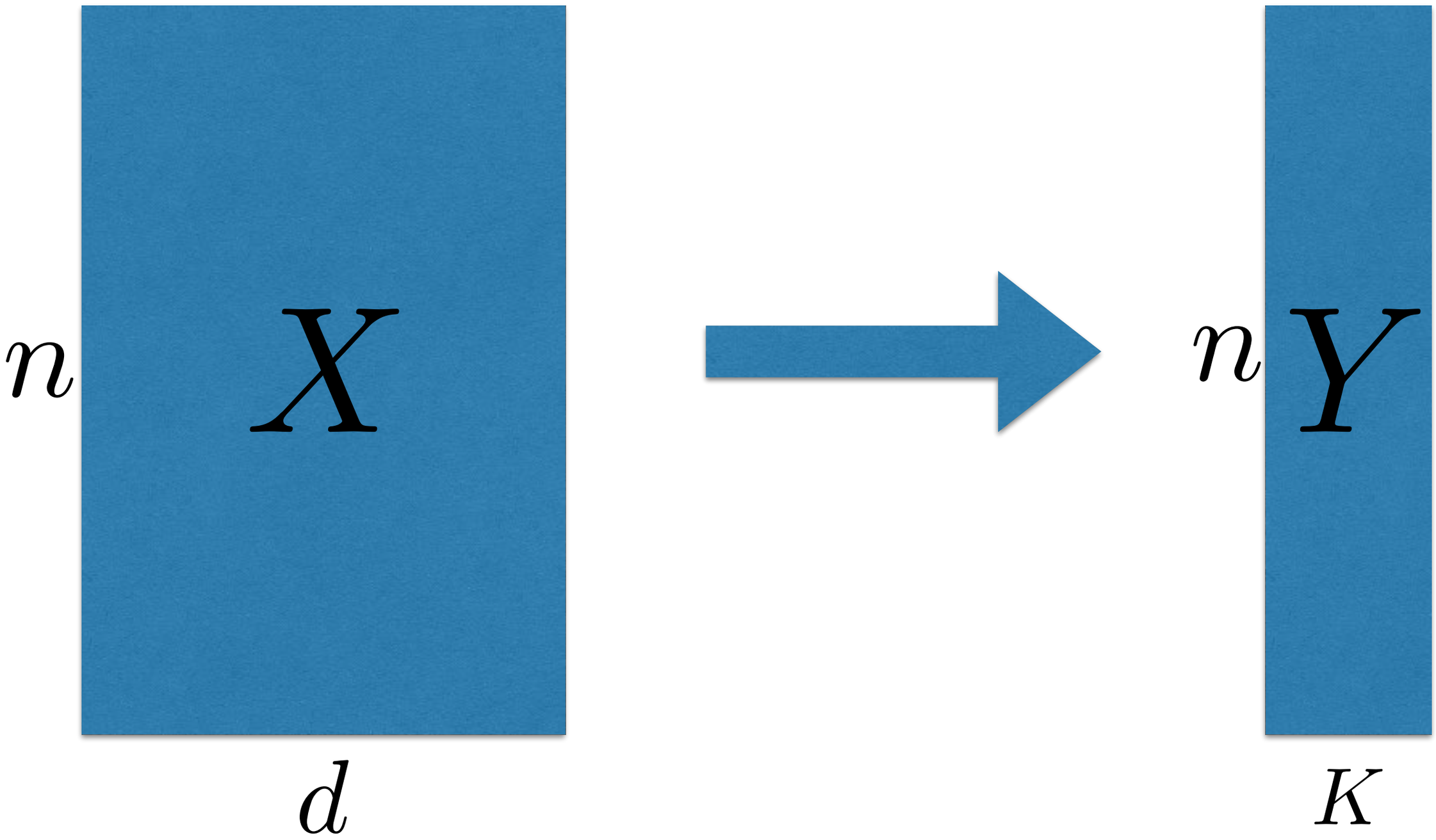
Machine Learning for Data Science (CS4786)
Lecture 27

Last Lecture

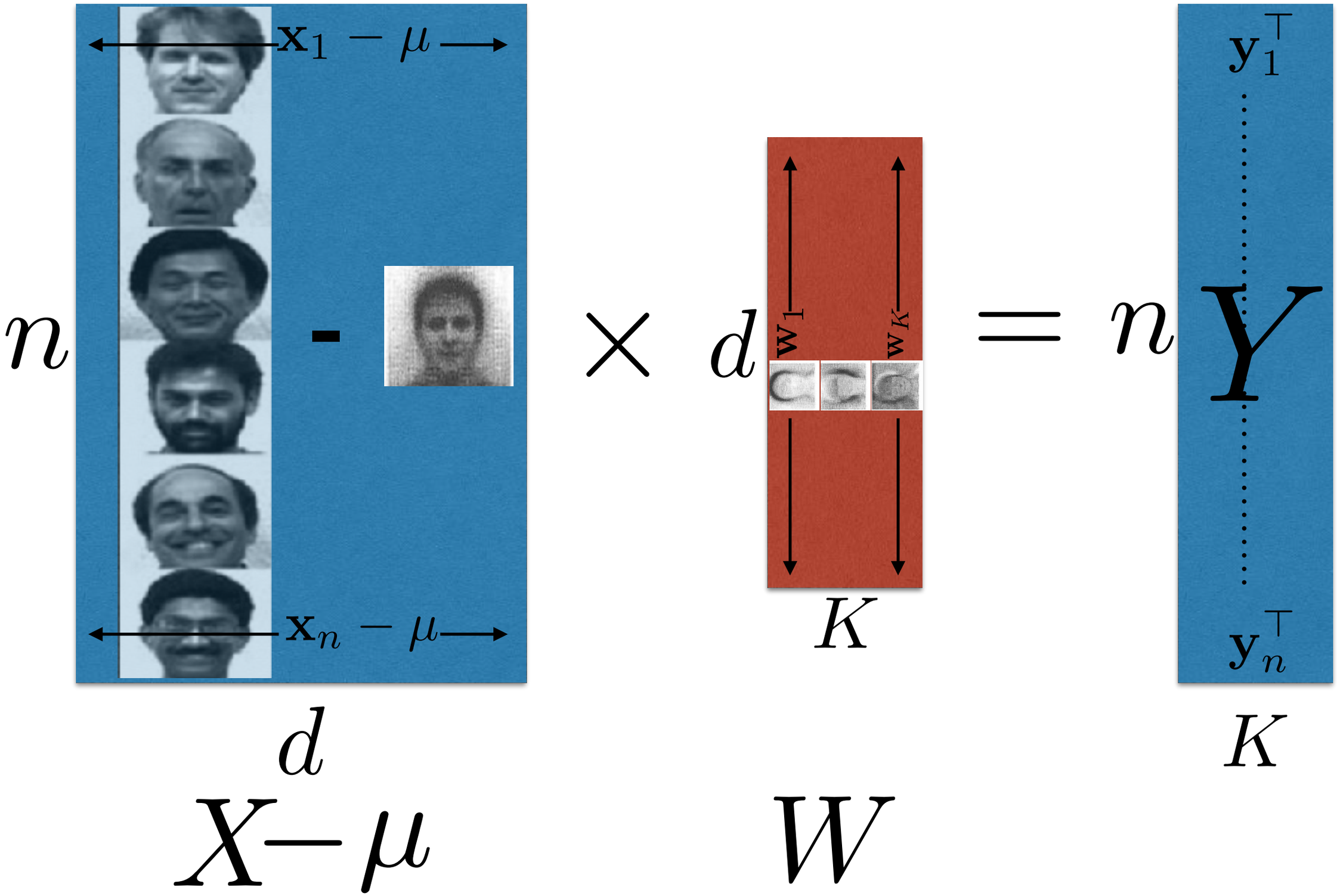
What have we covered so far?

Dimensionality Reduction

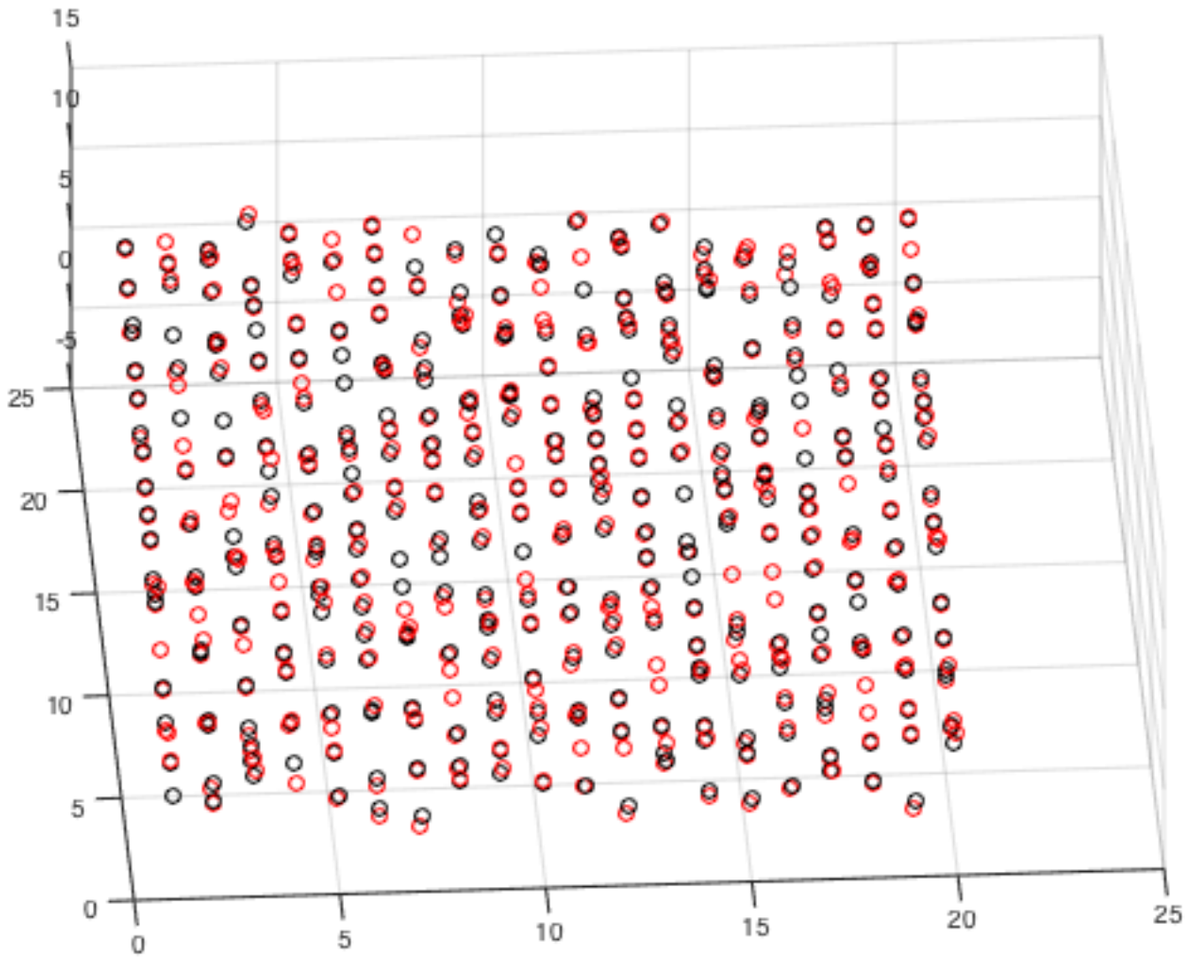


Dim Reduction: Linear Transformation

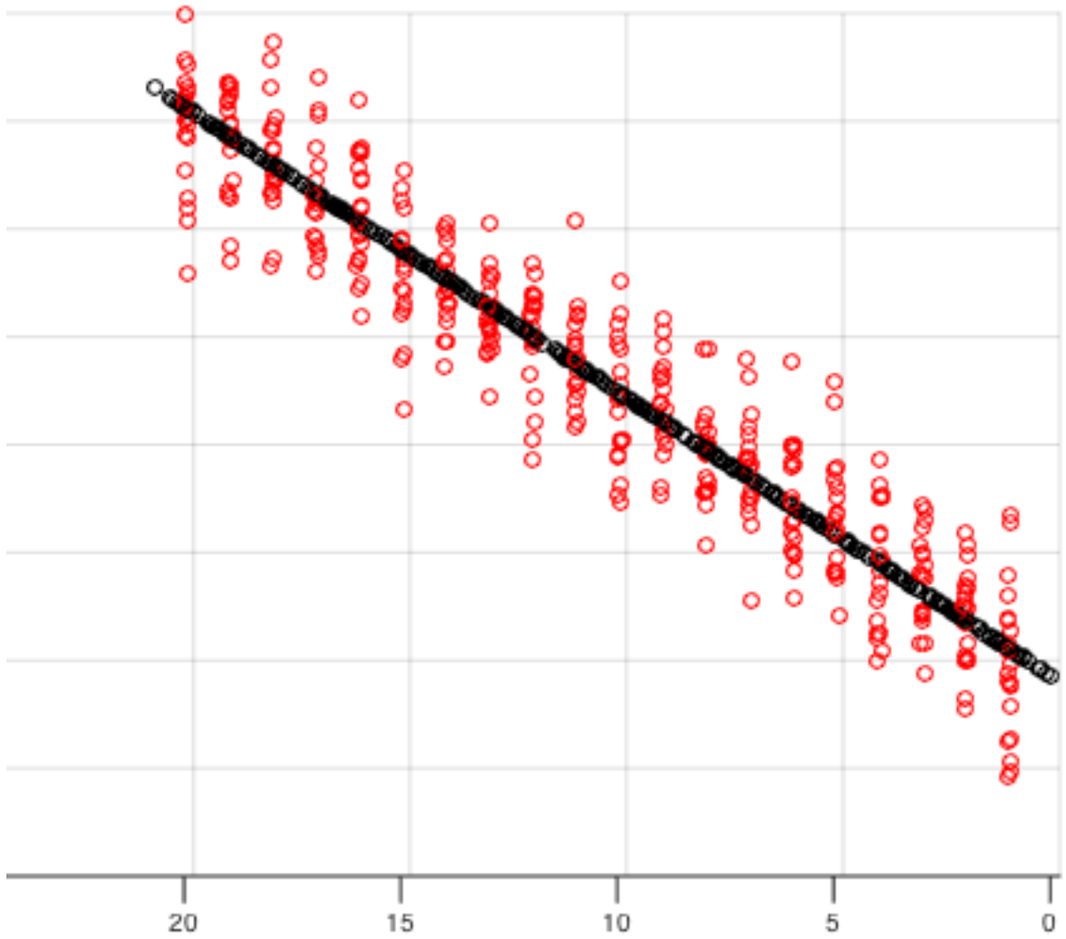
$$\mathbf{y}_i^\top = \mathbf{x}_i^\top W$$



Maximize Total Spread



Minimize Reconstruction Error



Top K eigenvectors of covariance matrix are the K directions

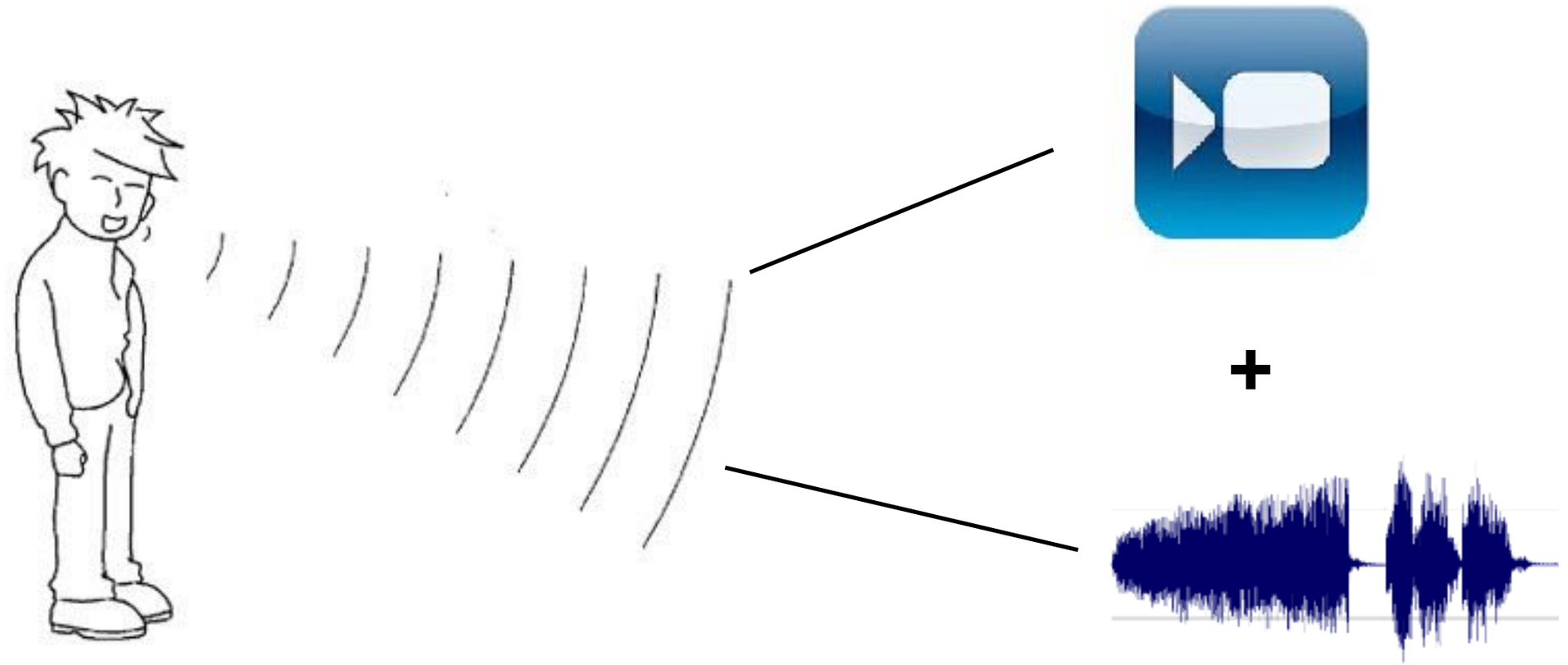
Pick a Random W

Preserve interpoint distances

$$Y = X \times \begin{bmatrix} +1 & \dots & -1 \\ -1 & \dots & +1 \\ +1 & \dots & -1 \\ & \vdots & \\ +1 & \dots & -1 \end{bmatrix}_{K} \Big/ \sqrt{\frac{d}{K}}$$

Common trick used to handle very large data!

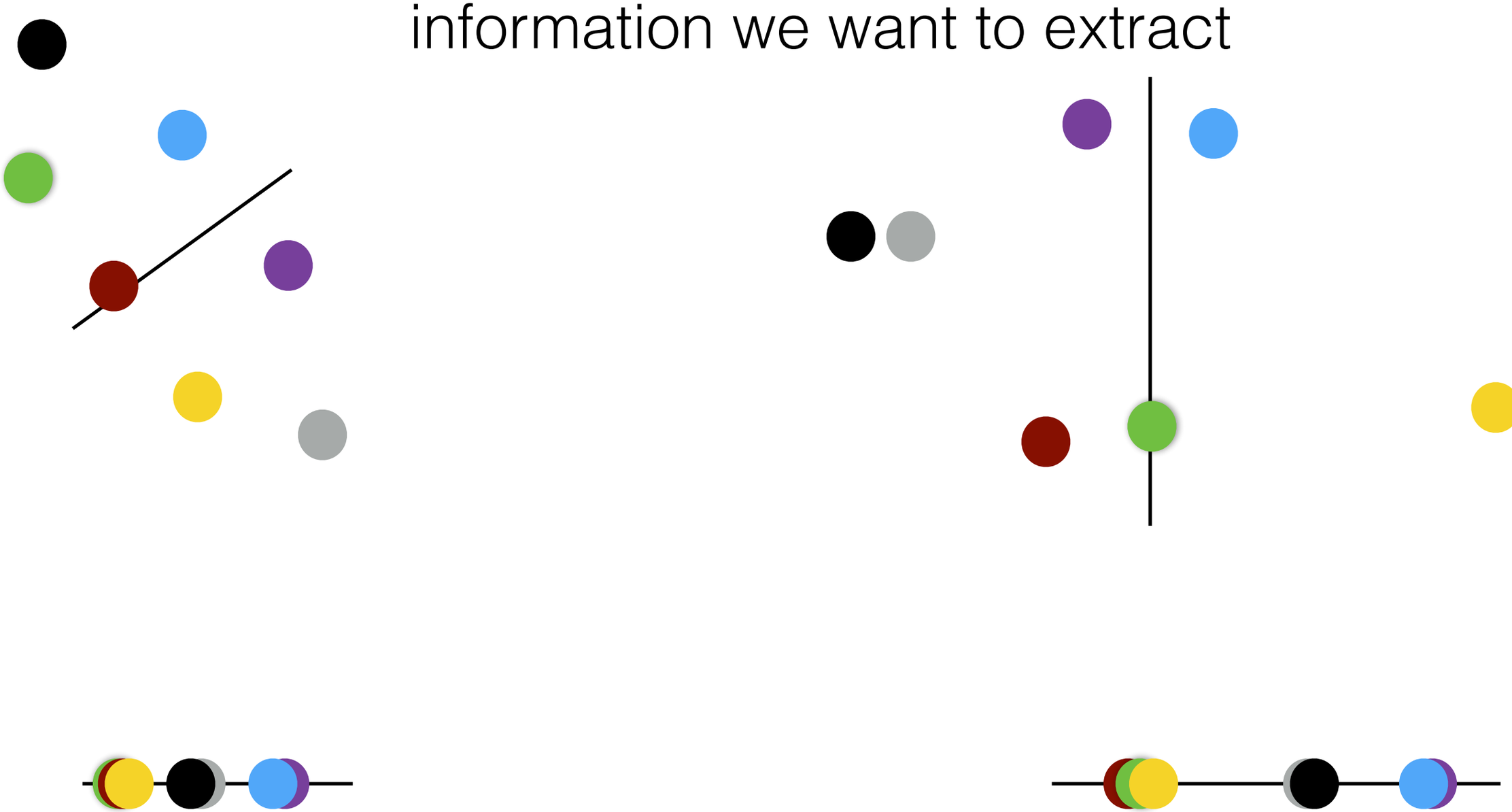
Example I: Speech Recognition



- Audio might have background sounds uncorrelated with video
- Video might have lighting changes uncorrelated with audio
- Redundant information between two views: the speech

Which Direction to Pick?

Data naturally split into two parts: both carry common information we want to extract



Direction has large correlation coefficient

Interpreting Deep Neural Networks with SVCCA

Tuesday, November 28, 2017

Posted by Maithra Raghu, Google Brain Team

Deep neural networks (DNNs) have driven unprecedented advances in areas such as vision, language understanding and speech recognition. But these successes also bring new challenges. In particular, contrary to many previous machine learning methods, DNNs can be susceptible to adversarial examples in classification, catastrophic forgetting of tasks in reinforcement learning, and mode collapse in generative modelling. In order to build better and more robust DNN-based systems, it is critically important to be able to interpret these models. In particular, we would like a notion of *representational similarity* for DNNs: can we effectively determine when the representations learned by two neural networks are same?

In our paper, "SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability," we introduce a simple and scalable method to address these points. Two specific applications of this that we look at are comparing the representations learned by different networks, and interpreting representations learned by hidden layers in DNNs. Furthermore, we are open sourcing the code so that the research community can experiment with this method.

Key to our setup is the interpretation of each neuron in a DNN as an *activation vector*. As shown in the figure below, the activation vector of a neuron is the scalar output it produces on the input data. For example, for 50 input images, a neuron in a DNN will output 50 scalar values, encoding how much it responds to each input. These 50 scalar values then make up an activation vector for the neuron. (Of course, in practice, we take many more than 50 inputs.)

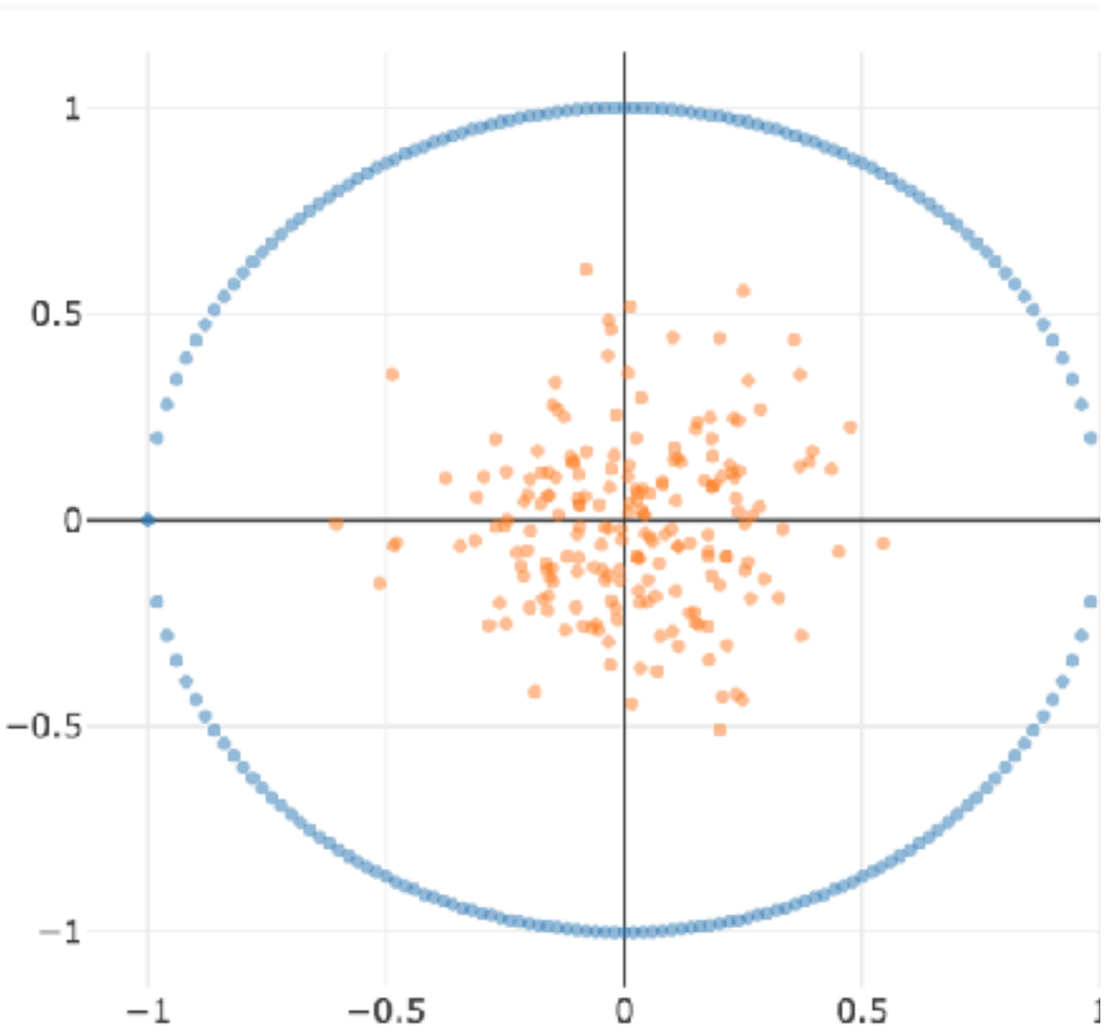
Kernel Trick

- We have have nice methods for linear dimensionality reduction. Can we use them?

- Lift to higher dimensions (introduces non-linearity)

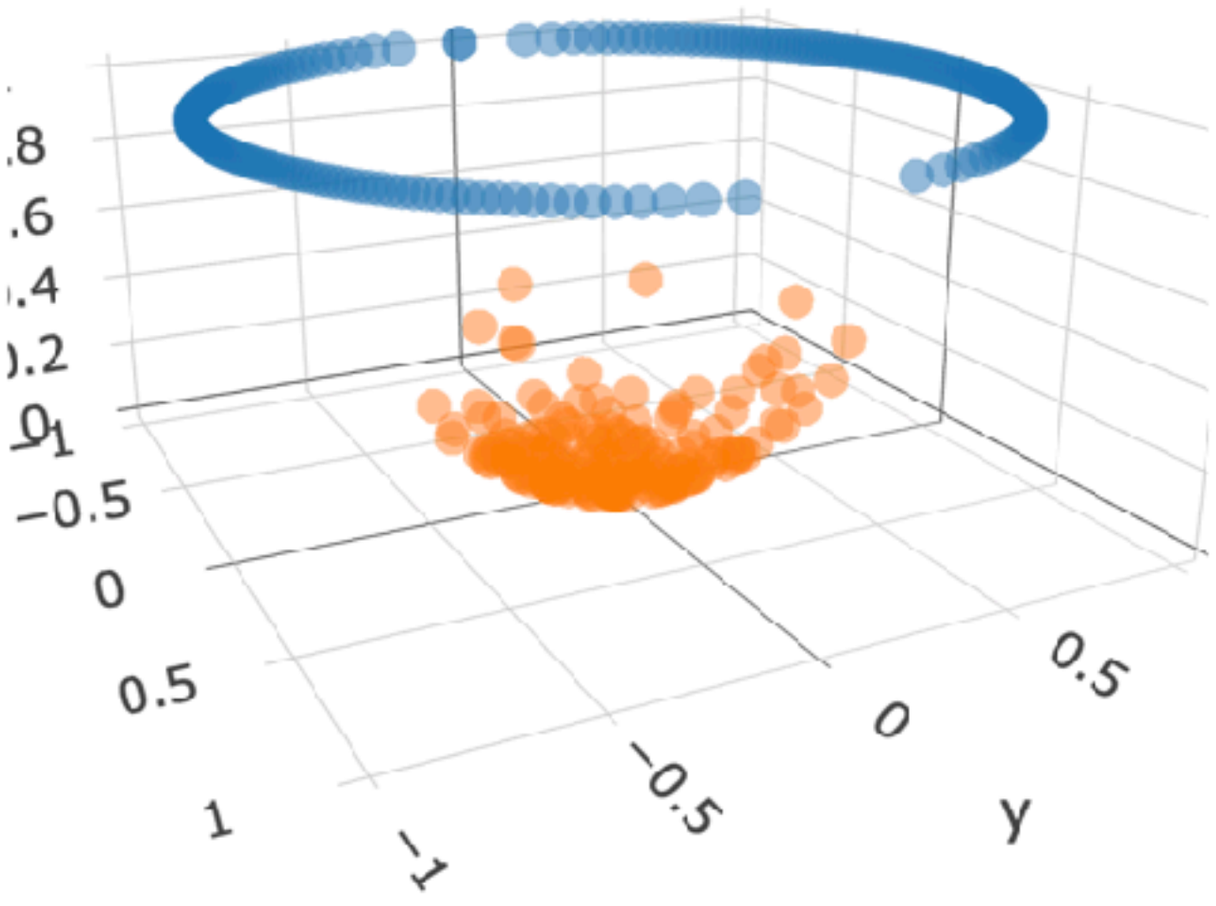
- Perform linear dimensionality reduction in this high dimensional space

Example



Original Data in 2D

(x,y)



Data Lifted to 3D

(x, y, x^2 + y^2)

Key Idea:

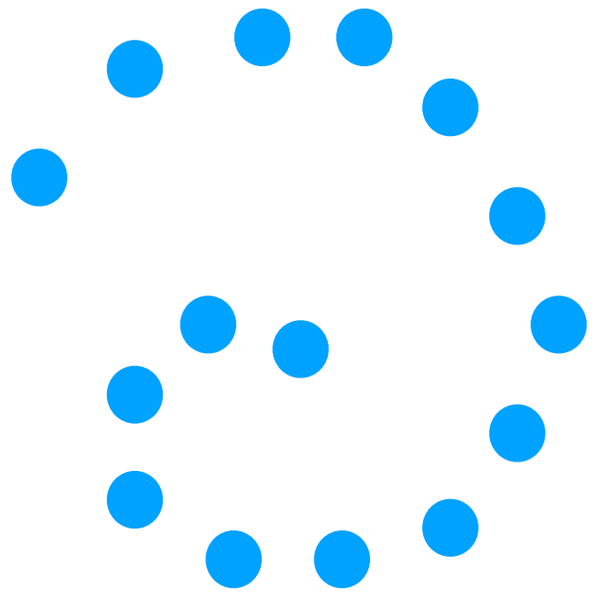
If an algorithm only depends on inner products, we can simply replace inner product in x space by inner product in $\phi(x)$ space

Inner product in $\phi(x)$ space can be written as kernel function

Making a resurface with scalable deep kernels!

Manifold Based Dimensionality Reduction

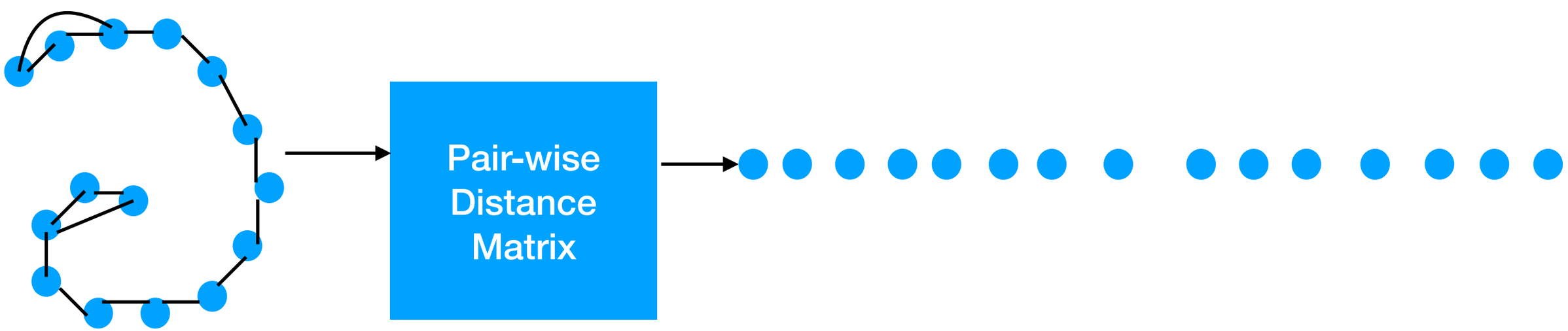
- Key Assumption: Points live on a low dimensional manifold
- Manifold: subspace that looks locally Euclidean
- Given data, can we uncover this manifold?



Can we unfold this?

Method I: Isomap

1. For every point, find its (k-) Nearest Neighbors
2. Form the Nearest Neighbor graph
3. For every pair of points A and B, distance between point A to B is shortest distance between A and B on graph
4. Find points in low dimensional space such that distances between points in this space is equal to distance on graph.



Stochastic Neighborhood Embedding

- Use a probabilistic notion of which points are neighbors.

- Close by points are neighbors with high probability, . . .
  Eg: For point $\mathbf{x}_t$, point $\mathbf{x}_s$ is picked as neighbor with probability

$$p_{t\to s} = \frac{\exp\left(-\frac{\|\mathbf{x}_s - \mathbf{x}_t\|^2}{2\sigma^2}\right)}{\sum_{u\neq t}\exp\left(-\frac{\|\mathbf{x}_u - \mathbf{x}_t\|^2}{2\sigma^2}\right)}$$

  Probability that points s and t are connected $P_{s,t} = P_{t,s} = \frac{p_{t\to s} + p_{s\to t}}{2n}$

- Goal: Find $\mathbf{y}_1, \ldots, \mathbf{y}_n$ with stochastic neighborhood distribution Q such that "P and Q are similar"

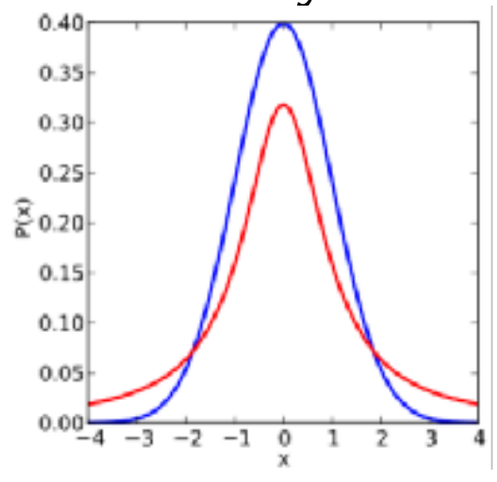
  i.e. minimize:

$$\mathrm{KL}(P\|Q) = \sum_{s,t} P_{s,t} \log\left(\frac{P_{s,t}}{Q_{s,t}}\right) = \sum_{s,t} P_{s,t}\log\left(P_{s,t}\right) - \sum_{s,t} P_{s,t}\log\left(Q_{s,t}\right)$$

Method II: t-SNE

- Instead for Q we use, student t distribution which is heavy tailed:

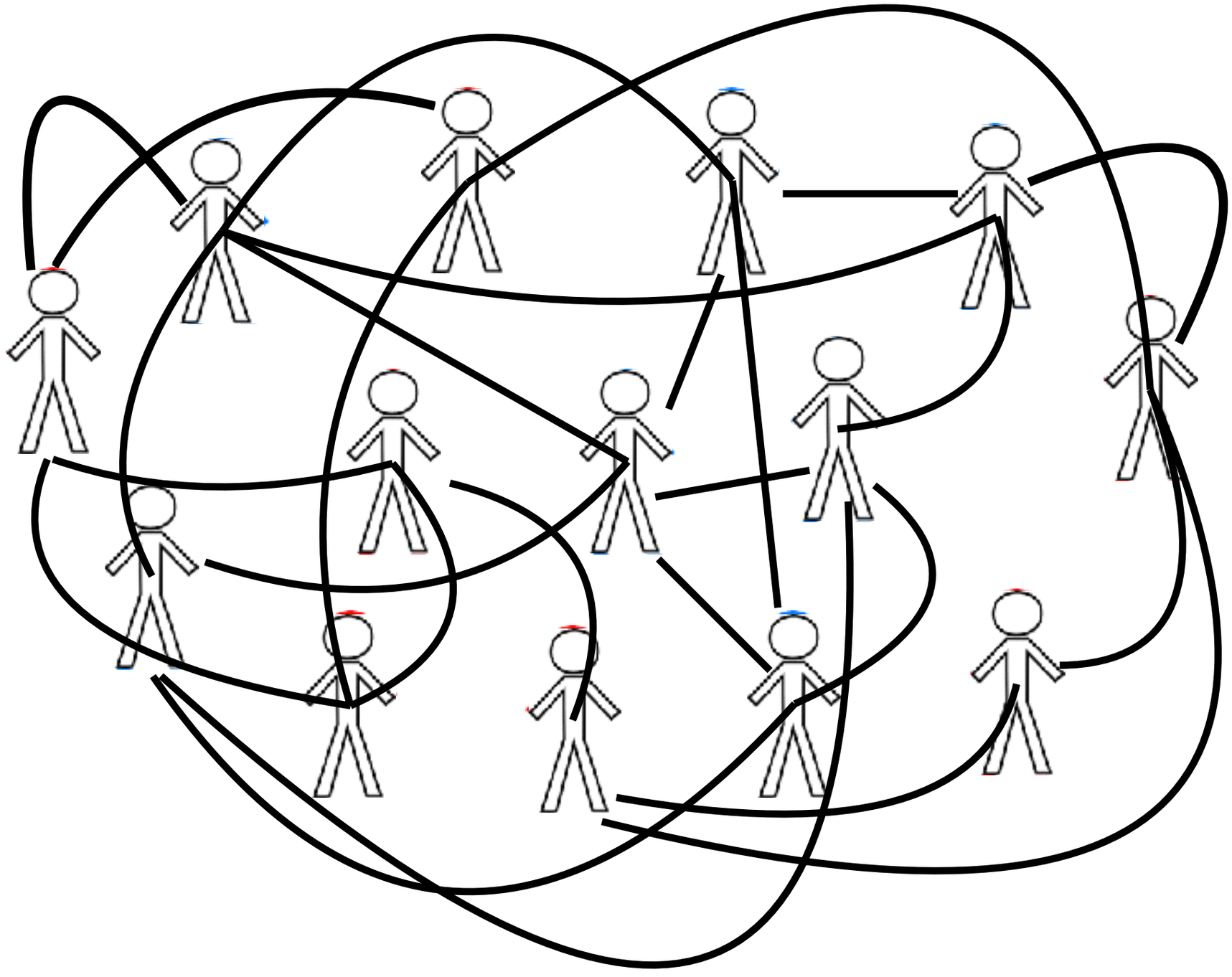
$$q_{t\to s} = \frac{\left(1 + \|\mathbf{y}_s - \mathbf{y}_t\|^2\right)^{-1}}{\sum_{u\neq t}\left(1 + \|\mathbf{y}_u - \mathbf{y}_t\|^2\right)^{-1}}$$



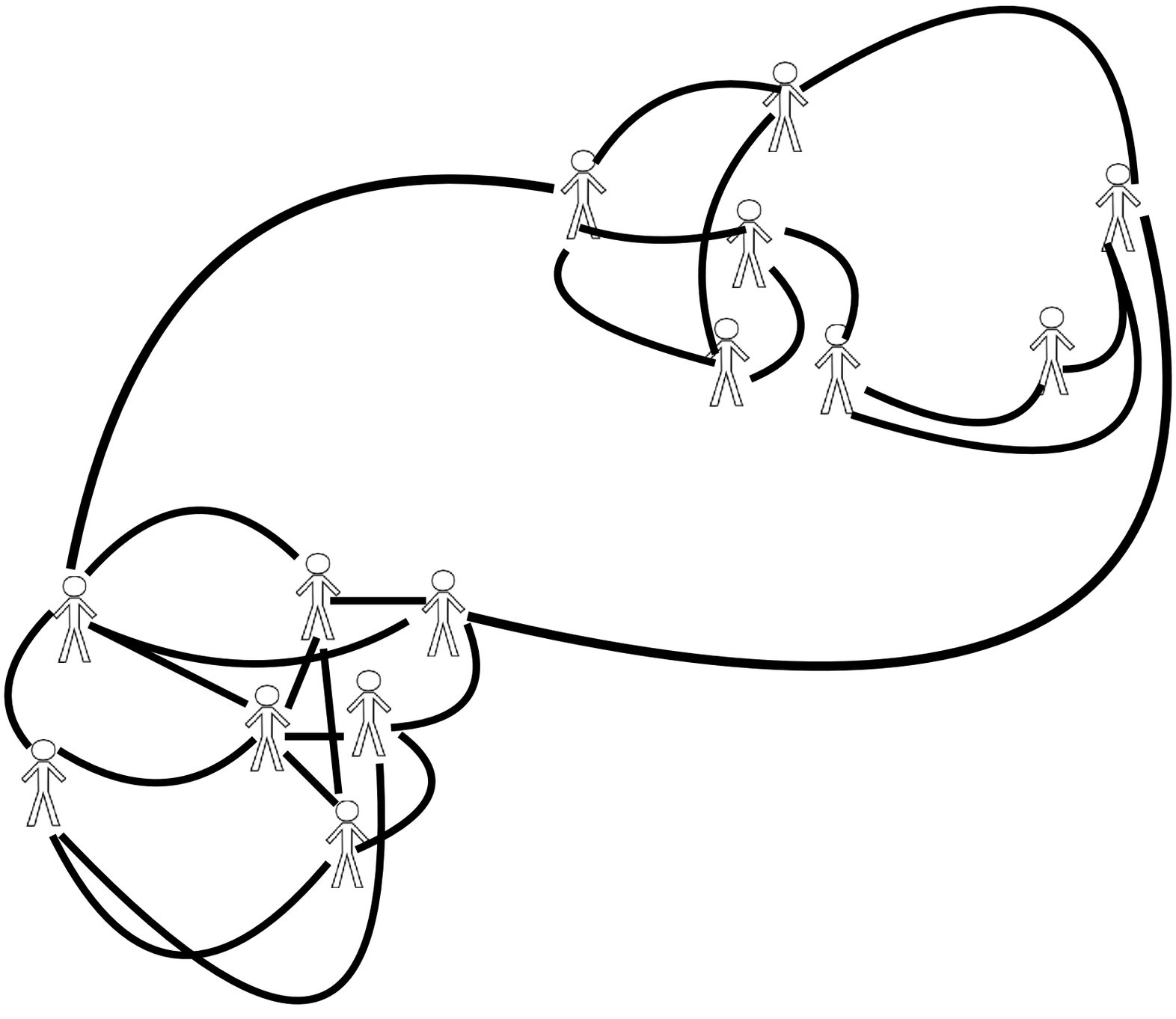
and then set $Q_{s,t} = \frac{q_{t\to s} + q_{s\to t}}{2n}$

Often used to visualize data points, different layers of a neural networks etc.

Motivating Example



Motivating Example



Normalized Spectral Embedding

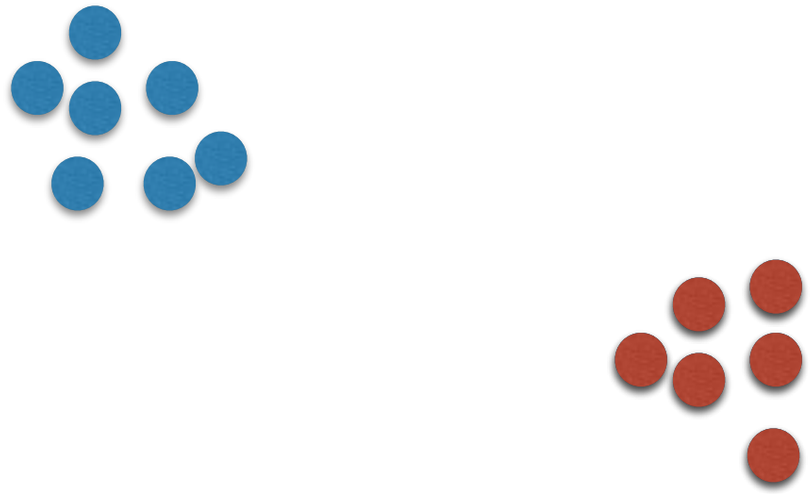
- Keep your Friends close $\text{minimize} \;\; y^\top L y$
- Points are centered at 0 $\sum_{i=1}^{n} D_{i,i} y_i = 0$
- Variance or spread should be large $\text{Maximize} \;\; y^\top D y$

$$\text{Minimize } u^\top D^{-1/2} L D^{-1/2} u \qquad \text{s.t. } \|u\| = 1 \;\&\; u \perp \text{diag}(D^{1/2})$$

$$\text{Solution: Second smallest eigen vector of } D^{-1/2} L D^{-1/2}$$

For graphs, spectral method is still the most robust one used!

Examples



What are the clusters?

K-means Clustering

Look for nice round clusters

- For all $j \in [K]$, initialize cluster centroids $\hat{\mathbf{r}}_j^1$ randomly and set $m = 1$
- Repeat until convergence (or until patience runs out)
  1. For each $t \in \{1, \ldots, n\}$, set cluster identity of the point

$$\hat{c}^m(\mathbf{x}_t) = \underset{j \in [K]}{\operatorname{argmin}} \|\mathbf{x}_t - \hat{\mathbf{r}}_j^m\|$$

  2. For each $j \in [K]$, set new representative as

$$\hat{\mathbf{r}}_j^{m+1} = \frac{1}{|\hat{C}_j^m|} \sum_{\mathbf{x}_t \in \hat{C}_j^m} \mathbf{x}_t$$

  3. $m \leftarrow m + 1$

Still the most widely used clustering algorithm!

Single Link Clustering

Look for tightly connected clusters

- Initialize n clusters with each point $\mathbf{x}_t$ to its own cluster
- Until there are only K clusters, do
  1. Find closest two clusters and merge them into one cluster
  2. Update between cluster distances (called proximity matrix)

Linkage clustering in general is still the favorite hierarchical clustering method!

Probabilistic Models

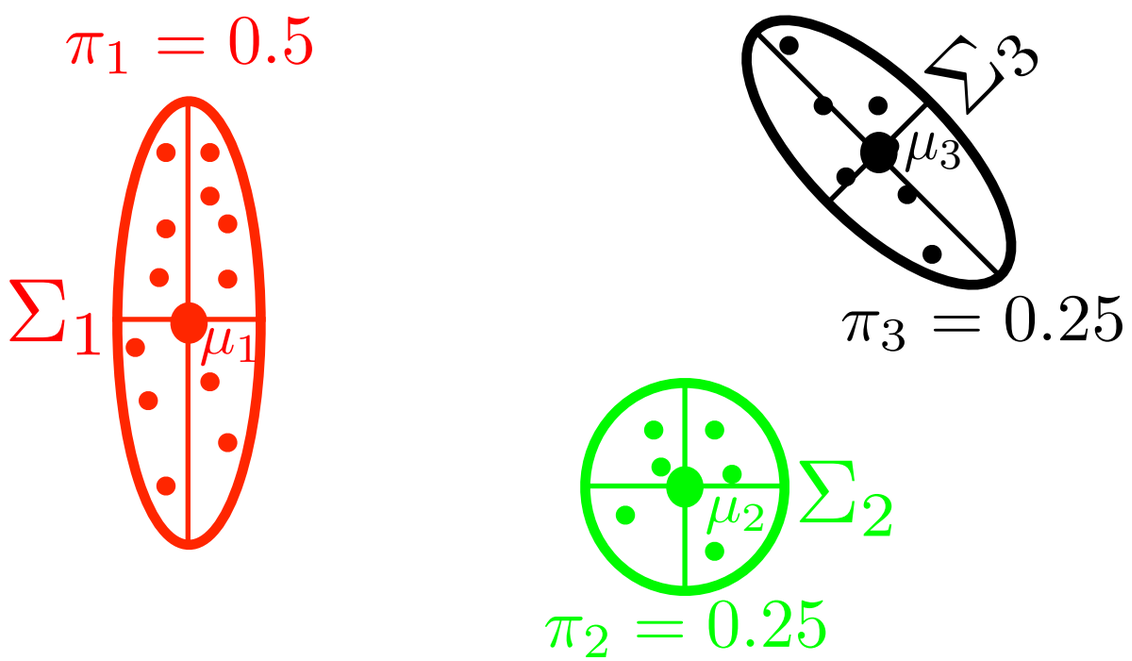
- Θ consists of set of possible parameters
- We have a distribution P_θ over the data induced by each $\theta \in \Theta$
- Data is generated by one of the $\theta \in \Theta$
- Learning: Estimate value or distribution for $\theta^* \in \Theta$ given data

Gaussian Mixture Models

Each $\theta \in \Theta$ is a model.

- Gaussian Mixture Model
  - Each θ consists of mixture distribution $\pi = (\pi_1, \ldots, \pi_K)$, means $\mu_1, \ldots, \mu_K \in \mathbb{R}^d$ and covariance matrices $\Sigma_1, \ldots, \Sigma_K$
  - For each t, independently:

$$c_t \sim \pi, \qquad x_t \sim N(\mu_{c_t}, \Sigma_{c_t})$$



Expectation Maximization Algorithm

- For demonstration we shall consider the problem of finding MLE (MAP version is very similar)
- Initialize $\theta^{(0)}$ arbitrarily, repeat unit convergence:

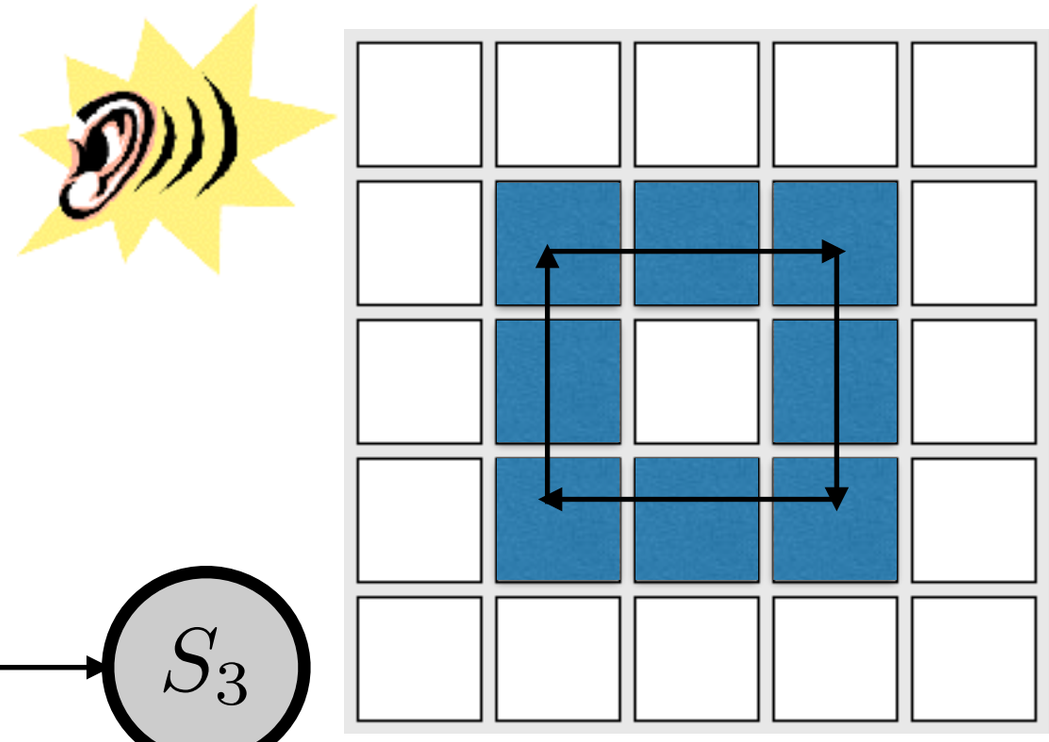
(E step) For every t, define distribution Q_t over the latent variable c_t as:

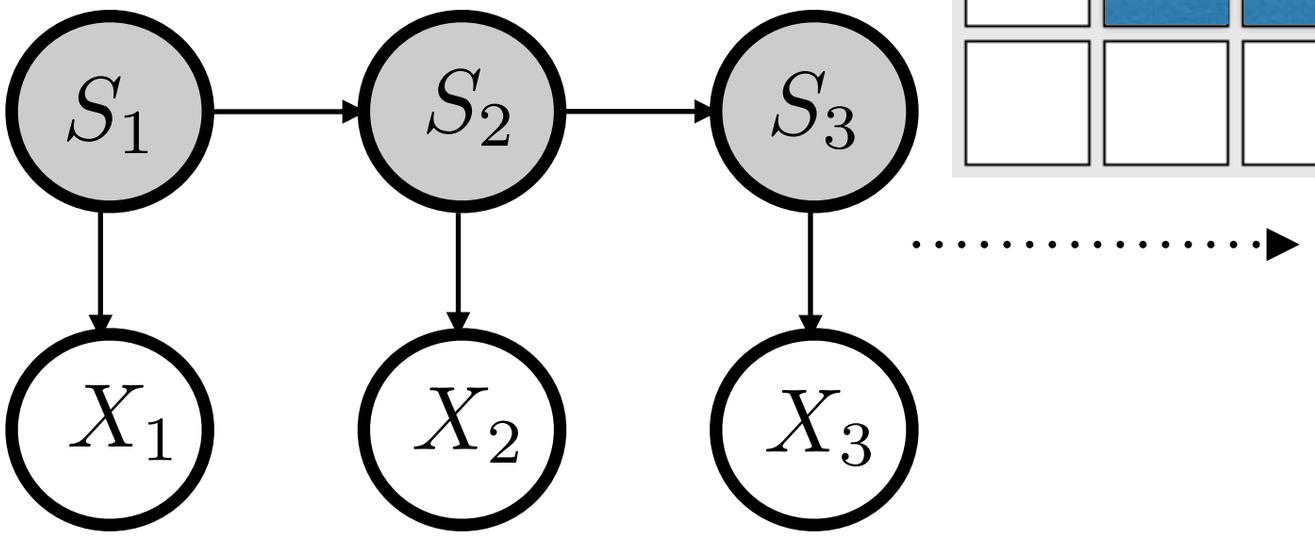
$$Q_t^{(i)}(c_t) = P(c_t|x_t, \theta^{(i-1)})$$

(M step)

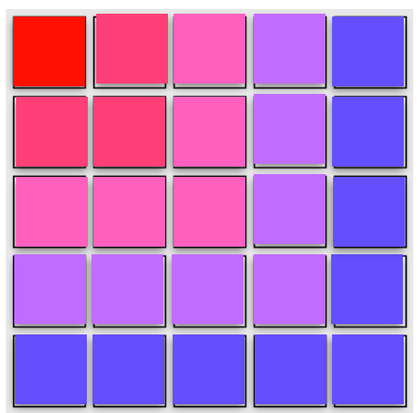
$$\theta^{(i)} = \text{argmax}_{\theta \in \Theta} \sum_{t=1}^{n} \sum_{c_t} Q_t^{(i)}(c_t) \log P(x_t, c_t|\theta)$$

Example: Hidden Markov Model





What you hear:



Example: Hidden Markov Model

. Forward Backward Algorithm

. EM for parameter estimation (Baum Welch)

. Approximate Inference (via sampling)

    Rejection and importance sampling

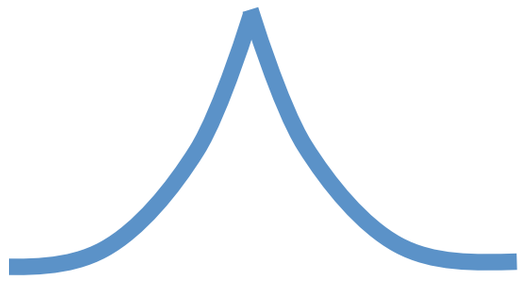
    Particle filtering for HMM

. More generally techniques lift to graphical models

Defining Privacy

Dataset +

Learning Algorithm

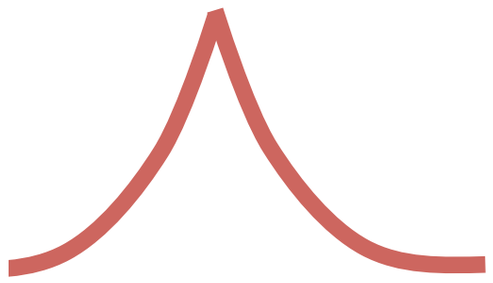


Distribution of outcome

Similar

Dataset +

Learning Algorithm



Distribution of outcome

Its in your apple watch and iPhone!

Threshold-out Algorithm

Input:

Data S, holdout H, threshold $T > 0$, tolerance $\sigma > 0$

Given function q:

Sample η, η' from $N(0,\sigma^2)$

If $|avg_H[q] - avg_S[q]| > T + \eta$:

output $avg_H[q] + \eta'$

Otherwise:

output $avg_S[q]$

Fairness Through Awareness

- Setup:
  - Variable T: indicated protected class or not
  - Output variable O: Indicates our prediction or outcome
  - Variable Y: indicates true/target/desired outcome
    (eg. Individual capable/qualified, individual can afford etc.)

Demographic Parity

$$P(O=1|T=1) = P(O=1|T=0)$$

Problem: when T=0, O can correlate with Y and if T=1, O can be random

Equalized Odds

For all o, y in {0,1}

$$P(O=o|Y=y,T=1) = P(O=o|Y=y,T=0)$$

- O is independent of T given Y

Sufficiency or Predictive Rate Parity

For all o, y in {0,1}

$$P(Y=y|O=o,T=1) = P(Y=y|O=o,T=0)$$

- Y is independent of T given O

Impossible to get any two simultaneously!

Fairness Through Awareness

- From legal requirements to providing uniform service
- To correct data collection biases
- Tools for ML techniques that can satisfy data-dependent constraints

Lessons Learnt

Relationships are key

- Between features (columns): Dimensionality reduction
- Between data points: clustering
- Between nodes in a graph: spectral clustering
- Between subsequent observations in a sequences: HMM
- Between variables in our model: Graphical models

No Free Lunch

- No model is universally good or better (remember the assignments)
- To make good models we need to make good assumptions
- Examples:
  - Probabilistic model generating the data
  - On relationship between various variables
  - Use the right latent variables to induce knowledge about the world

Bigger Picture

- Dimensionality reduction, clustering and more generally learning

There are no free lunches :(

Watch Out!

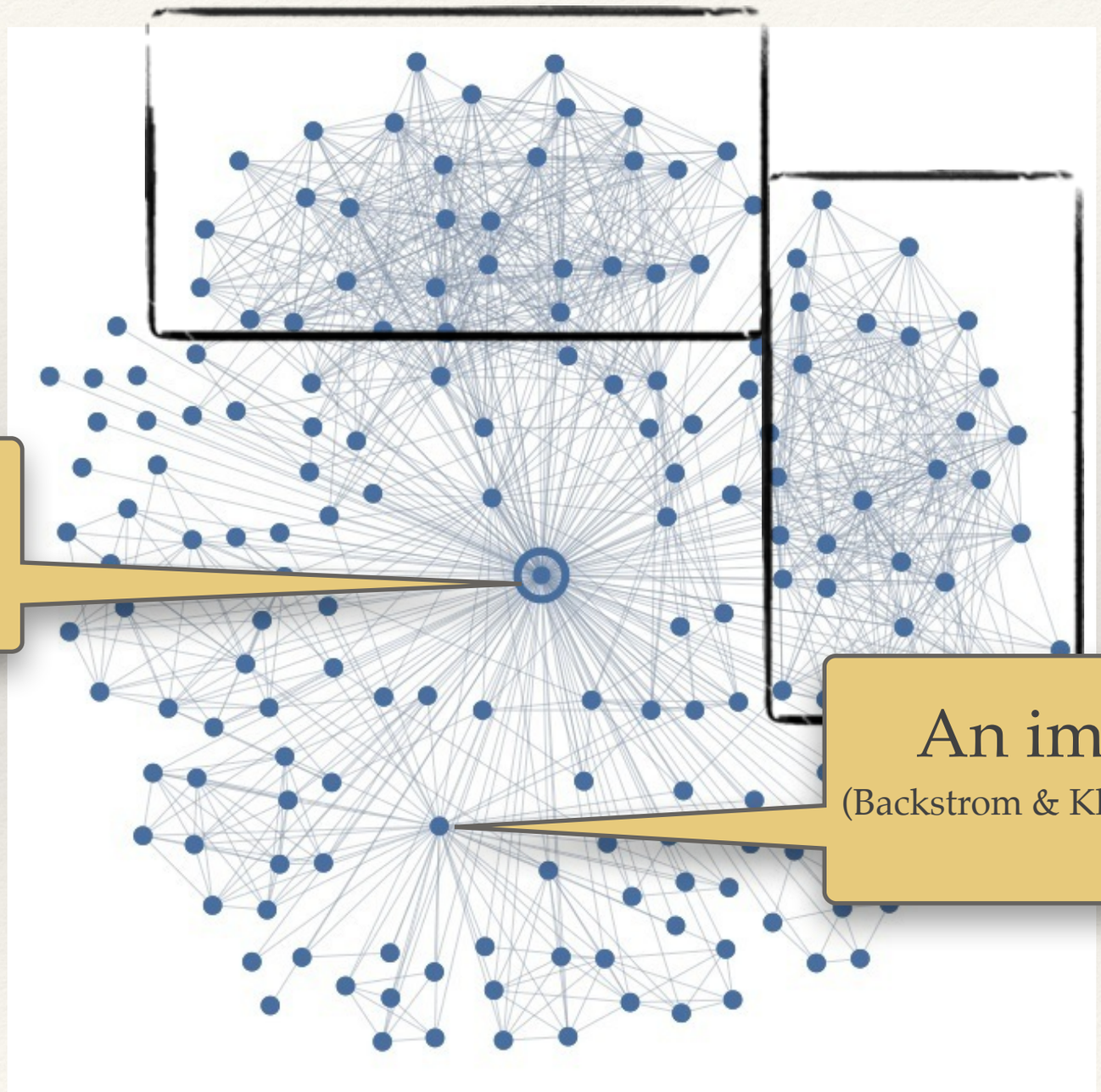
Good Feature Space is Key

- If you don't start with good feature space, you cant get good results
- Understand your problem, talk to practitioners and domain experts
  - Engineer features based on understanding problem Eg. Bag of words Vs N-grams
  - Engineer model based on understanding problem Eg. Convolutional networks

Watchout for Outliers

Methods can be thrown off by Outliers
Might have to remove outliers

Watchout for Outliers



Somebody on Facebook

An important outlier

(Backstrom & Kleinberg, best paper CSCW 2014)

http://overstated.net/wp/uploads/2009/03/asmith-connections.png

Outliers can also have useful information!

Its a big world out there!

Supervised Learning

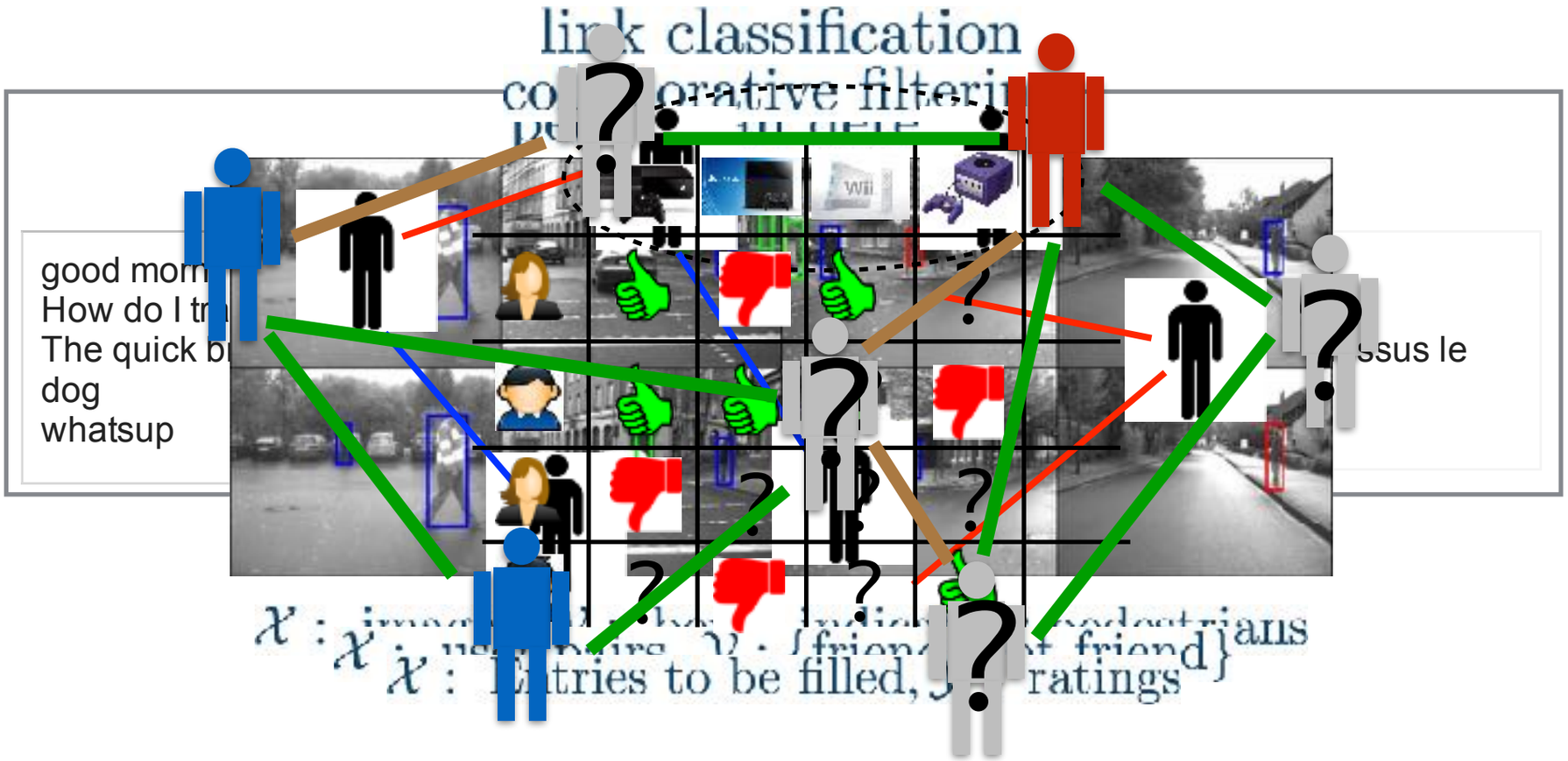
$$\text{Data} : (x_1, y_1)\,(x_2, y_2) \ldots (x_n, y_n)$$

$\mathcal{X}$: Input space

$\mathcal{Y}$: Output space

Goal : learn to predict y given x

Learning Problem



$\mathcal{X}$: Users, underlying social network , $\mathcal{Y}$: Preferences

Supervised Learning

- Training data: $(x_1, y_1), \ldots, (x_n, y_n)$ provided (typically assumed to be drawn from a fixed unknown distribution)
- Goal: Find a mapping h from input instances to outcome that minimizes $\mathbb{E}\Big[\ell(\hat{h}(x), y)\Big]$
  (ℓ is a loss function that measures error in prediction)

Generative Vs Discriminative Approaches

Generative approach:

- Input instances x_t's are generated based on/by y_t's
- We try to model $P(y, x) = P(x|y)P(y)$
- Example: Naive Bayes

Discriminative approach:

- We model $P(Y|X)$ or the boundary of classification
- Rationale: we are only concerned with predicting output y's given input x
- Example: linear regression, logistic regression

Probabilistic Story Vs Optimization Story

- Maximizing likelihood is same as minimizing negative log likelihood.
- Think of - log likelihood as loss function

$$-\log(P_\theta(Y|X)) \rightarrow \text{loss}(h_\theta(X), Y)$$

ie. θ parameterizes hypothesis for prediction or boundary

- MLE = Find hypothesis minimizing empirical loss on data
- Log Prior can be viewed as "regularization" of hypothesis

$$-\log(P(Y|X,\theta)) - \log(P(\theta)) \rightarrow \text{loss}(h_\theta(X), Y) + R(\theta)$$

- MAP = Find hypothesis minimizing empirical loss + regularization term
- Not all losses can be viewed as negative log likelihood

Semi-supervised Learning

- Can we use unlabeled examples to learn better?
- For instance, if we had a generative graphical model for the data: do example
- If we had prior information about the marginal distribution of X's and its relation to $P(Y|X)$

Active Learning

- Humans label the examples, can we get the learning algorithm in the loop?
- Learning algorithm picks the examples it wants labeled
- Eg. Margin based active learning, query points where model that fits observed data well so far disagree most

Domain Adaptation

- We learn a particular task on one corpus but want to use this learnt model to adapt with much fewer examples on another corpus
- Typical assumption: $P(Y|X)$ in both corpus remain fixed
- Marginal distributions change across the corpuses

Online Learning

For $t = 1$ to n

Input instance $x_t \in \mathcal{X}$ is revealed

Learner picks prediction $\hat{y}_t$

Output $y_t \in \mathcal{Y}$ is revealed and learner suffers loss $\ell(\hat{y}_t, y_t)$

End

Thanks!

- For all your patience and understanding
- For helping each other out on Piazza

Hope you had fun, you all worked hard!