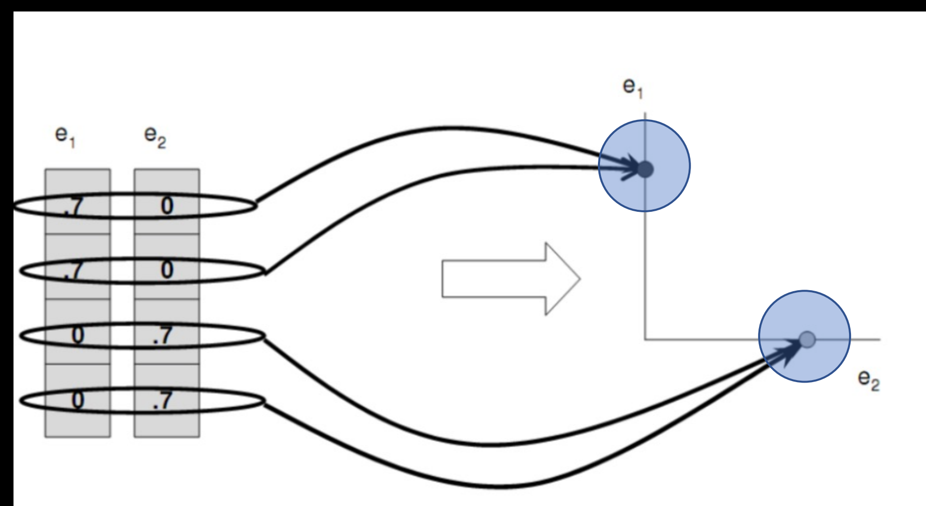
Embeddings I: Metric Learning

CSCI 4360/6360 Data Science II

Previously…

- Spectral clustering

1. Define graph (affinities -> Laplacian)
2. Compute eigenvectors (*embedding*)
3. Cluster embeddings trivially (K-means)

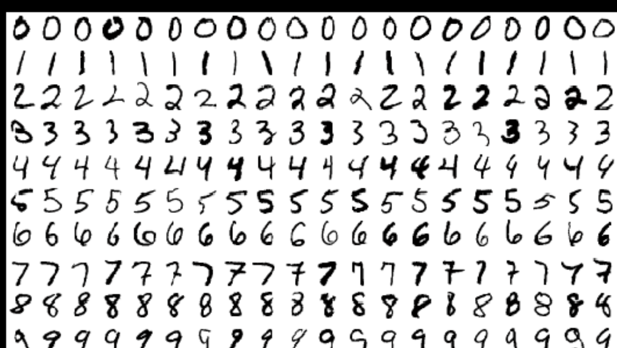


Embeddings

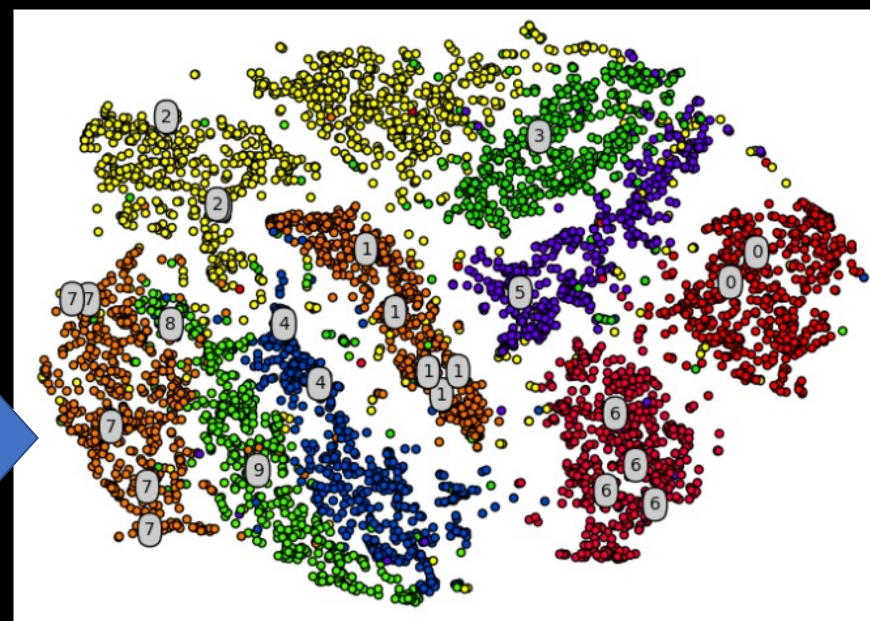
- What is an *embedding*?

$$f : X \to Y$$

- **Mapping**
- **Transformation**
- **Reveals / preserves "structure"**



Embedding



Embeddings

- From Vicki Boykis' "What Are Embeddings" (2023)
- Transformation
- Compression
- Representation

- *Transforms* multimodal input into representations that are easier to perform intensive computation on, in the form of **vectors**, tensors, or graphs [51]. For the purpose of machine learning, we can think of vectors as a list (or array) of numbers.
- *Compresses* input information for use in a machine learning **task** — the type of methods available to us in machine learning to solve specific problems — such as summarizing a document or identifying tags or labels for social media posts or performing **semantic search** on a large text corpus. The process of compression changes variable feature dimensions into fixed inputs, allowing them to be passed efficiently into downstream components of machine learning systems.
- *Creates an embedding space* that is specific to the data the embeddings were trained on but that, in the case of deep learning representations, can also generalize to other tasks and domains through **transfer learning** — the ability to switch contexts — which is one of the reasons embeddings have exploded in popularity across machine learning applications

Embedding Strategies (non-DL)

- Principal Components Analysis (PCA)
  - **Sparse & Kernel PCA (next Tuesday!)**
- Independent Components Analysis (ICA)
- Non-negative Matrix Factorization (NMF)
- Locally-linear Embeddings (LLE)
- **Dictionary Learning (next!)**

Linear

Sparse &

Non-Gaussian

Non-negative

Nonlinear

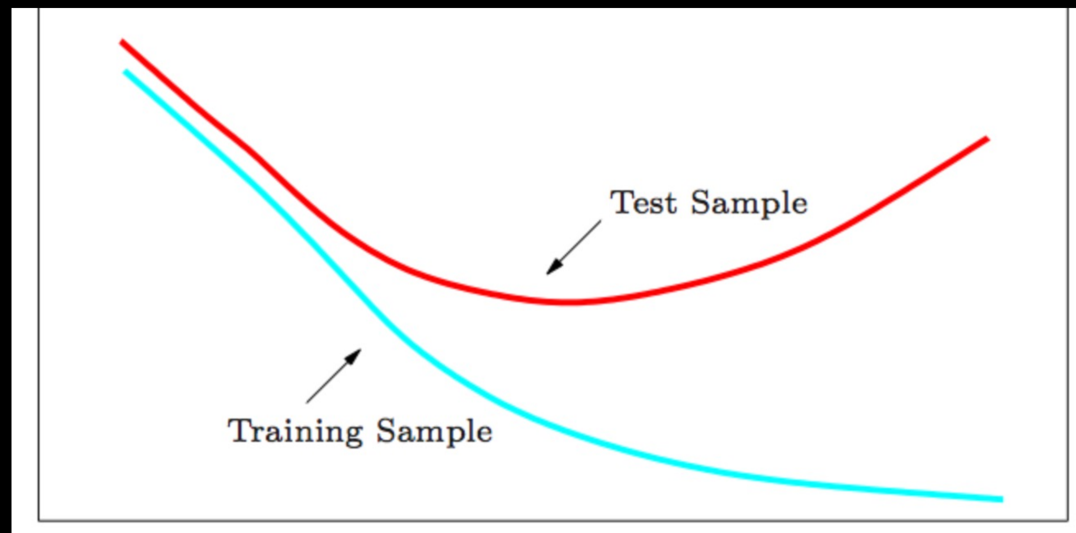
Sparse & Nonlinear

Million Dollar Question:

How do you know the embedding is **right?**

Embeddings

- If you're performing **classification**, it's pretty easy to know if your embedding is "right"
- Error decreases?
- Error increases?
- What about **unsupervised learning?**



Assumptions

- Choice of embedding -> Assumptions about the data
- **What if we knew something about the data?**
- **“Side information”:** we don’t know what classes/clusters the data belong to, but we do have some notion of similarity

Side Information

- Define a set *S*
  - for every pair x_i and x_j that are similar, we put this pair in *S*
- ”Similar” is user-defined; can mean anything
- Likewise have a set *D*
  - for every pair x_i and x_j that are **dis**similar, we put this pair in *D*
  - can consist of every pair not in *S*, or specific pairs if information is available

- We have this similarity information; what can we do with it?

Distance Metrics

- Goal: use side-information to **learn a new distance metric**
- Encode our side-information in a "metric" *A*
- Generalization of Euclidean distance
  - Note when *A* = *I*, this is regular Euclidean distance
  - When *A* is diagonal, this is a "weighted" Euclidean distance
  - When data are put through nonlinear basis functions ϕ, nonlinear metrics can be learned

$$d(\vec{x}, \vec{y}) = d_A(\vec{x}, \vec{y})$$

$$= ||\vec{x} - \vec{y}||_A$$

$$= \sqrt{(\vec{x} - \vec{y})^T A (\vec{x} - \vec{y})}$$

$$= \sqrt{(\phi(\vec{x}) - \phi(\vec{y}))^T A (\phi(\vec{x}) - \phi(\vec{y}))}$$

Distance Metrics

- Quick Review: **What constitutes a *valid distance metric?***

1: Non-negativity $d(\vec{x}, \vec{y}) \geq 0$

2: Symmetry $d(\vec{x}, \vec{y}) = d(\vec{y}, \vec{x})$

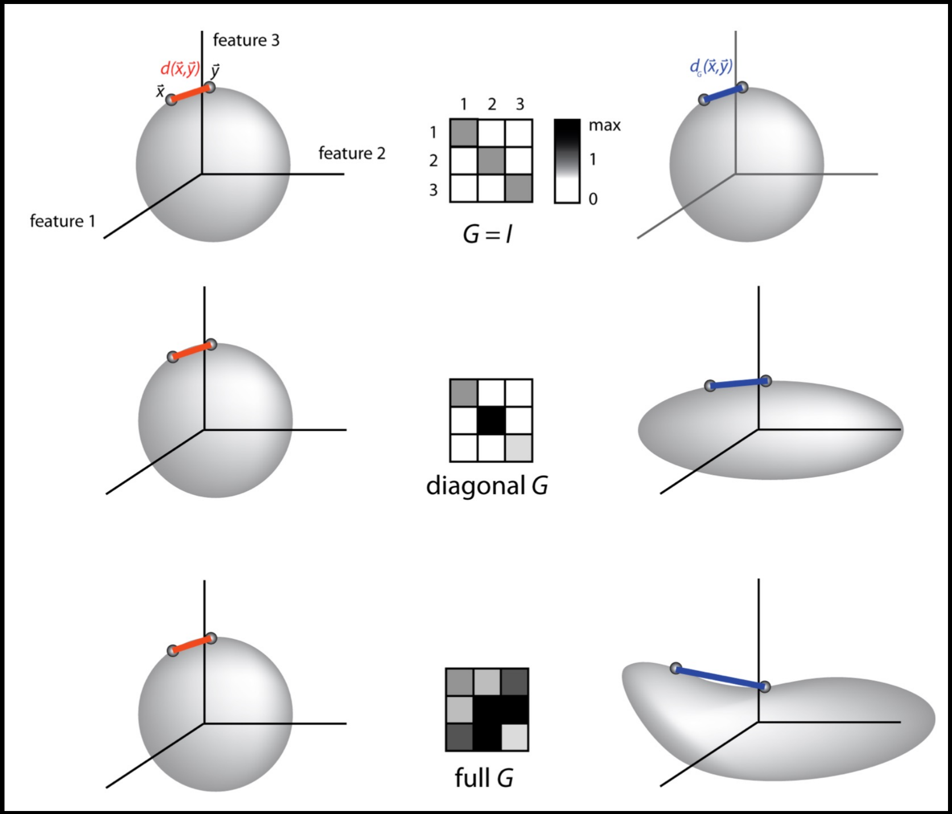
3: Triangle Inequality $d(\vec{x}, \vec{z}) \leq d(\vec{x}, \vec{y}) + d(\vec{y}, \vec{z})$

~~4: Identity of indiscernibles $d(\vec{x}, \vec{y}) = 0 \Leftrightarrow x = y$~~

"Pseudometric"

Form of a Metric

- Learning metric A (G in figure) also equivalent to replacing each point x with $A^{1/2}x$ and using standard Euclidean distance
- **It's an embedding!**
- Learning a **space** inhabited by your data
  - Bonus: easy to incorporate new data! (unlike LLE or others)



Learning a Metric (1)

- Goal: Define a metric *A* that respects constraint sets *S* and *D*
- Simple enough: constrain all pairs in *S* to have small distances
- Is that all?
- **Nope – trivially solved with *A* = *0***

$$\min_A \sum_{\vec{x},\vec{y} \in S} ||\vec{x} - \vec{y}||_A^2$$

Learning a Metric (2)

- Additional constraint: use pairs in *D* to guarantee non-zero distances
- (choice of 1 is arbitrary; any other constant *c* would have the effect of replacing *A* with $c^2 A$)
- Is that all?

- **Nope – need to ensure *A* is positive semi-definite (why?)**

$$\sum_{\vec{x},\vec{y} \in D} ||\vec{x} - \vec{y}||_A \geq 1$$

Aside!

- We used squared Euclidean distance in the first constraint

$$\min_A \sum_{\vec{x},\vec{y} \in S} ||\vec{x} - \vec{y}||_A^2$$

- But not in the second! Why?

$$\sum_{\vec{x},\vec{y} \in D} ||\vec{x} - \vec{y}||_A \geq 1$$

- Squared distance in 2^{nd} constraint would always result in **rank-1 *A***, i.e. the data would always be projected on a line
- (proof left as an exercise!)

Learning a Metric (3)

- A third constraint: keep *A* positive semi-definite
  - (this means the diagonal is always ≥ 0)
- If *A* is PSD, its eigenvectors and eigenvalues exist and are real
- Set any negative eigenvalues to 0
- Compute *A'*

$$A \succeq 0$$

$$A = X\Lambda X^T$$

$$\Lambda' = \mathrm{diag}(\max\{0, \lambda_1\}, ..., \{0, \lambda_n\})$$
$$A' = X\Lambda' X^T$$

Learning a Metric

- We have our constraints!
- How do we learn *A*?
- *(Hint)* Linear in parameters of *A*
- *(HINT)* First two constraints are verifiably convex

$$\min_{A} \sum_{\vec{x},\vec{y}\in S} ||\vec{x}-\vec{y}||_A^2$$

$$\sum_{\vec{x},\vec{y}\in D} ||\vec{x}-\vec{y}||_A \geq 1$$

$$A \succeq 0$$

Convex Optimization

- For diagonal *A*, this is easy

$$g(A) = g(A_{11}, \dots, A_{nn}) = \sum_{(x_i,x_j)\in\mathcal{S}} ||x_i - x_j||_A^2 - \log\left(\sum_{(x_i,x_j)\in\mathcal{D}} ||x_i - x_j||_A\right)$$

- (just a fancy reformulation of the original constraints)
- Minimizing *g* is equivalent to solving original problem, up to multiplication of *A* by a positive constant

- **Gradient descent!** (step-size intrinsically enforces PSD of *A*)

Convex Optimization

- Trickier for full *A*
- Gradient ascent + iterative projections
- For this to work, constraints needed to be reversed

Iterate

  Iterate

    $A := \arg\min_{A'} \{||A' - A||_F : A' \in C_1\}$

    $A := \arg\min_{A'} \{||A' - A||_F : A' \in C_2\}$

  until A converges

  $A := A + \alpha(\nabla_A g(A))_{\perp \nabla_A f}$

until convergence

Constraint Reformulation

Previous

$$\min_A \sum_{\vec{x},\vec{y} \in S} ||\vec{x} - \vec{y}||_A^2$$

$$\sum_{\vec{x},\vec{y} \in D} ||\vec{x} - \vec{y}||_A \geq 1$$

$$A \succeq 0$$

Current

$$\sum_{\vec{x},\vec{y} \in S} ||\vec{x} - \vec{y}||_A^2 \leq 1$$

C_1

$$\max_A \sum_{\vec{x},\vec{y} \in D} ||\vec{x} - \vec{y}||_A$$

$g(A)$

$$A \succeq 0$$

C_2

GA + IP

Iterate

  Iterate

    $A := \arg\min_{A'}\{||A' - A||_F : A' \in C_1\}$

    $A := \arg\min_{A'}\{||A' - A||_F : A' \in C_2\}$

  until A converges

  $A := A + \alpha(\nabla_A g(A))_{\perp \nabla_A f}$

until convergence

Iterate

  Iterate

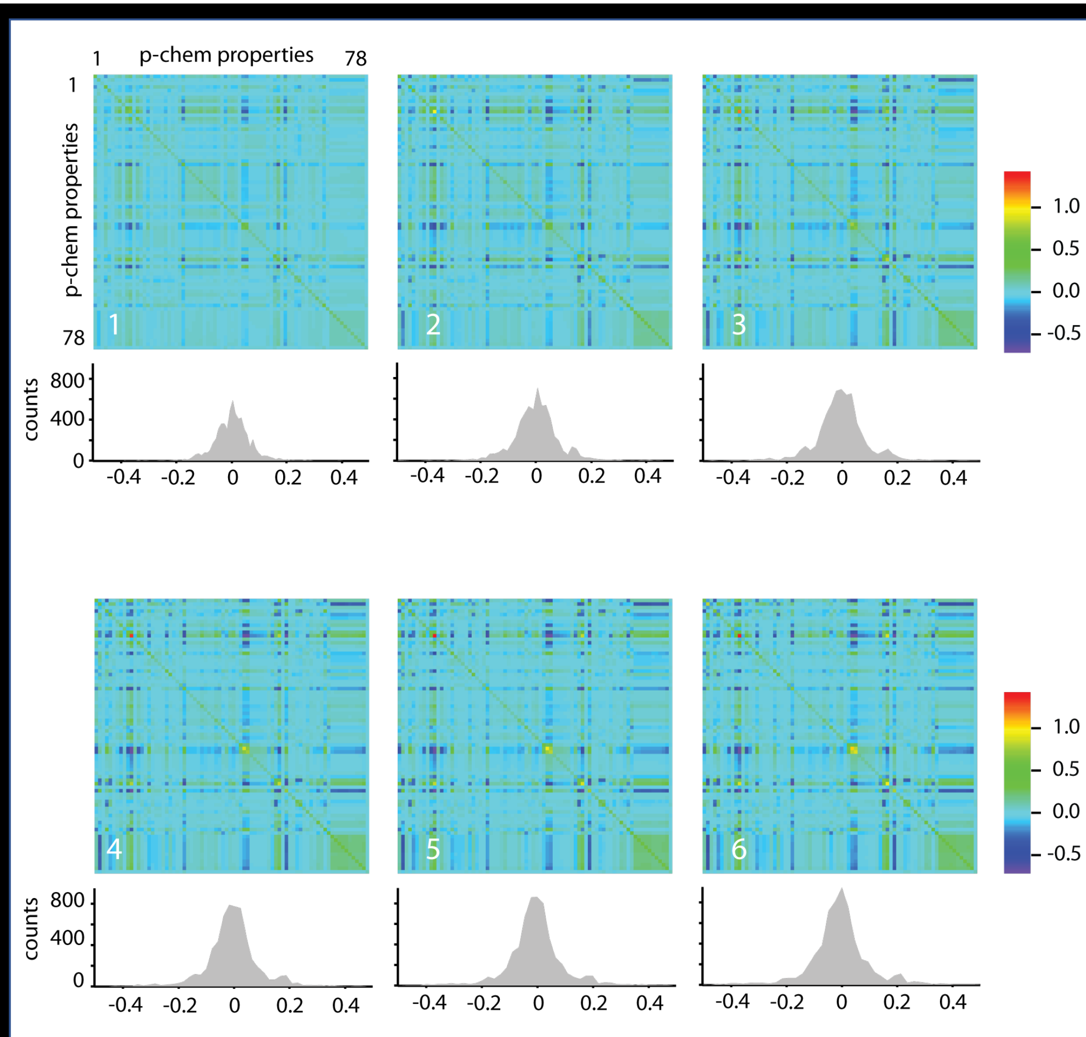
$$\sum_{\vec{x},\vec{y} \in S} ||\vec{x} - \vec{y}||_A^2 \leq 1 \quad + \quad A \succeq 0$$

  until A converges

$$A := A + \alpha \nabla \max_A \sum_{\vec{x},\vec{y} \in D} ||\vec{x} - \vec{y}||_A$$

until convergence

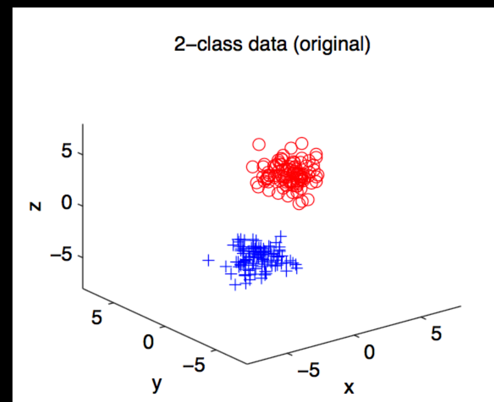
GA + IP

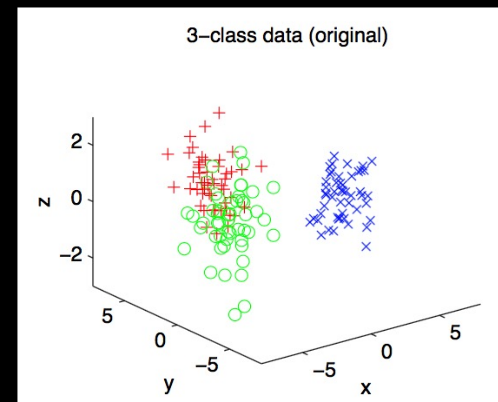


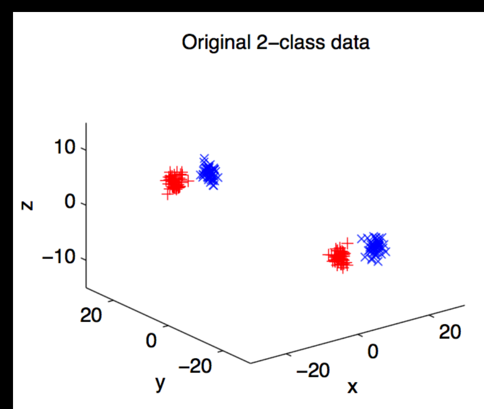
p-chem properties
1
78
counts
800
400
0
-0.4
-0.2
0
0.2
0.4
1
2
3
4
5
6
1.0
0.5
0.0
-0.5

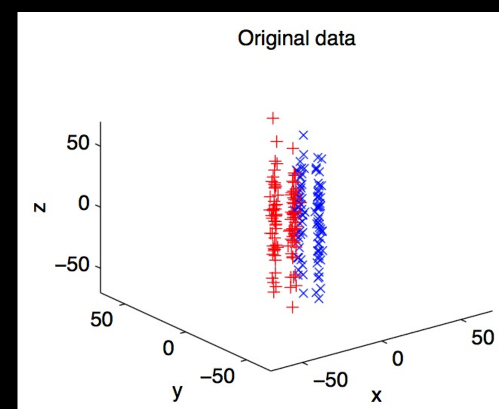
Experiments

- Generated artificial 3D data
  - 2 class
  - 3 class
  - Separated by y-axis
  - Separated by z-axis

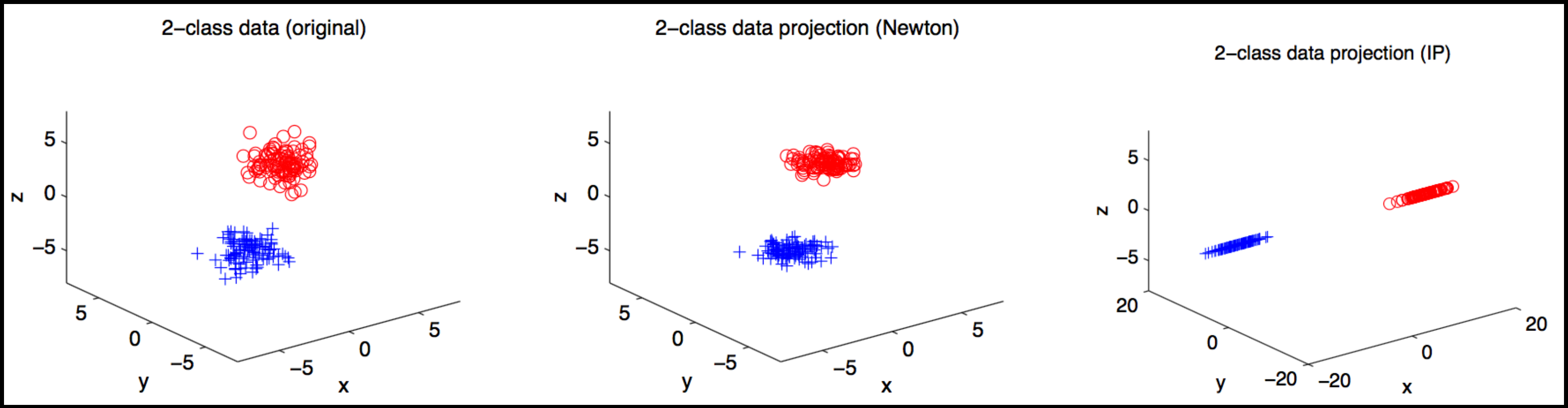




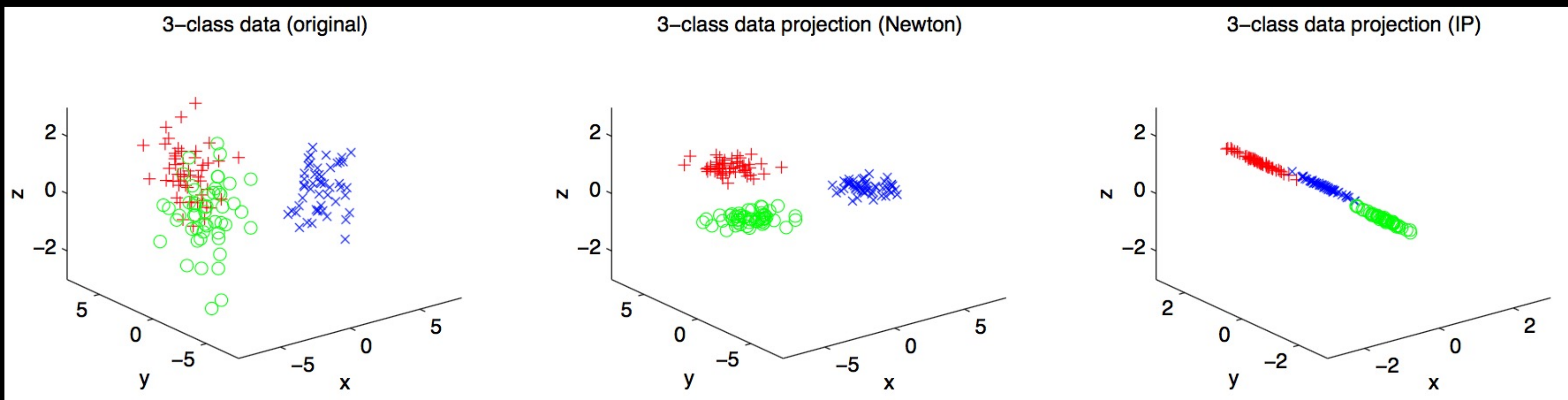




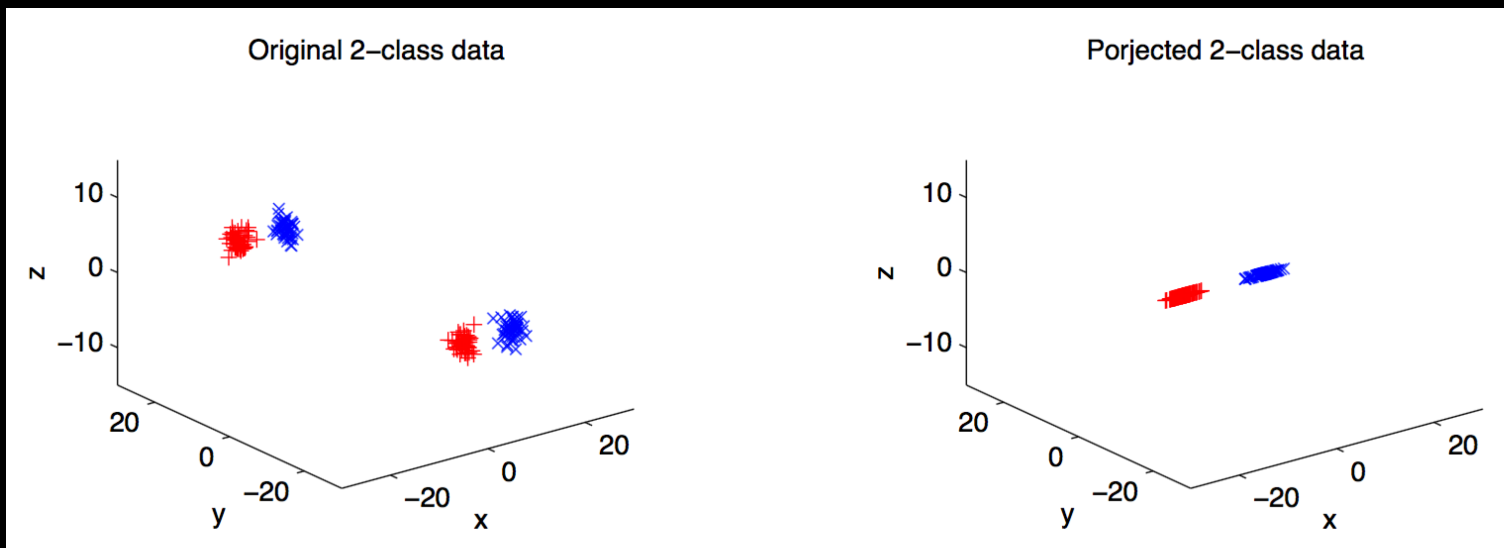
Experiments



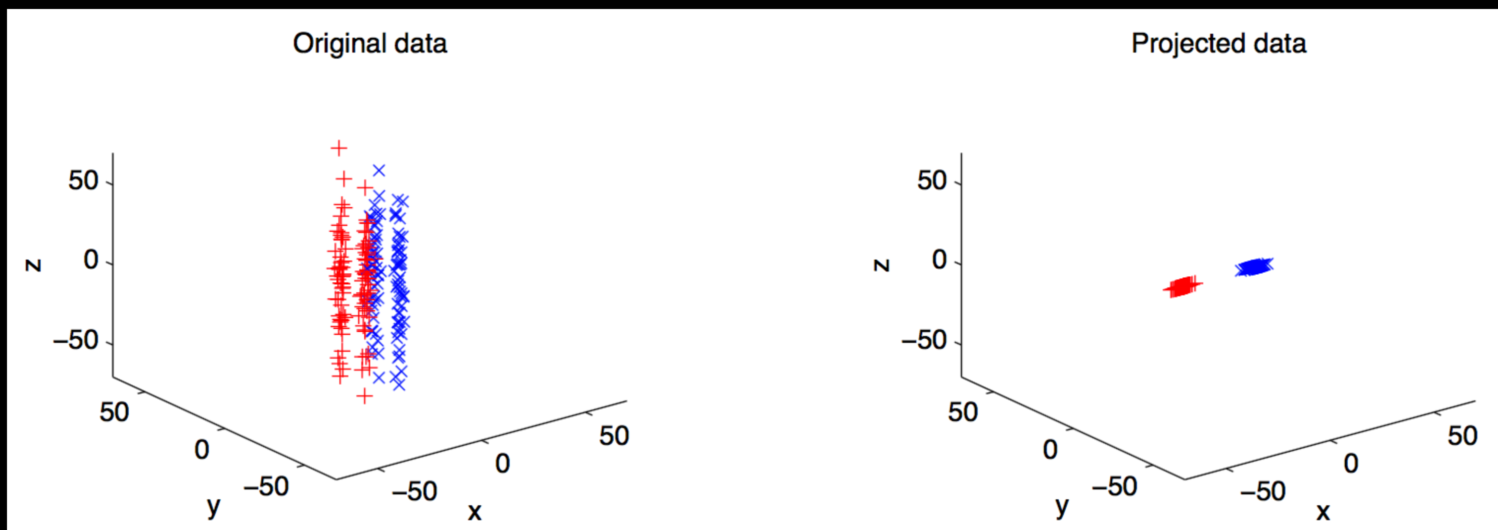
Experiments



Experiments

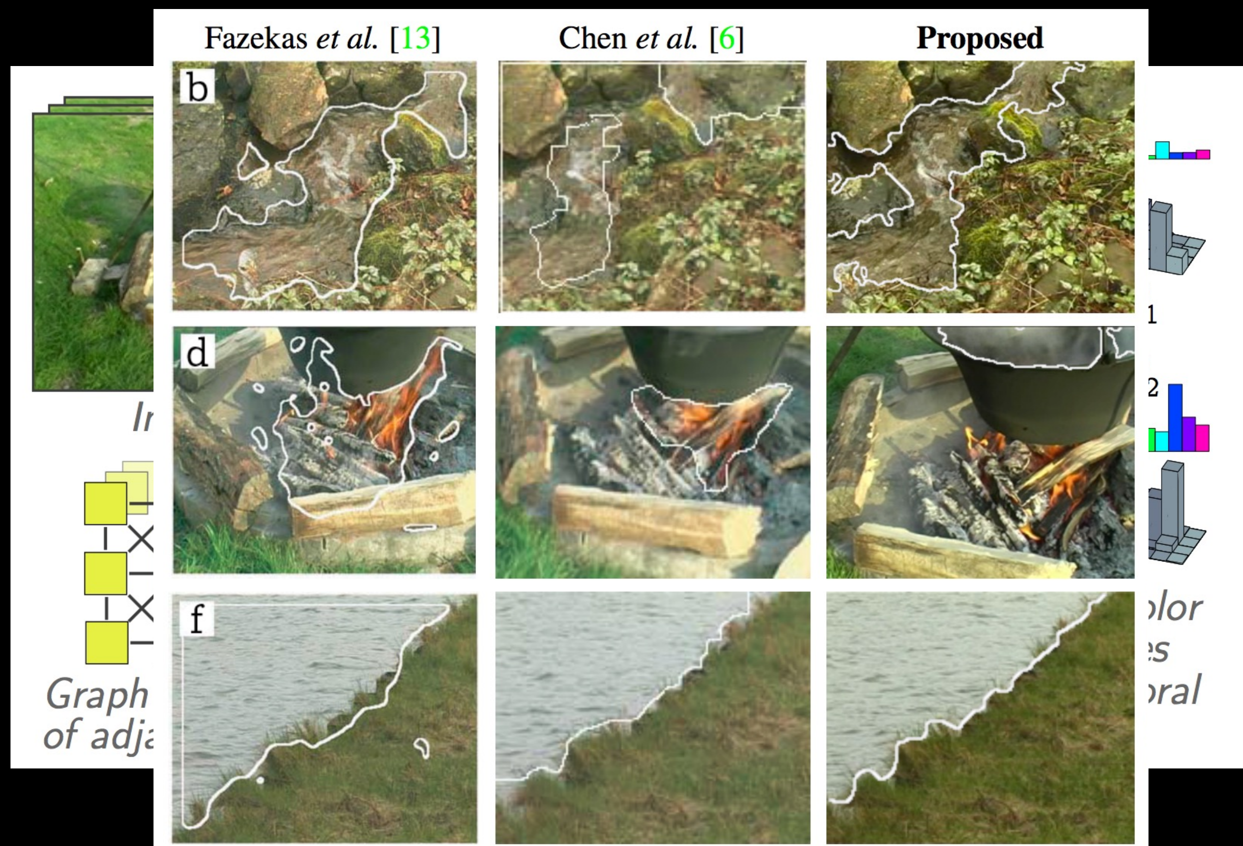


Experiments



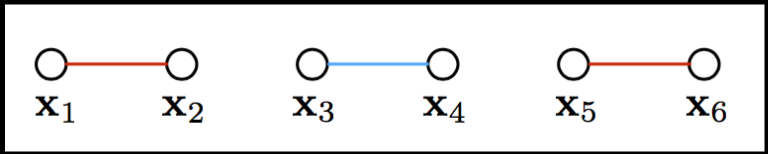
Other Applications

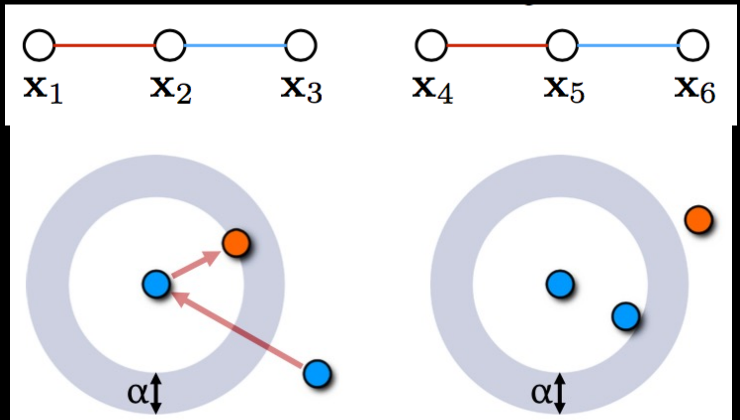
- Video scene segmentation
- Identifying dynamic textures in videos



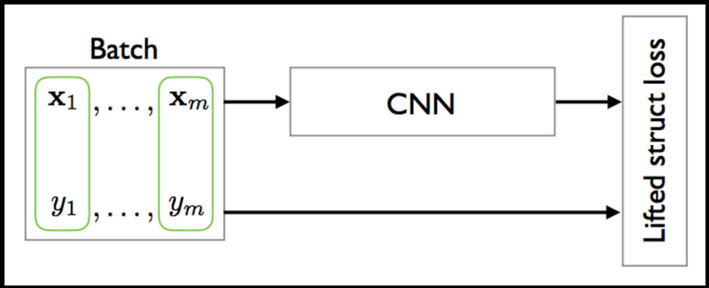
Other Formulations

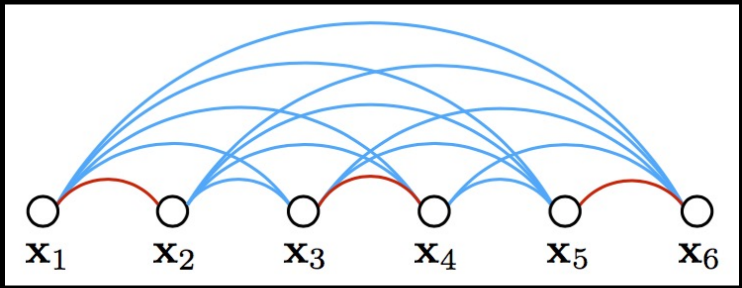
- Deep metric learning
- Differentiates different metric constraints
  - Contrastive
  - Triplet
  - Lifted structure





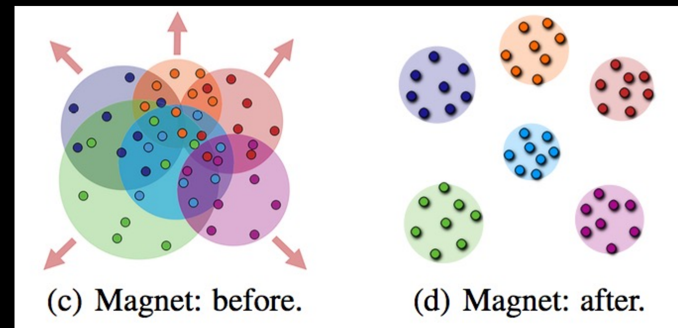
(a) Triplet: before. (b) Triplet: after.

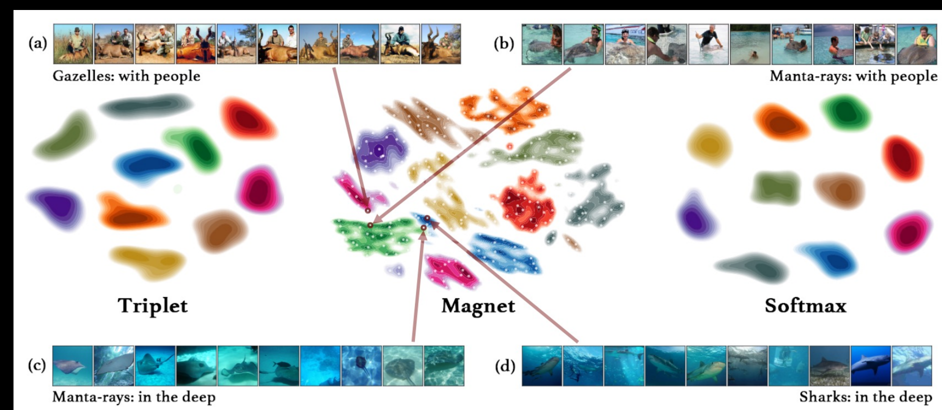


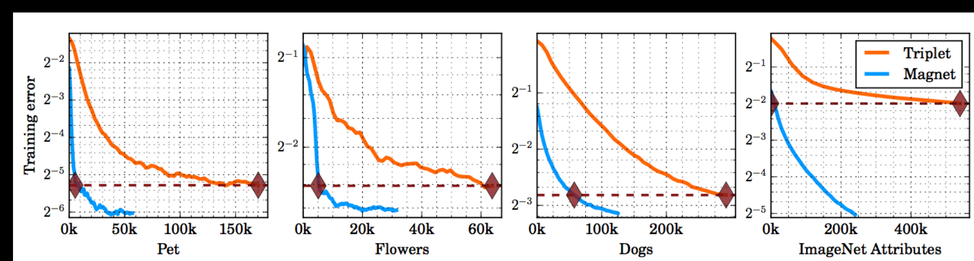


Other Formulations

- Adaptive densities
- Introduces "magnet loss" *(how does it work?)*
  - Optimizes over entire neighborhoods simultaneously
  - Reduces distribution overlap, rather than just pairs or triplets
- Requires ground-truth labels







Other Formulations

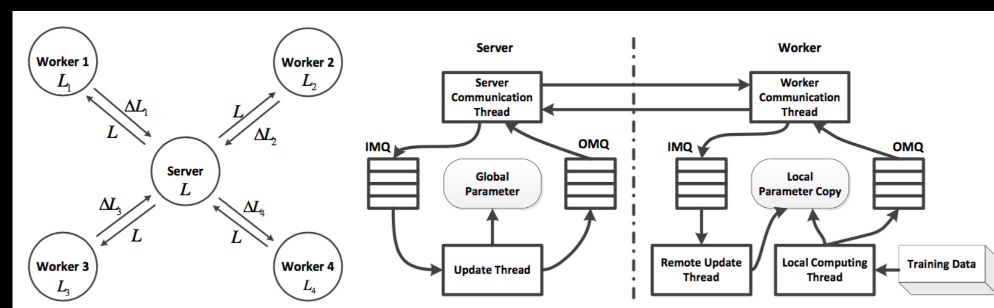
$$\begin{aligned} \min_L \quad & \sum_{(x,y)\in\mathcal{S}} \|L(x-y)\|^2 \\ s.t. \quad & \|L(x-y)\|^2 \geq 1, \forall (x,y) \in \mathcal{D} \end{aligned}$$

- Large-scale metric learning
  - If feature space is extremely large, iterative eigen-decompositions are a deal-breaker
  - Nested convex optimization is a deal-breaker

$$\begin{aligned} \min_L \quad & \sum_{(x,y)\in\mathcal{S}} \|L(x-y)\|^2 + \lambda \sum_{(x,y)\in\mathcal{D}} \xi_{x,y} \\ s.t. \quad & \|L(x-y)\|^2 \geq 1 - \xi_{x,y}, \xi_{x,y} \geq 0, \forall (x,y) \in \mathcal{D} \end{aligned}$$

$$\min_L \sum_{(x,y)\in\mathcal{S}} \|L(x-y)\|^2 + \lambda \sum_{(x,y)\in\mathcal{D}} \max(0, 1 - \|L(x-y)\|^2)$$

- Represent metric *A* = *L*^{T}*L*
  - Learn *L* directly, instead of *A*
- Use hinge loss to induce unconstrained optimization
- Parameter server for SGD-based metric updates



Questions?

Midterm Exam

- Some true/false
- Some short answer
- Some multi-part problems (like the homeworks)
- Some “coding” (very short)

- **Thursday, Oct 12** (normal lecture time, normal lecture location)

Final Projects

- Teams! (2-4 people)
- ~1.5 homeworks in scope/size
- Some kind of data science + machine learning problem
- Deliverables (proposal, update #1, update #2, final presentations)

- **Proposals: due Tuesday, Oct 17 by 11:59pm**
  - 1-page (maximum) with i) what problem you're working on, ii) how you intend to solve it, iii) what your validation plan is (how do you know it worked or didn't work?).
  - Only 1 submission needed per team, should include names of your teammates

IOB Symposium

AI and Deep Learning Applications in Bioinformatics

- https://iob.uga.edu/symposium2023/ (posted in #lounge)
- Oct 16-17

- **No class on Monday, Oct 16 or Tuesday, Oct 17**
- **Class as usual on Thursday, Oct 19**
  - A workshop!

References

- Xing, Eric P., Michael I. Jordan, Stuart J. Russell, and Andrew Y. Ng. "Distance metric learning with application to clustering with side-information" http://papers.nips.cc/paper/2164-distance-metric-learning-with-application-to-clustering-with-side-information.pdf
- Teney, Damien, Matthew Brown, Dmitry Kit, and Peter Hall. "Learning similarity metrics for dynamic scene segmentation" http://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Teney_Learning_Similarity_Metrics_2015_CVPR_paper.pdf
- Oh Song, Hyun, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. "Deep metric learning via lifted structured feature embedding" http://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Song_Deep_Metric_Learning_CVPR_2016_paper.pdf