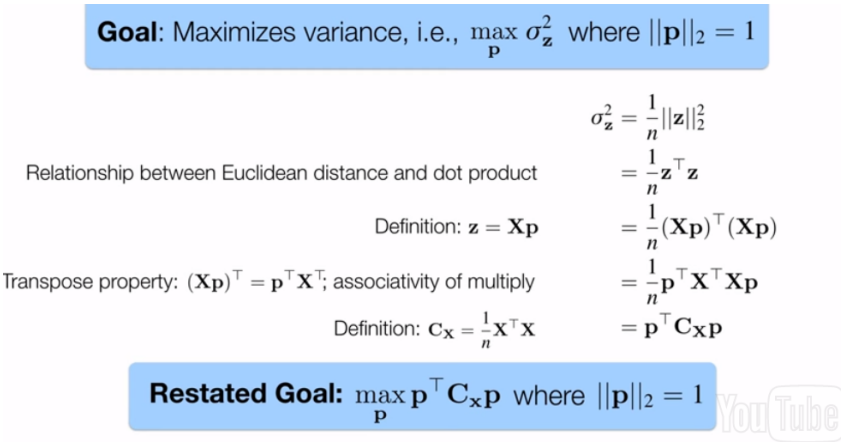
- Week 4
- **Neuroscience Introduction**
  - The brain
    - everything that happens between Stimuli and Behavior
  - Numbers of neurons
  - Studying the brain in humans
    - fMRI scanner
      - changes in blood oxygen
    - human brain
  - Relating neuronal responses to properties of an animal and its environment
  - Fine-scale sensory tuning
- **Exploratory Data Analysis**
- Model
  - Raw data
  - extracted signals
  - analysis
  - visualization
  - sharing
  - exploring
  - Interactive feedback
- Methods
  - Supervised methods
    - predict our data as a function of other data
  - Unsupervised methods
    - find structure in the data own its own
- Time series
  - supervised (regression and tuning
    - time series as a function of some other variable
  - unsupervised (dimensionality reduction and clustering)
    - identify simple and more compact representations
    - aid our understanding
- Clustering for preprocessing
  - raw data is complex and high-dimensional
  - clustering finds collections of inputs that are similar to one another
  - these groups of clusters may be the more meaningful “unit” of measurement
- Clustering to find waveforms associated with individual neurons based on their traces across multiple electrodes
- Dimensionality reduction
  - Raw data is complex and high-dimensional
  - Dimensionality reduction describes the data using a simpler, more compact representation
  - This representation may make interesting patterns in the data more clear or easier to see
- **PCA Overview**
- Raw data can be Complex, High-Dimensional
  - to understand a phenomenon we measure various related quantities
  - If we knew what to measure or how to represent our measurements we might find simple relationships
  - But in practice we often measure redundant signals, e.g., US and European shoe sizes
  - We also represent data via the method by which it was gathered, e.g. pixel representation of brain imaging data

- Dimensionality reduction
  - Issues
    - Measure redundant signals
    - Represent data via the method by which it was gathered
  - Goal: Find a ‘better’ representation for data
    - To visualize and discover hidden patterns
    - Preprocessing for supervised task, e.g., feature hashing
- Shoe Size
  - We take noisy measurements on european and american scale
    - modulo noise, we expect perfect correlation
  - How can we do ‘better’, i.e., find a simpler, compact representation?
    - pick a direction and project onto this direction
- Goal: Minimize Reconstruction Error
  - Minimize Euclidean distances between original points and their projections
  - PCA solution solves this problem
  - PCA - reconstruct 2D data via 2D data with single degree of freedom. Evaluate reconstructions (represented by blue line) by Euclidean distances
  - Linear Regression - predict y from x. Evaluate accuracy of predictions (represented by blue line) by vertical distances between points and the line
- Another Goal: Maximize Variance
  - To identify patterns we want to study variation across observations
  - Can we do ‘better’ i.e., find a compact representation that captures variation
  - PCA solution finds directions of maximal variance
- PCA Assumptions and Solution
- PCA Formulation
  - PCA: find lower-dimensional representation of raw data
    - X is n x d (raw data)
    - Z = XP is n x k (reduced representation, PCA ‘scores’)
    - P is d x k (columns are k principal components)
    - Linearity assumption (Z = XP) simplifies problem
    - Variance constraints
- Given training points
  - X matrix storing points
  - x_j^(i): jth feature for i’th point
  - mau_j: mean of j’th feature
  - Variance of 1st feature
    - the sum of the squared difference between each sample and the mean, divided by the number of samples
  - Variance of 1st feature (assuming zero mean)
    - drop the mau term since by assumption it equals zero
  - Covariance of 1st and 2nd features (assuming zero mean)
    - computing the product of the two feature values for each data point, taking the sum of these products, and finally dividing by n, the number of samples that we have
      - large positive covariance indicates that the two feature are highly correlated
      - large negative covariance indicates that the two features are highly anti correlated
  - Covariance Matrix
    - Covariance matrix generalizes this idea for many features
    - Cx = (1/n)X’X
    - d x d matrix

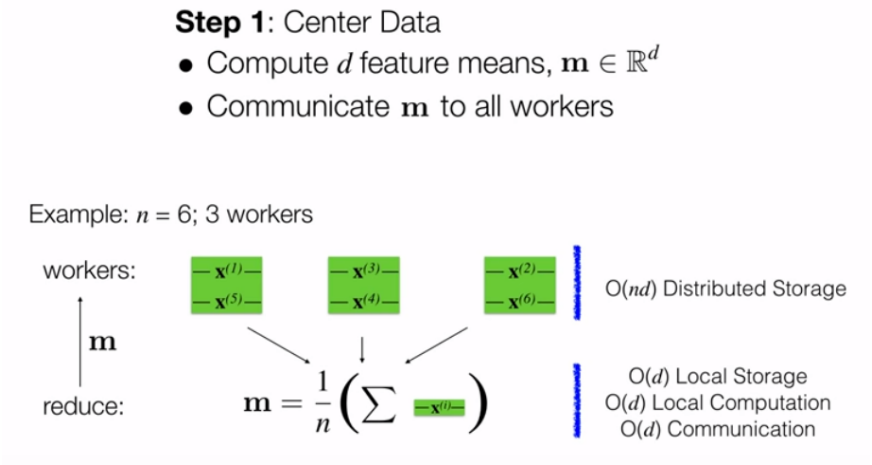
    - each entry stores pairwise covariance information about the d features
    - ith diagonal entry equals variance of i'th feature
    - ij'th entry is covariance between i'th and nth features
    - Symmetric (makes sense given definition of covariance)
- PCA Formulation
  - PCA: find lower-dimensional representation of raw data
    - X is n x d (raw data)
    - Z = XP is n x k (reduced representation, PCA 'scores')
    - P is d x k (columns are k principal components)
    - Variance / Covariance constraints
  - What constraints make sense in reduced representation?
    - No feature correlation, i.e., all off-diagonals in Cz are zero
    - Rank-ordered features by variance, i.e., sorted diagonals of Cz
    - Variance of the first feature in the reduced dimension is the largest, followed by the variance in the second feature, and so on
- PCA Solution
  - All covariance matrices have an eigendecomposition
    - Cx = UAU' (eigendecomposition)
    - U is d x d (column are eigenvectors, sorted by their eigenvalues)
    - A is d x d(diagonals are eigenvalues, off-diagonal are zero)
  - The d eigenvectors are orthonormal directions of max variance
    - associated eigenvalues equal variance in these directions
    - 1st eigenvector is direction of max variance (variance is lambda_1)
- Choosing k
  - How should we pick the dimension of the new representation?
  - Visualization: Pick top 2 or top 3 dimensions for plotting purposes
  - Other analyses: Capture 'most' of the variance in the data
    - Recall that eigenvalues are variances in the directions specified by eigenvectors, and that eigenvalues are sorted
    - Fraction of retained variance
      - can choose k such that we retain fraction of the variance, e.g.
- Other Practical Tips
  - PCA assumptions (linearity, orthogonality) not always appropriate
    - Various extensions to PCA with different underlying assumptions e.g., manifold learning, Kernel PCA, ICA
  - Centering is crucial, i.e., we must preprocess data so that all features have zero mean before applying PCA
  - PCA results dependent on scaling of data
    - Data is sometimes rescaled in practice before applying PCA
- PCA Algorithm
- Orthogonal and Orthonormal Vectors
  - Orthogonal vectors are perpendicular to each other
    - equivalently, their dot product equals zero
  - Orthonormal vectors are orthogonal and have unit norm
    - a are b are orthonormal, but b are d are not orthonormal
- PCA Iterative Algorithm
  - k = 1: Find direction of max variance, project onto this direction
    - locations along this direction are the new 1D representation
  - More generally, for i = {1,…,k}

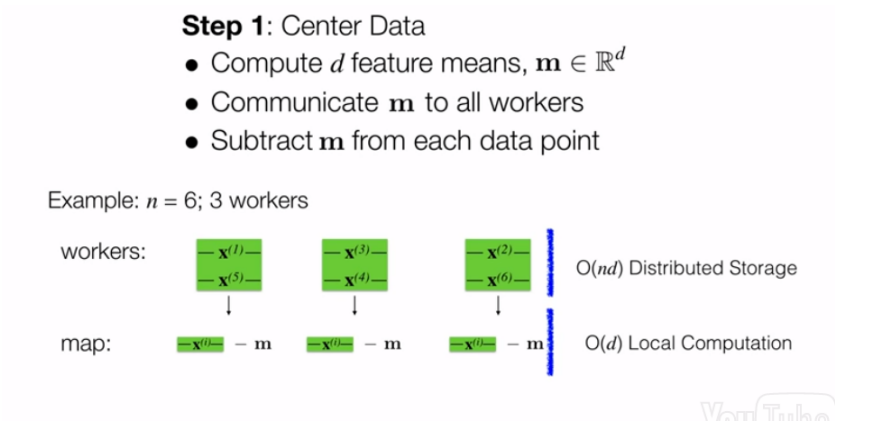
    - find direction of max variance that is orthonormal to previously selected directions, project onto this direction
    - locations along this direction are i the feature in new representation
- PCA Derivation
- Eigendecomposition
- All covariance matrices have an eigendecomposition
  - Cx = UAU' (eigendecomposition)
  - U is d x d (column are eignevectors, sorted by their eigenvalues)
  - A is d x d (diagonals are eigenvalues, off-diagonals are zero)
- Eigenvector/Eigenvalue equation: Cxu = lambda*u
  - By definition u'u = 1(unit norm)
- PCA Formulation
  - PCA: find lower-dimensional representation of raw data
    - X is n x d (raw data)
    - Z = XP is n x k (reduced representation, PCA 'scores')
    - P is d x k (columns are k principal components)
    - Variance / Covariance constraints
- PCA Formulation, k = 1
  - PCA: find one-dimensional representation of raw data
    - X is n x d (raw data)
    - z = Xp is n x 1 (reduced representation, PCA 'scores')
    - p is d x 1 (columns are k principal components)
    - variance = sigma Z squared / n = square Euclidean norm of the vector z / n
    - Goal: maximizes variance
      - resulting principal component p to be unit norm
    - noting the relationship between the Euclidean distance and the dot product, we can rewrite the variance as a dot product z transpose z
      - relationship between Euclidean distance and dot product = (1/n)z'z



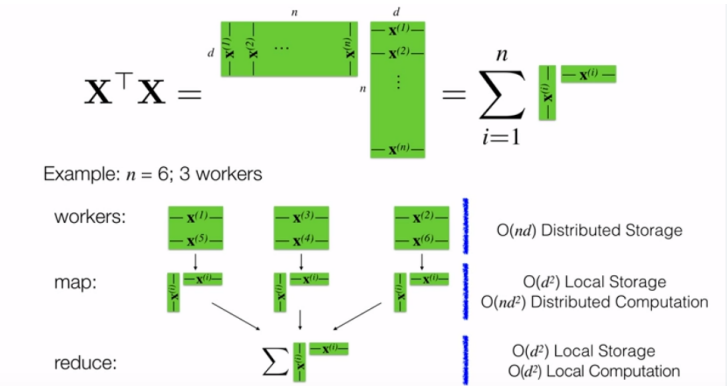

- Connection to Eigenvectors
  - Recall eigenvector/eigenvalue equation: Cx * u = lambda * u
    - By definition u'u = 1, and thus u' Cx * u = lambda
    - But this is the expression we're optimizing, and thus maximal variance achieved when p is top eigenvector of Cx
- Distributed PCA

- Computing PCA Solution
  - Given: n x d matrix of uncentered raw data
  - Compute k << d dimensional representation
  - Step 1: Center Data
    - computing the mean of each feature and subtracting the mean
  - Step 2: Compute Covariance or Scatter Matrix
    - (1/n) * X' * X versus X' * X
  - Step 3: Eigendecomposition
  - Step 4: Compute PCA Scores
- PCA at Scale
  - Case 1 Big n and small d
    - O(d squared) local storage, O(d cubed) local computation, O(dk) communication
    - Similar strategy as closed-form linear regression
  - Case 2 Big n and Big d
    - O(d) local storage and computation on workers, O(dk) communication
    - Iterative algorithm
- Step 0: Data Parallel storage
  - Example: n = 6; 3 workers
    - workers: O(nd) Distributed Storage
- Step 1: Center Data
  - Compute d feature means, m element of R^d
  - Example n = 6; 3 workers
    - workers
    - reduce
      - m = (1/n) * ( summation features)

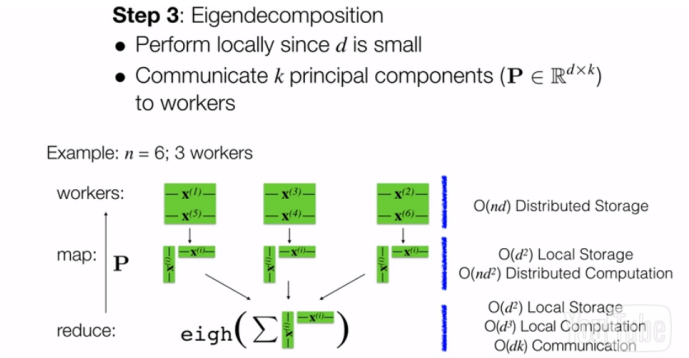


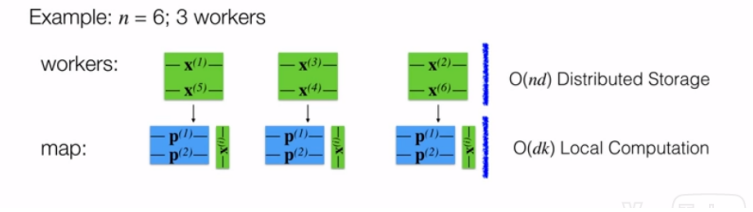
- Step 2: Compute Scatter Matrix (X'X)
  - Compute matrix product via outer products (just like we did for closed-form linear regression!)




- Step 3: Eigendecomposition
  - perform locally since d is small



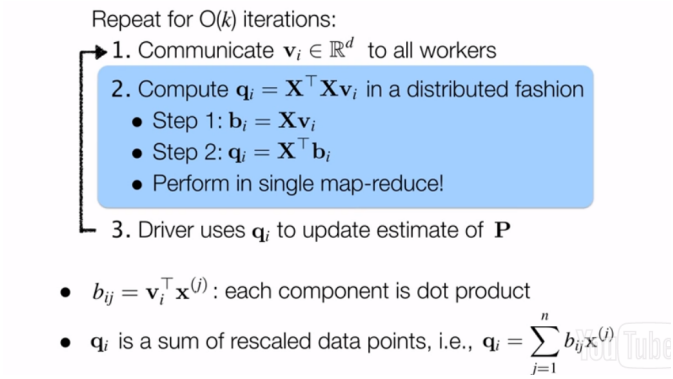

- Step 4: Compute PCA Scores
  - Multiply each point by principal components, P
  - workers -> map

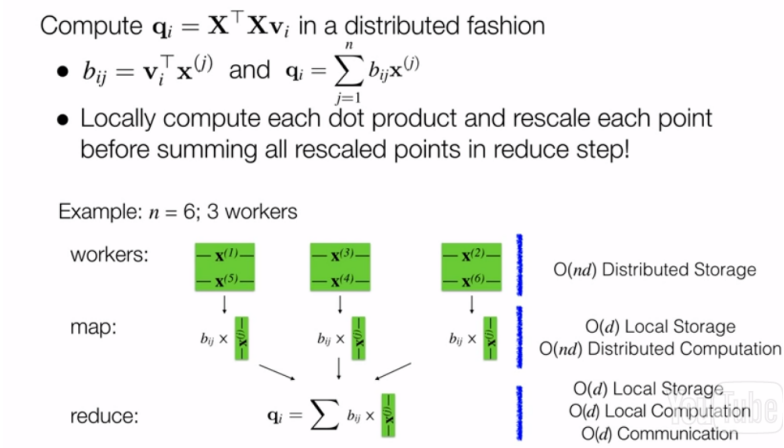



- Distributed PCA, Part 2 (Optional)
- PCA at Scale
  - Case 1: Big n and Small d
    - O(d^2) local storage, O(d^2) local computation, O(dk) communication
    - Similar strategy as closed-form linear regression
  - Case 2: Big n and Big d
    - O(d) local storage and computation on workers, O(dk) communication

- Iterative algorithm

- An Iterative Approach
  - We can use algorithms that rely on a sequence of matrix-vector products to compute top k eigenvectors (P)
    - E.g., Krylov subspace or random projection methods
  - Krylov subspace methods (used in MLlib) iteratively compute X'Xv for some v element R^d provided by the method
    - Requires O(k) passes over data, O(d) local storage on workers
    - We don't need to compute the covariance matrix!
    - Step1: communicate vi element R^d to all workers
    - Step2: Compute qi = X'Xvi in a distributed fashion
      - Step1: b_i = Xvi
      - Step2: qi = X'bi
      - Perform in single map-reduce




  - q = trainData.map(rescaleByBi).reduce(sumVectors)