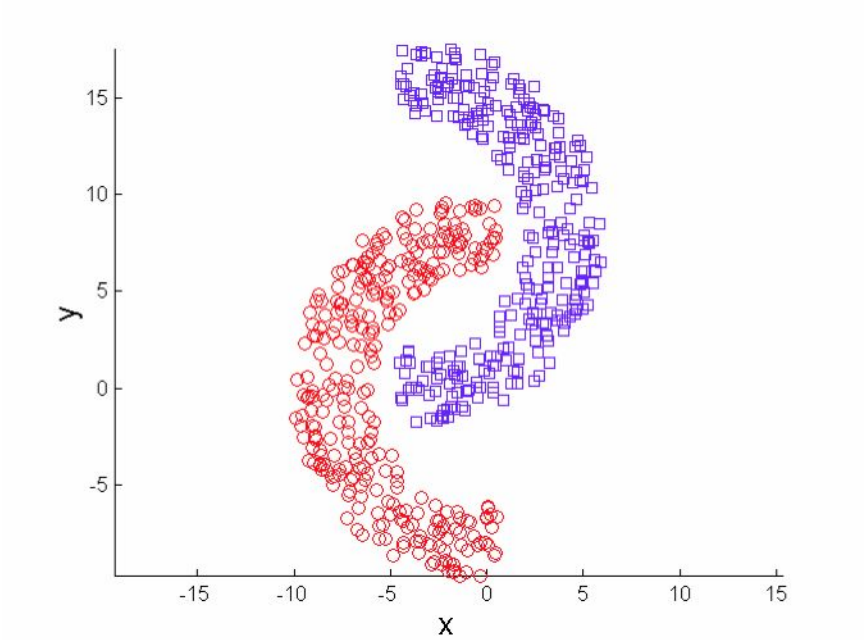
What happened here is a bit subtler. In its quest to minimize the within-cluster sum of squares, the k-means algorithm gives more "weight" to larger clusters. In practice, that means it's happy to let that small cluster end up far away from any center, while it uses those centers to "split up" a much larger cluster.

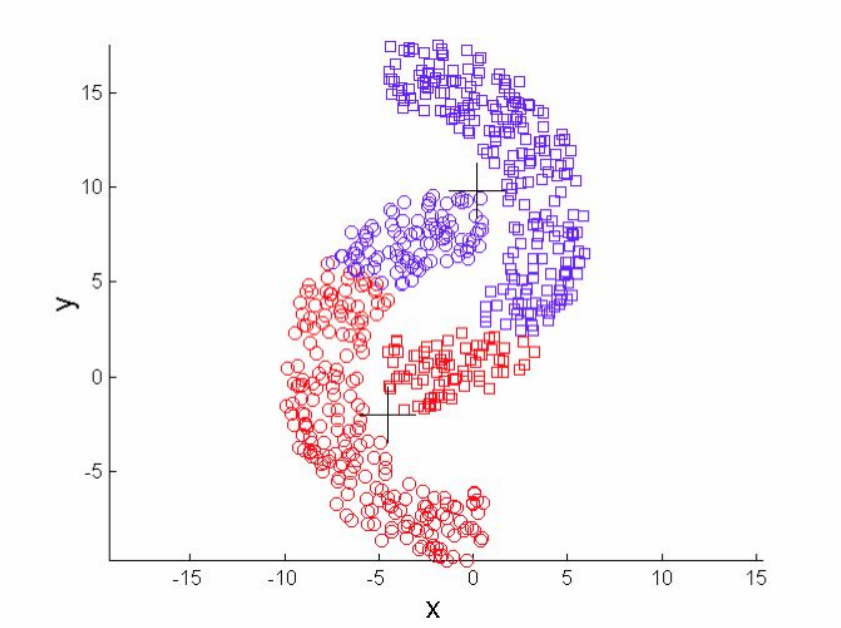
Limitations of K means continued.



Wants to minimize the spread here. Imagine the red balls were really far away.

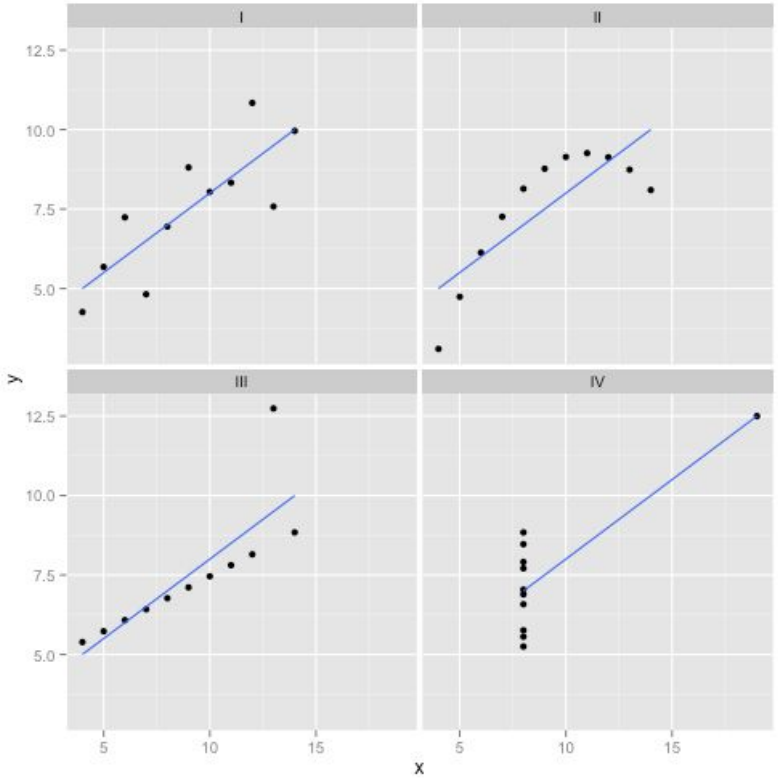
Limitations of K means continued.





Can’t separate points with nonlinear boundaries.

Square pegs into round holes



- Created in 1973 by statistician Francis Anscombe, this delightful concoction illustrates the folly of trusting statistical methods blindly. Each of the datasets has the same linear regression slope, intercept, p-value and R2- and yet at a glance we can see that only one of them,

- In **I**, it is appropriate for linear regression.
- In **II** it suggests the wrong shape,
- In **III** it is skewed by a single outlier- and in
- In **IV** there is clearly no trend at all!

Normally we solve this by evaluating on testing data, but we no longer can! Since we don't have labels!

Summary

- K means is a simple and effective clustering algorithm in many cases where you need to group things into a small number of categories.

- Like many algorithms, it's very simple to misuse it! It's essential to always have a good understanding of the underlying data before using any model. This is especially true in unsupervised learning, because we can't see how well the model generalizes to test data!

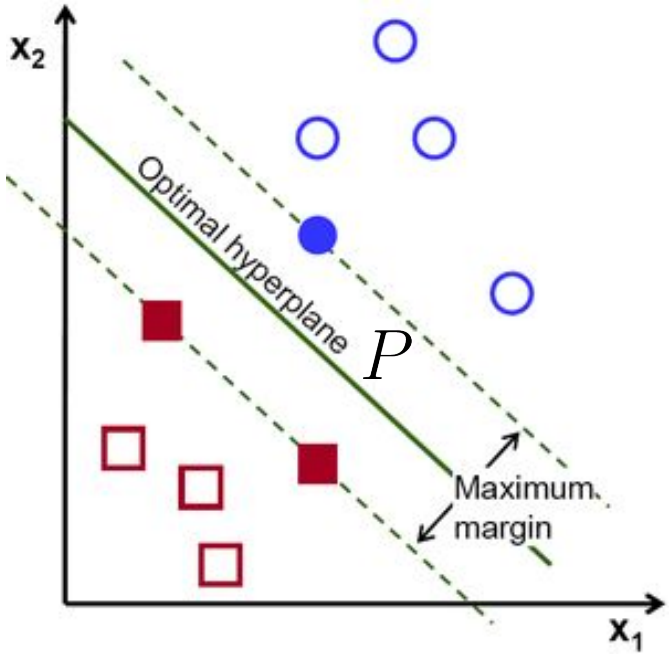
Support Vector Machines

Support Vector Machines

$$f(x) = \omega_0 + \omega^T \cdot x$$

$$|\omega_0 + \omega^T \cdot x| = 1$$

$$d(x, P) = \frac{|\omega_0 + \omega^T \cdot x|}{\|\omega\|} = \|\omega\|^{-1}$$



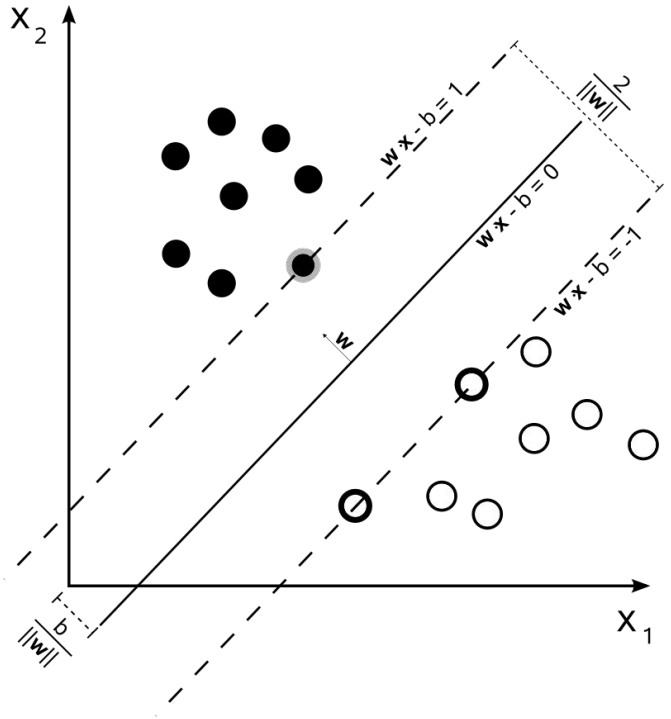
Goal: Create a maximally separating hyperplane between two classes, ie. choose P as to maximize d. Thus we which to maximize the norm of w.

Support Vector Machines

$$\text{minimize } \frac{1}{n}\sum_{i=1}^{n} \|w\|^2$$

$$\text{subject to } y_i(w \cdot x_i + b) \geq 1$$

- Note that maximizing $\|\omega\|^{-1}$ is the same as minimizing $\|\omega\|$
- This is a convex linear program and thus has a global unique solution.
- But we have just one problem! The constraint is probably not satisfied.



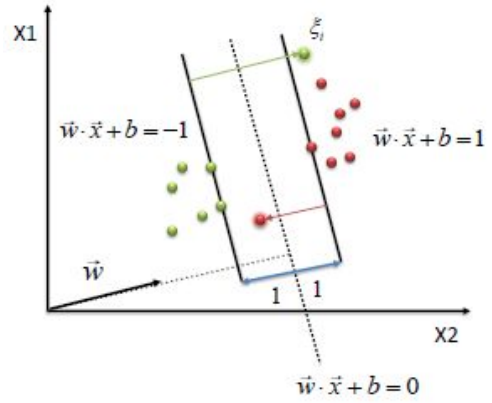
$$d(x, P) = \frac{|\omega_0 + \omega^T \cdot x|}{\|\omega\|} = \|\omega\|^{-1}$$

Support Vector Machines With Slack

$$\text{minimize } \|w\|^2 + C\frac{1}{n}\sum_{i=1}^{n}\zeta_i$$

$$\text{subject to } y_i(w \cdot x_i + b) \geq 1 - \zeta_i \text{ and } \zeta_i \geq 0, \text{ for all } i.$$

- Note that maximizing $\|\omega\|^{-1}$ is the same as minimizing $\|\omega\|$
- ζ_i is called the ***slack variable***. It accounts for the fact that data will never have perfect separation, so it relaxes the above constraint.
- C is the ***regularization strength***. If C = 0, the above problem doesn't care about getting the right answer at all.



slack variable:

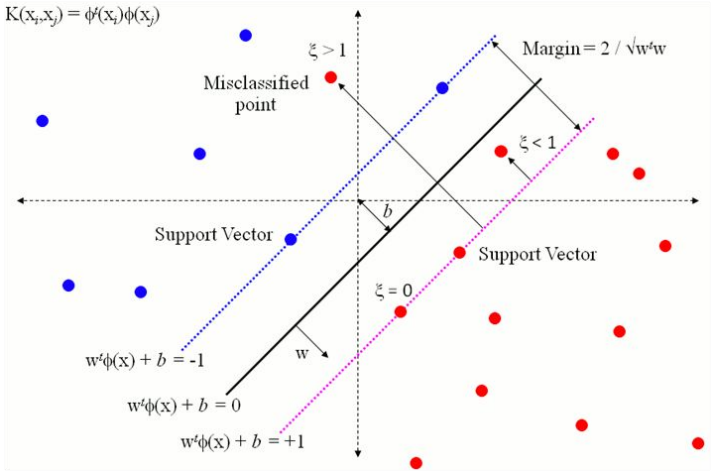
ξ_i

Allow some instances to fall off the margin, but penalize them

Support Vector Machines With Slack

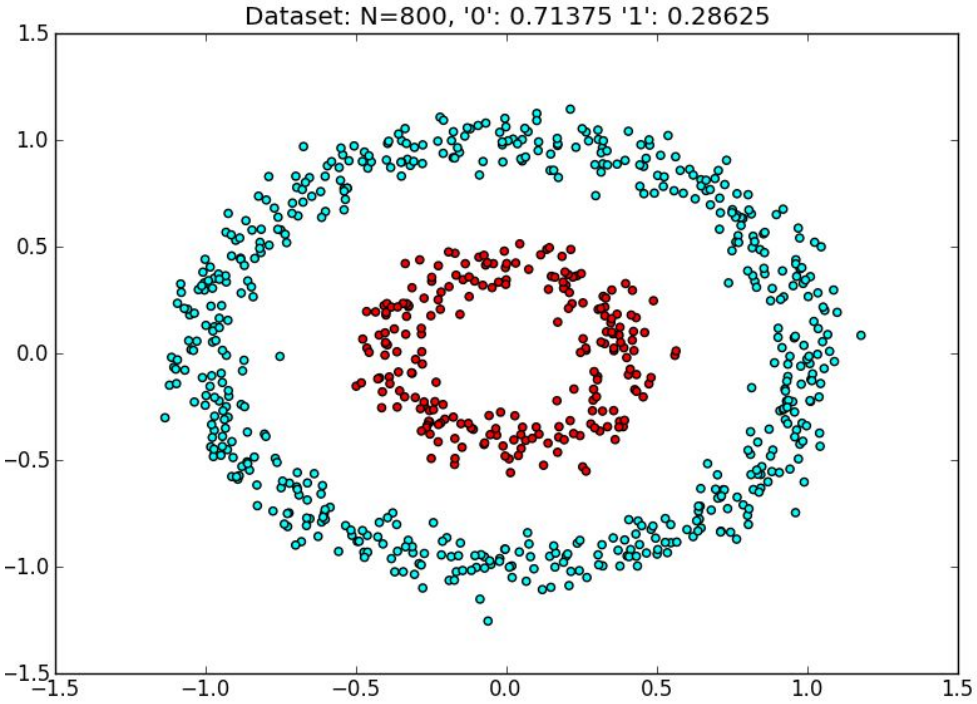
$$\text{minimize } \|w\|^2 + C\frac{1}{n}\sum_{i=1}^{n}\zeta_i$$

$$\text{subject to } y_i(w \cdot x_i + b) \geq 1 - \zeta_i \text{ and } \zeta_i \geq 0, \text{ for all } i.$$



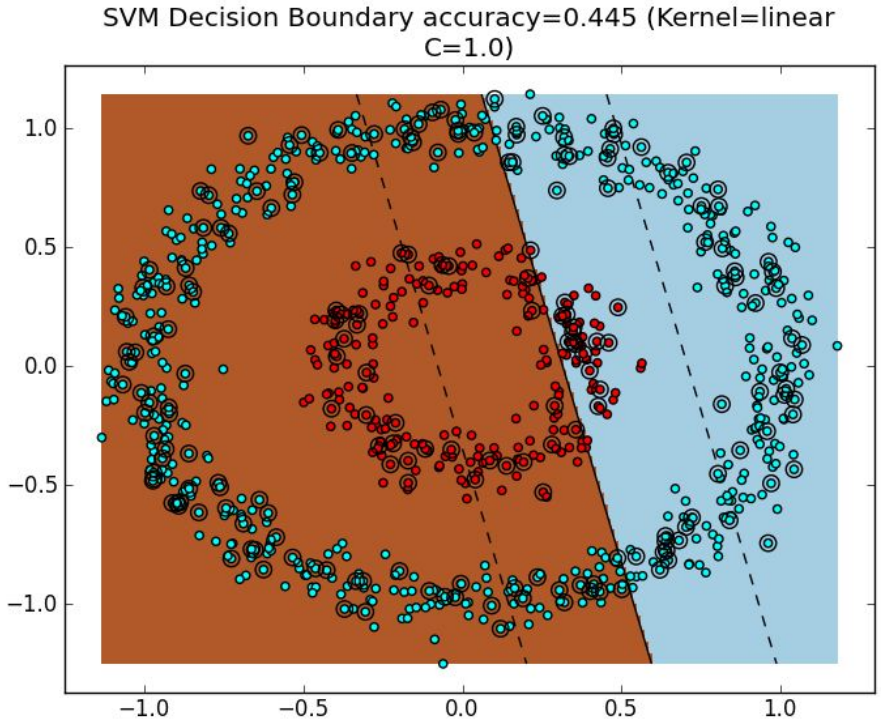
- If $0 \leq \zeta_i \leq 1$ then the point is on the right side, but not as far from the plane as we originally wanted.
- If $\zeta_i > 1$ then the point is incorrectly classified. It will be penalized more heavily.
- How much you want to penalize wrong answers depends on C.

Nonlinear example



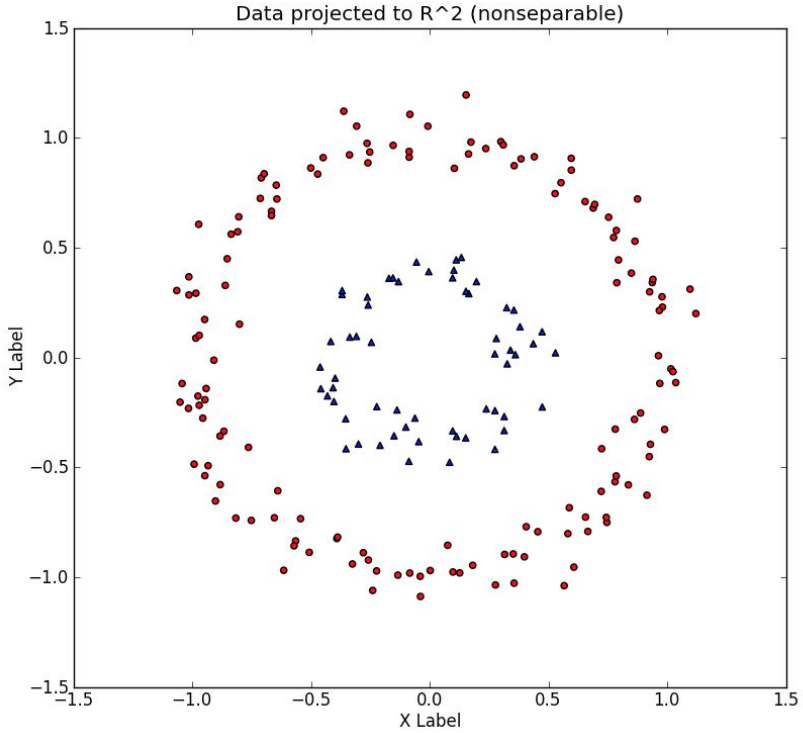
- In this example, a separating hyperplane won't work. So we're out of luck right?

Nonlinear example



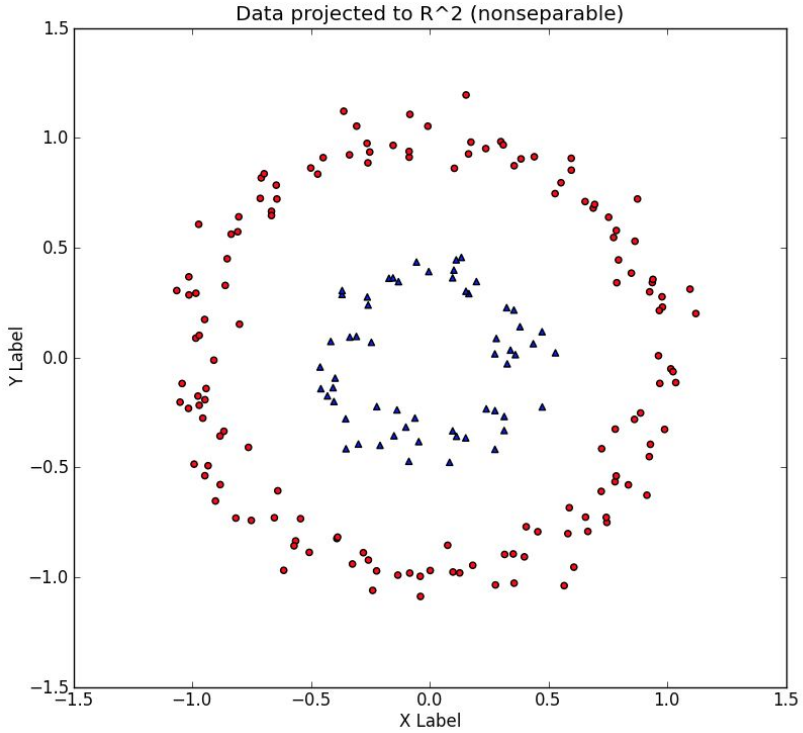
- In this example, a separating hyperplane won't work. So we're out of luck right?

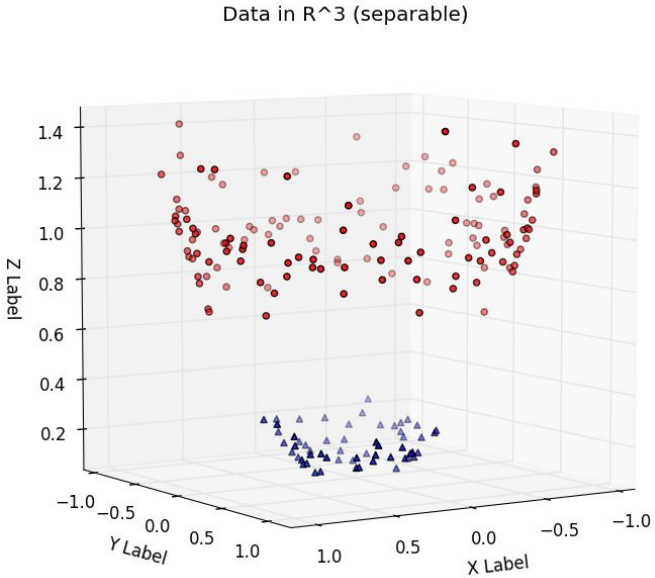
Kernel Trick



- What if we could learn a nonlinear decision boundary?
- What about mapping this to a higher dimensional space?

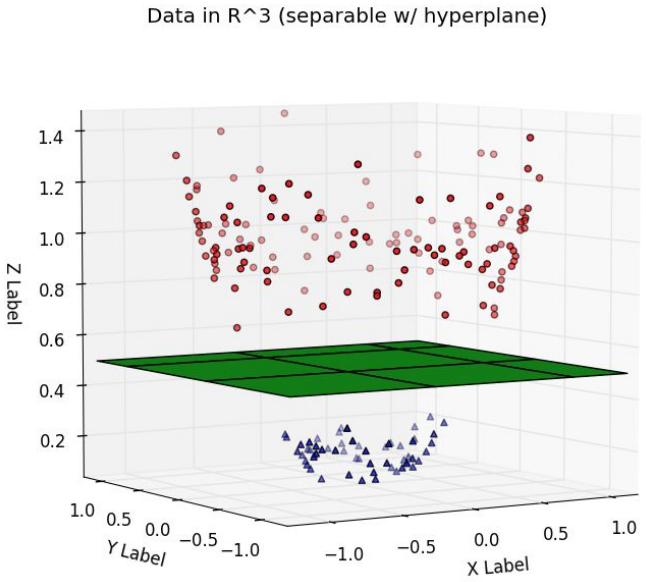
Kernel Trick

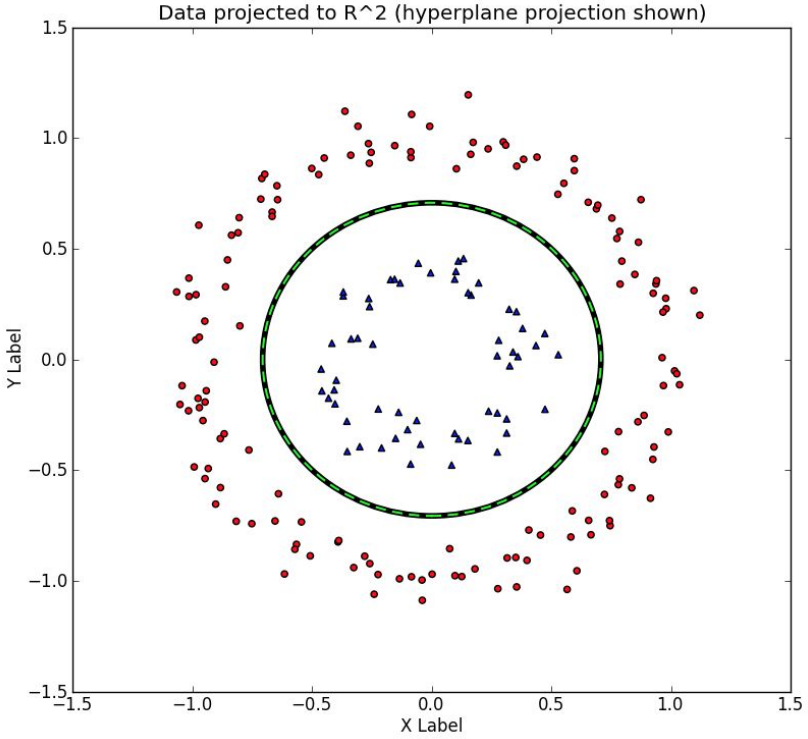




$$[x, y, x^2 + y^2]$$

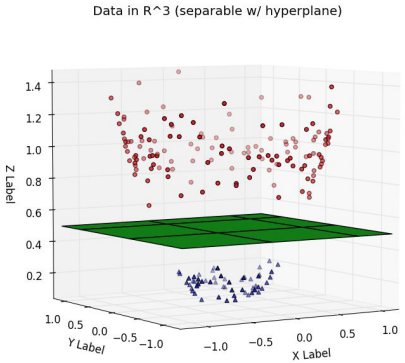
A solution in dimension 3

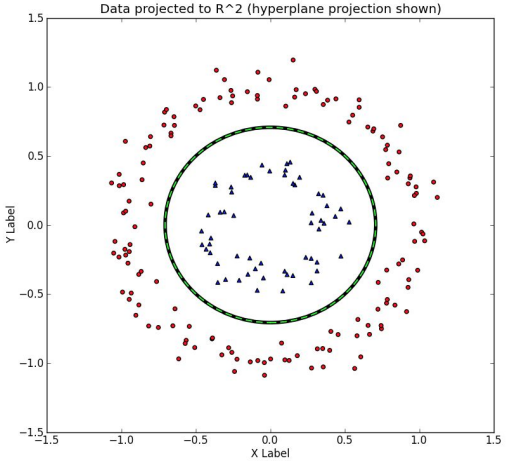




$$\phi((x, y)) = (x, y, x^2 + y^2)$$

A solution in dimension 3





$$\phi((x, y)) = (x, y, x^2 + y^2)$$

- One maps to a higher dimensional space where linear separation can be possible. Then find a solution, map back to lower dimensional space.
- But the trick is, we don't actually have to do this!
- How do we avoid mapping to higher dimensions to solve our problem?

Dual Problem

Original:

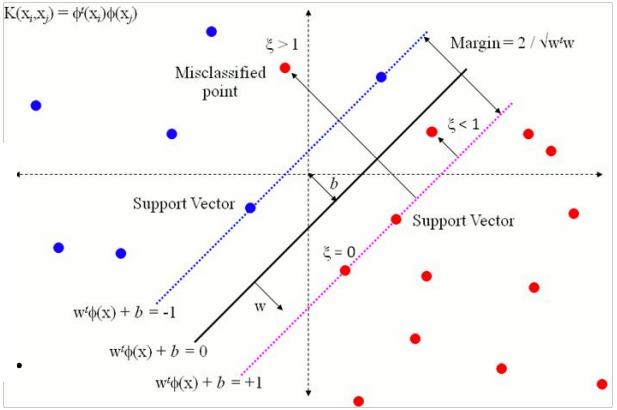
$$\text{minimize } \frac{1}{n}\sum_{i=1}^{n}\zeta_i + \lambda\|w\|^2$$

$$\text{subject to } y_i(w \cdot x_i + b) \geq 1$$

Dual Problem:

$$\text{maximize}_{\alpha} \quad \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j$$

$$\text{s.t.} \quad \forall i : \alpha_i \geq 0 \wedge \sum_{i=1}^{n} y_i \alpha_i = 0 \qquad w = \sum_{i} \alpha_i y_i \mathbf{x_i}$$



Why do we care about this at all?

Dual Problem

Original:

$$\text{minimize } \frac{1}{n}\sum_{i=1}^{n} \|w\|^2$$

$$\text{subject to } y_i(w \cdot x_i + b) \geq 1$$

Lagrangian:

$$\mathcal{L}(\omega, \omega_0, \alpha) := \sum_{k=1}^{d} \omega_k^2 - \sum_{i=1}^{N} \alpha_i [y_i(\omega \cdot \mathbf{x}_i + \omega_0) - 1]$$

$$\nabla_{\omega} \mathcal{L} = 0$$

$$\nabla_{\omega_0} \mathcal{L} = 0$$

We want to minimize the Lagrangian over the original variables, and maximize it over the Lagrange multipliers (since we expect the term in brackets to be positive).

Dual Problem

Lagrangian:

$$\mathcal{L}(\omega, \omega_0, \alpha) := \frac{1}{2}\sum_{k=1}^{d} \omega_k^2 - \sum_{i=1}^{N} \alpha_i [y_i(\omega \cdot \mathbf{x}_i + \omega_0) - 1]$$

$$\nabla_\omega \mathcal{L} = 0 \longrightarrow w = \sum_i \alpha_i y_i \mathbf{x_i} \qquad (1)$$

$$\nabla_{\omega_0} \mathcal{L} = 0 \longrightarrow \sum_i \alpha_i y_i = 0 \qquad (2)$$

Substituting (1) and (2) into the Lagrangian , we get, maximizing over alpha:

$$\text{maximize}_\alpha \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j$$

Dual Problem - Who Cares?

Dual Problem:

$$\text{maximize}_\alpha \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j$$

$$\text{s.t.} \quad \forall i : \alpha_i \geq 0 \wedge \sum_{i=1}^{n} y_i \alpha_i = 0$$

$$w = \sum_i \alpha_i y_i \mathbf{x_i}$$

This gives us an explicit expression for w.

Question: Why is this useful?

Dual Problem - Written in terms of Inner Products

Dual Problem:

$$\text{maximize}_\alpha \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j$$

$$\text{s.t.} \quad \forall i : \alpha_i \geq 0 \wedge \sum_{i=1}^{n} y_i \alpha_i = 0$$

Let's plug this in and see what happens:

$$w = \sum_i \alpha_i y_i \mathbf{x_i} \qquad \omega^T \cdot \mathbf{x} + \omega_0 = \sum_i \alpha_i y_i \langle \mathbf{x_i}, \mathbf{x} \rangle_V + \omega_0$$

$$\tilde{w} = \sum_i \alpha_i y_i \phi(\mathbf{x_i}) \qquad \omega^T \cdot \phi(\mathbf{x}) + \omega_0 = \sum_i \alpha_i y_i \langle \phi(\mathbf{x_i}), \phi(\mathbf{x}) \rangle_V + \omega_0$$

Dual Problem

$$\omega^T \cdot \phi(\mathbf{x}) + \omega_0 = \sum_i \alpha_i y_i \langle \phi(\mathbf{x_i}), \phi(\mathbf{x}) \rangle_V + \omega_0$$

$$\omega^T \cdot \mathbf{x} + \omega_0 = \sum_i \alpha_i y_i \langle \mathbf{x_i}, \mathbf{x} \rangle_V + \omega_0$$

Recall we defined K as:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_1) \rangle_V$$

This is only in terms of inner products! Hmmm! But wait, we defined K as a Kernel on our original variables.

Therefore:

$$\omega^T \cdot \phi(\mathbf{x}) + \omega_0 = \sum_i \alpha_i y_i K(\mathbf{x_i}, \mathbf{x}) + \omega_0$$

The Kernel Trick

$$K(\mathbf{x}_1, \mathbf{x}_2) = \langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_1) \rangle_V$$

$$\omega^T \cdot \phi(\mathbf{x}) + \omega_0 = \sum_i \alpha_i y_i K(\mathbf{x_i}, \mathbf{x}) + \omega_0$$

- It now remains to optimize over $\{\alpha_i\}$.
- But note that we are now solving the problem with the same number of variables, but in a higher dimension!
- This is known as the **kernel trick**. It holds as long as K can be realized as an inner product on some larger space (maybe infinite dimensional!). The precise conditions are in Mercer's Theorem.

Examples of Kernels

- Linear $\mathbf{x} \cdot \mathbf{v}$
- Polynomial $(r + \mathbf{x} \cdot \mathbf{v})^d \text{ for } r \geq 0, d \geq 0$
- Radial Basis Function $\exp(-\gamma \|\mathbf{x} - \mathbf{v}\|^2), \text{ for } \gamma > 0$
- Gaussian $\exp(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{v}\|^2)$

We only need the dot products!

It turns out that the SVM has no need to explicitly work in the higher-dimensional space at training or testing time. One can show that during training, the optimization problem only uses the training examples to compute **pairwise** dot products.

Why is this significant? It turns out that there exist functions that, given two vectors v and w in , they implicitly computes the dot product between v and w in a higher-dimensional data **without explicitly transforming v and w to higher dimensions** . Such functions are called **kernel** functions,

<u>The implications are:</u>

1. By using a kernel , we can implicitly transform datasets to a higher-dimensional using no extra memory, and with a minimal effect on computation time.
2.
    - The only effect on computation is the extra time required to compute . Depending on , this can be minimal.
    -
3. By virtue of (1), we can efficiently learn nonlinear decision boundaries for SVMs simply by **replacing all dot products in the SVM computation with our kernel, if it satisfies certain properties !**

What SVM special? Linear Regression

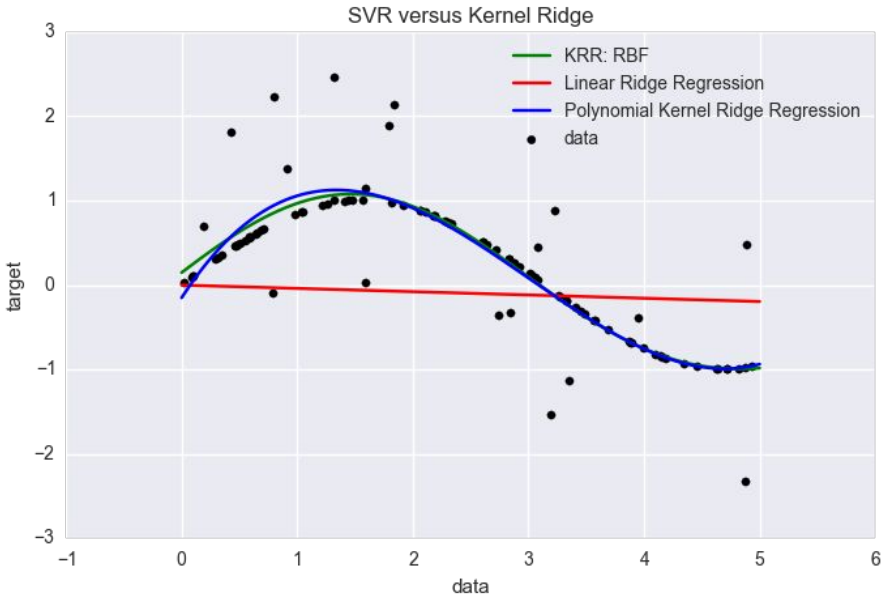
Original OLS:

Minimize: $\frac{1}{2}\sum_{i=1}^{N}(y_i - \mathbf{w}\cdot\mathbf{x_i})^2 \longrightarrow \mathbf{w} = (X^TX)^{-1}X^T\mathbf{y}$

$$K_{ij} = K(\mathbf{x_i}, \mathbf{x_j})$$

Minimize: $\sum_{i=1}^{N}(y_i - K(\mathbf{w}, \mathbf{x_i}))^2 \longrightarrow \mathbf{w} = (K^TK)^{-1}K^T\mathbf{y}$

Performance Comparison of Kernels



- Linear

$$\mathbf{X} \cdot \mathbf{V}$$

- Polynomial

$$(r + \mathbf{x} \cdot \mathbf{v})^d \text{ for } r \geq 0, d \geq 0$$

- Radial Basis Function

$$\exp(-\gamma \|\mathbf{x} - \mathbf{v}\|^2), \text{ for } \gamma > 0$$

- Gaussian

$$\exp(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{v}\|^2)$$

Conclusion - Representer Theorem

It turns out that almost all problems can be written as dot products, and therefore can be replaced with kernels! There is a powerful theorem the Representor Theorem which says:

Theorem 6.1. *Representer theorem: Let* Ω*:* $[0, +\infty) \to \mathbb{R}$ *be strictly increasing and let* ℓ*:* $(X \times Y \times \mathbb{R})^n \to \mathbb{R} \cup \{+\infty\}$ *by a loss function. Consider:*

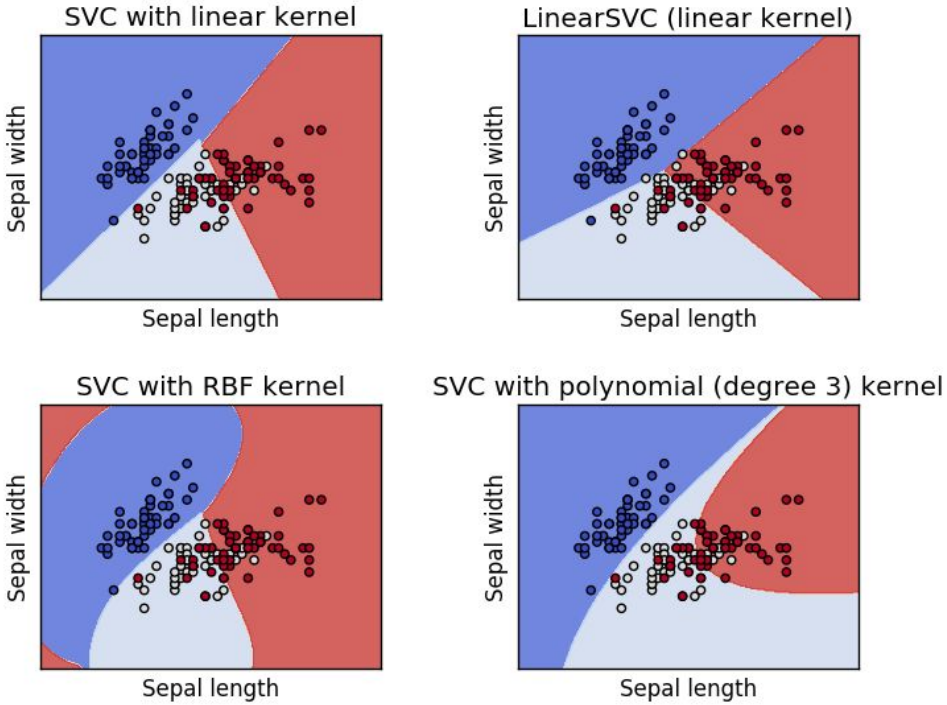
$$min_{f \in \mathcal{H}} \ell(x^{(i)}, y^{(i)}, f(x^{(i)})) + \lambda_n \Omega(\|f\|_{\mathcal{H}}^2) \tag{6.1}$$

where $\mathcal{H}$ *is an RKHS with kernel* $\mathbb{K}$*. Then any optimal solution has the following form:*

$$f(\cdot) = \sum_{i}^{n} \alpha_i \mathbb{K}(\cdot, x^{(i)}) \tag{6.2}$$

In the above form, the α_i are the data dependent weights and $\mathbb{K}(\cdot, x^i)$ is the kernel function centered at x^i.

Multiclass classification with different kernels



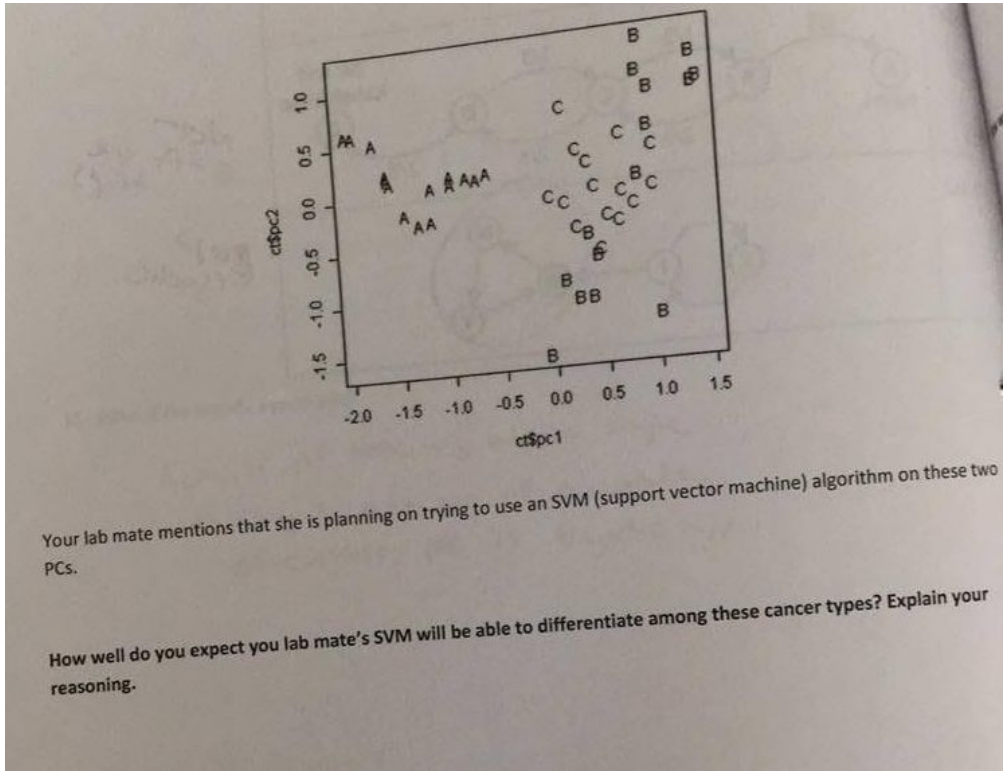
- For multiple classes, we use the *‘one vs all approach’*.
- Take one of the classes, split with respect to your desired kernel.
- Repeat this with the remaining sub-classes.

When to use which kernel?

Key Points (from Andrew Ng, Professor of Machine Learning at Stanford)

- Use linear kernel when number of features is larger than number of observations.
- Use gaussian kernel when number of observations is larger than number of features.
- If number of observations is larger than 50,000 speed could be an issue when using gaussian kernel; hence, one might want to use linear kernel.

Should this work for linear SVM?

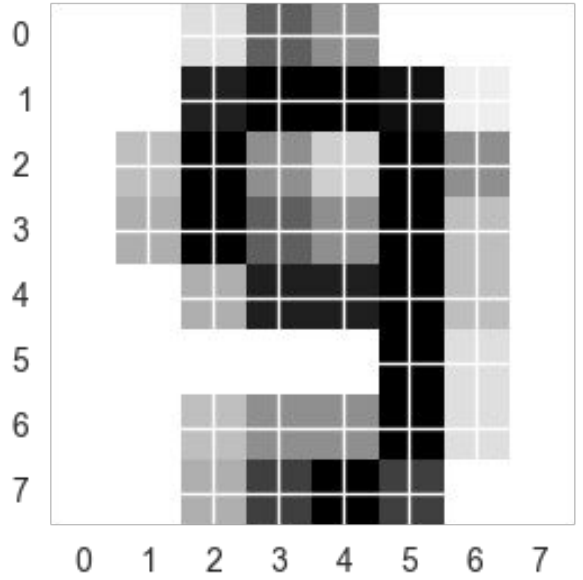


Work through this example, thinking carefully about how to use the one vs all approach, and which kernels would work vs which would not.

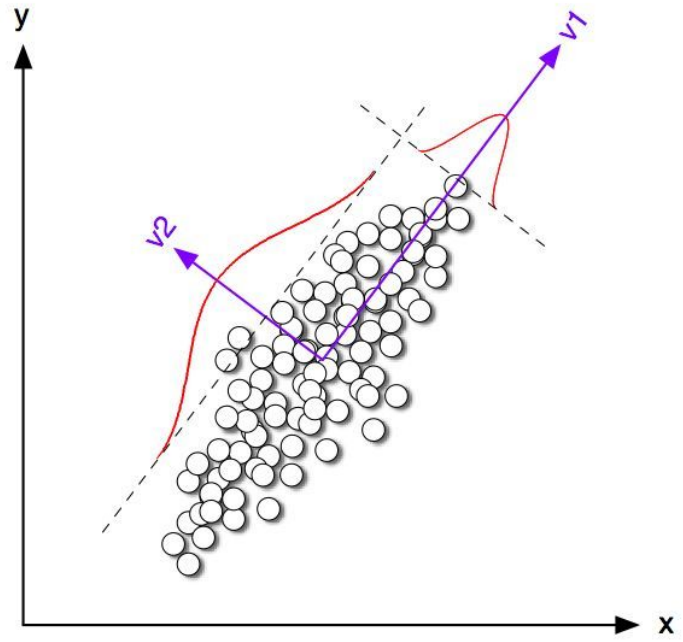
Principal Component Analysis

MNIST digit data set

- Recall each image is represented by 8x8 values which range from 0 to 255.
- The matrix is sparse, and not much information is included in many of the pixels, or *‘directions’* in the space for that matter.
- How do we reduce the dimension in such a way that saves the *“important”* information?



Recall the covariance matrix



$$[\mathrm{Corr}]_{ij} = \frac{\mathbf{x_i} \cdot \mathbf{x}_j}{\|\mathbf{x_i}\|\|\mathbf{x_j}\|} = \cos(\delta_{ij})$$

We wish to find the directions in 64 dimensional space which explain most of the variance. How do we do this?

Eigenvalues of Covariance Matrix

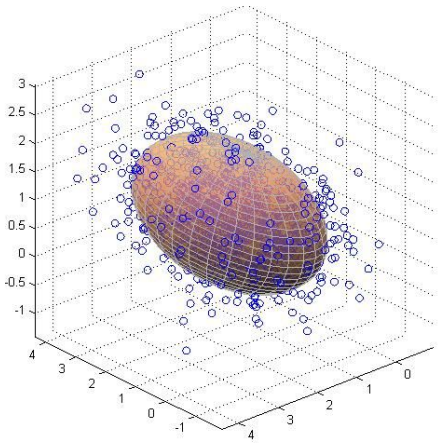
$$\lambda_1 = \max_{\|y\|=1} y^T A y$$

$$y^T A y = \langle y, Ay \rangle$$

$$y = v + \epsilon w$$

$$\nabla_\epsilon (y^T A y)\Big|_{\epsilon=0} = \langle w, Av \rangle + \langle v, Aw \rangle = \lambda \langle v, w \rangle$$

$$2\langle Av, w \rangle = \lambda \langle v, w \rangle \longrightarrow Av = \tilde{\lambda} v$$



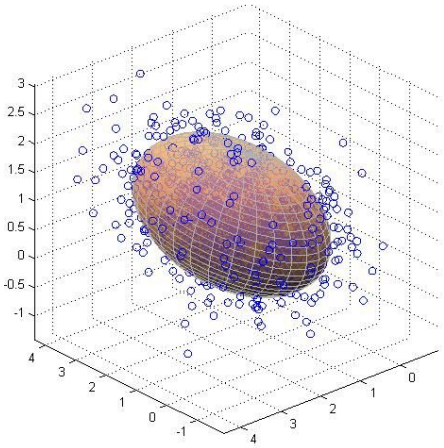
The others are obtained by maximizing in the orthogonal complement to the vector v.

Eigenvalues of Covariance Matrix

- The others are obtained by maximizing in the orthogonal complement to the vector v.
- In the end we have a collection of eigenvalues,

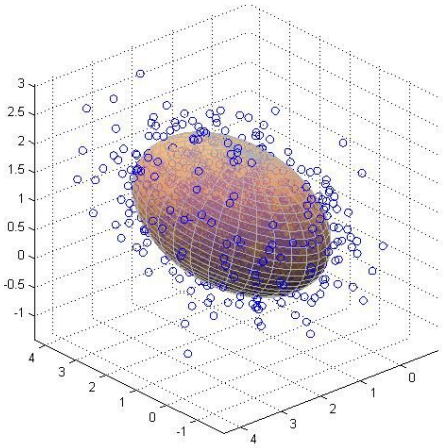
$$\lambda_1 > \lambda_2 > \cdots > \lambda_n$$

- The corresponding eigenvectors, $v_1, v_2, \cdots, v_n$ are known as the **Principal Components.**



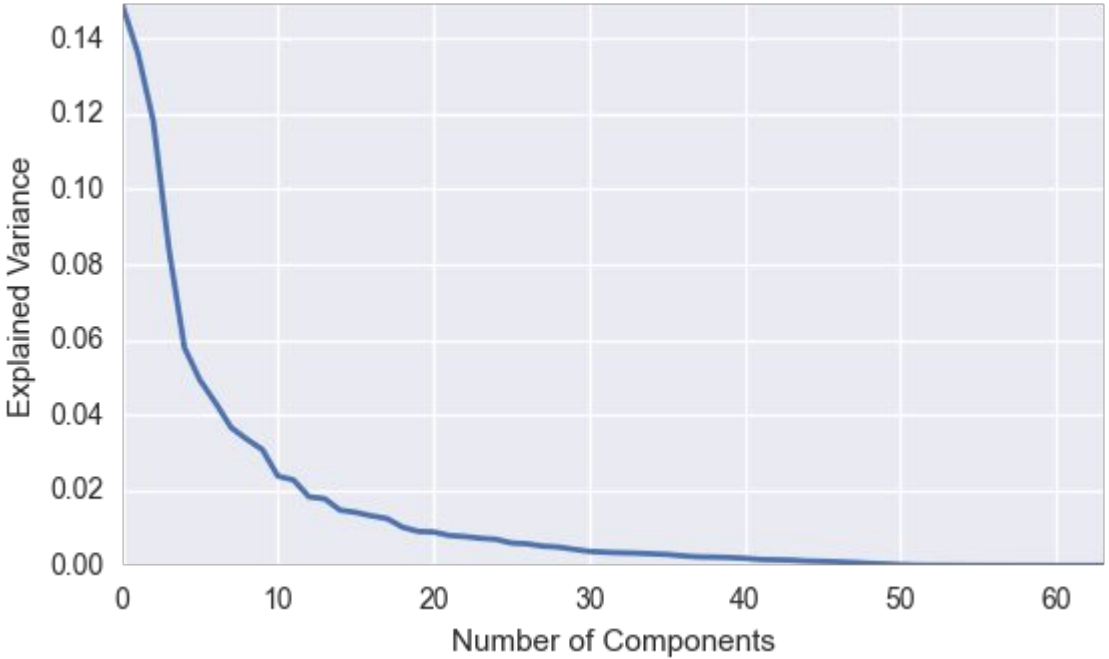
Dimensionality Reduction

$$y_k = f(v_1, v_2, \cdots, v_k)$$



Thus we simply are trying to make predictions based on a reduced subset of the variables.

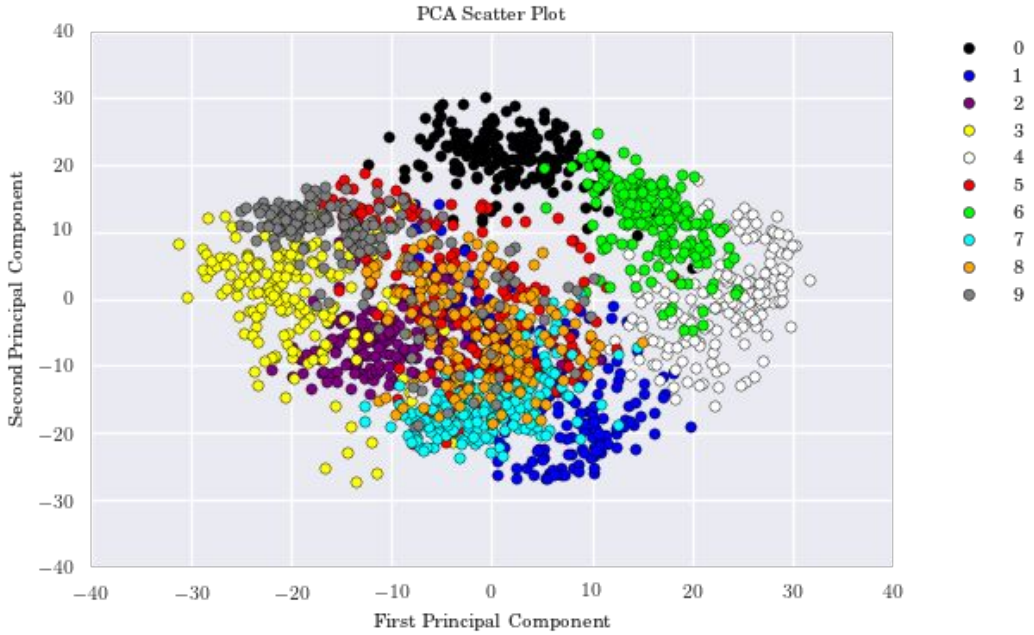
PCA on MNSIT dataset - explained variance.



The plot ot the left shows variance explained as a function of the number of principal components.

These are just the normalized eigenvalues

Top 2 components



We already see here that there is quite a lot of clustering of certain numbers based on only two of the components - we started off with 64!

Eigenvalue projections

Original number of features: 1850

	precision	recall	f1-score	support
Ariel Sharon	0.78	0.54	0.64	13
Colin Powell	0.81	0.87	0.84	60
Donald Rumsfeld	0.95	0.67	0.78	27
George W Bush	0.83	0.98	0.90	146
Gerhard Schroeder	0.95	0.80	0.87	25
Hugo Chavez	1.00	0.53	0.70	15
Tony Blair	0.97	0.78	0.86	36
avg / total	0.87	0.86	0.85	322

After eigenvalue decomposition: 150 features

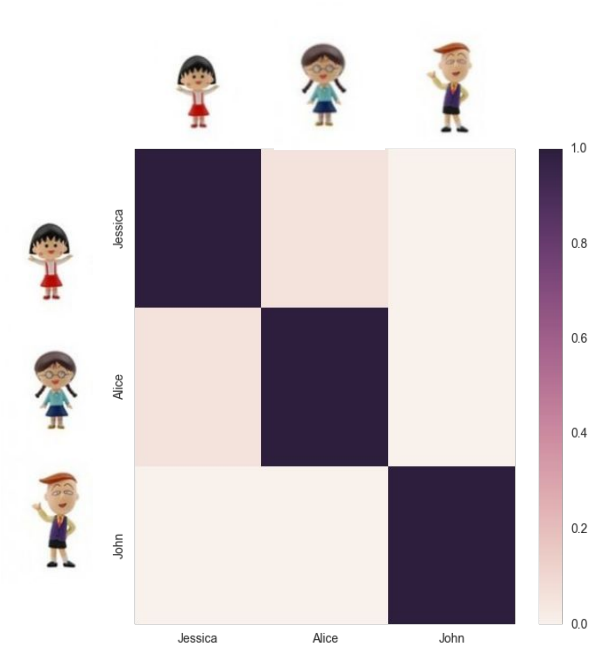
- Predictions made with 150 features

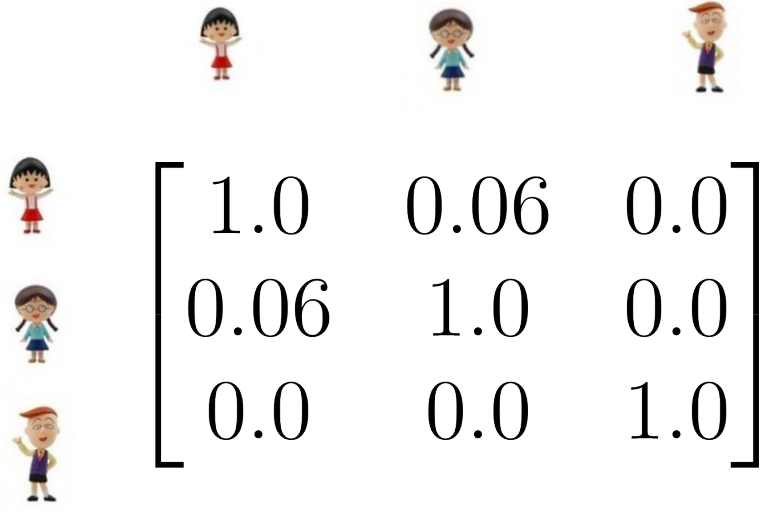
K Nearest Neighbors (KNN)

K Nearest Neighbors

$$[\mathrm{Corr}]_{ij} = \frac{\mathbf{x_i} \cdot \mathbf{x}_j}{\|\mathbf{x_i}\|\|\mathbf{x_j}\|} = \cos(\delta_{ij})$$

$$XX^T$$





$$\begin{bmatrix} 1.0 & 0.06 & 0.0 \\ 0.06 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

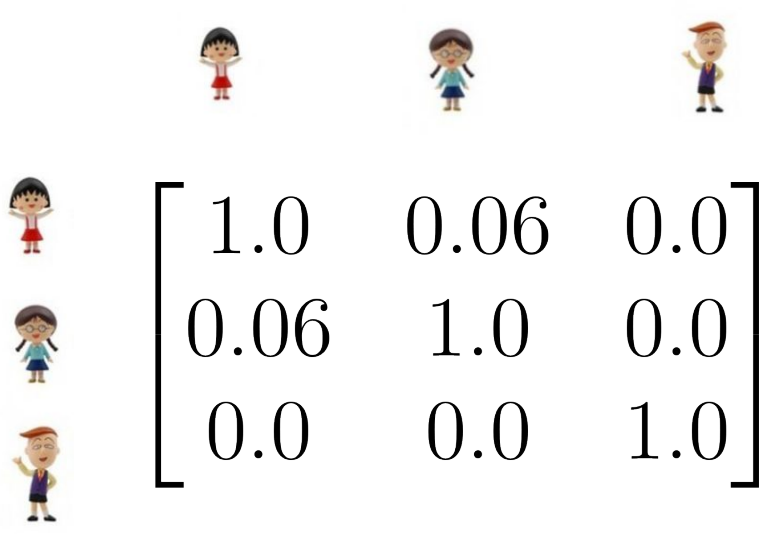
K Nearest Neighbors

1. Compute a distance between items/users.

2. Rank according to highest correlation/smallest cosine distance, or smallest euclidian distance (it's important to think about which distance to use).

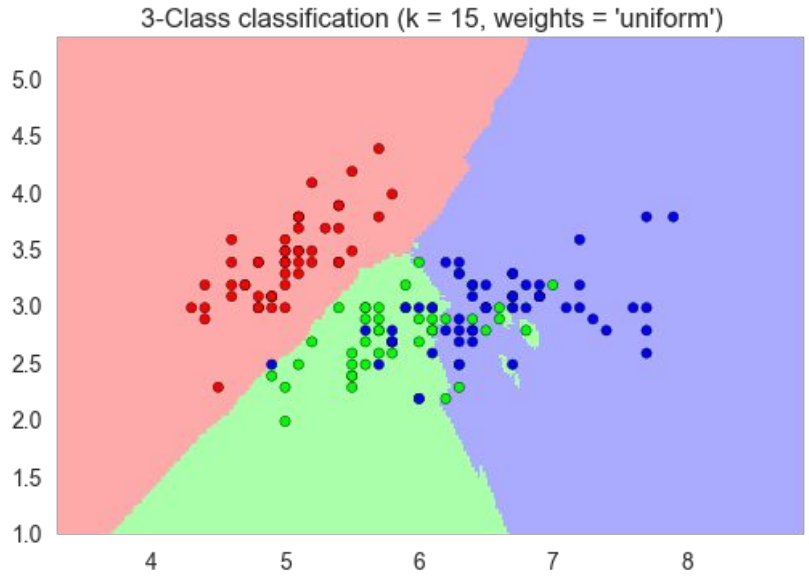
3. Choose a fixed K. Each member in the top K belongs to a particular group (note there will probably be more than K groups now).

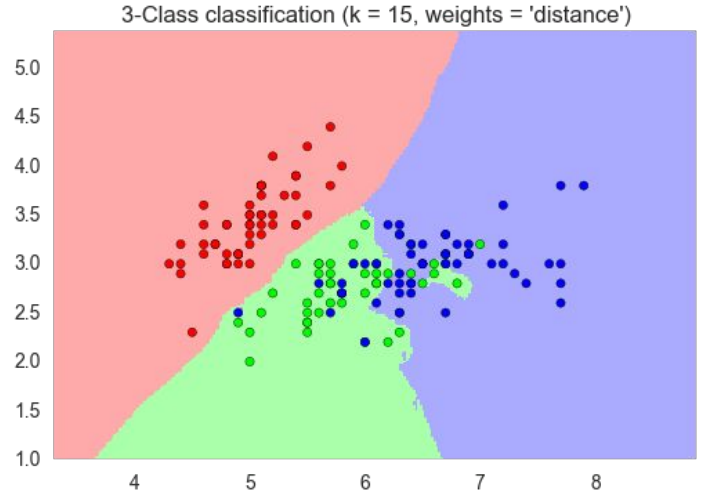
$$[\mathrm{Corr}]_{ij} = \frac{\mathbf{x_i} \cdot \mathbf{x}_j}{\|\mathbf{x_i}\|\|\mathbf{x_j}\|} = \cos(\delta_{ij})$$



$$\begin{bmatrix} 1.0 & 0.06 & 0.0 \\ 0.06 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Example

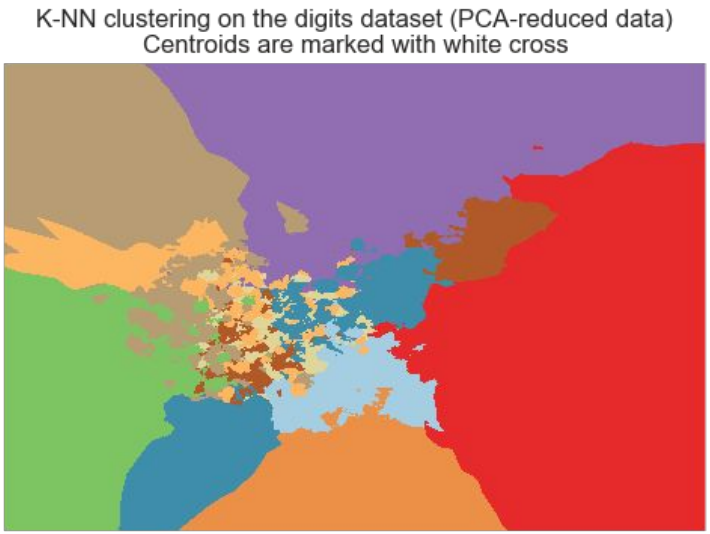


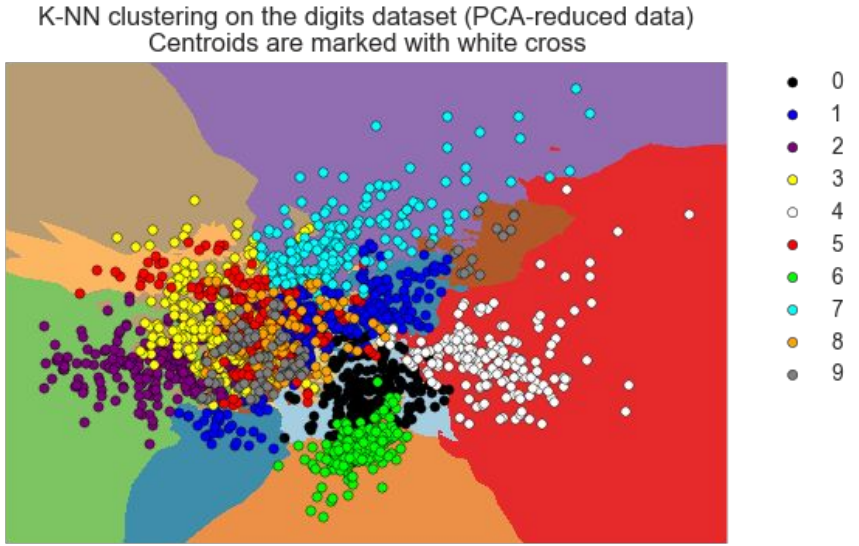


The default value, weights = 'uniform', assigns equal weights to all points. weights = 'distance' assigns weights proportional to the inverse of the distance from the query point. Alternatively, a user-defined function of the distance can be supplied, which will be used to compute the weights.

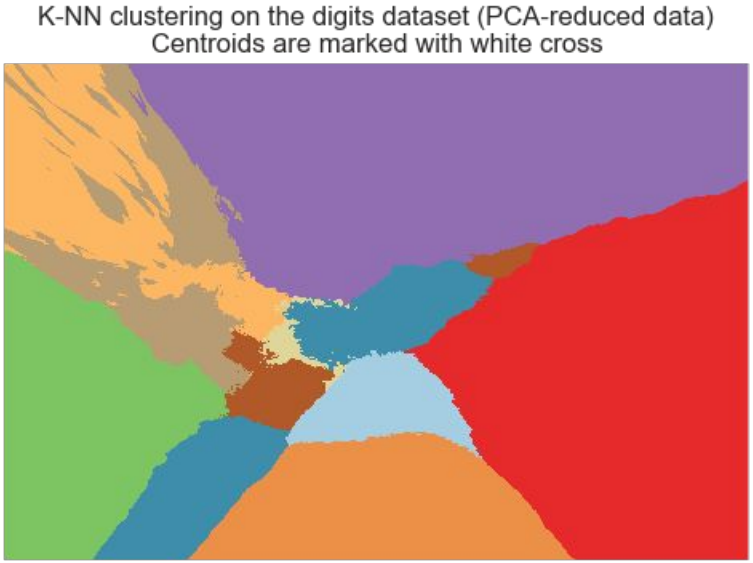
Question: Why is cosine distance here the wrong choice?

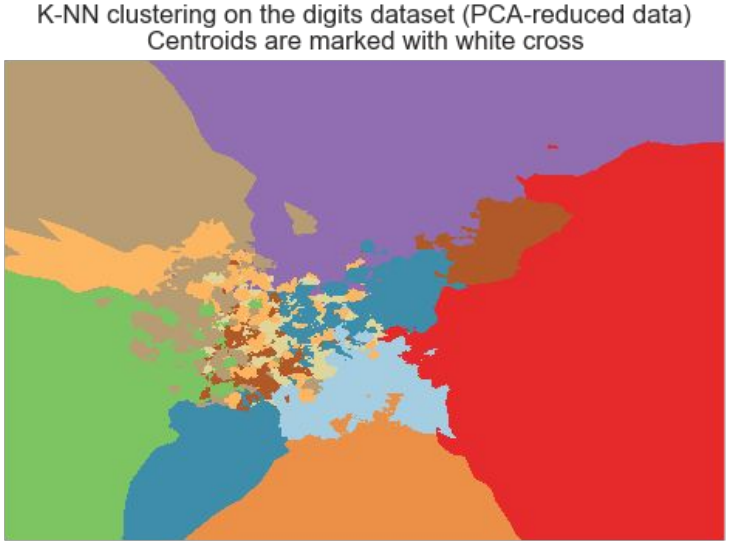
K nearest neighbors prediction (k=5)





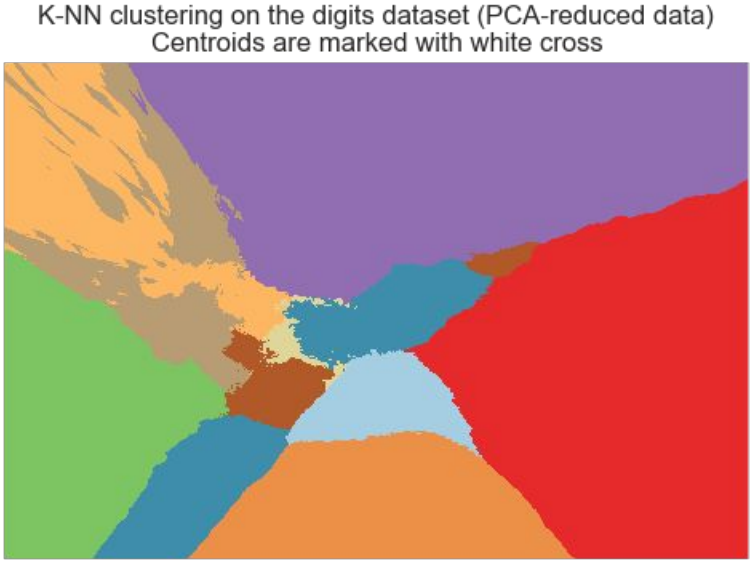
K nearest neighbors prediction (k=50 vs k=5)

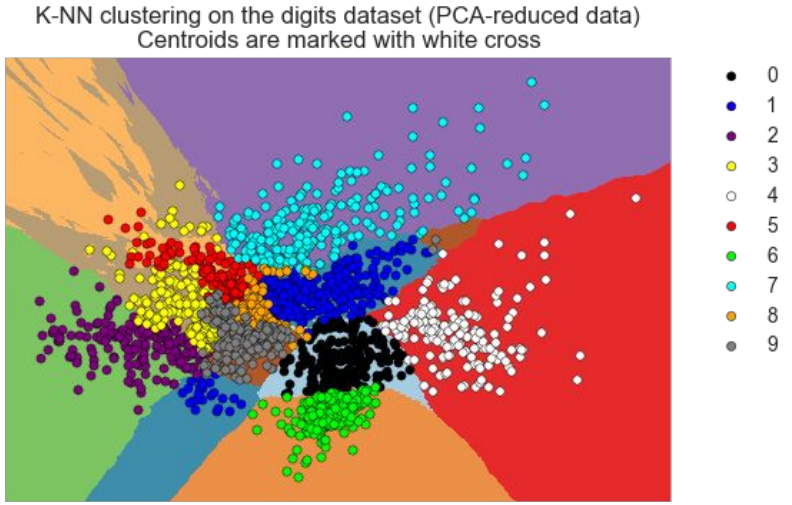




Notice how more neighbors results in smoothing the boundaries.
This can be seen as a regularization term.

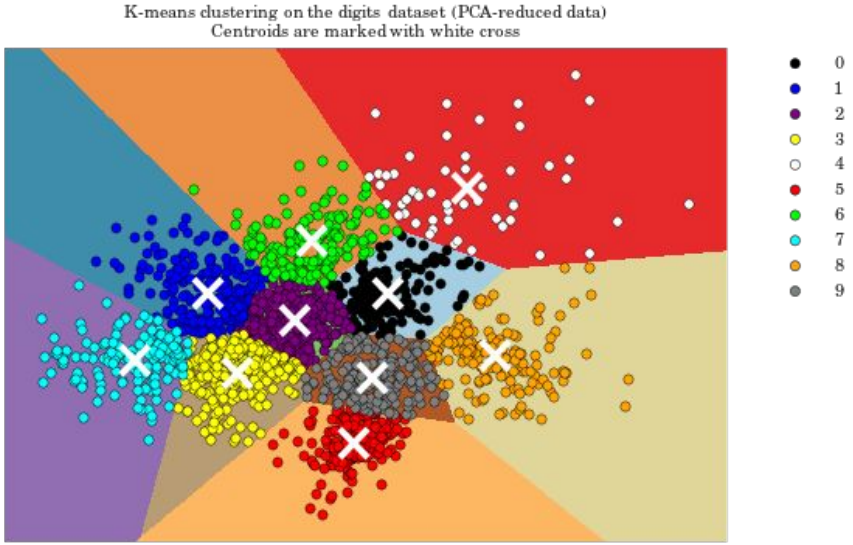
K nearest neighbors prediction (k=5)

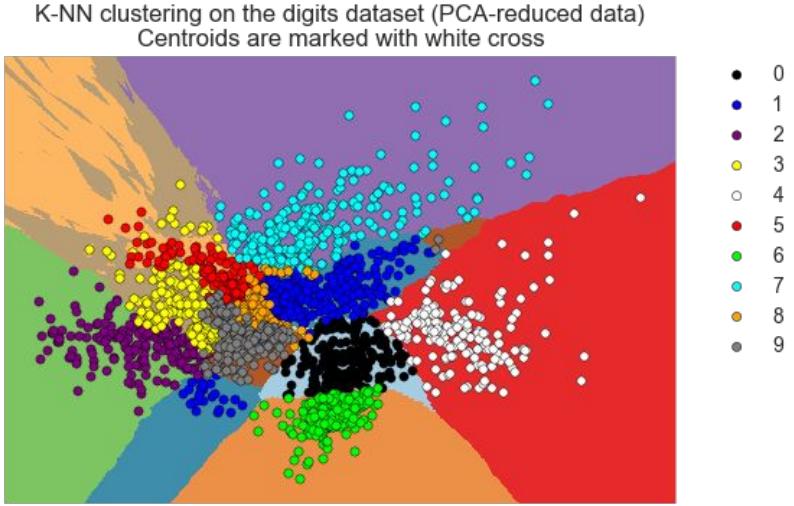




Notice how the assignments are much less noisy.

K means vs KNN





KNN and K Means have different goals - KNN is for classification, while K means is for clustering. Both can be used for either though if you are looking at labeled data.

Don’t confuse KNN and K Means

- K Means is a clustering algorithm for unlabeled data. The K is the number of clusters. For KNN, the K is the number of neighbors to use - therefore the Ks are completely unrelated.
- K means can be used to group a collection of items into a smaller set of groups. For instance

Don’t confuse KNN and PCA

- KNN assigns rows in your dataframe to the same group if they are the K closest elements.

- PCA reduces the actual number of columns in your dataframe.

Summary

- K Nearest Neighbors provides a simple clustering method which also allows for non-linear boundaries.

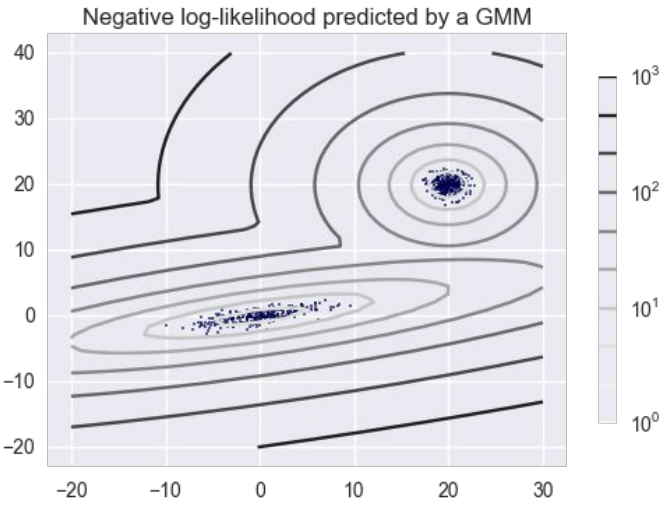
- The choice of metric is important! In the last example, cosine distance normalizes everything to the unit sphere, and we want to account for the magnitude. For recommendation algorithms, sometimes you don’t want to. Think carefully about this.

- Can be computationally difficult when there are many features, which is why we usually combine it with some sort of dimensionality reduction.

Gaussian Mixture Models

Gaussian Mixture Models

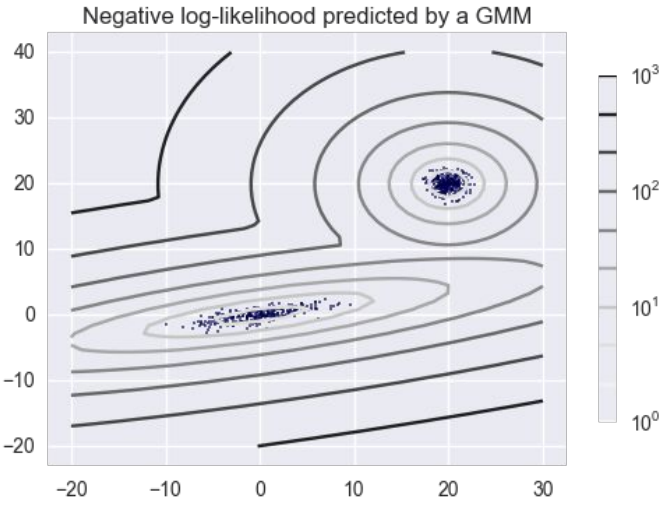
$$X \sim \pi\mathcal{N}(\mu_1, \sigma_1) + (1 - \pi)\mathcal{N}(\mu_2, \sigma_2)$$



A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.

Gaussian Mixture Models

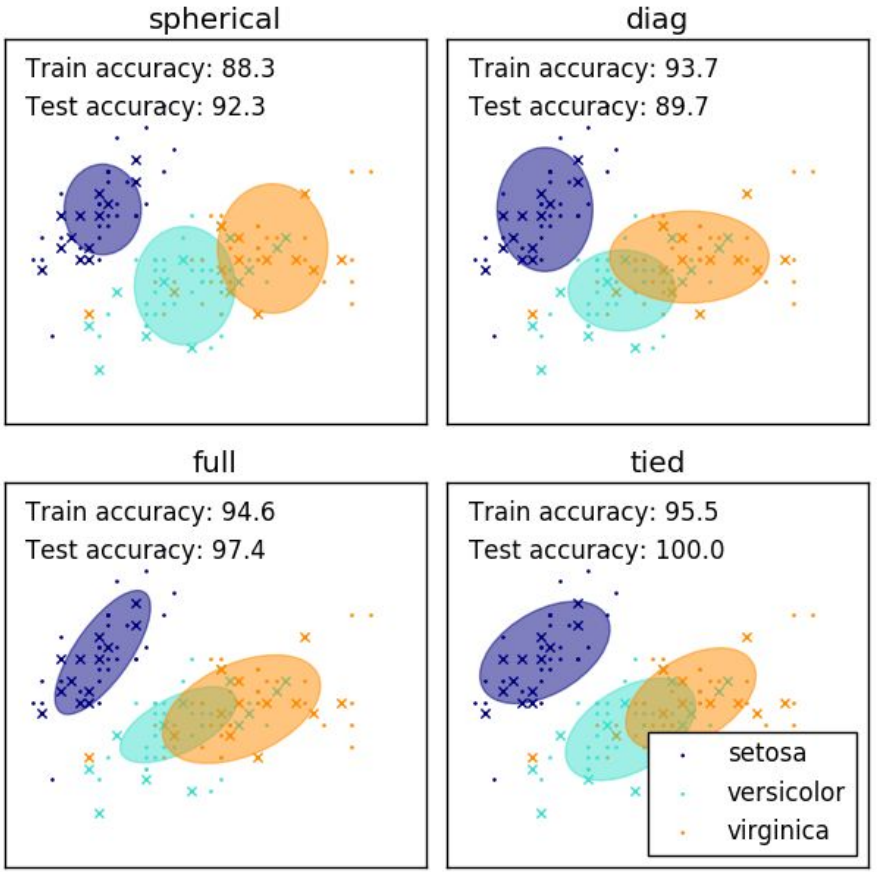
$$X \sim \pi\mathcal{N}(\mu_1, \sigma_1) + (1 - \pi)\mathcal{N}(\mu_2, \sigma_2)$$



- X is the collection of data (unlabeled).
- Are the two normal distributions that you're trying to fit the data to with means/variances μ_i, σ_i
- π represents the portion of the distributions

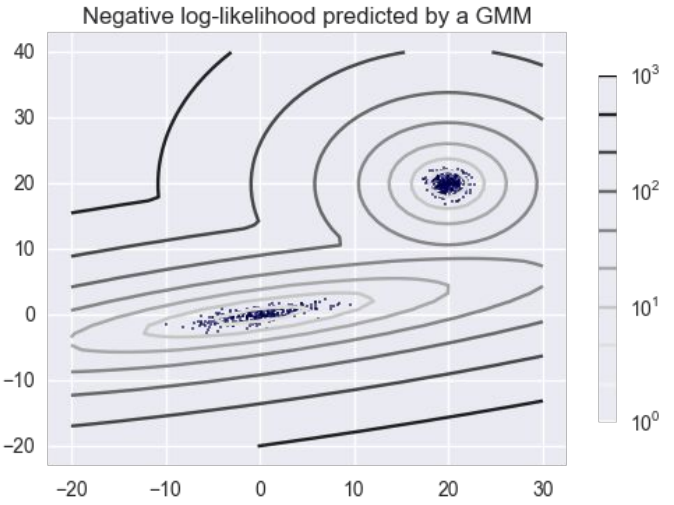
Some properties

- MM is *a lot* more **flexible** in terms of **cluster covariance.**
- k-means is actually a special case of GMM in which each cluster's covariance along all dimensions approaches 0.
- Think of rotated and/or elongated distribution of points in a cluster, instead of spherical as in k means.
- Allows for *mixed membership* - ie. a New York Times article may belong to several categories, not just one.



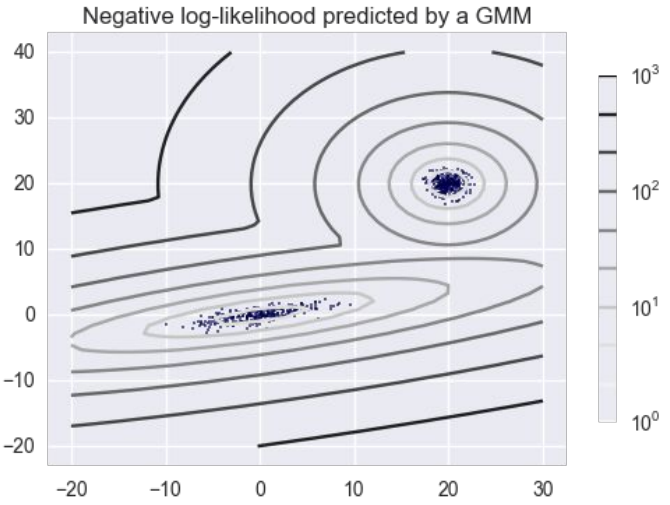
Likelihood function

$$X \sim \pi\mathcal{N}(\mu_1, \sigma_1) + (1 - \pi)\mathcal{N}(\mu_2, \sigma_2)$$



$$p(X|\mu_i, \sigma_i) = \prod_{k=1}^{N} \left(\frac{\pi}{\sqrt{2\pi\sigma_1^2}} e^{-(x_k-\mu_1)^2/\sigma_1^2} + \frac{1-\pi}{\sqrt{2\pi\sigma_2^2}} e^{-(x_k-\mu_2)^2/\sigma_1^2} \right)$$

Likelihood function



Goal:

$$\max_{\mu_i, \sigma_i} \log p(X|\mu_i, \sigma_i)$$

$$p(X|\mu_i, \sigma_i) = \prod_{k=1}^{N} \left(\frac{\pi}{\sqrt{2\pi\sigma_1^2}} e^{-(x_k-\mu_1)^2/\sigma_1^2} + \frac{1-\pi}{\sqrt{2\pi\sigma_2^2}} e^{-(x_k-\mu_2)^2/\sigma_1^2} \right)$$

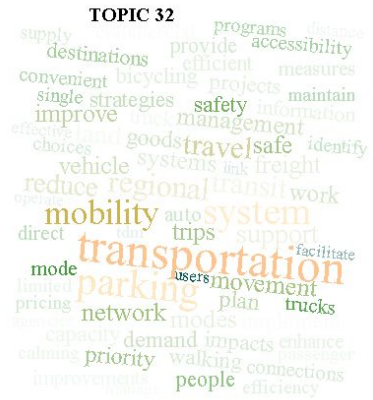
Gaussian Mixture Models

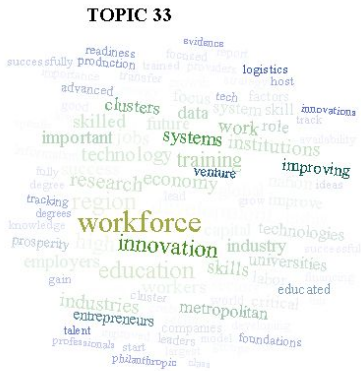
$$X \sim \pi\mathcal{N}(\mu_1, \sigma_1) + (1 - \pi)\mathcal{N}(\mu_2, \sigma_2)$$

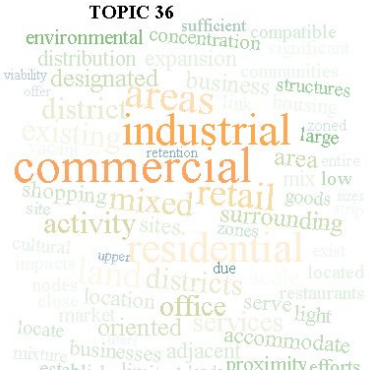
Latent Dirichlet Allocation (LDA)

Introduction to topic models

Topic Modeling







Latent Dirichlet Allocation (LDA)

- α is the parameter of the Dirichlet prior on the per-document topic distributions,
- β is the parameter of the Dirichlet prior on the per-topic word distribution,
- θ_m is the topic distribution for document *m*.
- φ_k is the word distribution for topic *k*.
- z_{mn} is the topic for the *n*-th word in document *m*, and
- w_{mn} is the specific word.