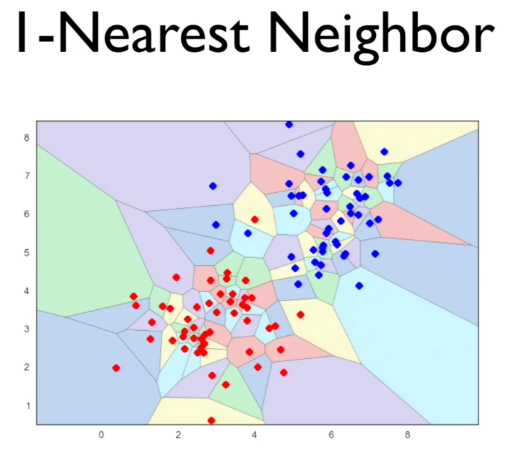
Classification, kNN, Cross-validation, Dimensionality Reduction, Part 1 & 2

[[Link](https://www.springboard.com/workshops/data-science-career-track/learn#/curriculum/18804)]

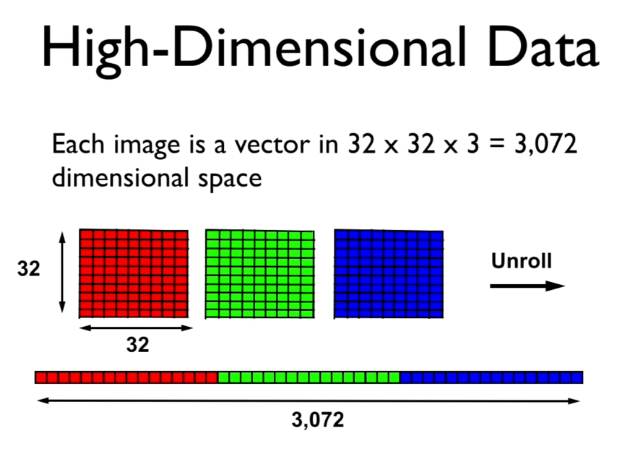


1-NN Properties

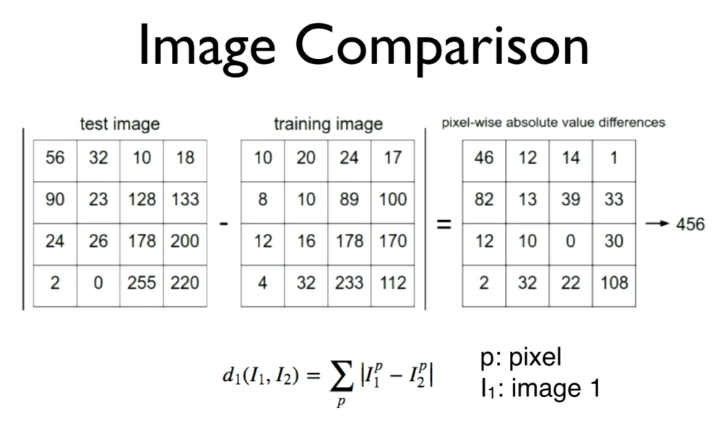
* Simple and quite good for low dimensional data
* “Rough” decision boundary, may have “islands”
* **Training** complexity for N data points?
  + Order 1
* **Testing** complexity for M data points?
  + For each of the M data points, we must run through all the N data points, and figure out which is the closest neighbor
  + O(N\*M)
* Variance? High
* Bias? Low
* How do you reduce the variance (“roughness” of the decision boundary)?
  + Increase the value of k
* As K **increases**, the bias **increases** but the variance **falls**

k-NN properties

* Gets rid of “islands”
* If k is too large, the boundary may become to smooth
* Lower variance, but increased bias
* How do we choose the ideal k?
  + Cross-validation
    - Average the parameters with the best performance on validation data

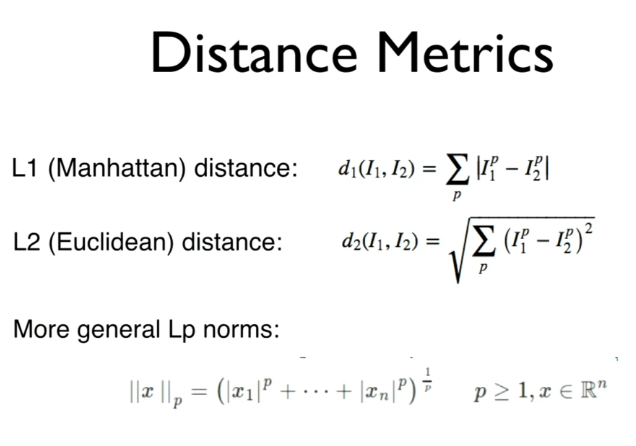


For an image with 32 X 32 pixels



How is distance calculated between 2 images?

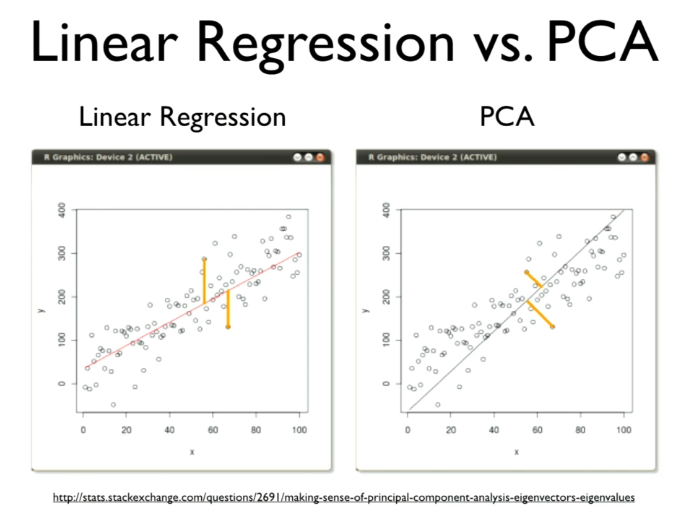
This distance is called L1 distance, or the Manhattan distance, which is the absolute difference between two vectors.



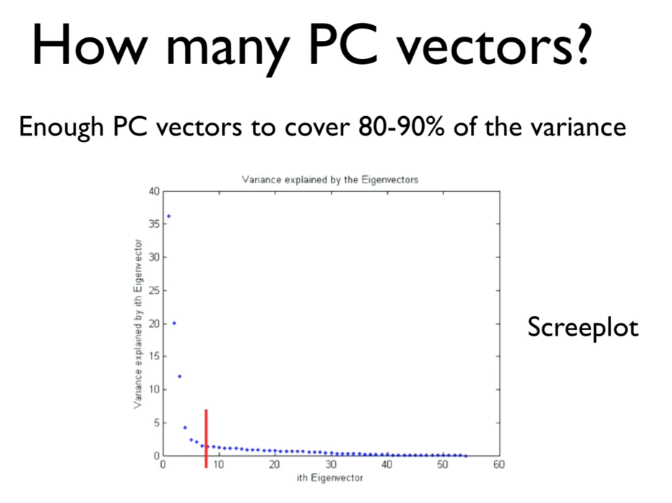
* Using pixel by pixel difference as the input features is very simple but does not produce high performance models.
* Hence, we need to look for other features.
  + One example, **SIFT** features:
    - Scale invariant
    - Rotation invariant
* Curse of Dimensionality
  + When dimensionality increases, the volume of the space increases so fast that the available data becomes sparse
  + Statistically sound result requires the sample size N to grow exponentially with d

Dimensionality Reduction [55:00]

* Basic idea:   
  Project the high-dimensional data onto a lower-dimensional subspace that best “fits” the data
* Principal Component Analysis (PCA)
  + Uses:
    - Dimensionality reduction for supervised learning
    - Compression
    - Visualization



* PCA uses orthogonal distance i.e., perpendicular distance.
* PCA Algorithm
  + Subtract the mean from data (center X)
  + (Typically) scale each dimension by its variance
    - Helps to pay less attention to magnitude of dimensions
  + Compute covariance matrix S
    - S =
  + Compute k largest eigenvectors of S
  + These eigenvectors are the k principal components
  + Note: Step 1 and 2 is basically standardization



* + However,   
    most people compute the eigenvectors using an algorithm called SVD (Singular Value Decomposition), which is much more robust than the process above
  + The point is, the way PCA is calculated might differ, but the fundamental idea is the same.
  + How many PC vectors?
  + Enough to cover 80-90% of variance

MDS (Multi-Dimensional Scaling)

* Like PCA
* A different goal:
  + Find a set of points whose pairwise distances match a given distance matrix
* Classical MDS
* Given n X n matrix of pairwise distances between data points
* Compute n X k matrix X with coordinates of distances with some linear algebra magic
* Perform PCA on this matrix X