Machine Learning

CSCI 5622 Fall 2020

Prof. Claire Monteleoni

Today

- Intro. to Unsupervised Learning
  - Intro. to Matrix Completion, Approximation
  - Intro. to Embedding
  - Intro. to Clustering (if time)

With credit to S. Dasgupta, T. Jaakkola, M. Mohan

Unsupervised learning

There are 4 Unsupervised Learning tasks:

1. Clustering
2. Embedding
3. Learning a generative model
4. Completion (estimating missing values), Approximation

- (Sparse) Matrix (or Tensor) Completion
- Matrix Approximation (see also Dictionary Learning, Matrix Factorization, Compressive Sensing)
  - December 11th guest lecture by Stephen Becker
- Deep Learning approaches
  - Autoencoders for Dictionary Learning
  - DL for in-painting tasks

Recovering and exploiting latent structure

One approach to unsupervised learning is to

- assume some latent structure in the data,
- and then exploit this assumption with algorithms to recover it.

Sparse Matrix Completion

Example task: “collaborative filtering,” predicting a user’s rating of a movie, given their past movie ratings, and those of other users.

- One could try nearest-neighbor techniques, however the data is extremely sparse; each user is a vector of ratings, but each user has only rated a small subset of all movies (many zeros).

➔ Most distance computations between pairs of users will be meaningless (largely disjoint sets of movies rated).

Approach: instead use ratings data of all users to jointly fill in zeros (complete the matrix) and predict ratings on users’ unseen movies.

- Sparse matrix completion works when data has simplifying structure, “low intrinsic dimensionality”
  - e.g. perhaps a small set of genres is predictive of user movie ratings (projected onto genre subspace, users can be grouped by similar prefs).

Sparse Matrix Completion

Sparse matrix completion techniques exploit notions of low intrinsic dimensionality (or latent structure).

→ E.g. if an n x d matrix has rank k:

- if k << n, then the rows can be summarized by a small set of representatives (e.g. via clustering).
- if k << d, then there's a lower dimensional subspace in which the rows can be represented (e.g. via embedding, dimensionality reduction).

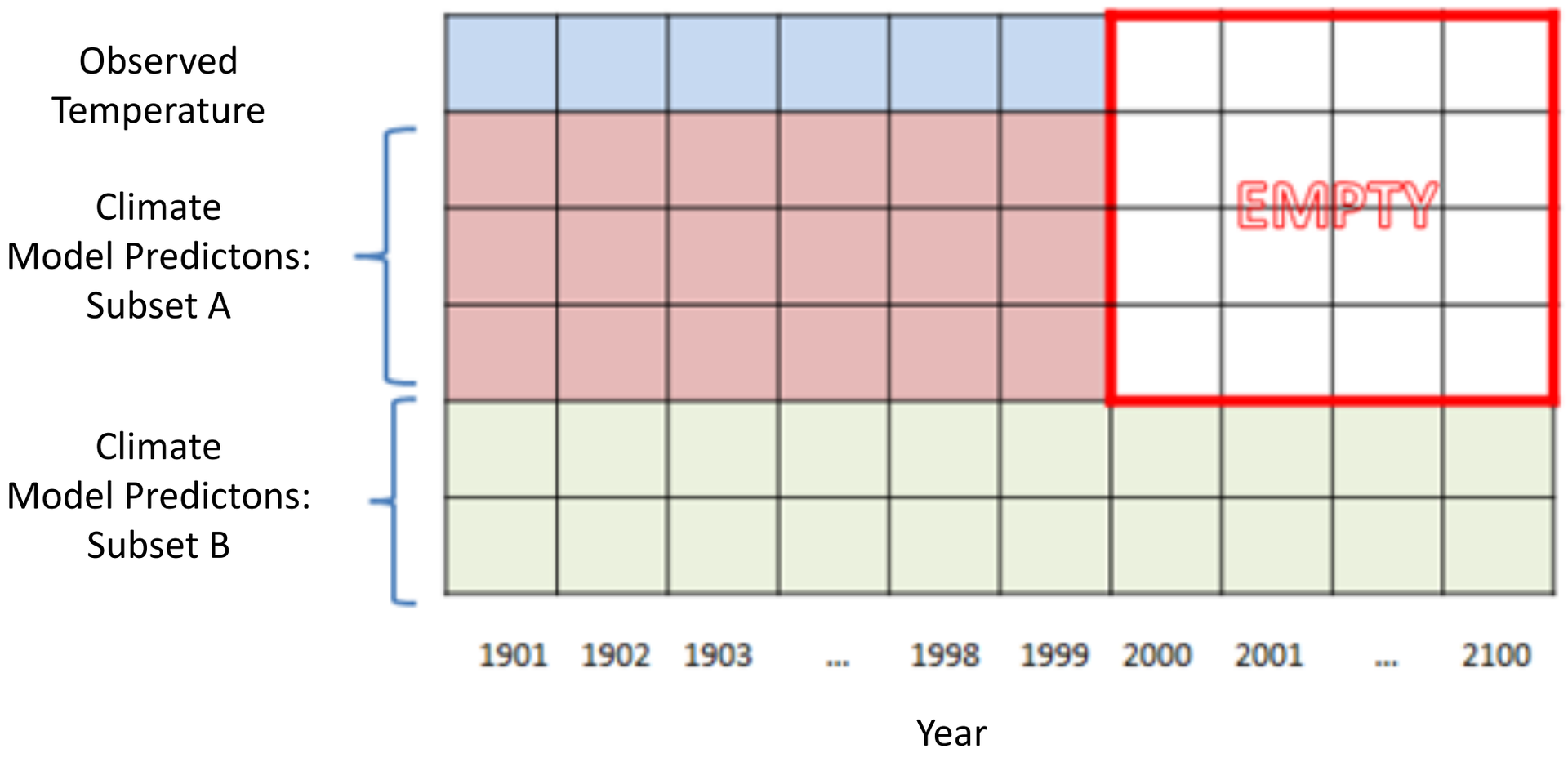
E.g., OptSpace algorithm [Keshavan, Montanari & Oh, '10]

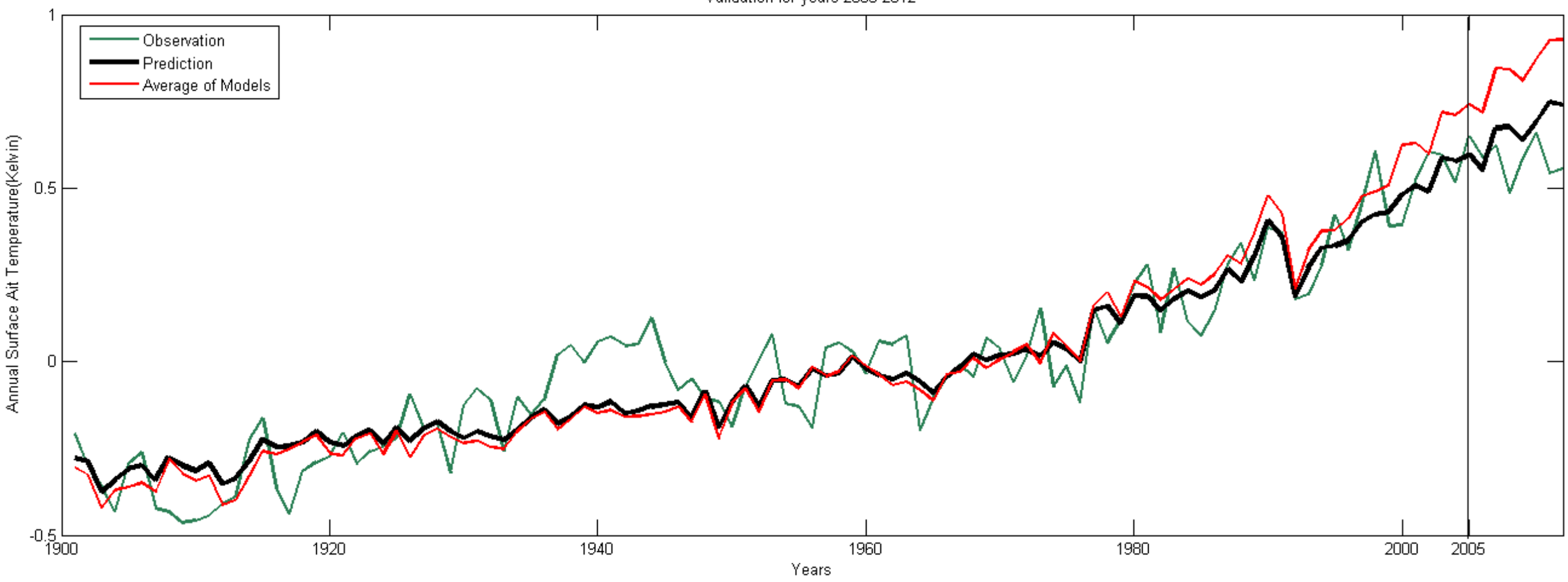
- Minimizes the the nuclear norm (sum of the singular values) of the recovered matrix.
- Uses a combination of spectral techniques and manifold optimization.

Climate Prediction via Matrix Completion

[Ghafarianzadeh & M, Late-Breaking Paper, AAAI 2013]

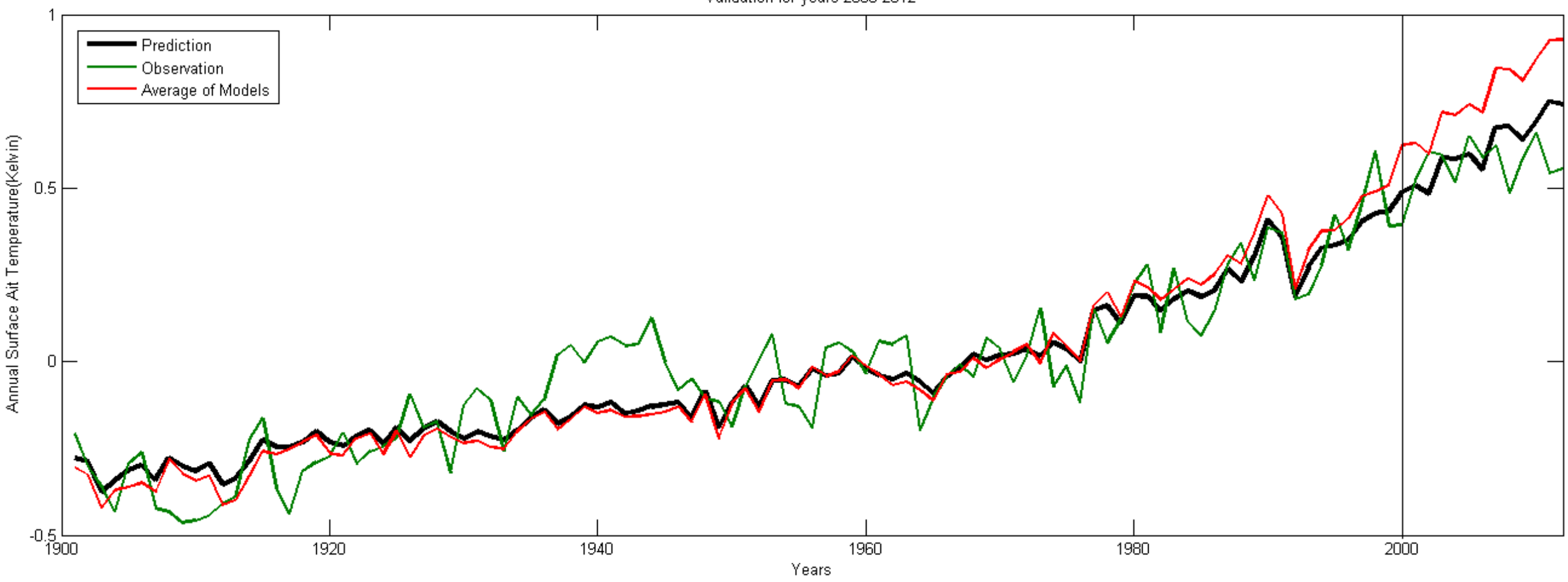
- Recover a sparse matrix of multi-model ensemble predictions and historical observations (OptSpace alg. [Keshavan et al. IEEE Trans. IT, 2010])
- Interpret recovered future (temperature) values as predictions
- Empirical success of the method suggests latent structure





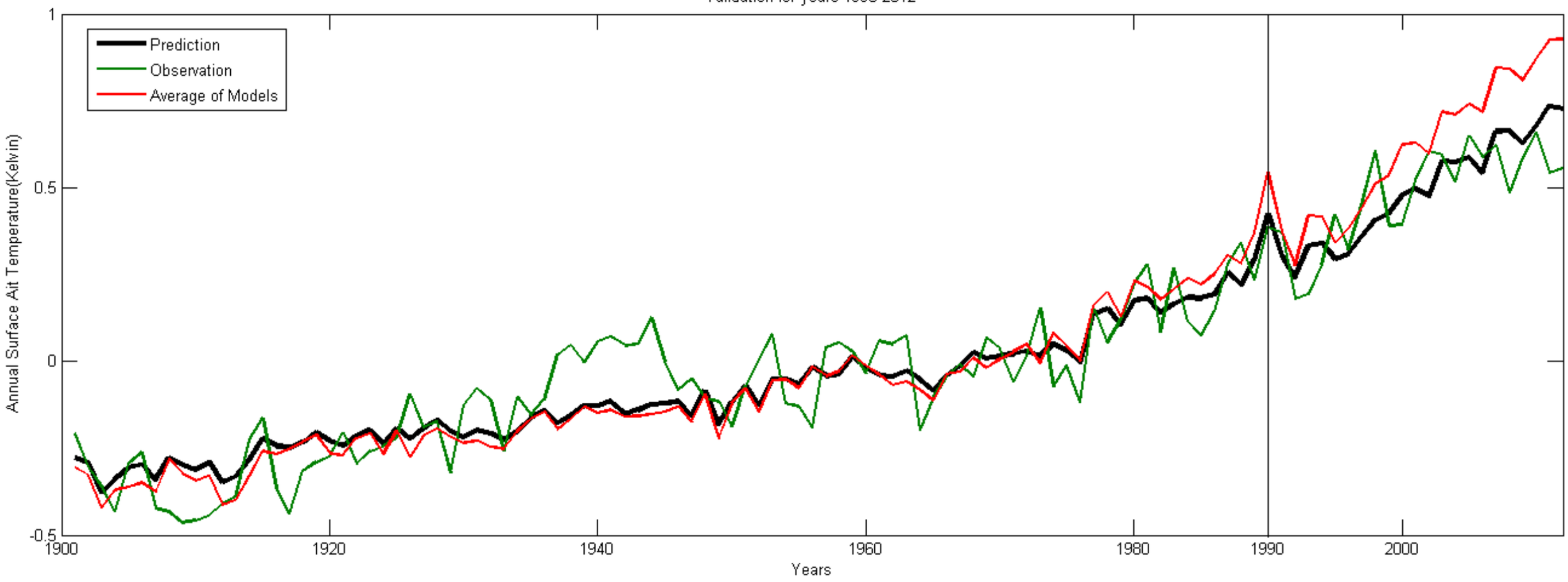
Green: observation, Red: mean prediction of climate models, Black: matrix completion

Validation period: 2005-2012



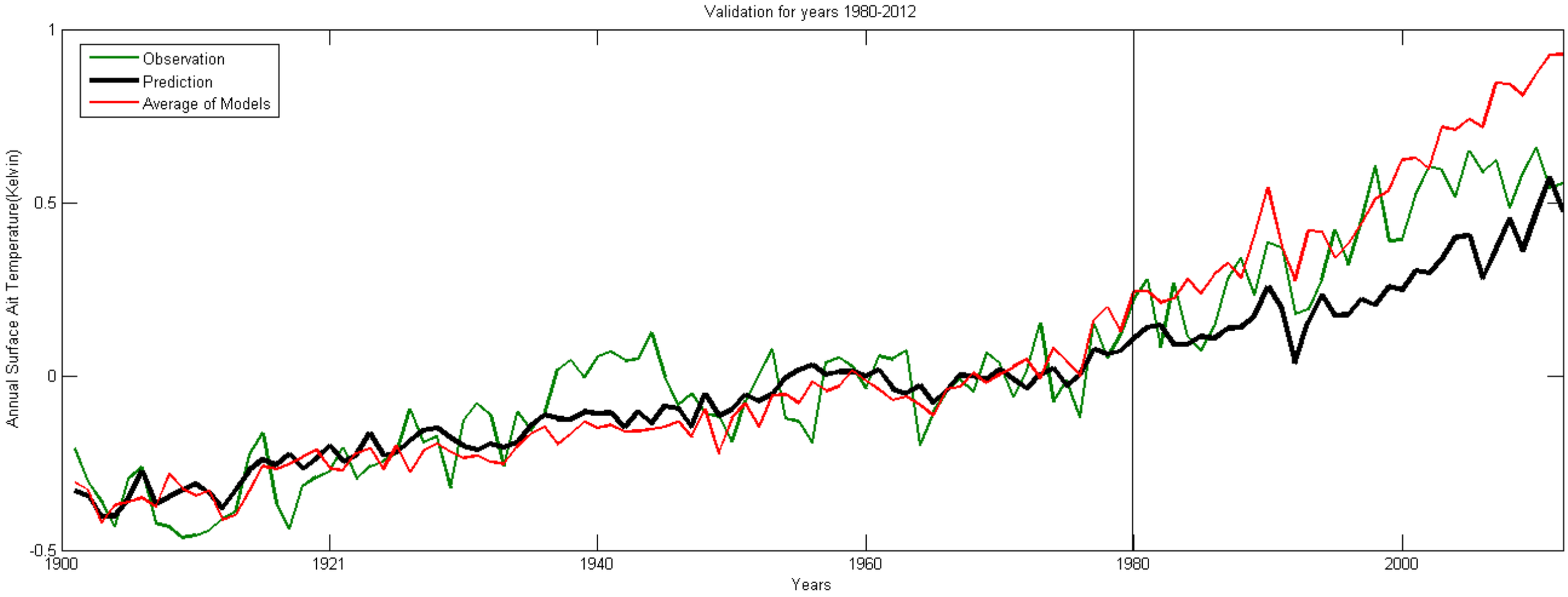
Green: observation, Red: mean prediction of climate models, Black: matrix completion

Validation period: 2000-2012



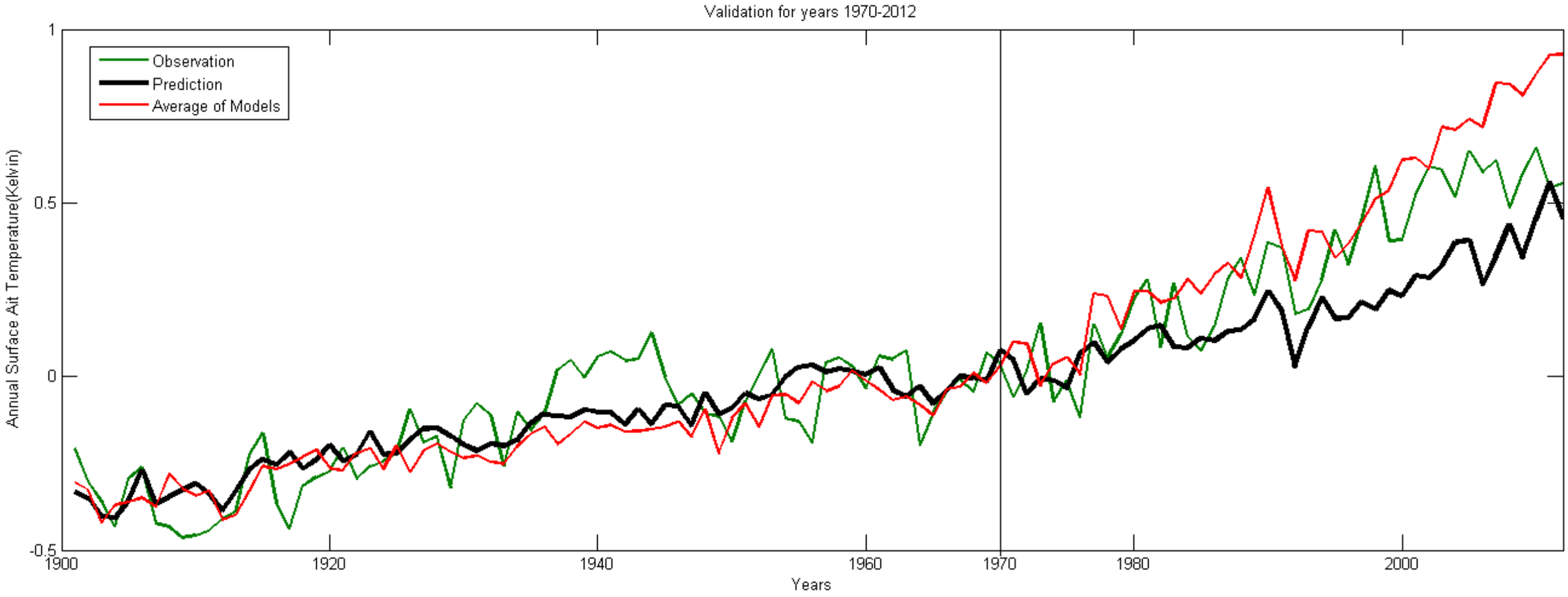
Green: observation, Red: mean prediction of climate models, Black: matrix completion

Validation period: 1990-2012



Green: observation, Red: mean prediction of climate models, Black: matrix completion

Validation period: 1980-2012

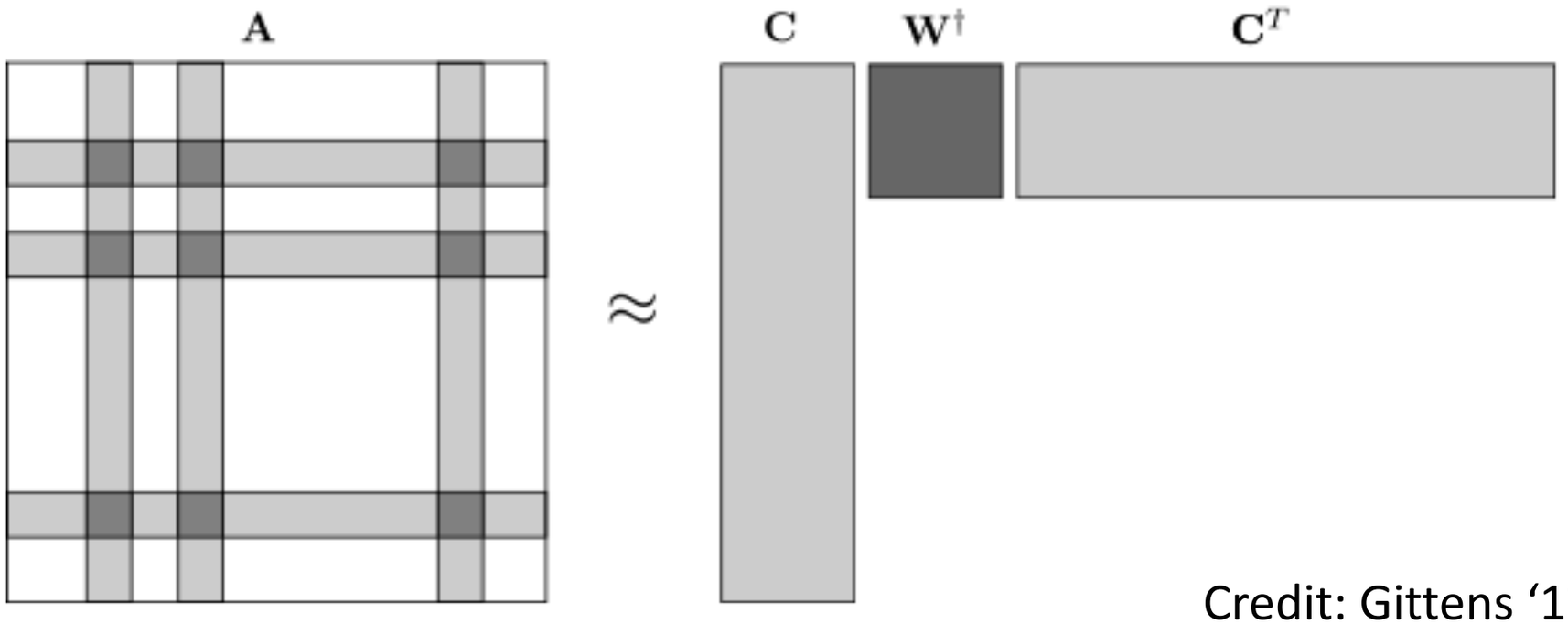


Green: observation, Red: mean prediction of climate models, Black: matrix completion

Validation period: 1970-2012

Matrix Approximation

E.g. the Nyström method:



Credit: Gittens '13.

SPSD Sketching Model. Let $\mathbf{A}$ be an $n \times n$ positive semi-definite matrix, and let $\mathbf{S}$ be a matrix of size $n \times \ell$, where $\ell \ll n$. Take $\mathbf{C} = \mathbf{AS}$ and $\mathbf{W} = \mathbf{S}^T\mathbf{AS}$. Then $\mathbf{CW}^\dagger\mathbf{C}^T$ is a low-rank approximation to $\mathbf{A}$ with rank at most ℓ.

[Gittens & Mahoney '13]

Dimensionality Reduction

Why reduce the number of features in a data set?

1. It reduces storage and computation time.
2. High-dimensional data often has a lot of redundancy.
3. Remove noisy or irrelevant features.

Example: are all the pixels in an image equally informative?

If we were to choose a few pixels to discard, which would be the prime candidates?

Dimensionality reduction

- How to handle the big data setting?
- What if some of the features are redundant (e.g. deterministic functions of other features, or dependent random variables), or simply not helpful for classification?

- Dimensionality reduction is the study of how to address these issues, as well as more general issues:
  - Perhaps the relevant data lives in a lower dimensional subspace. This lower dimension is known as the intrinsic dimension of the data.
  - This could be a subset of the dimensions (or features) of the input data, or perhaps some manifold of low intrinsic dimension.
  - Meanwhile, to reduce the complexity of learning a classifier, you might want to compute in a lower dimension.

Dimensionality reduction

Many techniques, including:

- Random projections (see RP-tree)
- Autoencoders: neural-networks approach to unsupervised learning
- Spectral embedding e.g. PCA, SVD
- …
- Feature selection: dimensionality reduction techniques that select a subset of the input dimensions

Spectral Methods

- Spectral methods can be used for both clustering and embedding

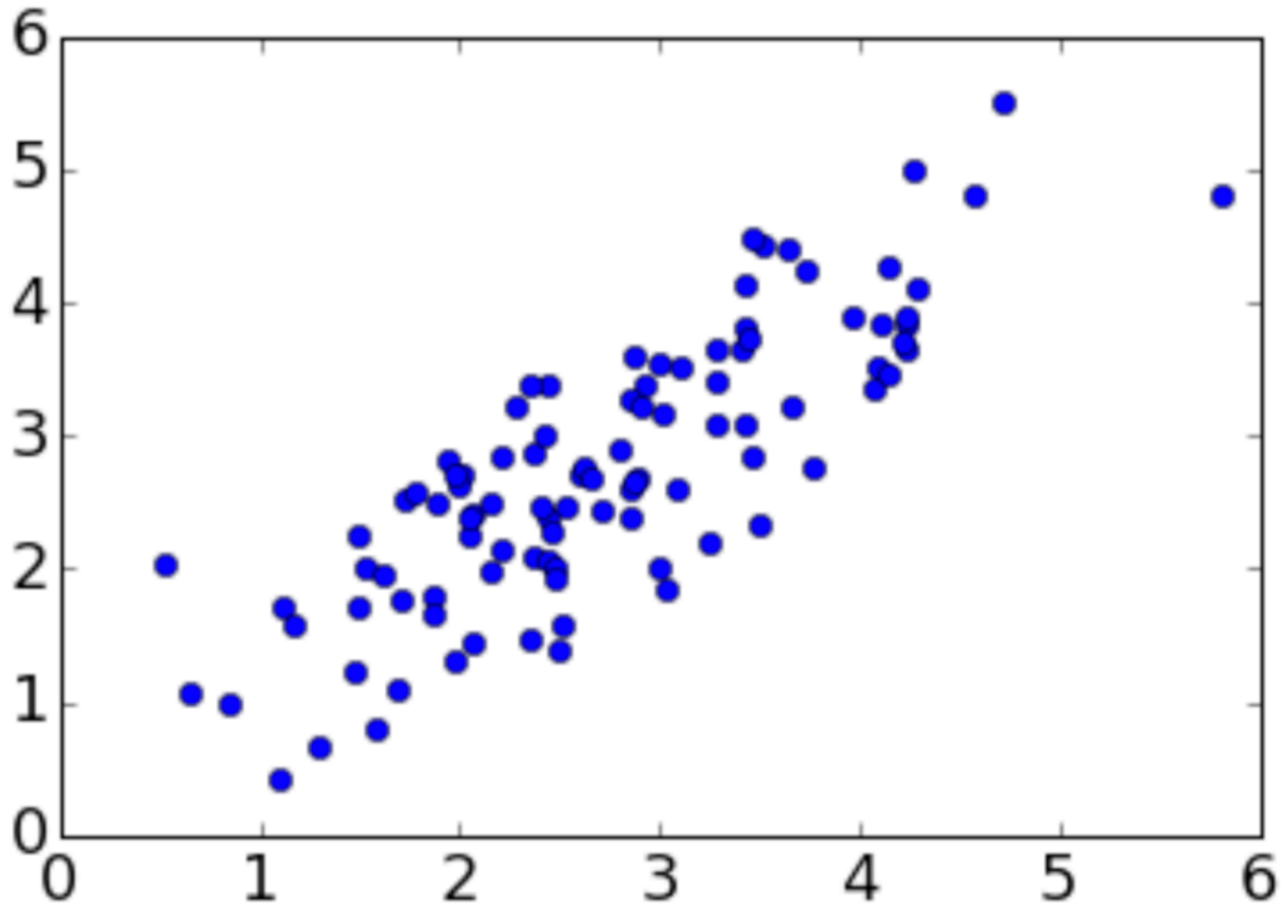
- If you're interested in the theory:
  - Reference text: Kannan & Vempala, "Spectral Algorithms."

Spectral Methods for Embedding

- Principal Components Analysis (PCA)
- Singular Value Decomposition (SVD)
  - Similar to PCA, when the matrix is non-square

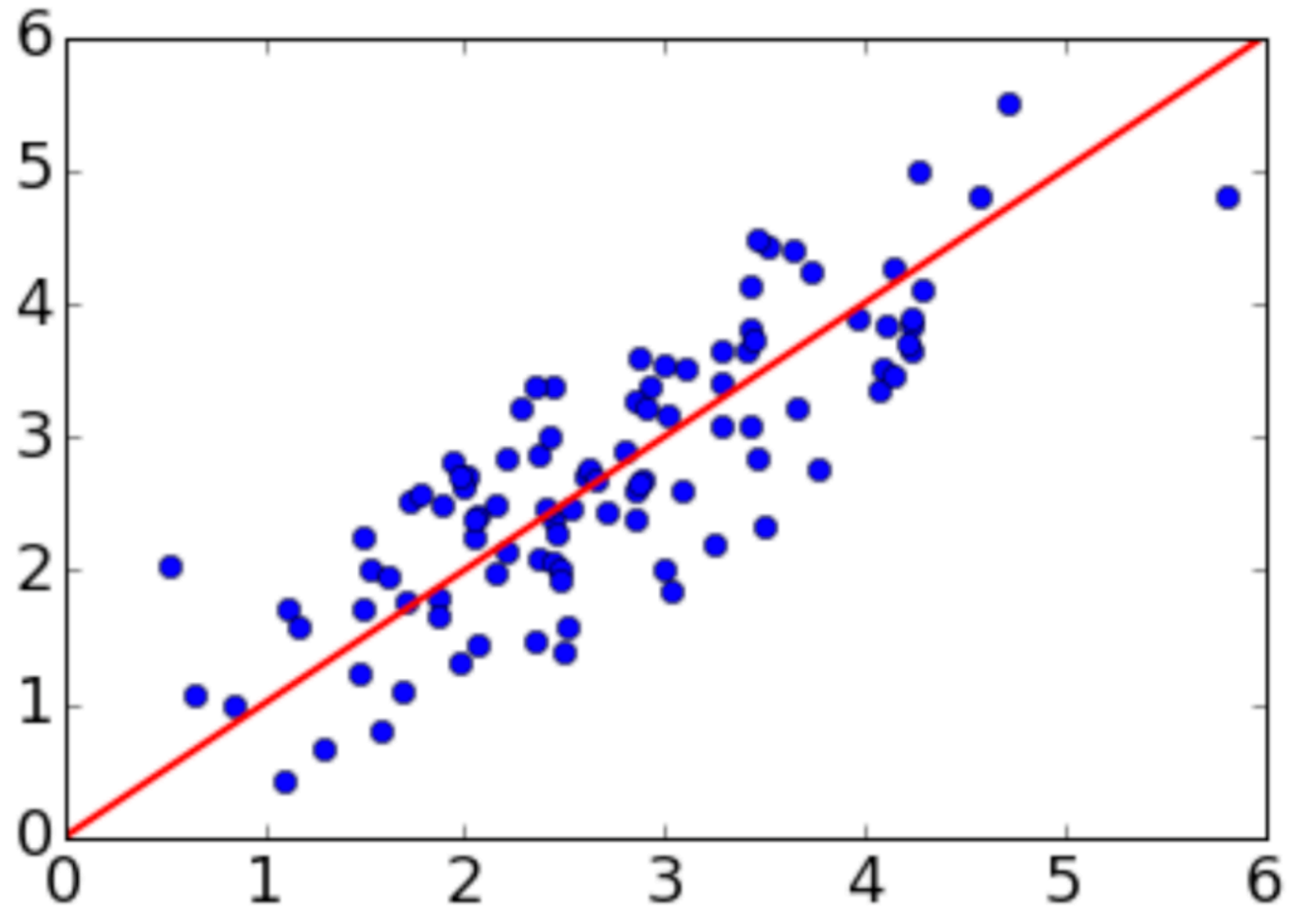
Principal Components Analysis

- Q: How can we (approximately) represent this entire data set using a single feature?



Principal Components Analysis

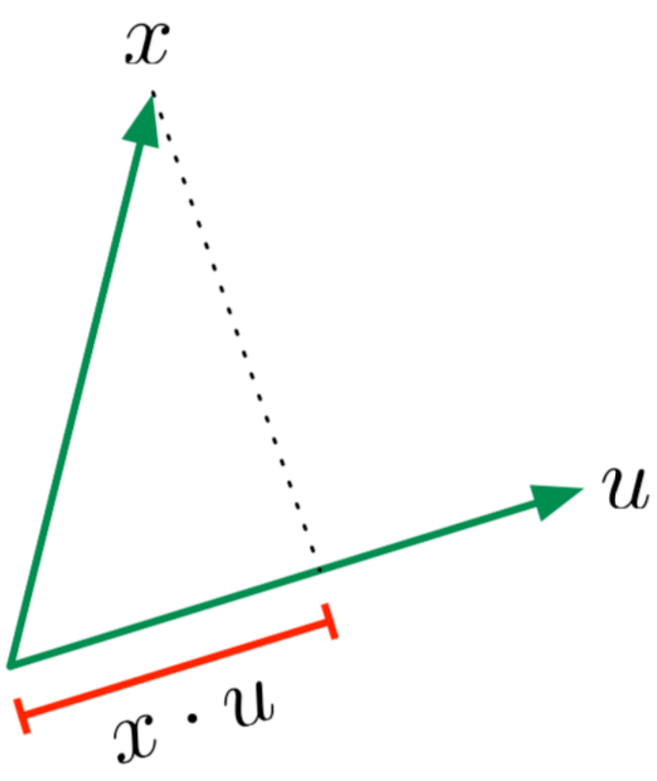
- Q: How can we (approximately) represent this entire data set using a single feature?



- A: Use the direction of maximum variance

Projection (review)

What is the projection of $x \in \mathbb{R}^d$ in the **direction** $u \in \mathbb{R}^d$?
Assume u is a unit vector (i.e. $\|u\| = 1$).



Projection is

$$x \cdot u = u \cdot x = u^T x = \sum_{i=1}^{d} u_i x_i.$$

Best 1-dimensional projection

Suppose we need to map our data $x \in \mathbb{R}^d$ into just **one** dimension:

$$x \mapsto u \cdot x \quad \text{for some unit direction } u \in \mathbb{R}^d$$

What is the direction u of maximum variance?

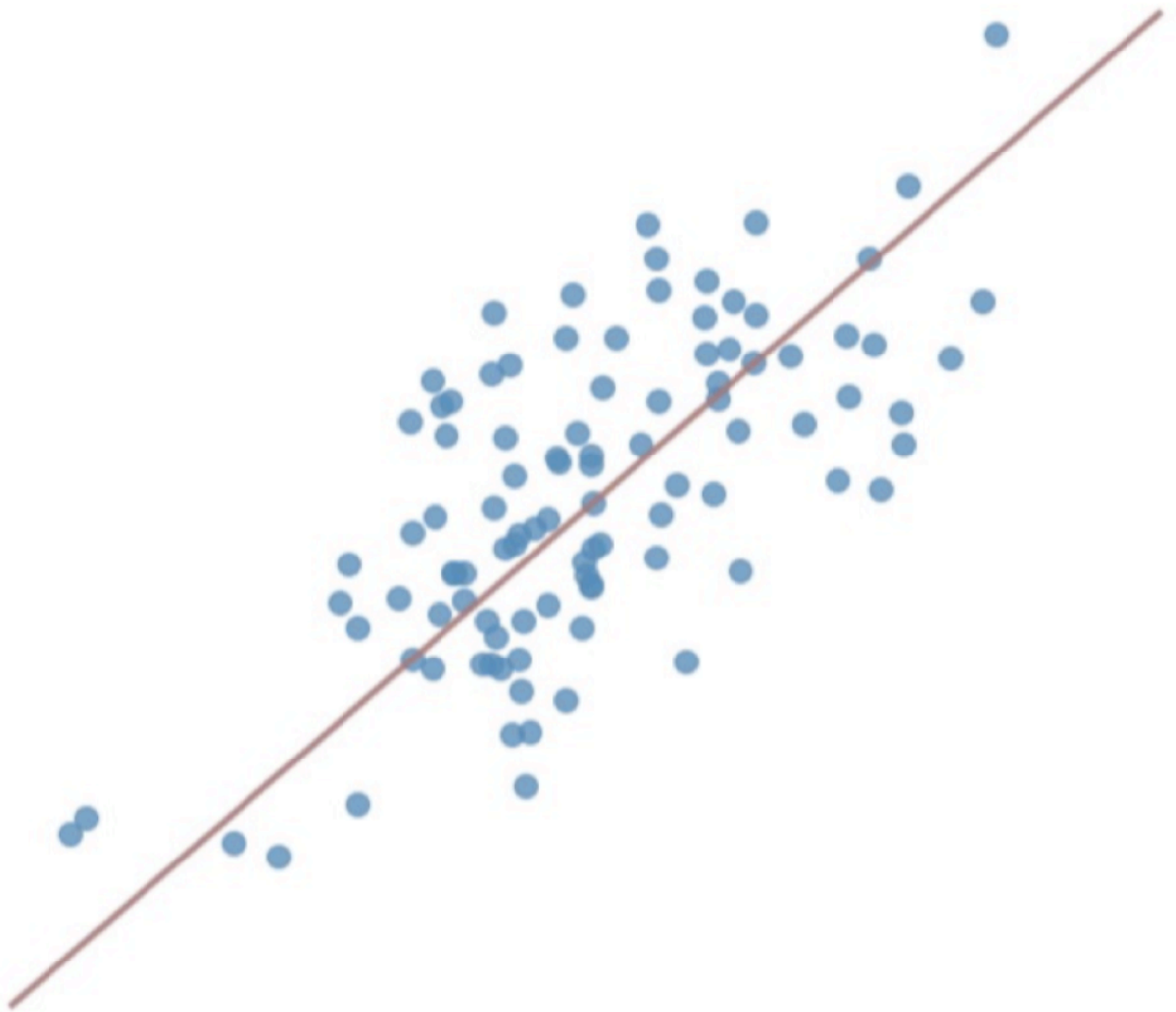
Useful fact 1:

- Let Σ be the $d \times d$ covariance matrix of X.
- The variance of X in direction u is given by $u^T \Sigma u$.

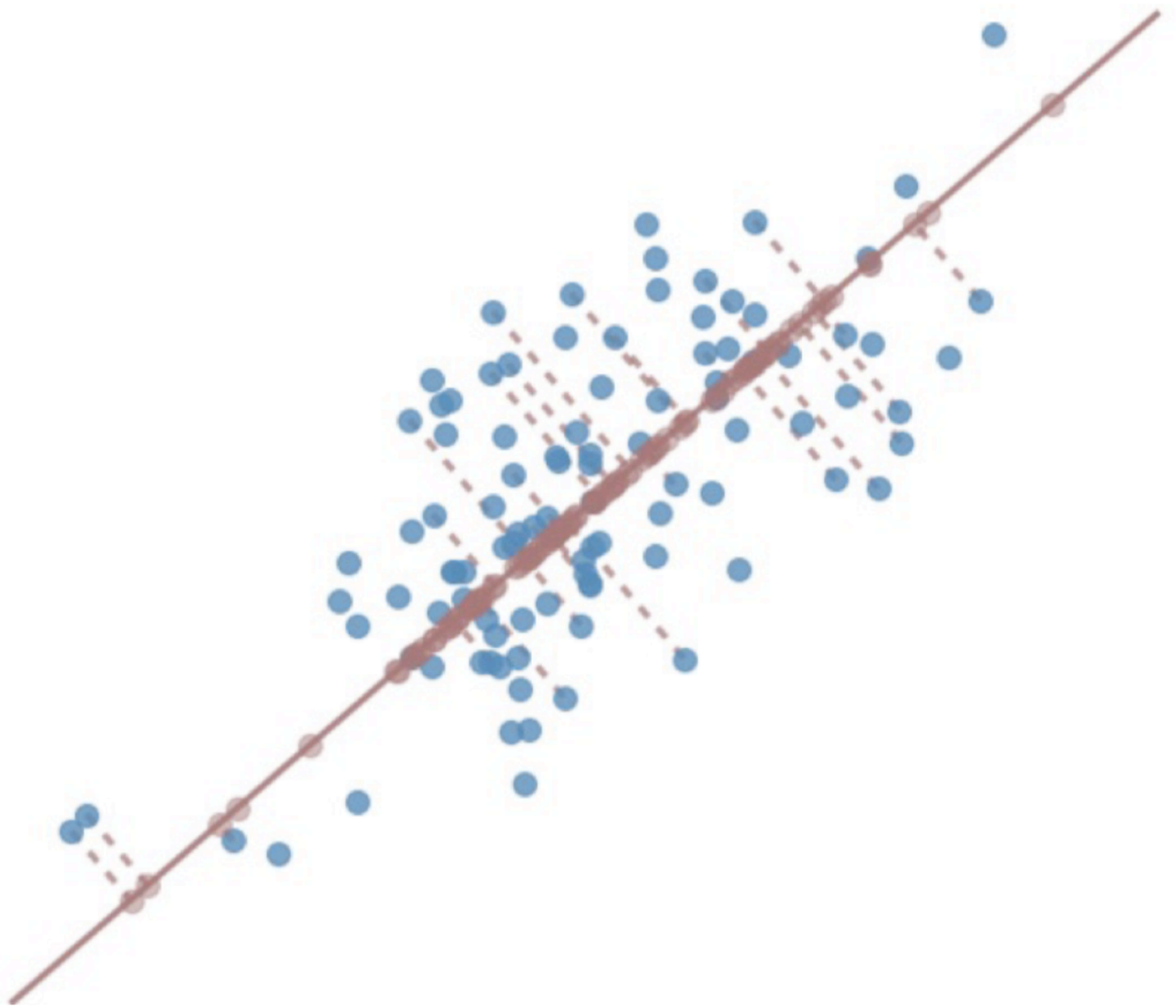
Useful fact 2:

- $u^T \Sigma u$ is maximized by setting u to the first **eigenvector** of Σ.
- The maximum value is the corresponding **eigenvalue**.

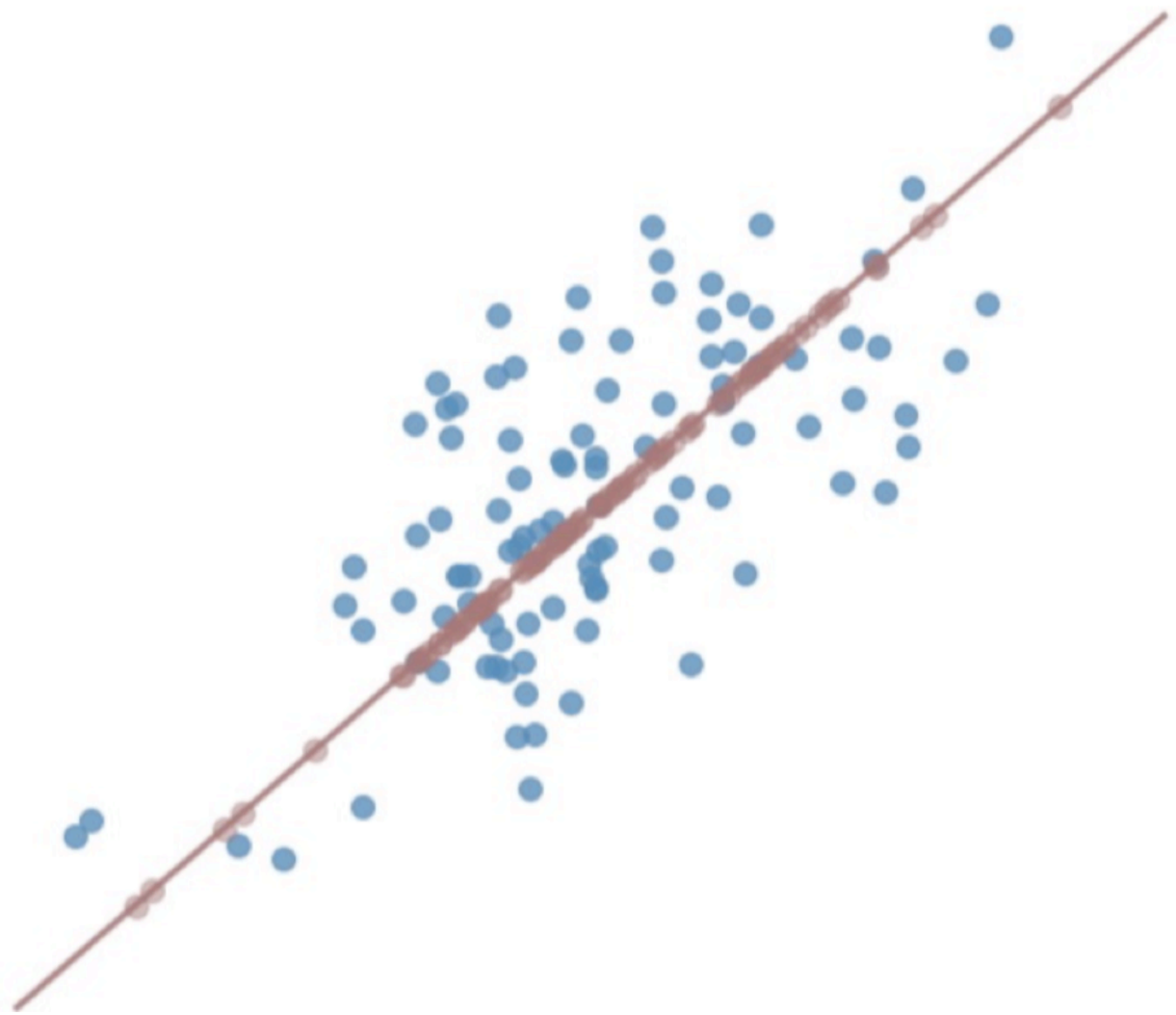
Projection onto direction of highest variance



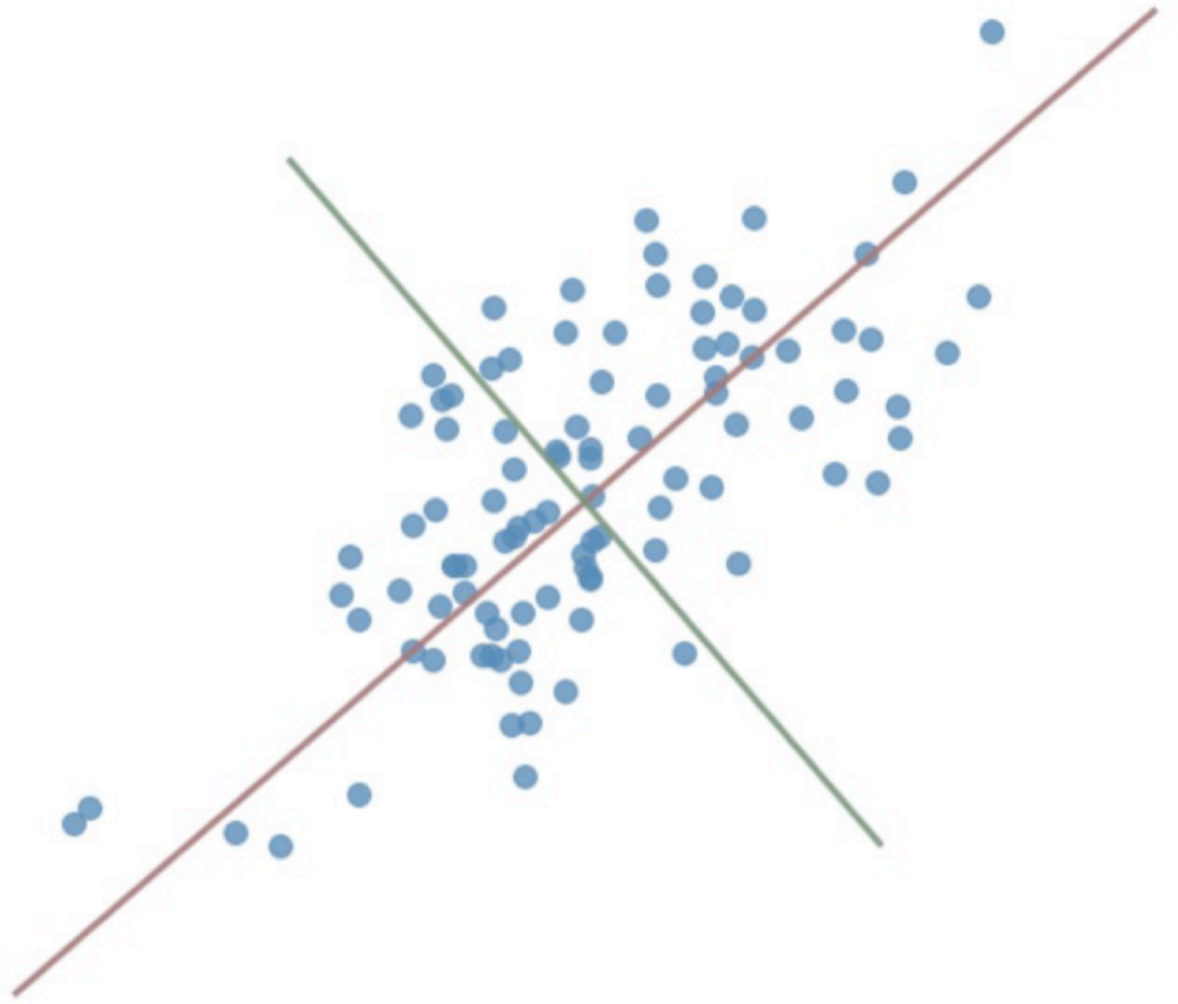
Projection onto direction of highest variance



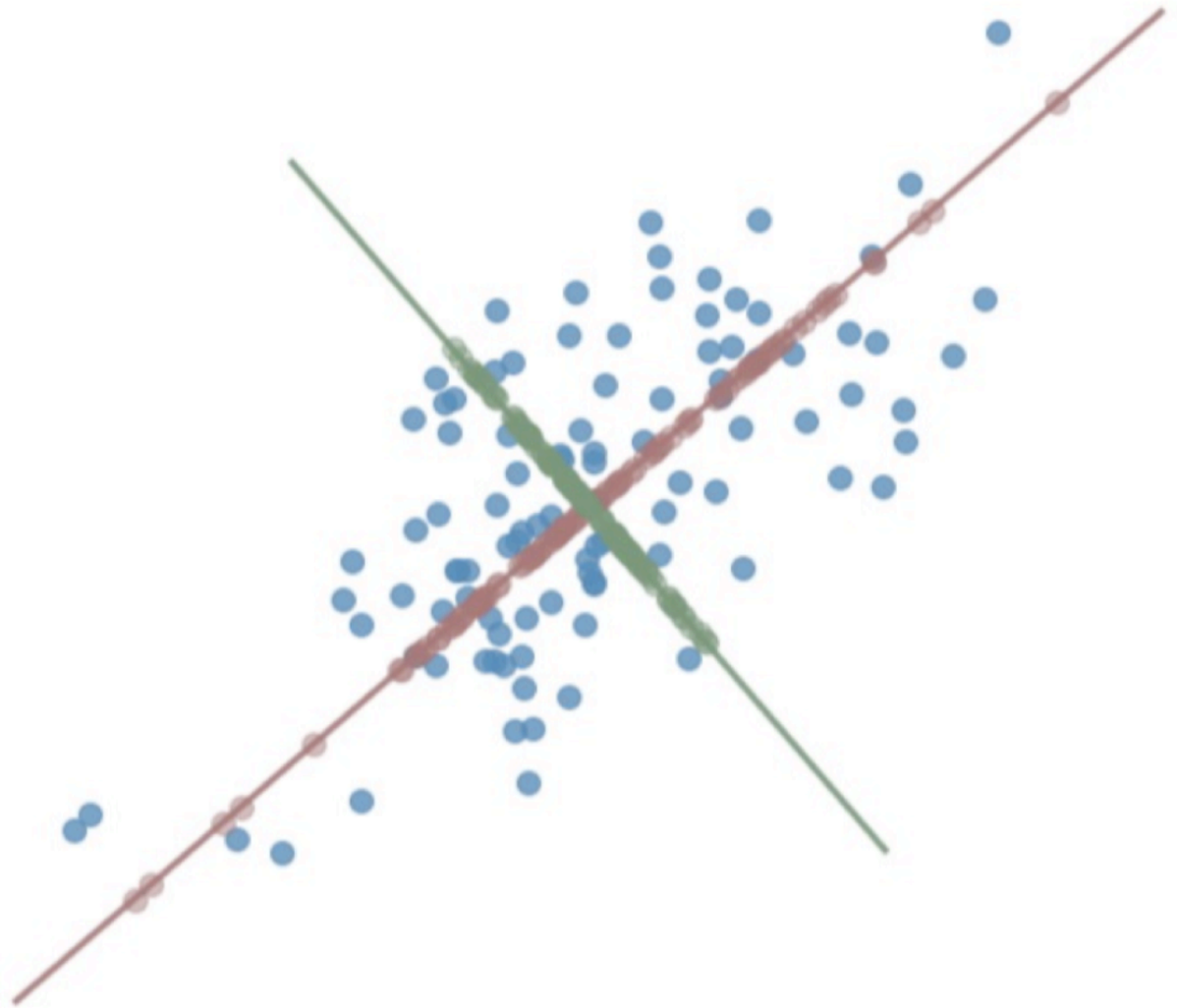
Projection onto direction of highest variance



Projection onto **top 2** directions of highest variance



Projection onto **top 2** directions of highest variance



Best k-dimensional projection

Let Σ be the $d \times d$ covariance matrix of X.
In $O(d^3)$ time, we can compute its **eigendecomposition**, consisting of

- real **eigenvalues** $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$
- corresponding **eigenvectors** $u_1, \ldots, u_d \in \mathbb{R}^d$ that are orthonormal (unit length and at right angles to each other)

Fact: Suppose we want to map data $X \in \mathbb{R}^d$ to just k dimensions, while capturing as much of the variance of X as possible. The best choice of projection is:

$$x \mapsto (u_1 \cdot x, u_2 \cdot x, \ldots, u_k \cdot x),$$

where u_i are the eigenvectors described above.

Principal Components Analysis (PCA)

Let Σ be the $d \times d$ covariance matrix of X.
In $O(d^3)$ time, we can compute its **eigendecomposition**, consisting of

- real **eigenvalues** $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$
- corresponding **eigenvectors** $u_1, \ldots, u_d \in \mathbb{R}^d$ that are orthonormal (unit length and at right angles to each other)

Fact: Suppose we want to map data $X \in \mathbb{R}^d$ to just k dimensions, while capturing as much of the variance of X as possible. The best choice of projection is:

$$x \mapsto (u_1 \cdot x, u_2 \cdot x, \ldots, u_k \cdot x),$$

where u_i are the eigenvectors described above.

This is called the k-PCA embedding of the data set X.

Dimensionality reduction

- One powerful technique is to take a set of random projections.

- Training:
  - Project the data set onto a random, lower dimensional subspace
  - Learn a classifier
  - Repeat multiple times (re-randomizing each time).
- Prediction:
  - On a new unlabeled example, compute its projections from above (do not re-randomize)
  - Classify the example under each classifier learned above.
  - Compute an aggregate classification. (Need to specify how to do the aggregation, for the particular classification task).

Note: can be done analogously for regression tasks.

Dimensionality reduction

Many techniques, including:

- Random projections (see RP-tree)
- Autoencoders: neural-networks approach to unsupervised learning
- Spectral embedding e.g. PCA, SVD
- …
- Feature selection: dimensionality reduction techniques that select a subset of the input dimensions

Feature selection

- Techniques to find a relevant subset of the dimensions or features are called Feature selection.

classification

$$y = \mathrm{sign}\left(\underline{\theta} \cdot \underline{\phi}(\underline{x}) \right) \in \{-1, 1\}$$

E.g. Linear models:

regression

$$y = \underline{\theta} \cdot \underline{\phi}(\underline{x}) \in \mathcal{R}$$

etc.

feature coordinate

$$\underline{\phi}(\underline{x}) = \begin{bmatrix} \phi_1(\underline{x}) \\ \cdots \\ \phi_d(\underline{x}) \end{bmatrix}$$

- We seek to identify a few feature coordinates that the class label (or regression output) primarily depends on
- This is often advantageous in order to improve generalization (as feature selection exerts additional complexity control, or regularization) or to gain interpretability.

Simple feature selection

- There are a many approaches to dimensionality reduction. Some common feature selection techniques:
- 1-norm regularization
  - replaces the L2-norm regularizer with the L1-norm: encourages some of the coordinates to be set exactly to zero
  - E.g. LASSO, we saw in Regression unit.
- Information analysis
  - rank individual features according to their mutual information with the class label
  - limited to discrete variables
- Iterative subset selection
  - iteratively add (or prune) coordinates based on their impact on the (training) error

Information analysis

- Suppose the feature vector is just the input vector x whose coordinates take values in {1,...,k}
- Given a training set of size n, we can evaluate an empirical estimate of the mutual information between the coordinate values and the binary label

$$\hat{I}(Y, X_i) = \sum_{y \in \{-1,1\}} \sum_{x_i=1}^{k} \hat{P}(y, x_i) \log \frac{\hat{P}(y, x_i)}{\hat{P}(y)\hat{P}(x_i)}$$

empirical estimates

$$\hat{P}(y, x_i) = \frac{1}{n} \sum_{t=1}^{n} \delta(y, y_t)\delta(x_i, x_{ti})$$

$$\hat{P}(y) = \frac{1}{n} \sum_{t=1}^{n} \delta(y, y_t)$$

$$\hat{P}(x_i) = \frac{1}{n} \sum_{t=1}^{n} \delta(x_i, x_{it})$$

$$\begin{cases} 1, & \text{if } x_i = x_{it} \\ 0, & \text{otherwise} \end{cases}$$

Information analysis

- Suppose the feature vector is just the input vector x whose coordinates take values in {1,...,k}
- Given a training set of size n, we can evaluate an empirical estimate of the mutual information between the coordinate values and the binary label

$$\hat{I}(Y, X_i) = \sum_{y \in \{-1,1\}} \sum_{x_i=1}^{k} \hat{P}(y, x_i) \log \frac{\hat{P}(y, x_i)}{\hat{P}(y)\hat{P}(x_i)}$$

- Pro: provides a ranking of the features to include
- Con: weights redundant features equally (includes neither or both)
- Pro: Filter method, i.e. you can run it before doing learning, so it's not tied to the particular classifier being learned

  Con: → However, may select features that the particular classifier cannot use, or omit combinations of features particularly useful for the classifier

Subset selection

- An algorithm for finding a good subset of feature coordinates (feature functions) to use

$$\phi_1(\underline{x}), \ldots, \phi_d(\underline{x}) \qquad \underline{\phi}_S(\underline{x}) = \{\phi_j(\underline{x})\}_{j \in S}$$

for each subset S, $|S| = k$, evaluate

$$J(S) = \min_{\underline{\theta}_S} \frac{1}{2} \sum_{t=1}^{n} \left(y_t - \underline{\theta}_S \cdot \underline{\phi}_S(\underline{x}_t)\right)^2$$

each feature subset is assessed on the basis of the resulting training error

$$\hat{S} = \operatorname*{argmin}_S J(S)$$

find the best subset

- k is used for (statistical) complexity control
- computationally hard (exponential in k!)
- This is a wrapper method, as opposed to a filter method, e.g. you need to learn the classifier (or regressor) to evaluate this method.

Greedy subset selection

- A greedy algorithm for finding a good subset of feature coordinates (feature functions) to use

$$\phi_1(\underline{x}), \ldots, \phi_d(\underline{x}) \qquad \underline{\phi}_S(\underline{x}) = \{\phi_j(\underline{x})\}_{j \in S}$$

$$S = \emptyset$$

for each $j \notin S$ evaluate — try each new coordinate

$$J(S \cup j) = \min_{\underline{\theta}_{S \cup j}} \frac{1}{2} \sum_{t=1}^{n} \left(y_t - \underline{\theta}_{S \cup j} \cdot \underline{\phi}_{S \cup j}(\underline{x}_t) \right)^2$$

re-estimate all the parameters in the context of the new coordinate

$$\hat{j} = \operatorname*{argmin}_{j \notin S} J(S \cup j)$$

find the best coordinate to include

$$S \leftarrow S \cup \{\hat{j}\}$$

- each new feature is assessed in the context of those already included
- the method is not guaranteed to find the optimal subset

Forward-fitting

- We can also choose feature coordinates without re-estimating the parameters associated with already included coordinates

$$\phi_1(\underline{x}), \ldots, \phi_d(\underline{x}) \qquad \underline{\phi}_S(\underline{x}) = \{\phi_j(\underline{x})\}_{j \in S}$$

$$S = \emptyset, \quad \underline{\hat{\theta}}_\emptyset = 0$$

for each j evaluate

fixed at this stage

$$J(\underline{\hat{\theta}}_S, j) = \min_{\theta_j} \frac{1}{2} \sum_{t=1}^{n} \left(y_t - \underline{\hat{\theta}}_S \cdot \underline{\phi}_S(\underline{x}_t) - \theta_j \phi_j(\underline{x}_t) \right)^2$$

re-estimate only the parameter associated with the new coordinate

$$\hat{j} = \operatorname*{argmin}_j J(\underline{\hat{\theta}}_S, j)$$

select the best new feature

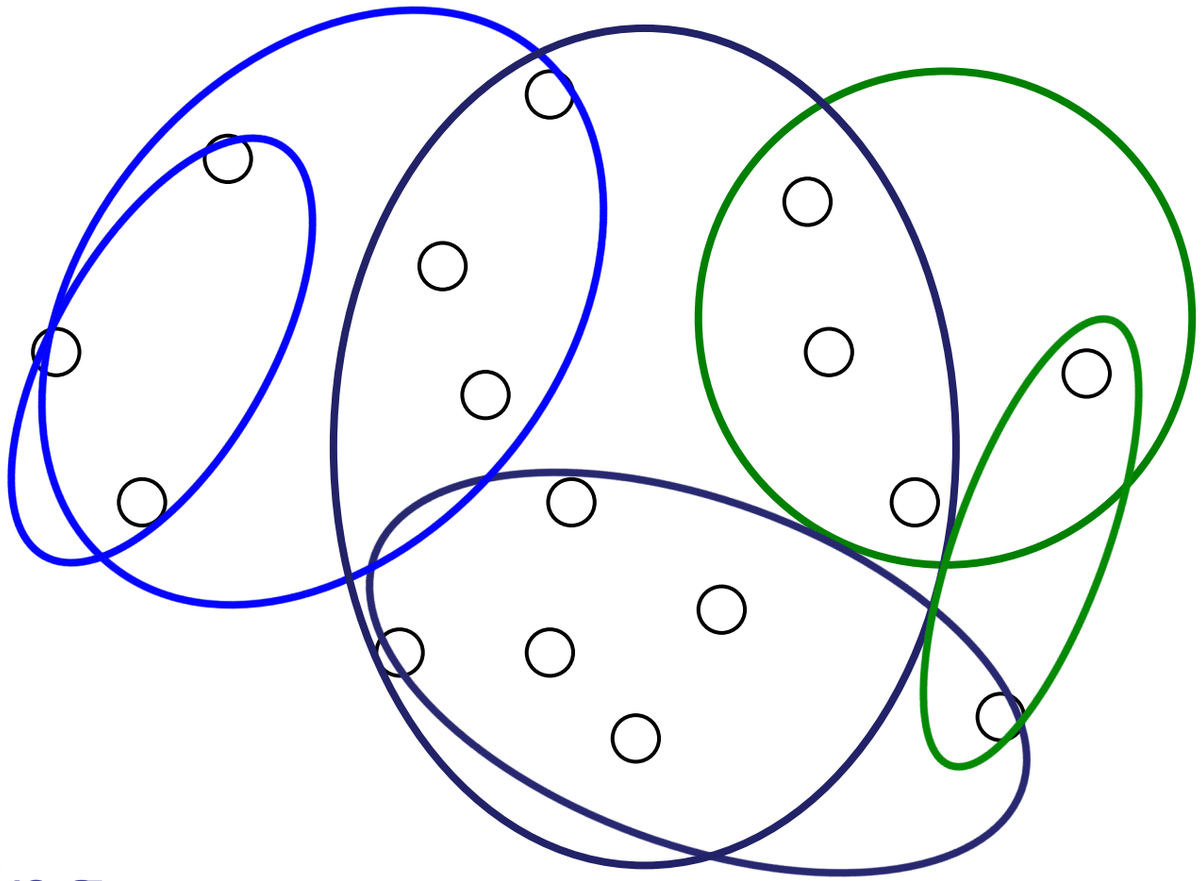
$$\underline{\hat{\theta}}_{S \cup j} = \{\underline{\hat{\theta}}_S, \hat{\theta}_{\hat{j}}\}, \quad S \leftarrow S \cup \{\hat{j}\},$$

- same feature may be included more than once

Intro. to Clustering

What can be done without any labels?

Unsupervised learning



E.g. Clustering

Intro to clustering

Why use clustering?

- Preprocessing in a supervised learning pipeline
  - Quantization or discretization of a continuous data space
  - Summarization of big data
  - Cluster data points and then apply supervised learning separately to each cluster
  - Cluster the features, as a form of embedding

- Exploratory data analysis
  - Discover “groups” in the data (e.g. communities in social networks)
  - Discover hierarchical or nested structure

Clustering is a huge field

- Probabilistic methods
  - Mixture models
    - E.g. EM, and line of results depending on variance, etc.
  - Topic models
- Spectral clustering methods
- Clustering in metric spaces
- Among others, …

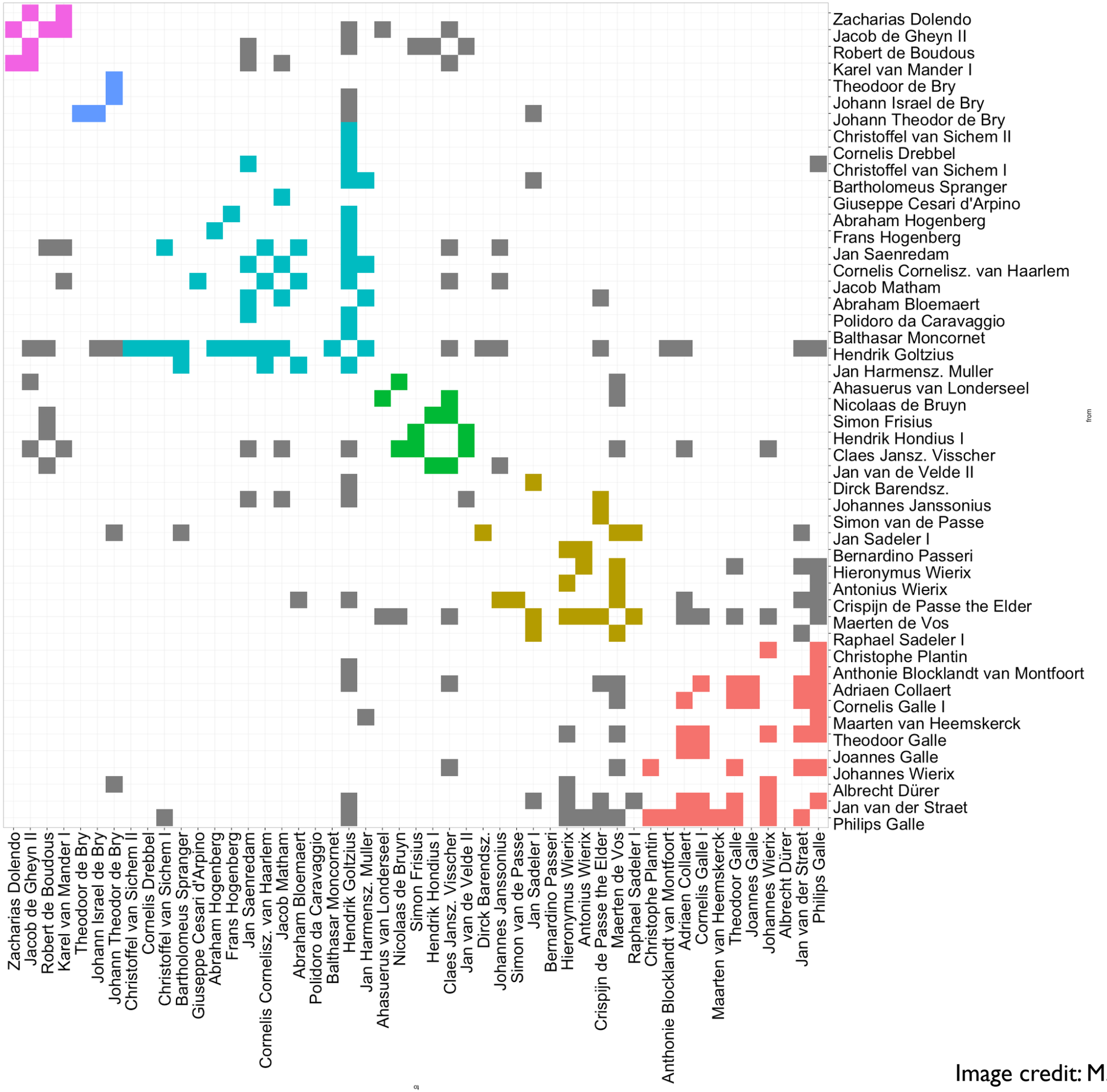


Image credit: M. Lincoln

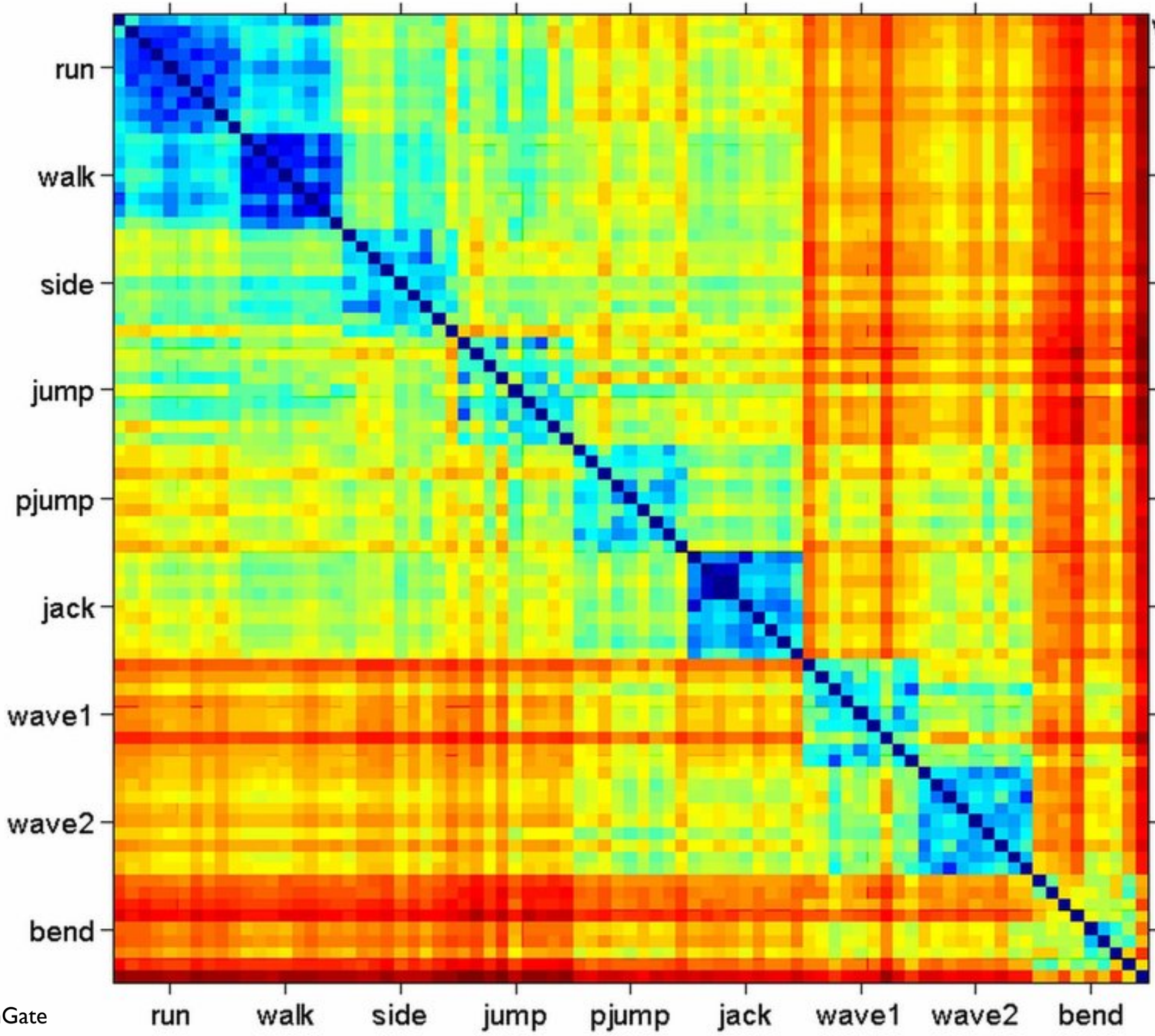


Image credit: ResearchGate

Spectral clustering

[Ng, Jordan, Weiss, '02]

Input: data set $S = \{s_1, s_2 \ldots, s_n\} \in \mathbb{R}^d$,

number of clusters *k*, kernel function $\kappa : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$

Compute:

Affinity matrix: $A \in \mathbb{R}^{n \times n}$ s.t. $A_{ij} = \delta[i \neq j]\kappa(s_i, s_j)$

$D \in \mathbb{R}^{n \times n}$ s.t. $D_{ij} = \delta[i = j] \sum_{j=1}^{n} A_{ij}$

Graph Laplacian:: $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

Top *k* eigenvectors of *L* := $X \in \mathbb{R}^{n \times k}$

Normalize rows of X; Cluster rows of X; Return clusters, {C_1, …, C_k}

Cluster assignment of data point *i* is cluster assignment of row X_i.

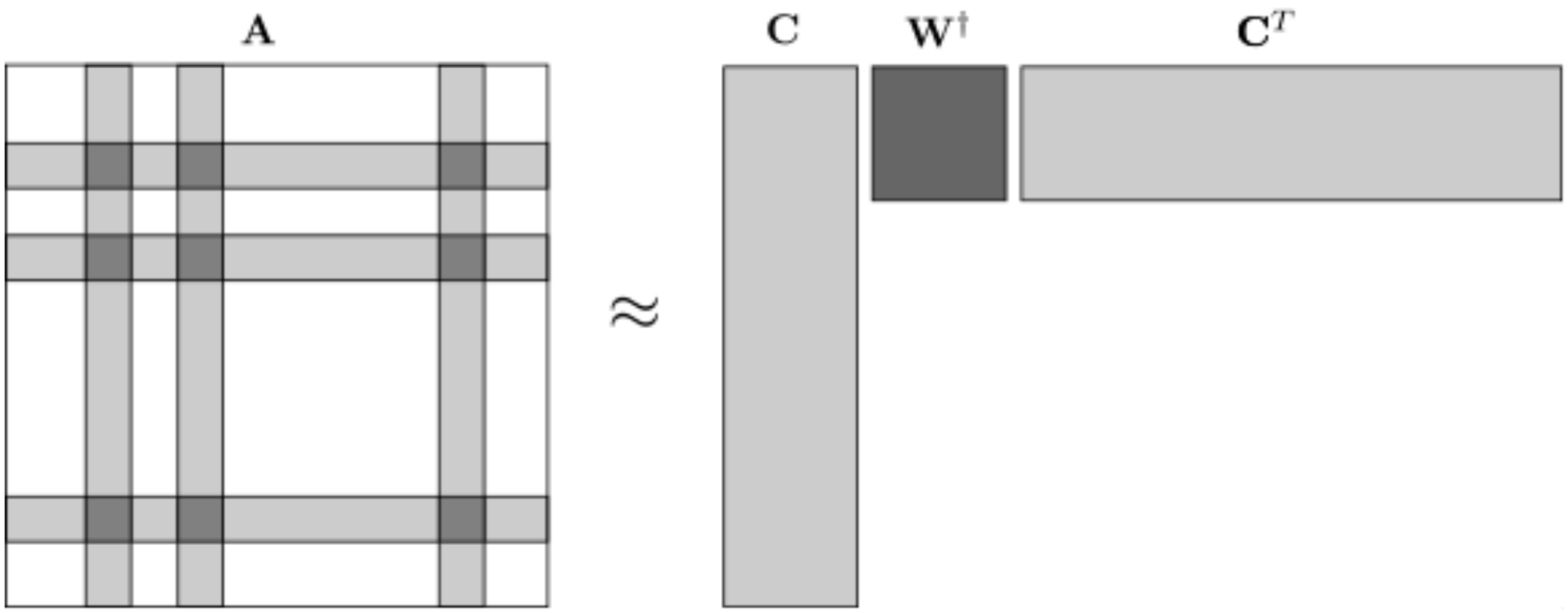
Scaling up spectral clustering to big data

- Spectral clustering exhibits performance advantages in a variety of applications
  - Graph data
  - You can represent any data with affinity matrix, given a similarity function (e.g. a Kernel).

- However computing with the affinity matrix becomes computationally prohibitive as *n* grows

- How can we create light-weight approximations to the affinity matrix?

Approximating the affinity matrix



Credit: Gittens ‘13.

SPSD Sketching Model. Let $\mathbf{A}$ be an $n \times n$ positive semi-definite matrix, and let $\mathbf{S}$ be a matrix of size $n \times \ell$, where $\ell \ll n$. Take $\mathbf{C} = \mathbf{AS}$ and $\mathbf{W} = \mathbf{S}^T\mathbf{AS}$. Then $\mathbf{CW}^\dagger\mathbf{C}^T$ is a low-rank approximation to $\mathbf{A}$ with rank at most ℓ.

Mahoney ‘13]

Clustering in metric spaces

Definition 1. A metric space $(\mathcal{X}, \rho)$ consists of a set $\mathcal{X}$ and a distance function $\rho : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that satisfies the three properties of a metric:

1. Reflexivity: $\rho(x, y) \geq 0$ with equality iff $x = y$
2. Symmetry: $\rho(x, y) = \rho(y, x)$
3. Triangle inequality: $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$

k-center clustering objective

k-CENTER CLUSTERING

Input: Finite set $S \subset \mathcal{X}$; integer k.

Output: $T \subset \mathcal{X}$ with $|T| = k$.

Goal: Minimize $\text{cost}(T) = \max_{x \in S} \rho(x, T)$.

Algorithm: Farthest-first retrieval

pick any $z \in S$ and set $T = \{z\}$
while $|T| < k$:
    $z = \arg\max_{x \in S} \rho(x, T)$
    $T = T \cup \{z\}$

k-means clustering objective

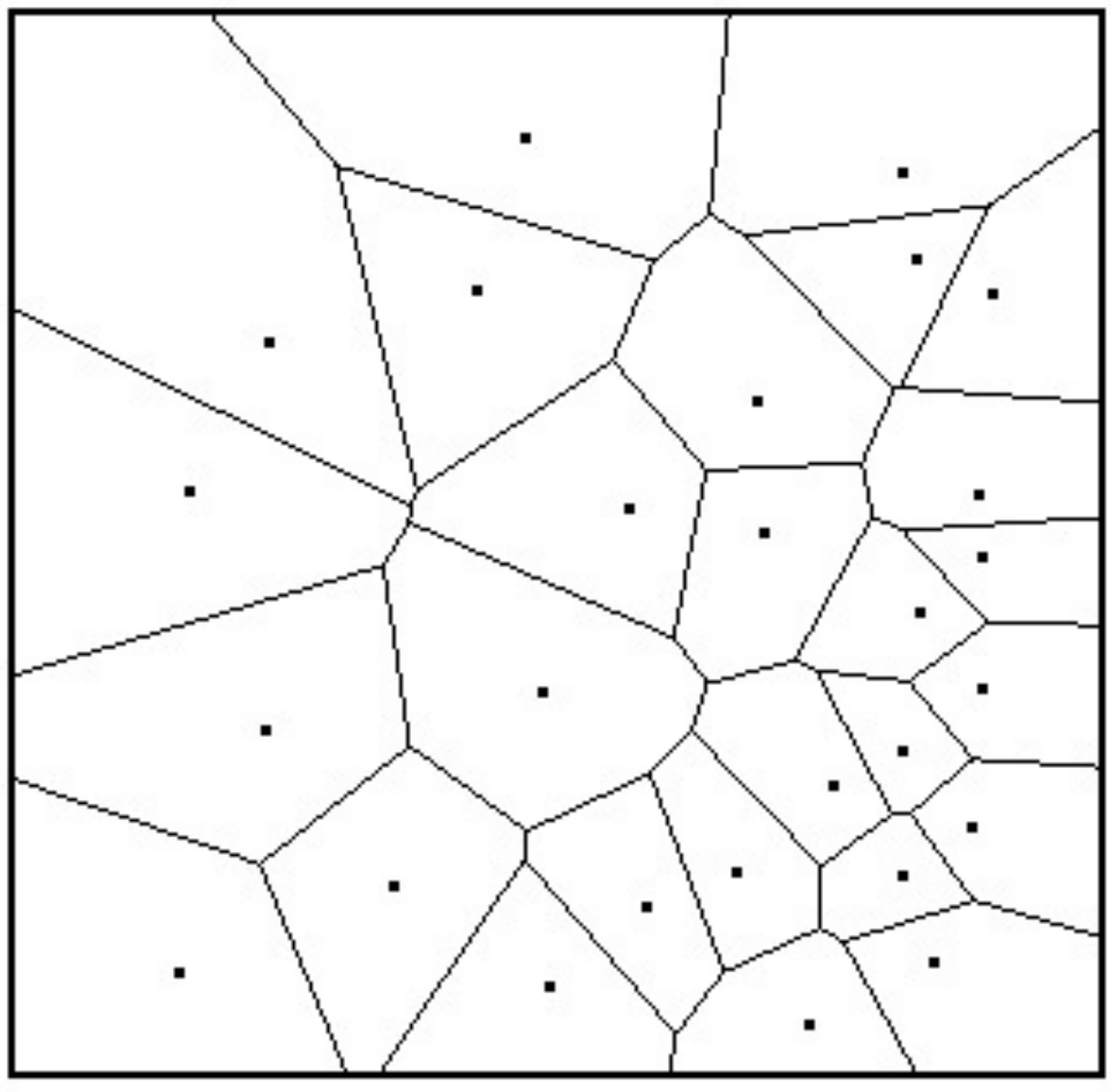
Clustering algorithms can be hard to evaluate without prior information or assumptions on the data.

With *no* assumptions on the data, one evaluation technique is w.r.t. some objective function.

A widely-cited and studied objective is the k-means clustering objective: Given set, $X \subset R^d$, choose $C \subset R^d$, $|C|$ = k, to minimize:

$$\phi_C = \sum_{x \in X} \min_{c \in C} \|x - c\|^2$$

Voronoi regions



k-means clustering objective

A widely-cited and studied objective is the k-means clustering objective: Given set, $X \subset R^d$, choose $C \subset R^d$, $|C| = k$, to minimize:

$$\phi_C = \sum_{x \in X} \min_{c \in C} \|x - c\|^2$$

Unfortunately this in an NP-hard optimization problem, even when d=2!

Even the algorithm called “the k-means clustering algorithm” (a.k.a. Lloyd’s algorithm) does not have an approximation guarantee!

*without assumptions on the data

Algorithm: k-means++:

Choose first center c_1 uniformly at random from X,
and let $C = \{c_1\}$.

Repeat (k-1) times:

Choose next center $c_i = x' \in X$ with prob.

$$\frac{D(x', C)^2}{\sum_{x \in \mathcal{X}} D(x, C)^2}$$

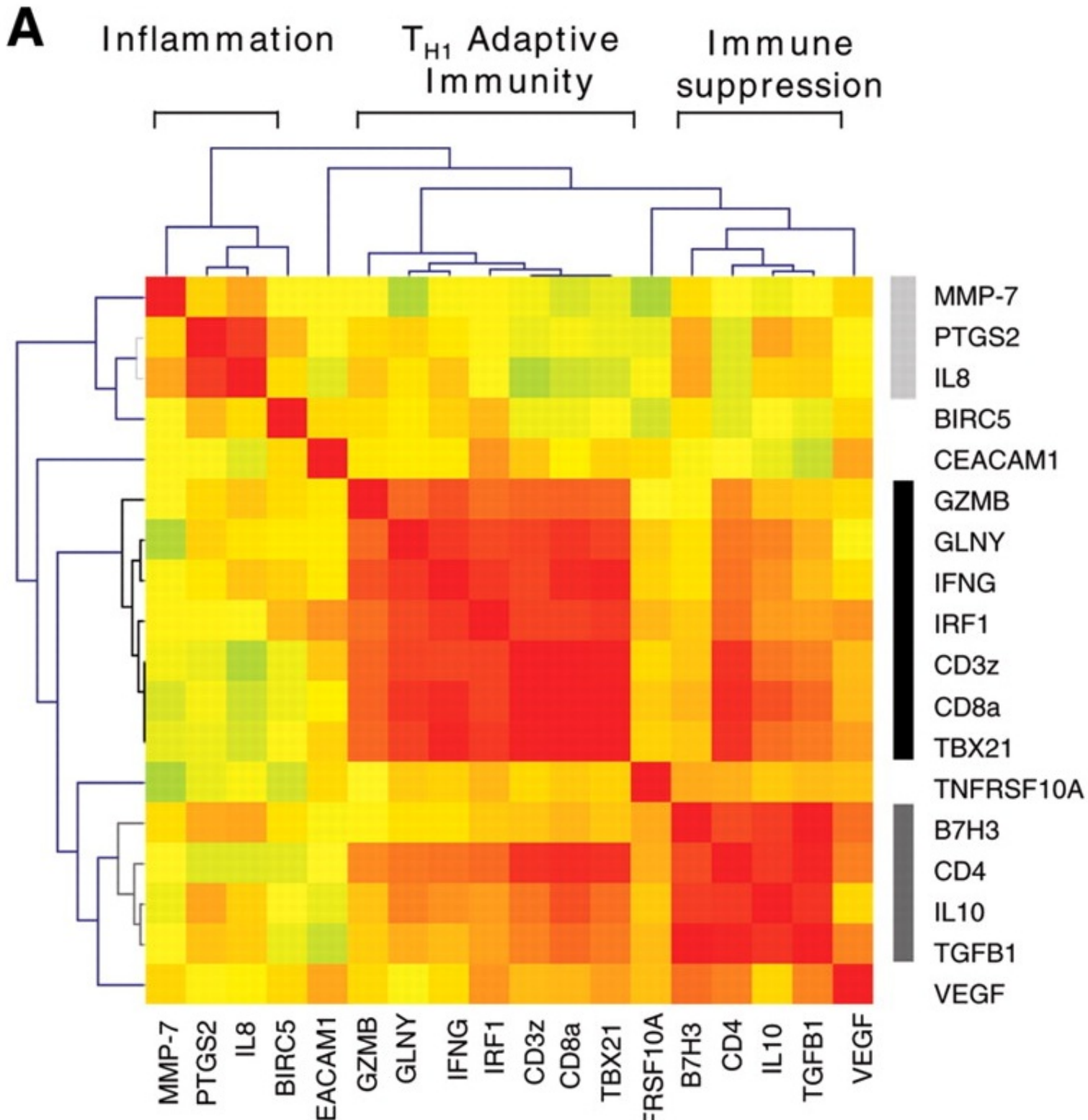
$C \leftarrow C \cup \{c_i\}$

where:

$$D(x, C) = \min_{c \in C} \|x - c\|$$

Theorem (Arthur & Vassilvitskii '07):
Returns an O(log *k*)- approximation to the *k*-means objective,
in expectation.

Hierarchical clustering

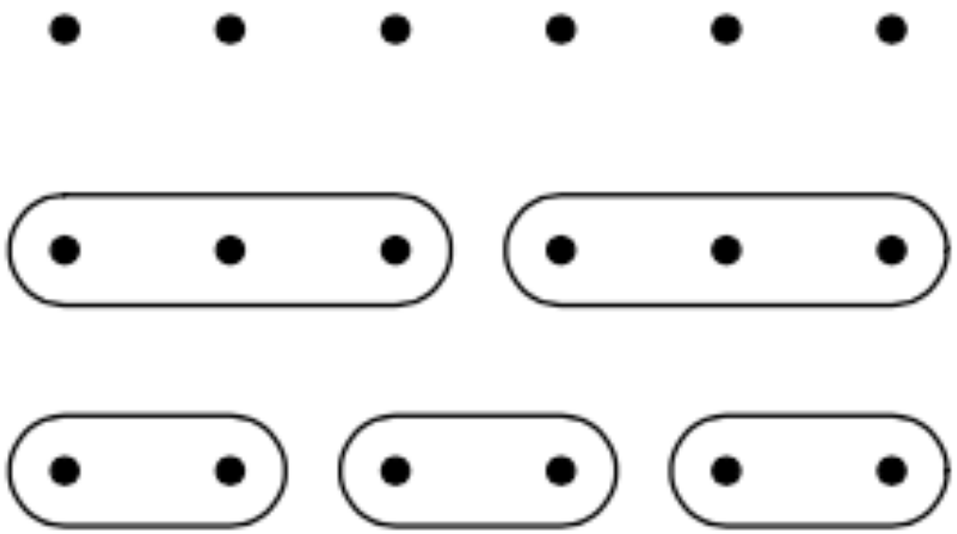


igure credit:

Hierarchical clustering

It is not always possible to construct a hierarchical clustering whose induced k-clustering is optimal for all k.

E.g.



So instead, the goal is to provide an algorithms that approximate the optimum, for all k.

Figure credit:

Hierarchical Agglomerative Clustering (HAC)

Input: a set of *n* points in some metric space $(\mathcal{X}, \rho)$

Algorithm:

Initialize *n* clusters, each a single data point

While the total number of clusters is > 1:

    Merge the two “closest” clusters into one cluster

Deep Unsupervised Learning

- How to address clustering using Deep Unsupervised Learning?

- Use a clustering objective as your loss function!
  - This is a function of your training data and network output

Semi-supervised learning

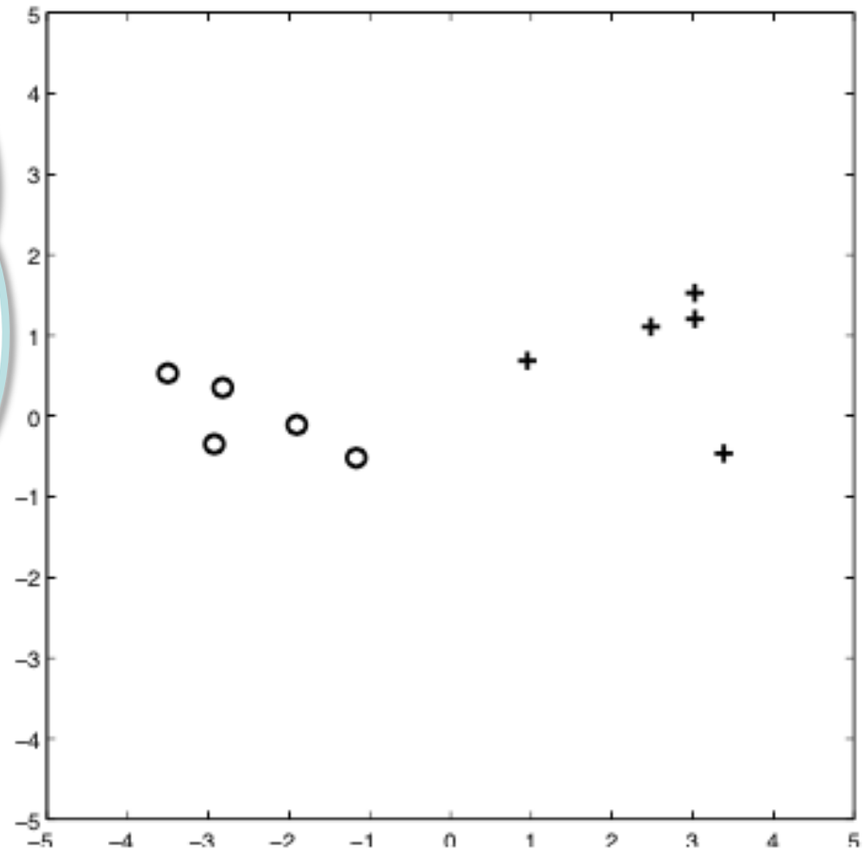
- Semi-supervised learning is an umbrella term for problems in which some of the data is labeled and some of the data is unlabeled.

- Typically this is determined in advance, and algorithms do not get to interact with the label-generating process.

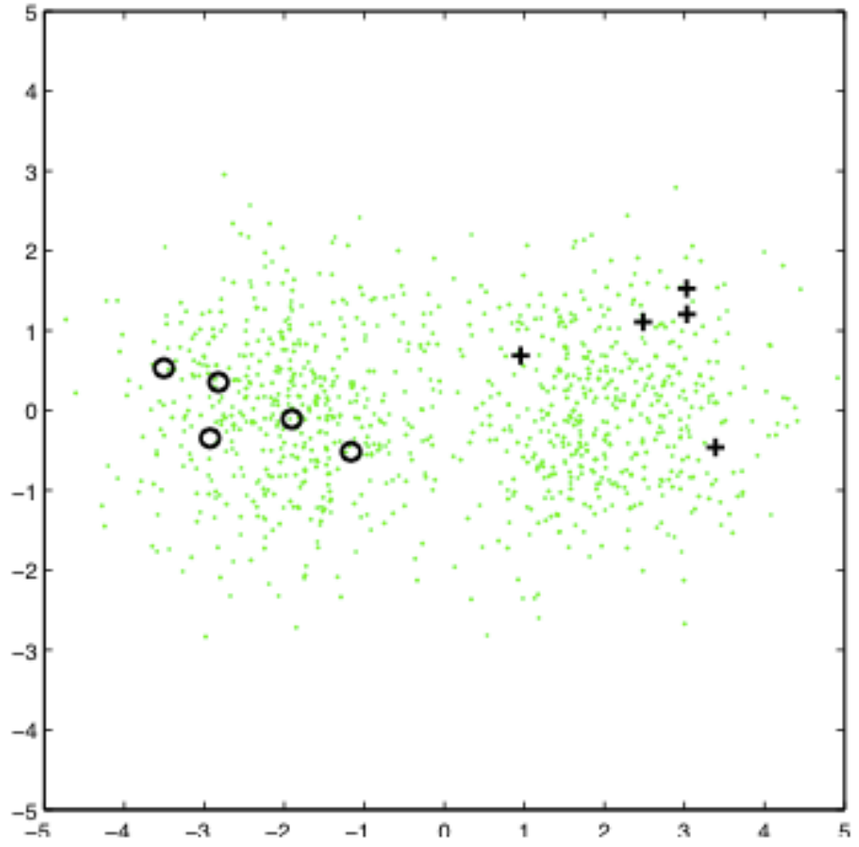
- Semi-supervised learning can be studied in the standard setting, in which we need to learn a classifier/regressor to generalize to future unseen data.

- OR, the goal could be transduction: only need to output labels on the unlabeled points in the input set.

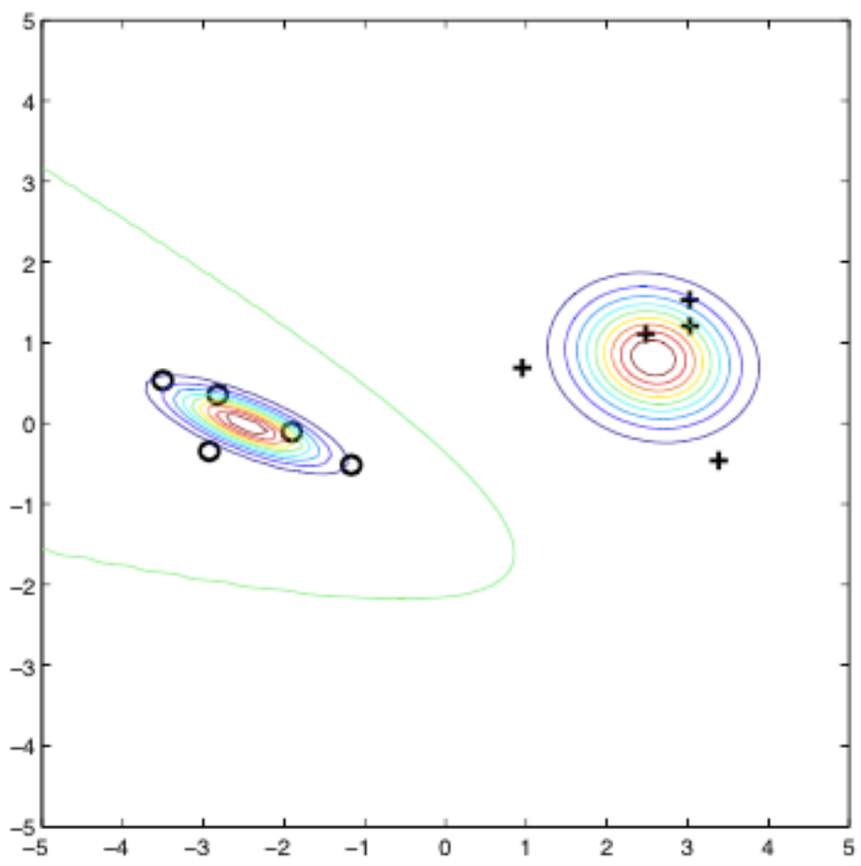
How can unlabeled data help?



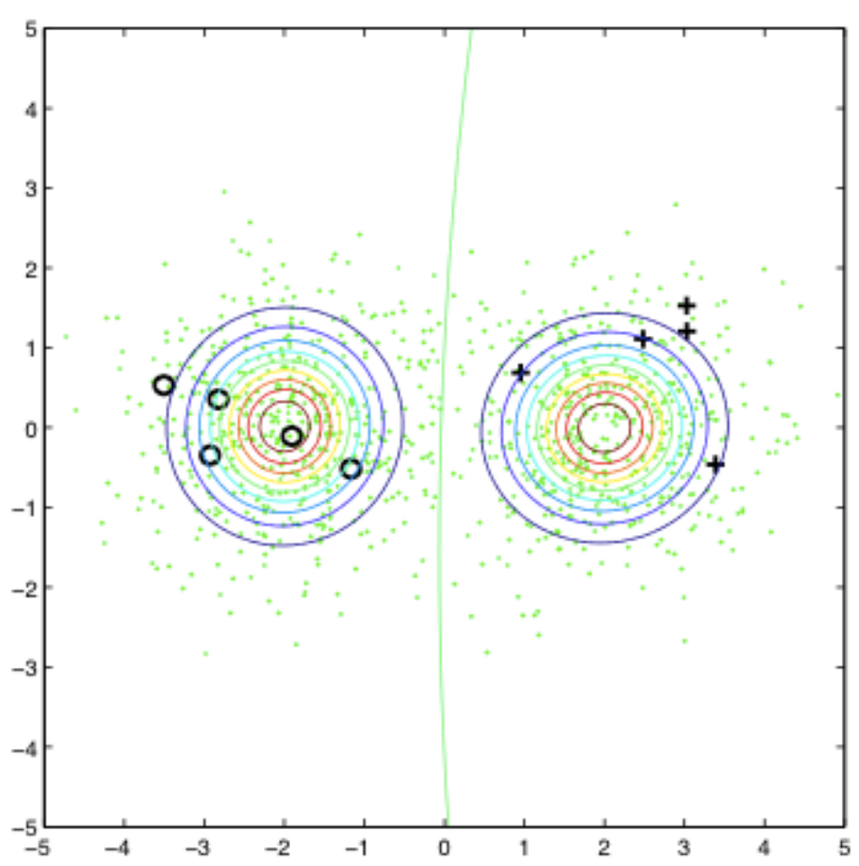
(a) labeled data



(b) labeled and unlabeled data (small dots)



(c) model learned from labeled data



(d) model learned from labeled and unlabeled data

Credit: X. Zhu '08