DIMENSIONALITY REDUCTION AND VISUALIZATION

The curse of dimensionality

https://erikbern.com/2015/10/20/nearest-neighbors-and-vector-models-epilogue-curse-of-dimensionality.html

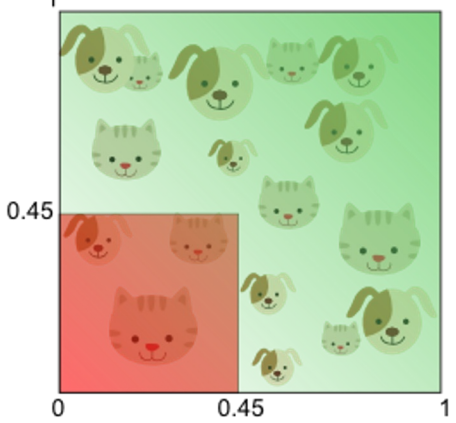
The Curse of Dimensionality

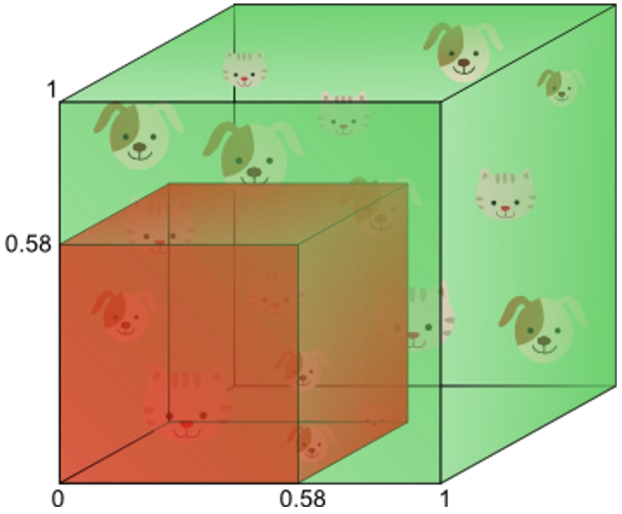
- Harder to visualize or see structure of
  - Verifying that data come from a straight line/plane needs n+1 data points
- Hard to search in high dimension – More runtime
- Need more data to get a get a good estimation of the distribution

Wind in the morning X ∈ {Calm, Windy}
PM2.5 level in the afternoon Y ∈ {Low, Med, High}
PM2.5 level in the evening Z ∈ {Low, Med, High}
argmax P(Z | Y, X) = argmax P(Y,X | Z) P(Z)

Day	X	Y	Z
1	W	L	M
2	C	M	M
3	W	H	M
4	W	M	H
5	C	M	L
6	W	M	L
7	C	L	H
8	W	H	L

count(Z,Y,X)	Z=L	Z=M	Z=H
X=W,Y=L	0	1	0
X=W,Y=M	1	0	1
X=W,Y=H	1	1	0
X=C,Y=L	0	0	1
X=C,Y=M	1	1	0
X=C,Y=H	0	0	0

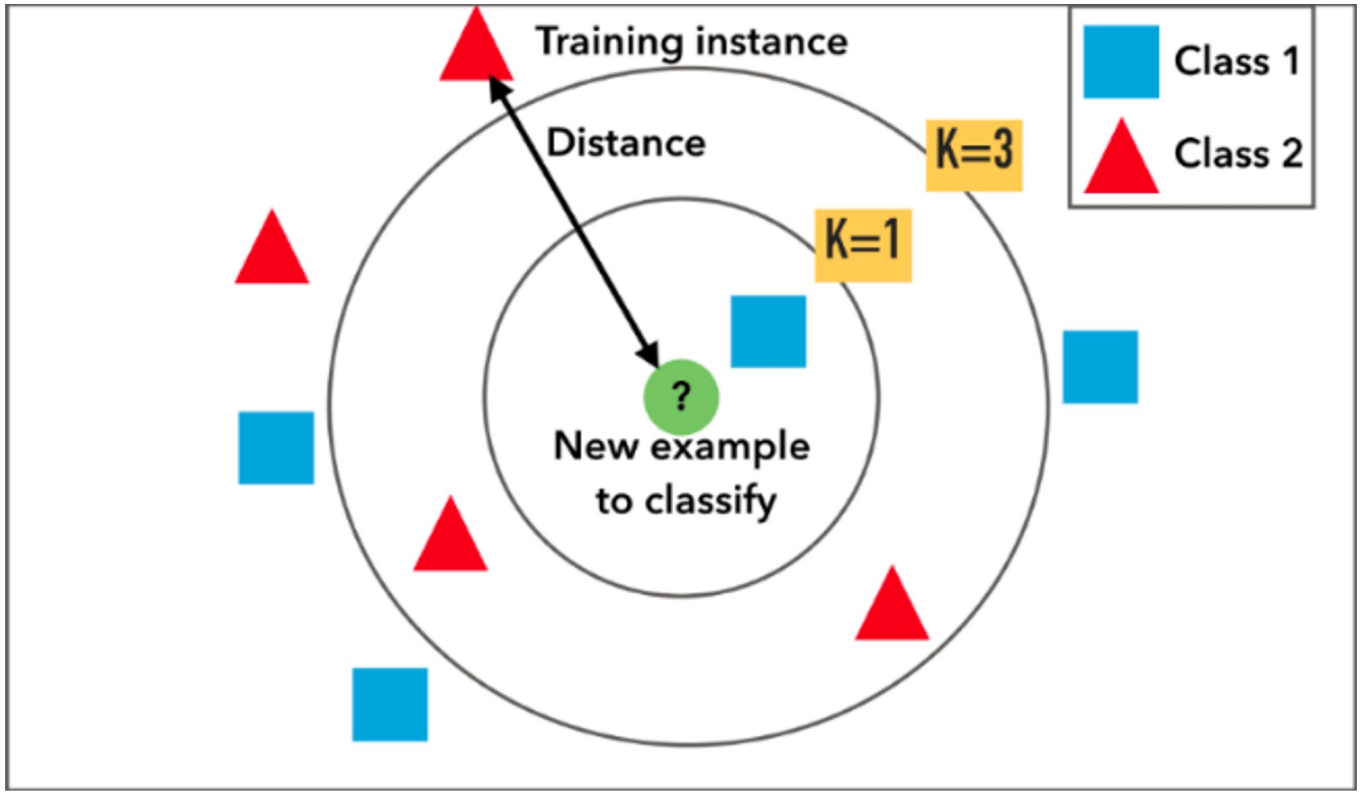




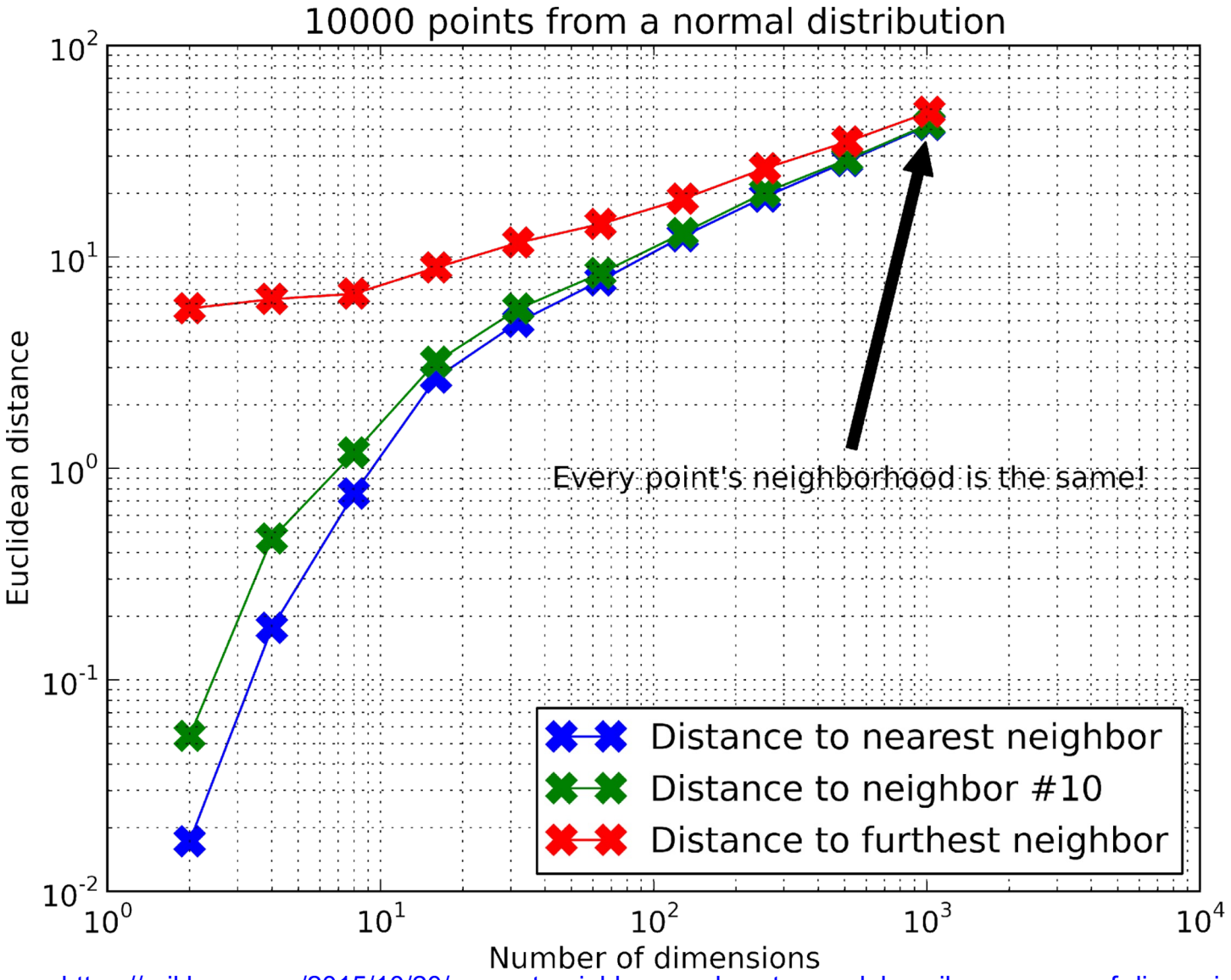
http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

K-Nearest Neighbor Classifier

- Use the k-nearest neighbors as the classification decision
  - Use majority vote



What's wrong with knn in high dimension?



https://erikbern.com/2015/10/20/nearest-neighbors-and-vector-models-epilogue-curse-of-dimensionality.html

Combating the curse of dimensionality

- Feature selection
  - Keep only “Good” features
- Feature transformation (Feature extraction)
  - Transform the original features into a smaller set of features

Feature selection vs Feature transform

- Keep original features
  - Useful for when the user wants to know which feature matters
    - Hard to select good features automatically
- New features (a combination of old features)
- Usually more powerful
  - Harder to interpret the model

Feature selection

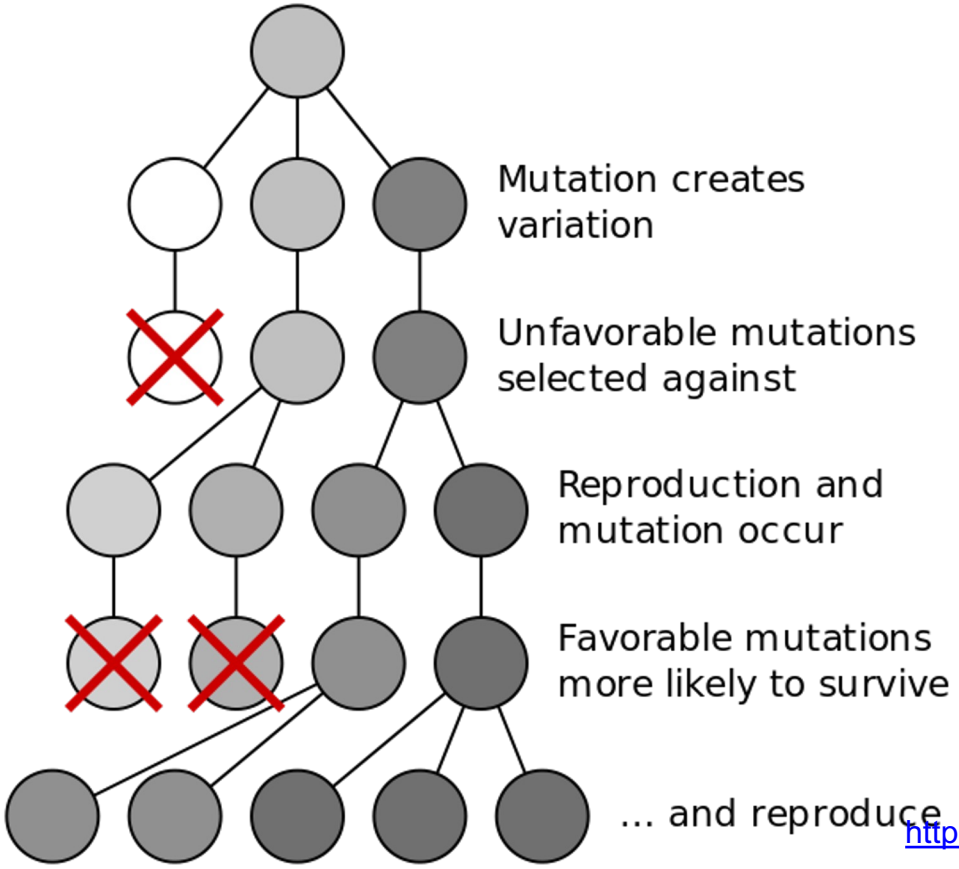
- Hackathon level (time limit days-a week)
  - Drop missing features
  - Low variance column
    - A feature that is a constant is useless. Tricky in practice
  - Forward or backward feature elimination
    - Greedy algorithm: create a simple classifier with n-1 features, n times. Find which one has the best accuracy, drop that feature. Repeat.

Feature selection

- Proper methods
  - Algorithm that handles high dimension well has built-in feature selection
    - Regression with L1 regularization
    - Tree-based classifiers: random forest, XGBoost
  - Genetic Algorithm

Genetic Algorithm

- A method based inspired by natural selection
  - No theoretical guarantees but often work



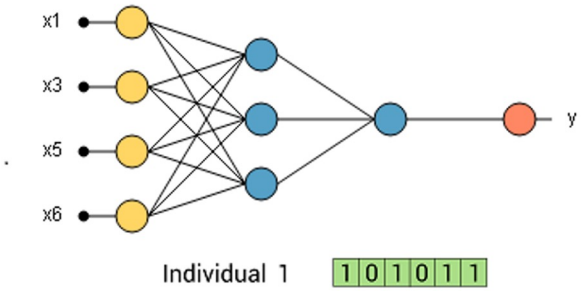
https://elitedatascience.com/dimensionality-reduction-algorithms

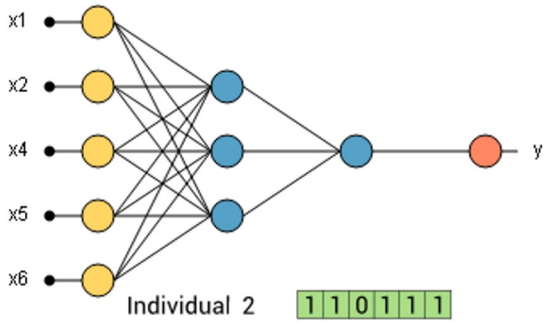
Genetic Algorithm

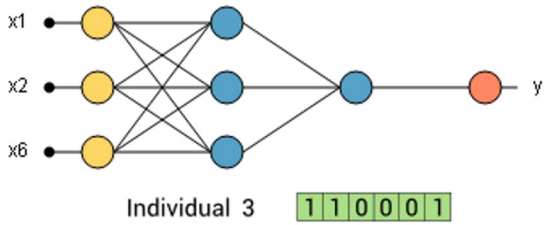
- Initialization
  - Create N classifiers, each using different subset of features
- Selection process
  - Rank the N classifiers according to some criterion, kill the lower half
- Crossover
  - The remaining classifier breeds offsprings by selecting traits from the parents
- Mutation
  - The offsprings can have mutations by random in order to generate diversity
- Repeat till satisfied

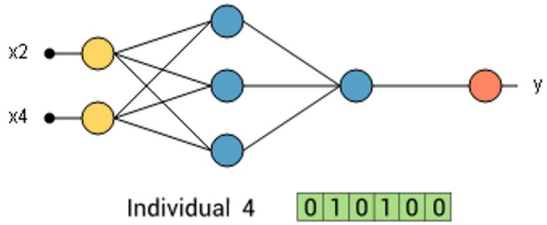
Initialization

- Create N classifiers
- Randomly select a subset of features to use









Examples from https://www.neuraldesigner.com/blog/genetic_algorithms_for_feature_selection

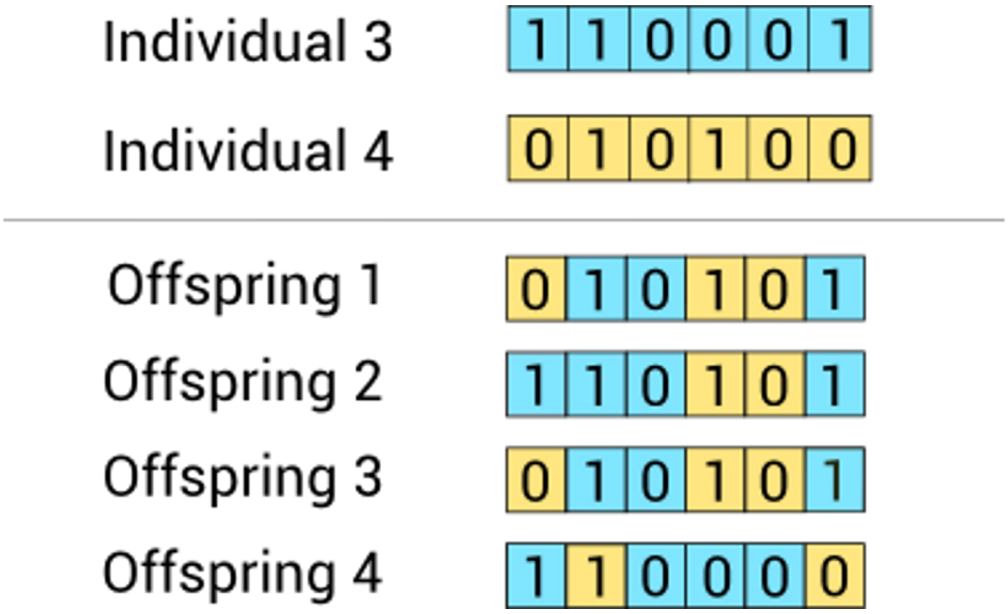
Selection process

- Score the classifiers and kill the lower half (the amount to kill is also a parameter)

	Selection error	Rank
~~Individual 1~~	0.9	1
Individual 2	0.6	3
~~Individual 3~~	0.7	2
Individual 4	0.5	4

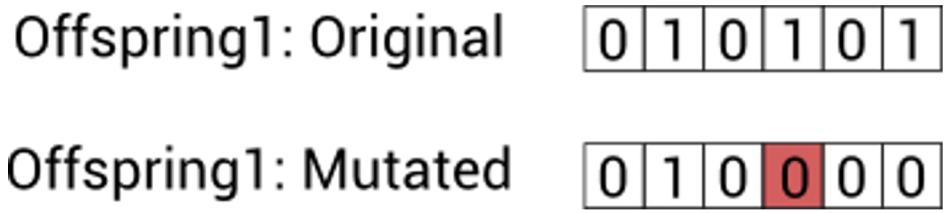
Crossover

- Breed offsprings by randomly select genes from parents



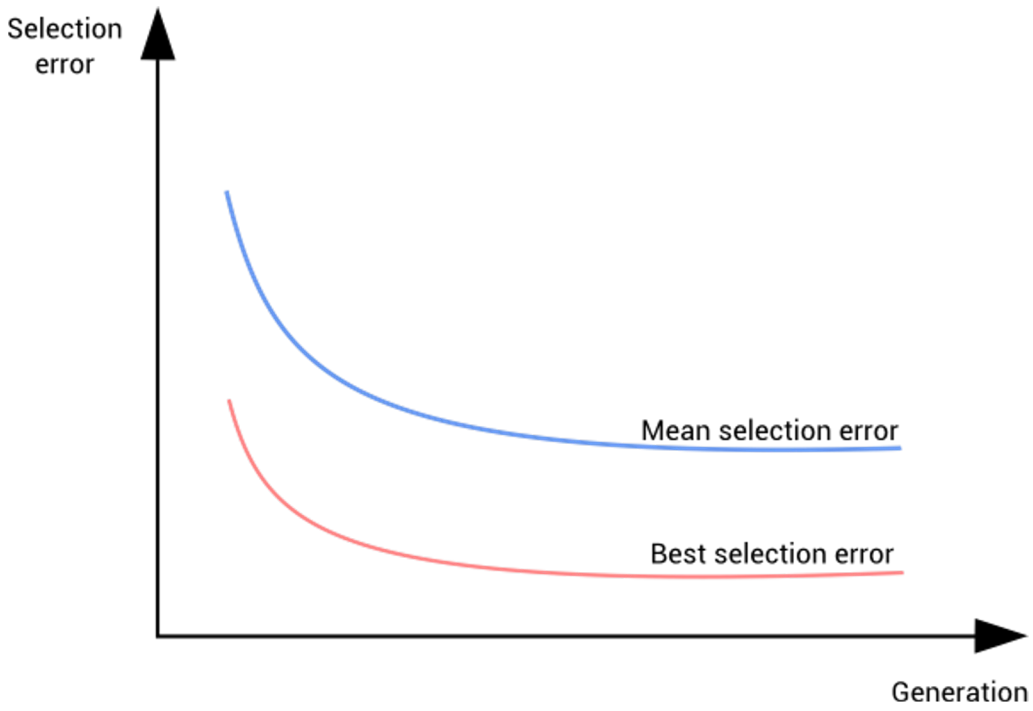
Mutation

- Offspring can mutate with some probability to introduce diversity
- Mutation rate is usually 1/k where k is the number of features.
  - On average you mutate once per individual



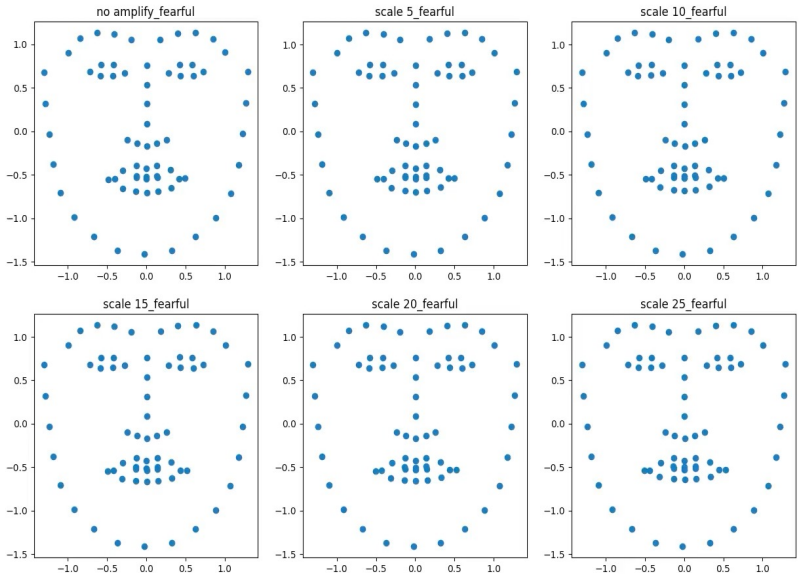
Performance

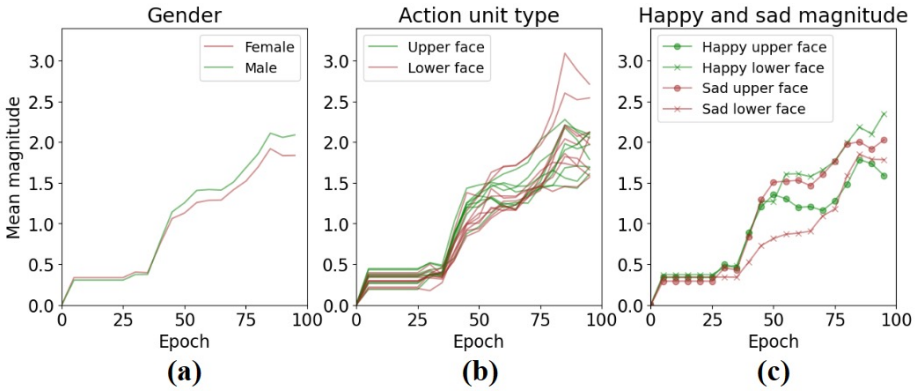
- Usually performs well. The general population usually gets better (mean). The best performing (individual) also gets better after each generation



Other examples of genetic algorithm

- Tuning hyperparameters in a neural network
  - https://blog.coast.ai/lets-evolve-a-neural-network-with-a-genetic-algorithm-code-included-8809bece164
- Tune augmentation algorithms





Generation ->

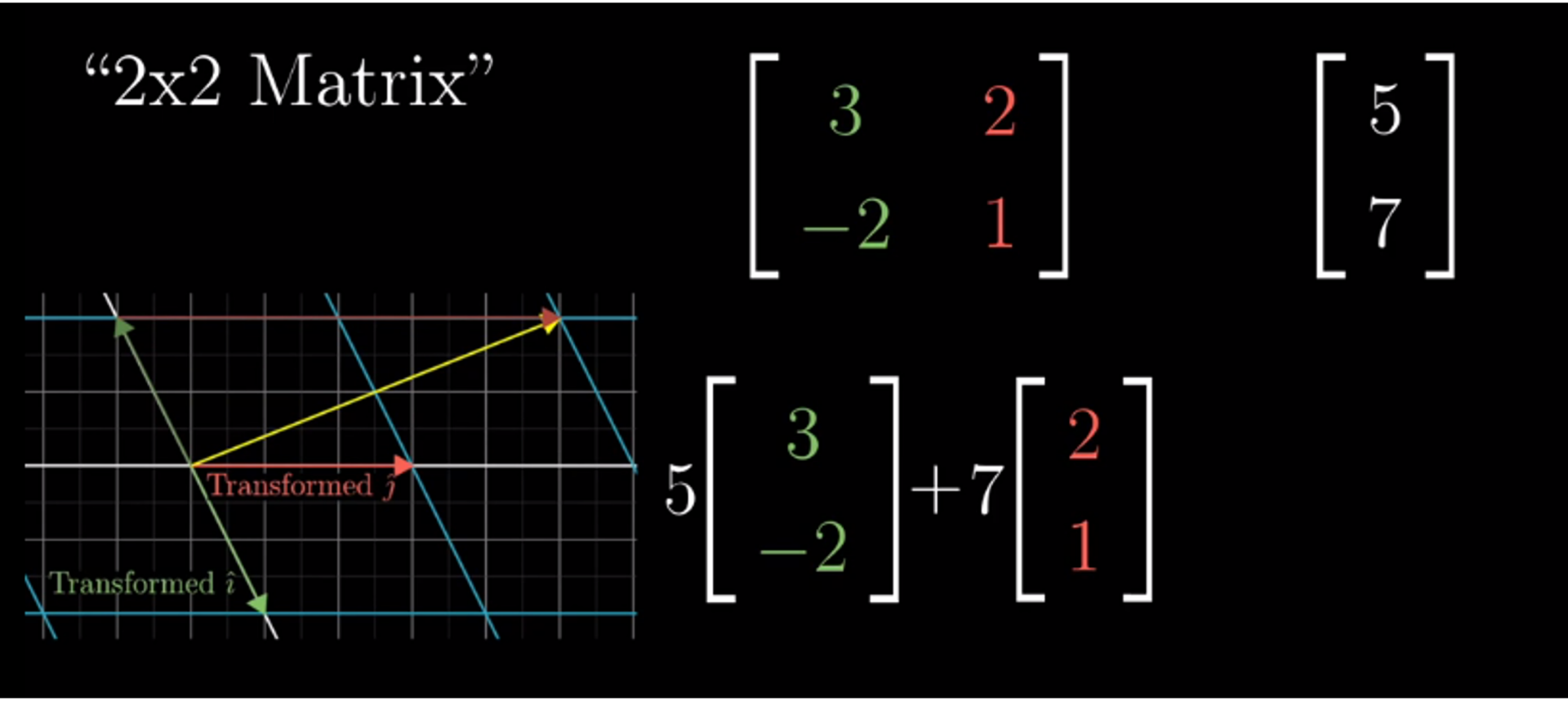
ATTRIBUTE-AWARE AMPLIFICATION OF FACIAL FEATURE SEQUENCES FOR FACIAL EMOTION RECOGNITION, to appear ICASSP 2024

Feature transformation

- Principal Component Analysis
- Linear Discriminant Analysis (NOT Latent Dirichlet Allocation)
- Random Projections

Linear Algebra Review

- Matrix as a sequence of column vectors



https://www.youtube.com/watch?v=kYB8IZa5AuE&list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab&index=4

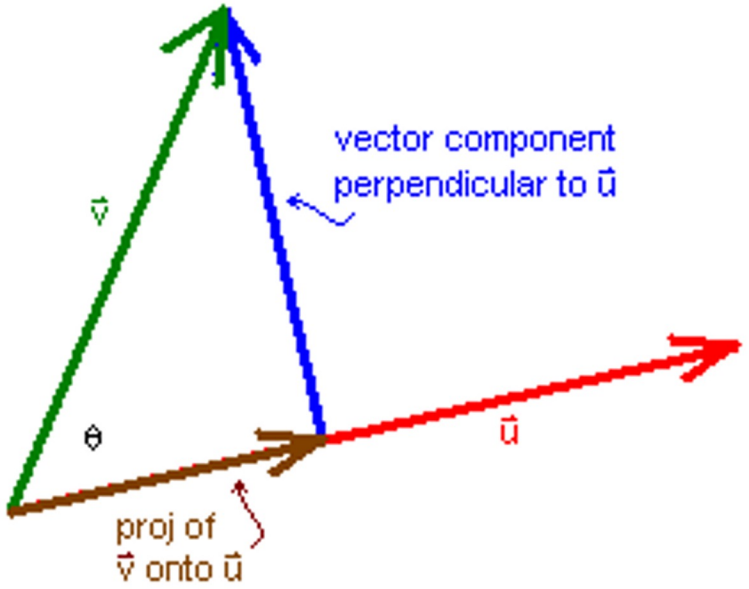
Linear Algebra Review

- View Eigendecomposition (ED) and Singular Value Decomposition (SVD) as rotations and stretches

D has eigenvalues on the diagonal
Q is a matrix where i^{th} column is the q^{th} eigenvector

Linear Algebra Review

- Projection as a change of basis
- Change basis from x,y coordinates to be on u



$u^t v$

Covariance matrix

- Given a set of RVs (features dimension), X_1 X_2 … X_n
- The covariance matrix is a matrix which has the covariance of the i and j RV in position (i,j)

$$\Sigma = \begin{bmatrix} \mathrm{E}[(X_1 - \mu_1)(X_1 - \mu_1)] & \mathrm{E}[(X_1 - \mu_1)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_1 - \mu_1)(X_n - \mu_n)] \\ \mathrm{E}[(X_2 - \mu_2)(X_1 - \mu_1)] & \mathrm{E}[(X_2 - \mu_2)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_2 - \mu_2)(X_n - \mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n - \mu_n)(X_1 - \mu_1)] & \mathrm{E}[(X_n - \mu_n)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_n - \mu_n)(X_n - \mu_n)] \end{bmatrix}.$$

Positive semi-definite and covariance matrixes

Definitions for real matrices [edit]

An $n \times n$ symmetric real matrix M is said to be **positive-definite** if $\mathbf{x}^{\mathrm{T}} M\mathbf{x} > 0$ for all non-zero $\mathbf{x}$ in $\mathbb{R}^n$. Formally,

$$M \text{ positive-definite} \iff \mathbf{x}^{\mathrm{T}} M\mathbf{x} > 0 \text{ for all } \mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$$

An $n \times n$ symmetric real matrix M is said to be **positive-semidefinite** or **non-negative-definite** if $\mathbf{x}^{\mathrm{T}} M\mathbf{x} \geq 0$ for all $\mathbf{x}$ in $\mathbb{R}^n$. Formally,

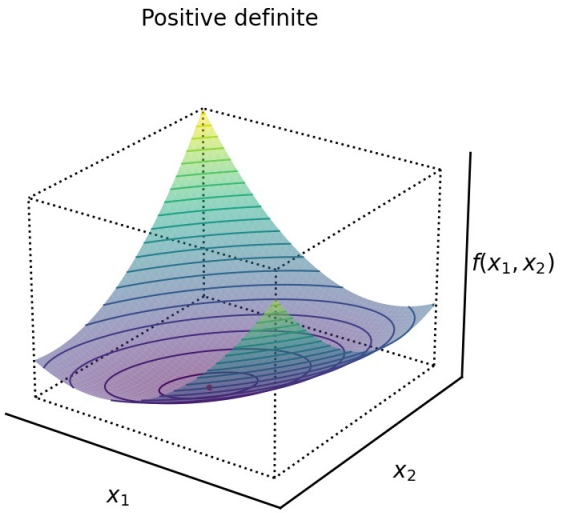
$$M \text{ positive semi-definite} \iff \mathbf{x}^{\mathrm{T}} M\mathbf{x} \geq 0 \text{ for all } \mathbf{x} \in \mathbb{R}^n$$

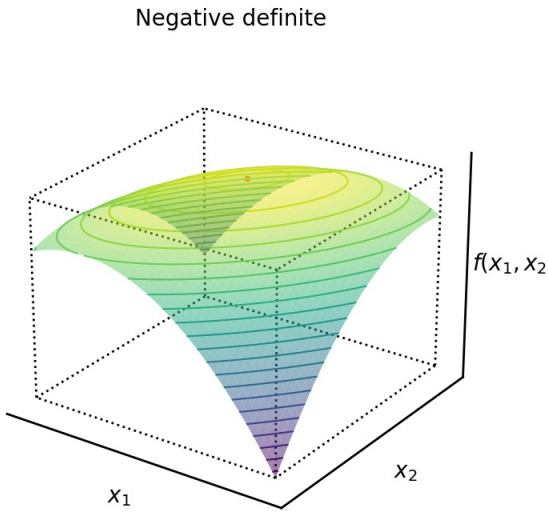
https://en.wikipedia.org/wiki/Definite_matrix

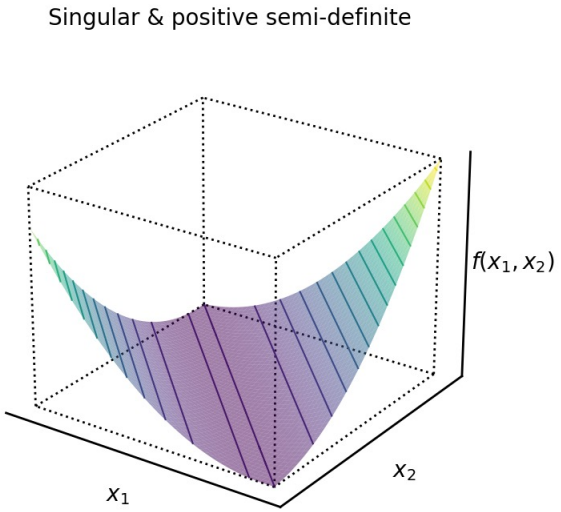
- Covariance matrix is positive semi-definite and symmetric.
  - Semi-definite because sometimes the variance can be zero.

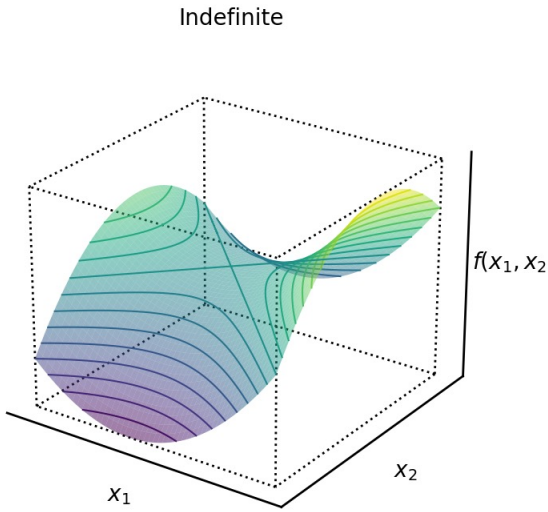
Positive semi-definite

$$f(x_1, x_2) = \frac{1}{2} \begin{bmatrix} x_1 & x_2 \end{bmatrix} \mathbf{A} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \mathbf{b}^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + c$$









https://gregorygundersen.com/blog/2022/02/27/positive-definite/

Positive semi-definite and eigen decomposition

Definitions for real matrices [edit]

An $n \times n$ symmetric real matrix M is said to be **positive-definite** if $\mathbf{x}^{\mathrm{T}} M\mathbf{x} > 0$ for all non-zero $\mathbf{x}$ in $\mathbb{R}^n$. Formally,

$$M \text{ positive-definite} \iff \mathbf{x}^{\mathrm{T}} M\mathbf{x} > 0 \text{ for all } \mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$$

An $n \times n$ symmetric real matrix M is said to be **positive-semidefinite** or **non-negative-definite** if $\mathbf{x}^{\mathrm{T}} M\mathbf{x} \geq 0$ for all $\mathbf{x}$ in $\mathbb{R}^n$. Formally,

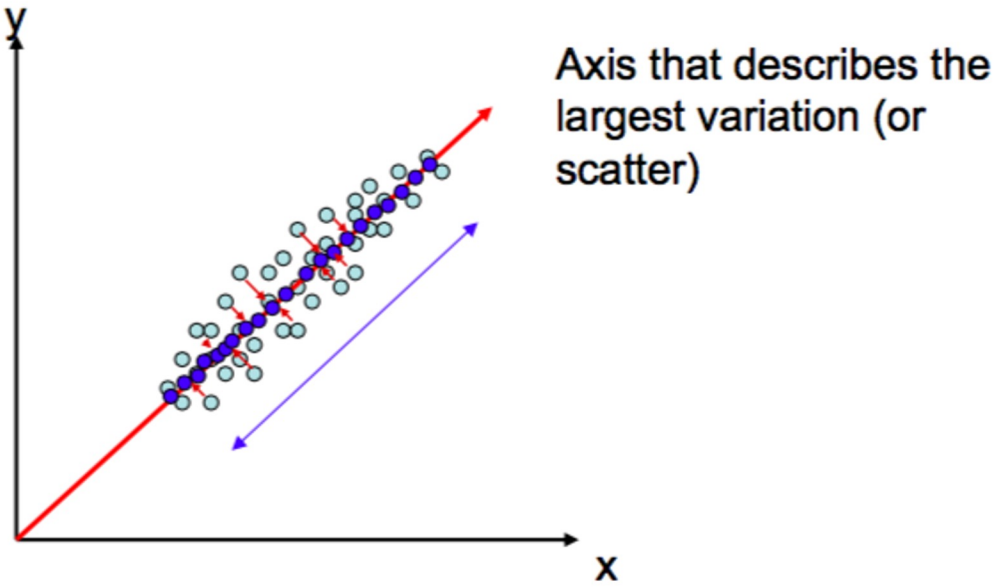
$$M \text{ positive semi-definite} \iff \mathbf{x}^{\mathrm{T}} M\mathbf{x} \geq 0 \text{ for all } \mathbf{x} \in \mathbb{R}^n$$

https://en.wikipedia.org/wiki/Definite_matrix

- Covariance matrix is positive semi-definite and symmetric.
  - Symmetric -> real eigen values, eigen vectors are mutually orthogonal
    $q_i^t q_j = 0$ for i != j
  - Positive semi-definite -> eigen values are nonnegative
  - Positive definite -> eigen values are positive -> invertible

What is PCA?

- We want to reduce the dimensionality but keep useful information
  - What is useful information? Variation
- We want to find a projection (a transformation) that describe maximum variation



Formulation

- Maximize the variance after projection ie
  - argmax Var($\mathbf{w}^t \mathbf{x}$)
- Subject to w is a unit vector

- $\Sigma\mathbf{w} = \lambda\mathbf{w}$ <- eigenvector

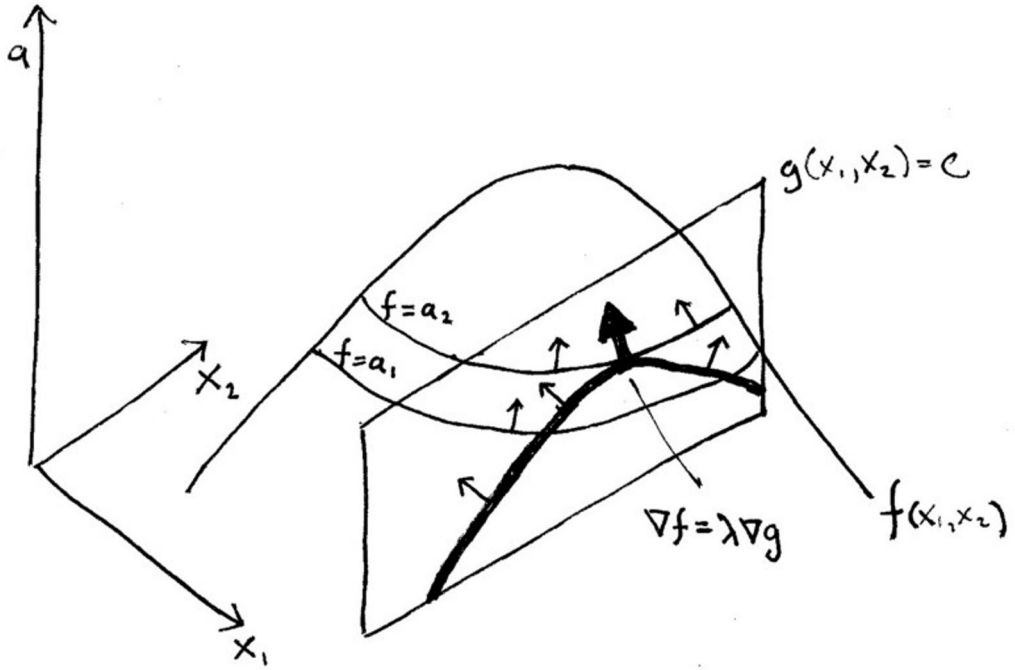
Trace properties

1. tr (a) = a
2. trA = trA^T
3. tr(A+B) = trA+trB
4. tr(aA) = atr(A)

5. $$\nabla_A trAB = B^T$$
6. $$\nabla_{A^T} f(A) = (\nabla_A f(A))^T$$
7. $$\nabla_A trABA^TC = CAB + C^TAB^T$$
8. $$\nabla_{A^T} trABA^TC = B^TA^TC^T + BA^TC$$

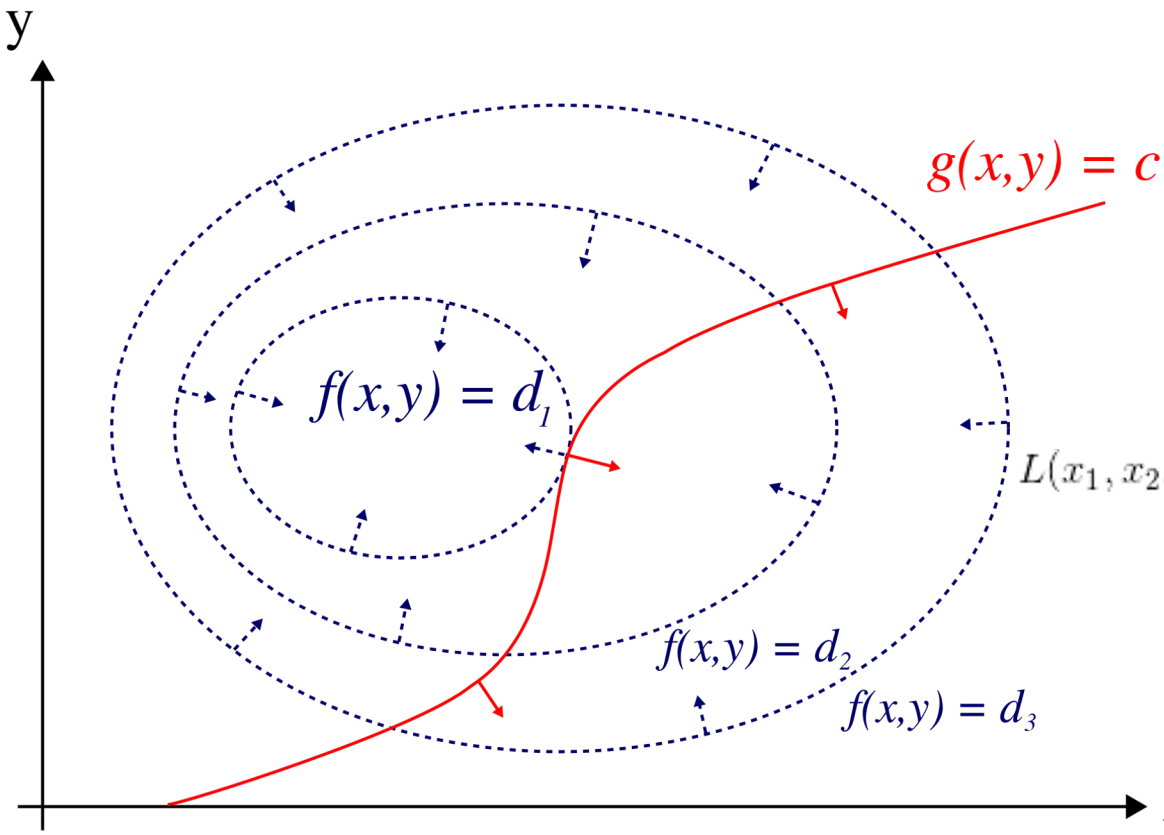
Lagrange Multiplier

https://medium.com/@andrew.chamberlain/a-simple-explanation-of-why-lagrange-multipliers-works-253e2cdcbf74



$$L(x_1, x_2, \lambda) = f(x_1, x_2) - \lambda(g(x_1, x_2) - c)$$

$$\nabla L = \begin{bmatrix} \frac{\partial f}{\partial x_1} - \lambda \frac{\partial g}{\partial x_1} \\ \frac{\partial f}{\partial x_2} - \lambda \frac{\partial g}{\partial x_2} \\ g(x_1, x_2) - c \end{bmatrix} = 0$$



$$L(x_1, x_2, \lambda) = f(x_1, x_2) - \lambda(g(x_1, x_2) - c)$$

$$\nabla L = \begin{bmatrix} \frac{\partial f}{\partial x_1} - \lambda \frac{\partial g}{\partial x_1} \\ \frac{\partial f}{\partial x_2} - \lambda \frac{\partial g}{\partial x_2} \\ g(x_1, x_2) - c \end{bmatrix} = 0$$

https://en.wikipedia.org/wiki/Lagrange_multiplier#/media/File:LagrangeMultipliers2D.svg

So we got to eigenvectors

- A dxd covariance matrix has d eigenvectors/values pair. Do we use all of them?
- Which pair to use?

Selecting eigenvectors

- Remember the variance of projected data is

$$\omega^T \Sigma \omega. \quad (1)$$

- And our solution yielded

$$\Sigma\omega = \lambda\omega \quad (2)$$

- Plug (2) in (1) and we get

$$\begin{aligned} \text{projected variance} &= \omega^T \Sigma \omega = \omega^T \lambda\omega \\ &= \lambda\, \omega^T \omega \qquad (\text{remember } \|\omega\|=1) \\ &= \lambda \end{aligned}$$

PCA

- The direction vector captures the variance corresponding to the eigenvalue
- So we want the higher eigenvalues
  - How many?

Matrix rank

- A square dxd matrix has full rank (e.g. rank d) if the columns are linearly independent.
- The number of linearly independent columns is the rank of the matrix

- A covariance matrix of size dxd will have have at most N-1 rank where N is the number of training samples
  - 640x640 images = ~400000 dimensions
  - 1000 training images
  - The covariance matrix will be at most rank 999. The missing rank is because of the mean.

https://stats.stackexchange.com/questions/178587/why-is-the-rank-of-covariance-matrix-at-most-n-1

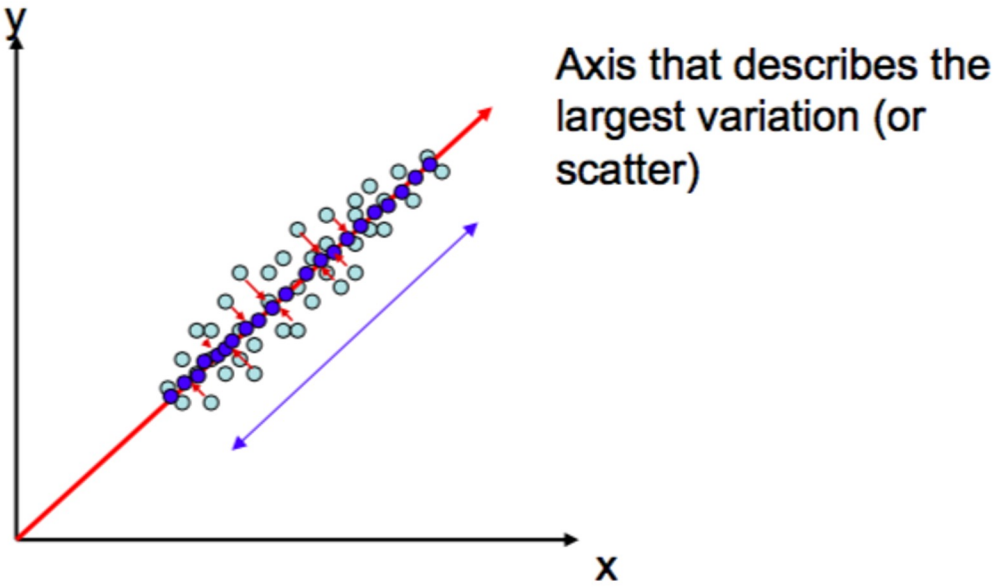
PCA

- The direction vector captures the variance corresponding to the eigenvalue
- So we want the higher eigenvalues
- Take the eigenvalues with non-zero eigenvalues (at most N-1 non-zero eigenvalues)

Eigenfaces

What is PCA?

- We want to reduce the dimensionality but keep useful information
  - What is useful information? Variation
- We want to find a projection (a transformation) that describe maximum variation



Means

- In PCA, we model variance. (Variation around the mean)
- In our projection we need to remove the mean

$$\mathbf{p} = \mathbf{V}^{\mathbf{T}}(\mathbf{x} - \mathbf{m})$$

- The mean is the mean of all your training data
- If we want to reconstruct the data we need to add back the mean

$$\mathbf{x} = \sum_{i=1}^{N} p_i \mathbf{v}_i + \mathbf{m} = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + .. + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} + \mathbf{m} = \mathbf{V}\mathbf{p} + \mathbf{m}$$

Basis decomposition

- Let's consider our projection w_i which is the eigenvectors to be a basis vector v_i
- We can represent any vector as a sum of basis vectors as follows:

$$\mathbf{x} = \sum_{i=1}^{N} p_i \mathbf{v}_i = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + .. + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} = \mathbf{Vp}$$

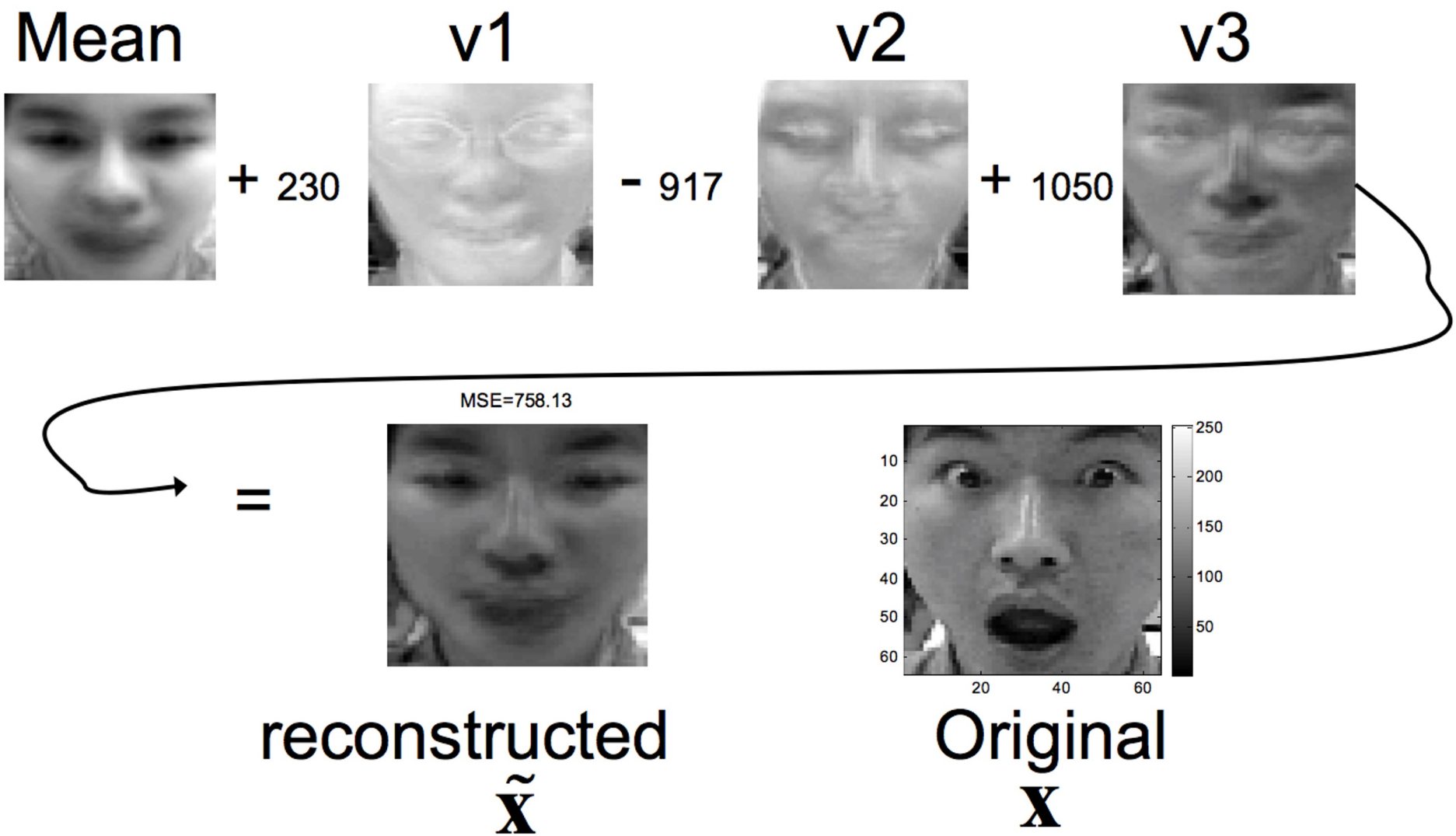
Finding the weights

$$\mathbf{x} = \sum_{i=1}^{N} p_i \mathbf{v}_i = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + .. + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} = \mathbf{V}\mathbf{p}$$

- If v_i are orthogonal, the projection of x onto v_i gives p_i

$$\mathbf{V}^{\mathbf{T}}\mathbf{x} = \begin{bmatrix} - & \mathbf{v}_1 & - \\ - & \mathbf{v}_2 & - \\ - & \mathbf{v}_3 & - \end{bmatrix} \begin{bmatrix} | \\ \mathbf{x} \\ | \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

Reconstruction with eigenfaces



Practical issues

- If your data has different magnitudes in different dimensions, normalize each dimension before PCA

- If we have 640x640 images = ~400000 dimensions.
- What is the size of the covariance matrix?
- 400000 x 400000

Practical issues

- You have N training examples.
- For the case where N << 400000, we only have N-1 eigenvalues we care about anyway

Gram Matrix

$$\Sigma = E(x - \mu)(x - \mu)^T = XX^T$$

Covariance matrix is the outer-product of the input matrix

Must solve $\Sigma v = \lambda v$

$XX^T v = \lambda v$ (pre-mult by X^T) (1)

$X^T X X^T v = \lambda X^T v$ ($v' = X^T v$) (2)

Solve eigenvalue problem $X^T X v' = \lambda v'$

- $X^T X$ is a gram of inner-product matrix. Its size is NxN where N is the number of data samples.

But how to get v from v'?

- From previous slide, equation (1) and (2)
  - $XX^Tv = \lambda v$ (1)
  - $v' = X^Tv$ (2)
- Substitute (2) into (1)
  - $Xv' = \lambda v$

- Thus, v = Xv'. We don't care about the scaling term because we will always scale the eigenvector so that it is orthonormal i.e. ||v|| = 1.

How many eigenvectors?

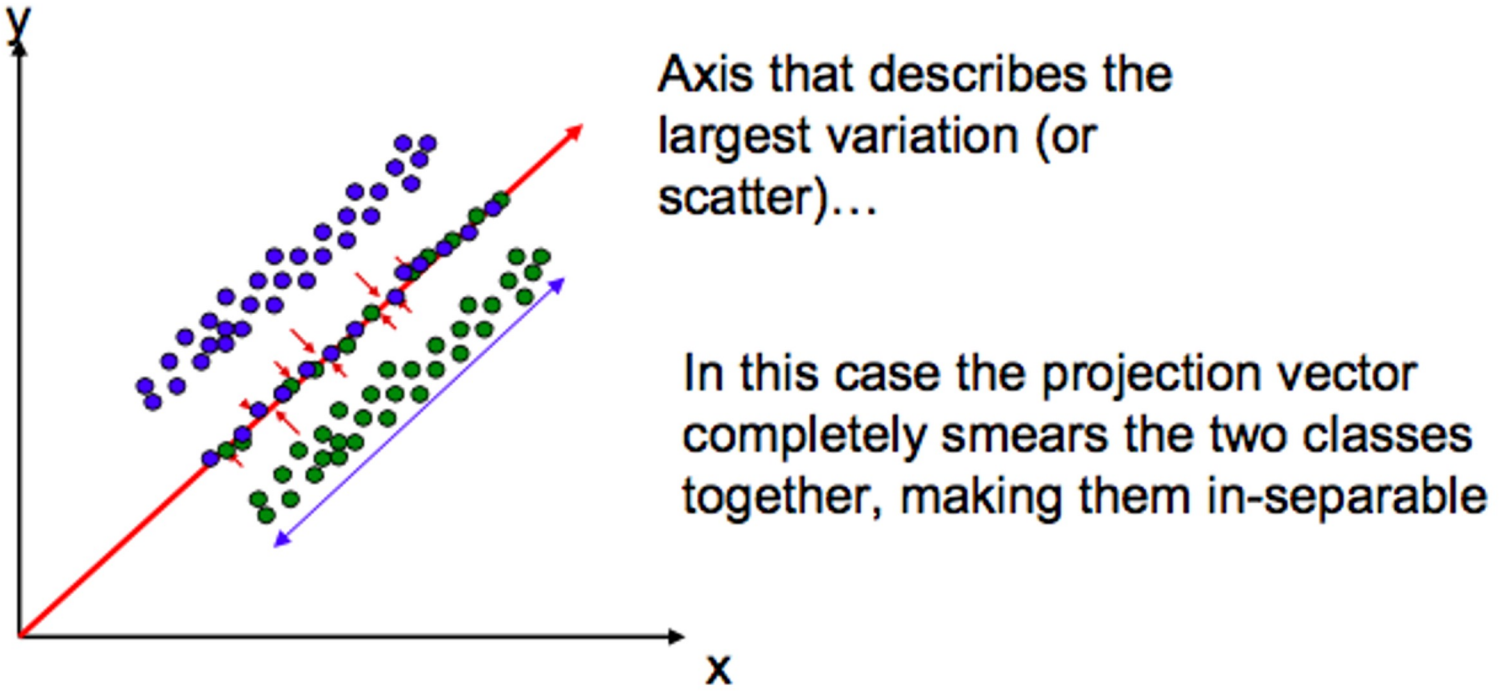
- Select based on amount of variance explained
  - Sum of eigenvalues exceeds some percent of total
- Reconstruction error

$$\mathbf{x} = \sum_{i=1}^{N} p_i \mathbf{v}_i = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + .. + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} = \mathbf{Vp}$$

  - Select enough v so that the difference between original x and reconstructed x is small
- Select based on downstream task

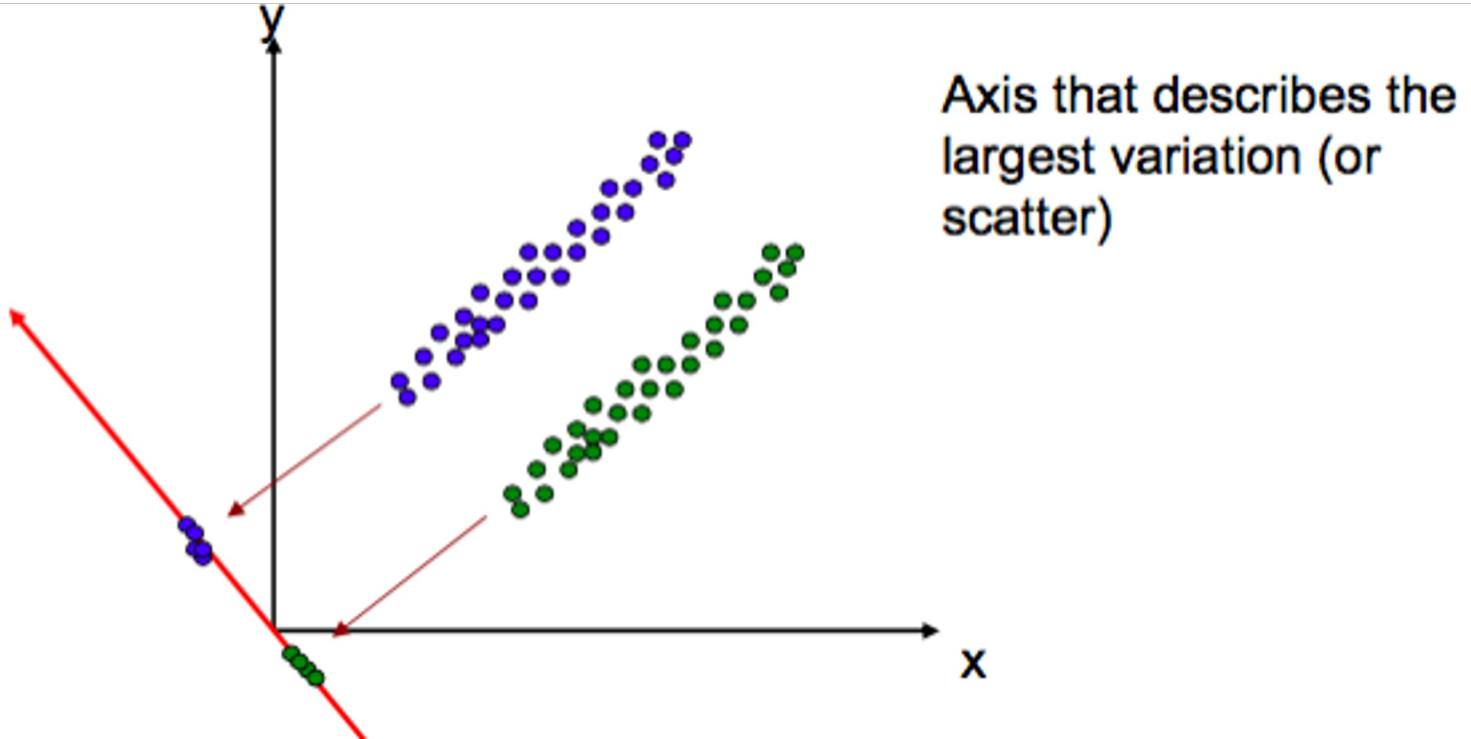
PCA for classification

- PCA does not cares about the class labels



What is LDA

- Find the projections that separate the classes.
- Assumes unimodal Gaussian model for each class
  - Maximize the distance between the means and minimize the variance of each class -> best classification performance



Simple 2 class case

- We want to maximize the distance between the projected means:

  e.g. maximize $|(\tilde{\mu}_1 - \tilde{\mu}_2)|^2$

Where $\tilde{\mu}_1$ is the projected mean μ_1 of class onto LDA direction vector $\mathbf{w}$, i.e.

$$\tilde{\mu}_1 = \mathbf{w}^T \boldsymbol{\mu}_1$$

and for class 2: $\tilde{\mu}_2 = \mathbf{w}^T \boldsymbol{\mu}_2$ thus

$$|(\tilde{\mu}_1 - \tilde{\mu}_2)|^2 = |(\mathbf{w}^T \boldsymbol{\mu}_1 - \mathbf{w}^T \boldsymbol{\mu}_2)|^2$$

$$= \mathbf{w}^T (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\ (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \mathbf{w}$$

$$= \mathbf{w}^T \mathbf{S}_B \mathbf{w}$$

Between class scatter matrix S_B

$$(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = (\mathbf{w}^T\boldsymbol{\mu}_1 - \mathbf{w}^T\boldsymbol{\mu}_2)^2$$
$$= \mathbf{w}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\ (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\mathbf{w}$$
$$= \mathbf{w}^T\mathbf{S}_B\mathbf{w}$$

We want to maximize w^TS_Bw where S_B is the between class scatter matrix defined as:

$$\mathbf{S}_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$$

We also want to minimize within class scatter

- The variance or scatter of each class. We also want to minimize them.

$$\tilde{s}_1^{\,2} = \sum_{i=1}^{N_1} (\tilde{x}_i - \tilde{\mu}_1)^2$$

Minimize the total scatter $\tilde{s}_1^{\,2} + \tilde{s}_2^{\,2}$

Within class scatter

- Lets expand on scatter s_1, s_2.

$$\tilde{s}_1^2 = \sum_{i=1}^{N_1} (\tilde{x}_i - \tilde{\mu}_1)^2$$

$$= \sum_{i=1}^{N_1} (\mathbf{w}^T \mathbf{x}_i - \mathbf{w}^T \boldsymbol{\mu}_1)^2$$

$$= \sum_{i=1}^{N_1} \mathbf{w}^T (\mathbf{x}_i - \boldsymbol{\mu}_1)(\mathbf{x}_i - \boldsymbol{\mu}_1)^T \mathbf{w}$$

$$= \mathbf{w}^T \mathbf{S}_1 \mathbf{w}$$

Total within class scatter

- We want to minimize $\tilde{s}_1^2 + \tilde{s}_2^2$

- This is the same as $\mathbf{w}^T\mathbf{S}_w\mathbf{w}$

$$\mathbf{S}_w = \sum_{i=1}^{C} \sum_{j=1}^{Ni} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T$$

C number of classes, Ni number of images from class i

Fisher Linear Discriminant Criterion

- We want to maximize between class scatter
- We want to minimize within class scatter

- We have an objective function as a ratio so we can achieve both!

$$J(\mathbf{w}) = \frac{|(\tilde{\mu}_1 - \tilde{\mu}_2)|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

LDA solution

- If you do calculus

$$\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w}$$

$$\mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{w} = \lambda\mathbf{w}$$

If S_w is non-singular and invertible.

- Generalized eigenvalue problem. The number of solutions is min(rankS_B, rankS_W) = C-1 or N-C
- For 2 class this simplifies to
  - Note this is only one projection direction

$$\mathbf{w} = \mathbf{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

LDA+PCA

- Original features dimension L
- S_B S_W are LxL matrixes
- rankS_B = C-1, rankS_W = N-C
- First do PCA to reduce dimension
- Then do LDA to maximize classification ability
- How many dimensions to PCA?
  - Do PCA to keep N-C eigenvectors -> Makes S_w full rank and invertible
  - Then, do LDA and compute C-1 projections in this N-C subspace
- PCA+LDA = Fisher projection

Random projection

- Original d-dimensional data is project to k-dimensional subspace
- Using a random k x d matrix R with unit norm columns
  - Johnson-Lindenstrauss lemma: If points in a vector space are projected onto a randomly selected subspace of suitably high dimension, then the distances between the points are approximately preserved
- Elements of R are usually selected from Gaussians.
  - Generally any zero mean unit variance distribution would satisfy Johnson-Lindenstrauss lemma.

Random projection notes

- R is not generally orthogonal.
  - But in a substantially large subspace, random vectors might be close to orthogonal.
- Looks weird but works…
- Can be used to initialized GMMs by finding good initializations in the projected space first

- Robust to outliers in the data, because it uses no data
- Faster to train than PCA if data has high dimension
  - $O(k^2 \times n + k^3)$ (covariance computation + eigen depcomposition) where k- features, n – samples
- See https://medium.com/data-science-in-your-pocket/random-projection-for-dimension-reduction-27d2ec7d40cd for usage guides

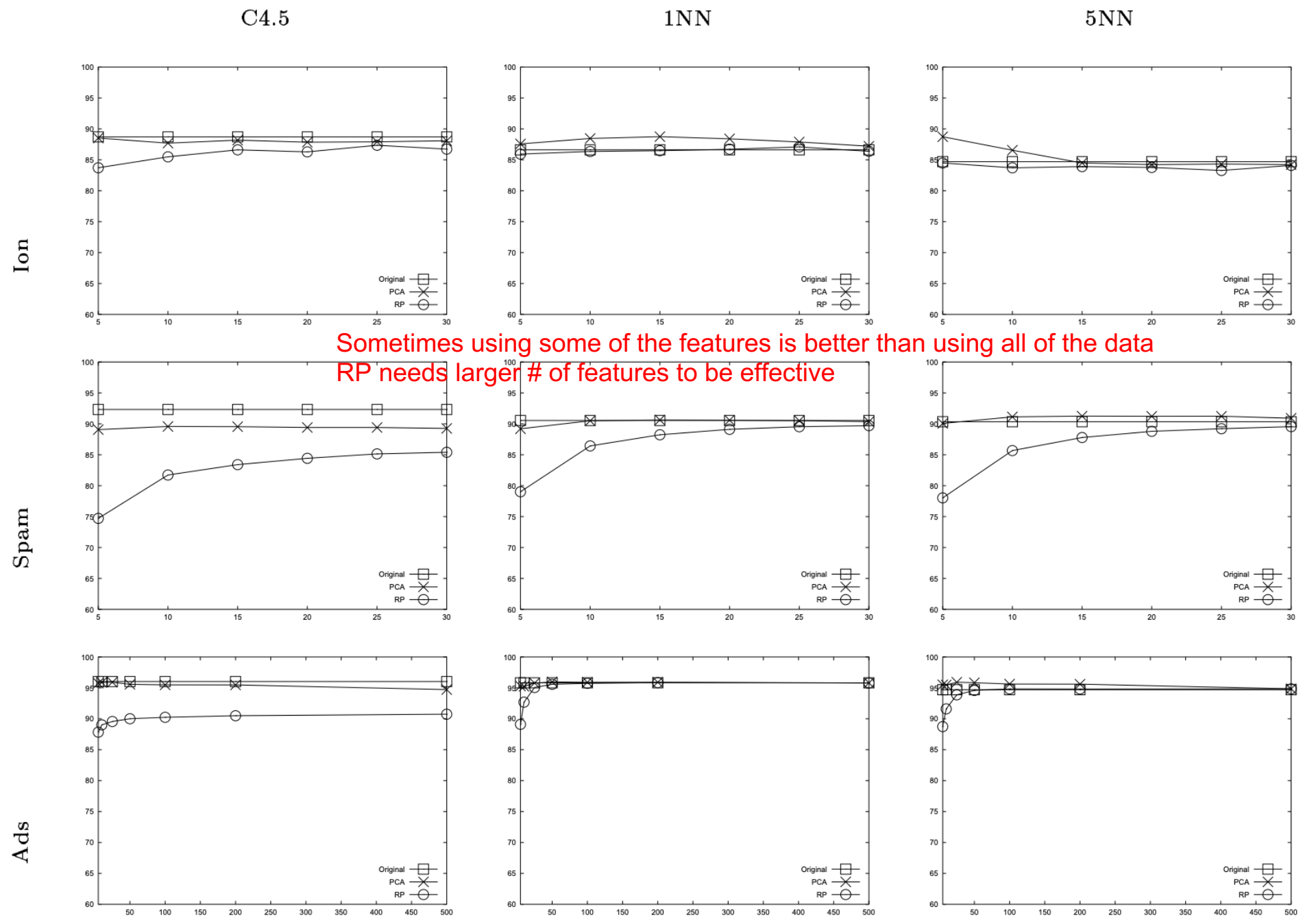
Workflow for dimensionality reduction in ML

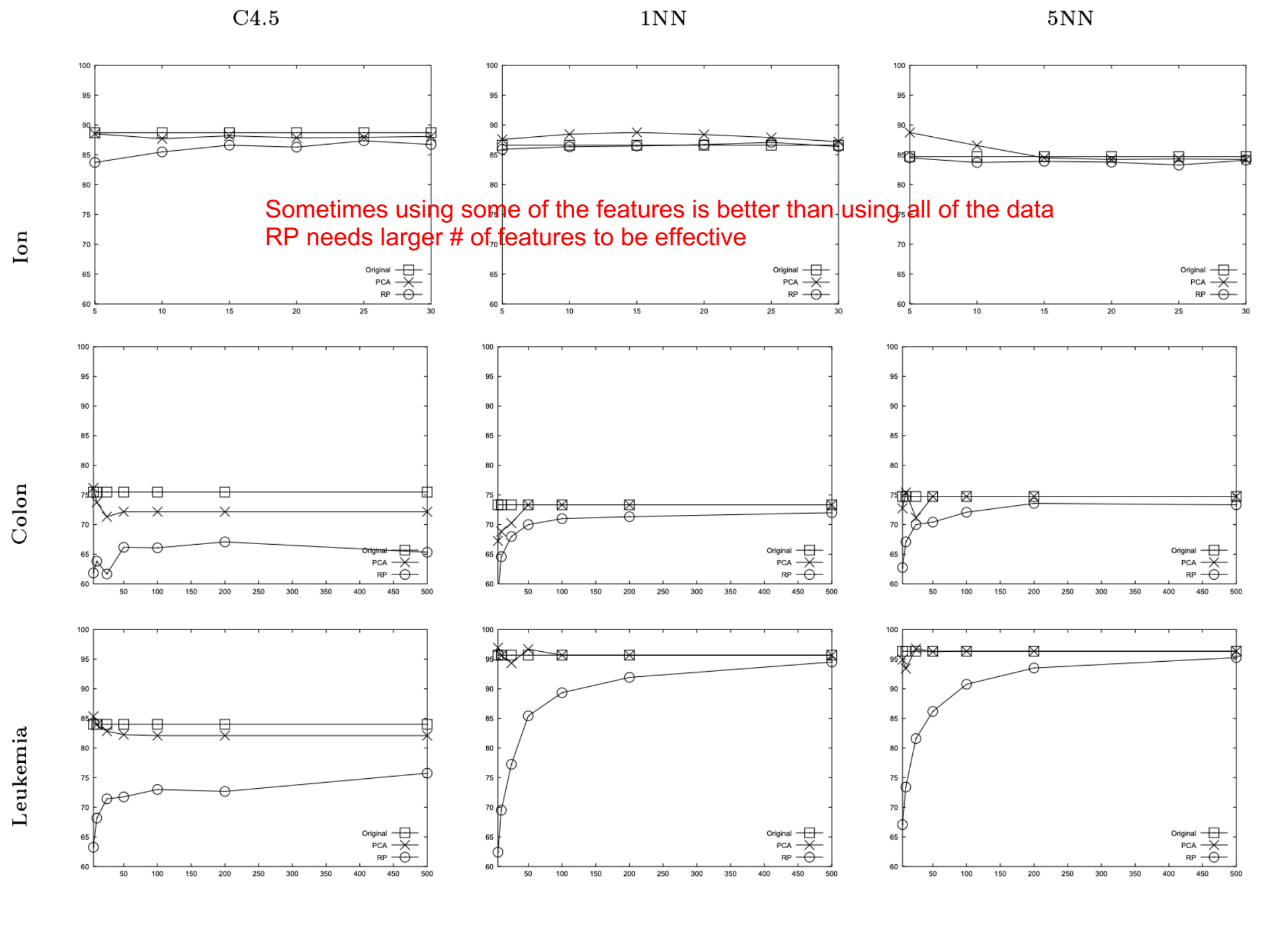
1. Reduce the number of features by projecting into lower dimensions (need to pick #of dimensions via cross validation)
2. Build a model using your favorite technique
3. ???
4. Profit

Table 1: Description of Datasets

Name	# Instances	# Attributes
Ionosphere	351	34
Colon	62	2000
Leukemia	72	3571
Spam	4601	57
Ads	3279	1554

Experiments with Random Projections for Machine Learning
http://archive.dimacs.rutgers.edu/Research/MMS/PAPERS/rp.pdf


C4.5
1NN
5NN
Ion
Spam
Ads
Original
PCA
RP
Sometimes using some of the features is better than using all of the data
RP needs larger # of features to be effective


C4.5
1NN
5NN
Ion
Colon
Leukemia
Sometimes using some of the features is better than using all of the data
RP needs larger # of features to be effective
Original
PCA
RP

Visualization

Methods covered

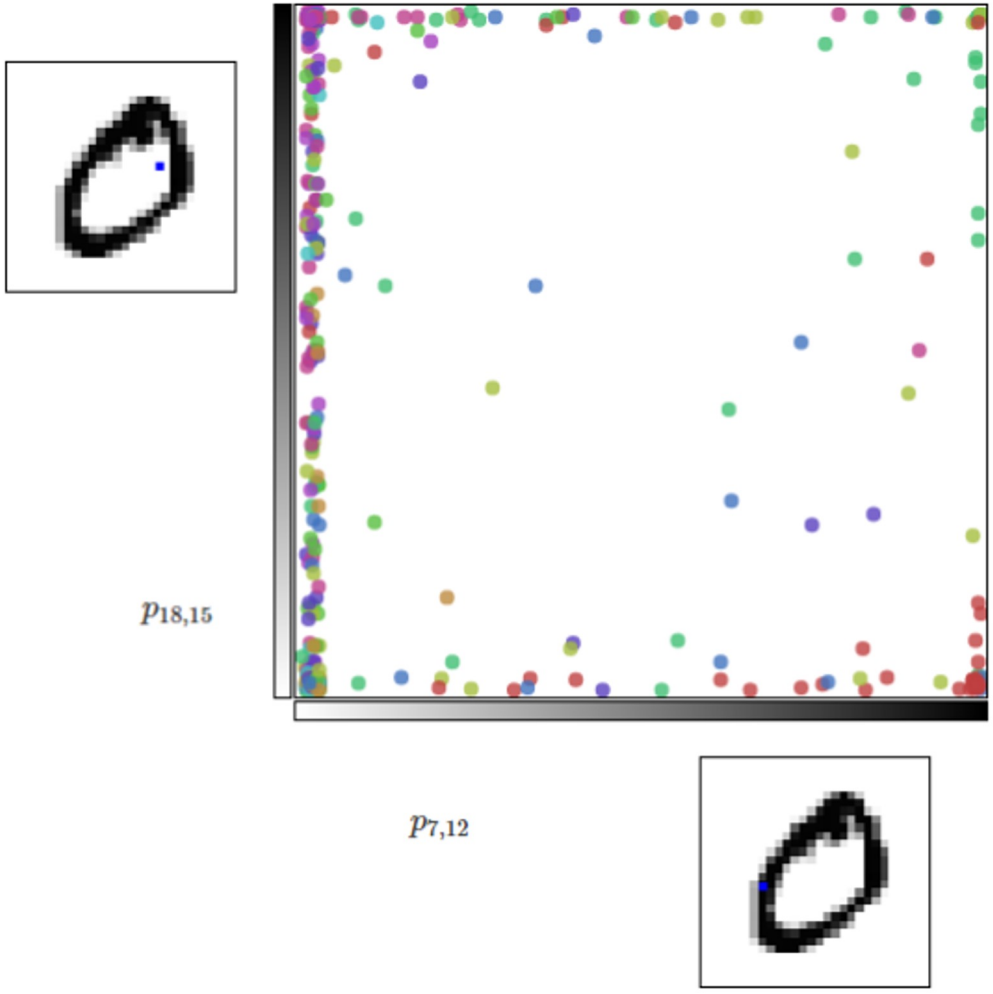
1. Reducing reconstruction error: PCA
2. Keeping direction of maximum separability: LDA
3. Preserving distance (globally): RP

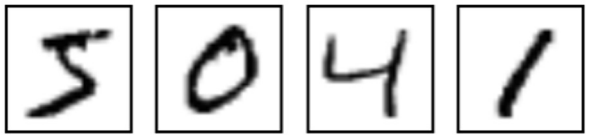
These are usually useful for downstream machine learning methods. (Classification/Regression/Clustering)

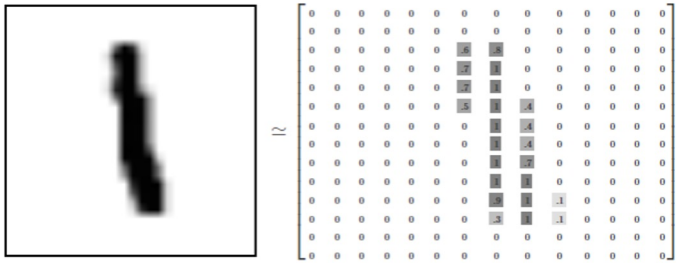
But what if we, as humans, want to get a sense of our data?

1. Interpretability (in some sense): ????
   - Check if our dimensionality reduction is good
   - Analyze clusters from k-mean, GMM, etc.

Visualizing MNIST

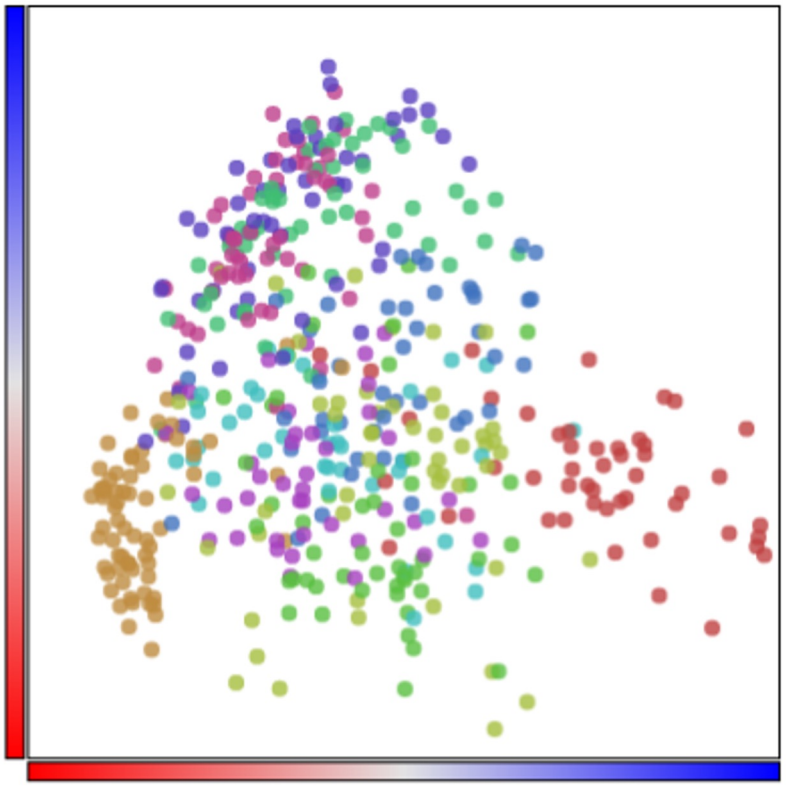


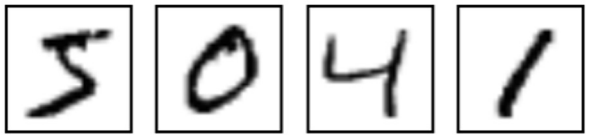


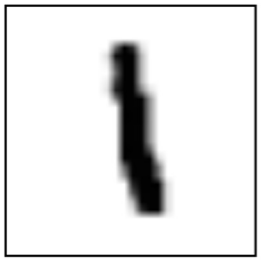


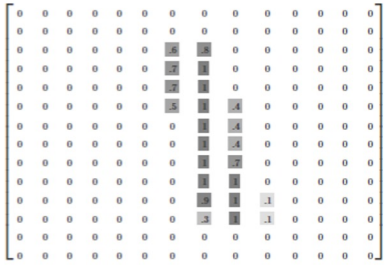
http://colah.github.io/posts/2014-10-Visualizing-MNIST/

PCA with MNIST









http://colah.github.io/posts/2014-10-Visualizing-MNIST/

t-distributed Stochastic Neighbor Embedding (t-SNE)

Preserves neighbor (preserves local distance).

- Things close together should be close together in the projected space
- Prefer using few projected dimensions (2-3)

Defining neighbors

Define $P_{j|i}$ probability that *i* would pick *j* as its neighbor

Assume *i* picks proportional to Gaussian centered at *i*

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2)/2\sigma_i^2}{\sum_{k \neq i} exp(-||x_i - x_k||^2)/2\sigma_i^2}$$

$P_{i|i} = 0$ since we don't want to have it pick itself

The variance is fixed to some value.

Defining neighbors

Define $p_{j|i}$ probability that *i* would pick *j* as its neighbor

Assume *i* picks proportional to Gaussian centered at *i*

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2)/2\sigma_i^2}{\sum_{k \neq i} exp(-||x_i - x_k||^2)/2\sigma_i^2}$$

When projected to set of points $\{y_i\}$, define $q_{j|i}$ the probability that *i* would pick *j* in embedding/latent space

$$q_{j|i} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq i} exp(-||y_i - y_k||^2)}$$

We set the variance in the y space to be 1/sqrt(2)

Defining neighbors

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2)/2\sigma_i^2}{\sum_{k \neq i} exp(-||x_i - x_k||^2)/2\sigma_i^2}$$

$$q_{j|i} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq i} exp(-||y_i - y_k||^2)}$$

We expect p and q to be the same -> small distance

How to measure distance between probability functions?

Kullback-Leibler (KL) divergence

KL divergence

Distance between two distributions

$$D_{KL}(P||Q) = \sum_i P(i) log \frac{P(i)}{Q(i)} = -\sum_i P(i) log \frac{Q(i)}{P(i)}$$

Note $D_{KL}(P||Q) \neq D_{KL}(Q||P)$ (Not a real distance)
Always positive. Equals 0 iff Q = P at every point.

P(head) = 0.5 P(tail) = 0.5

Q(head) = 0.7 Q(tail) = 0.3

$D_{KL}(P||Q)$ = 0.5*ln 0.5/0.7 + 0.5*ln 0.5/0.3 = 0.087

$D_{KL}(Q||P)$ = 0.7*ln 0.7/0.5 + 0.3*ln 0.3/0.5 = 0.082

Loss function

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2)/2\sigma_i^2}{\sum_{k \neq i} exp(-||x_i - x_k||^2)/2\sigma_i^2} \qquad q_{j|i} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq i} exp(-||y_i - y_k||^2)}$$

We expect p and q to be the same -> small distance

Loss function

All points i

KL computes over j

$$\sum_i D_{KL}(p_i||q_i)$$

$$D_{KL}(P||Q) = \sum_j P(j) log \frac{P(j)}{Q(j)}$$

Note P can be considered as the weight for the distance

Where p is large but q is small -> large penalty

p is small but q is large -> small penalty

D(p||q) focuses on local structure in p

What are we minimizing wrt?

How to minimize loss?

From SNE to t-SNE

Symmetric density

t-distributed

Symmetric with joint probability

$$D_{KL}(P||Q) = \sum_i P(i) log \frac{P(i)}{Q(i)}$$

$p_{j|i} \neq p_{i|j}$

A point i far away from everything will have small $p_{j|i}$

- Location of points in q no longer matter

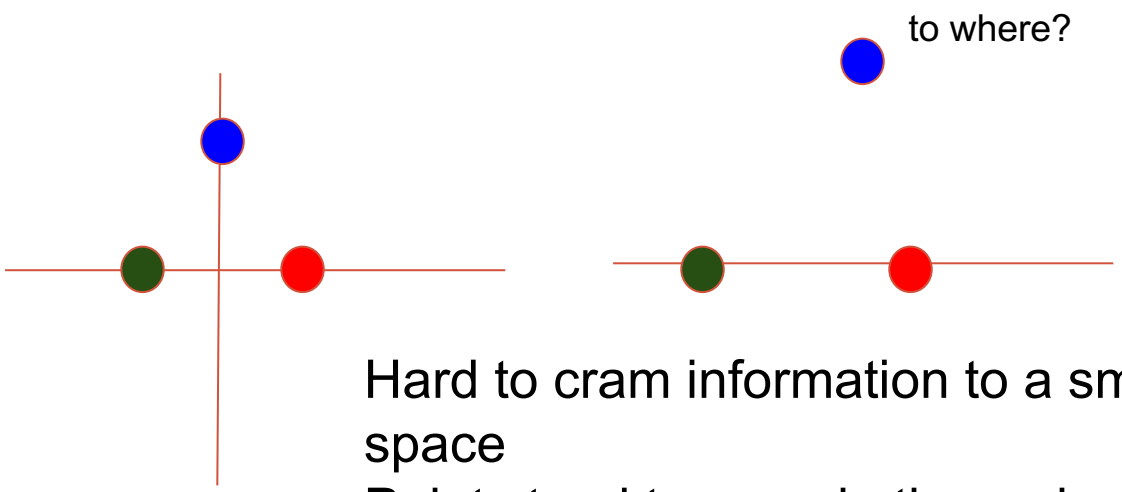
Create "joint probability" $p_{ij} = p_{ji} = (p_{i|j} + p_{j|i})/2$

- Each data point will contribute to the loss

Use instead of conditional probability in KL divergence

t-distributed

- “Crowding problem”
- In N dimension, you can have N+1 points at equal distance. But you cannot model this in smaller dimension



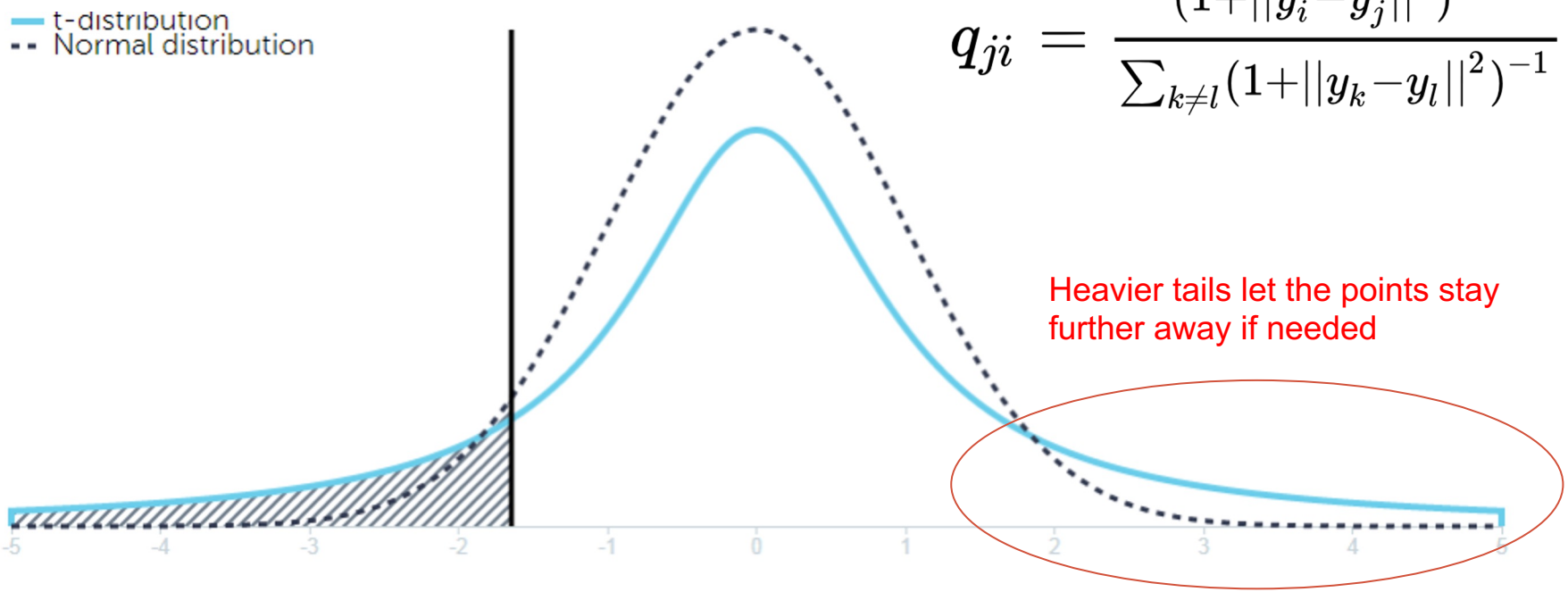
Hard to cram information to a smaller dimension space
Points tend to cram in the embedding space (no gap)

t-distributed

- Instead of Gaussian for q we use student's t distribution

$$q_{ji} \propto (1 + ||y_i - y_j||^2)^{-1}$$

$$q_{ji} = \frac{(1+||y_i-y_j||^2)^{-1}}{\sum_{k \neq l}(1+||y_k-y_l||^2)^{-1}}$$

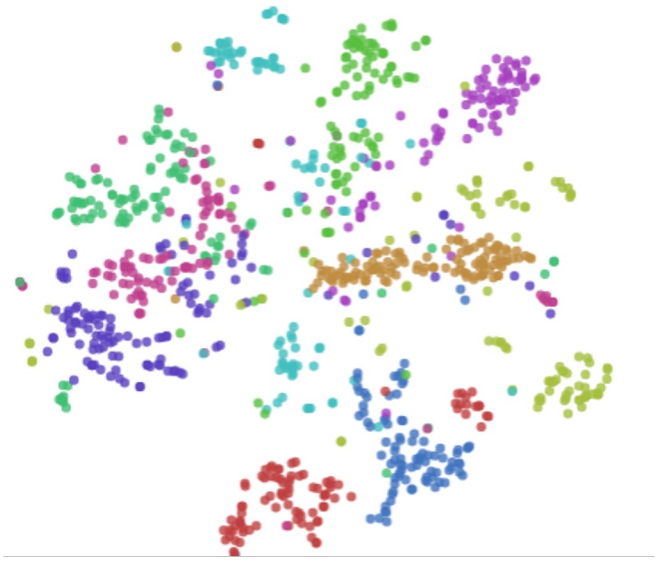


Heavier tails let the points stay further away if needed

http://rpsychologist.com/d3/tdist/

Crowding and t-SNE

Sammon's mapping
(crowding problem)



t-SNE

http://colah.github.io/posts/2014-10-Visualizing-MNIST/

Variance

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2)/2\sigma_i^2}{\sum_{k \neq i} exp(-||x_i - x_k||^2)/2\sigma_i^2}$$

How to set the variance of our original space?

A single variance for all points is not ideal.

- Want small variance for dense parts
- Want big variance for sparse parts

Set variance by amount of neighbors you want!

How to quantify amount of neighbors?

Perplexity

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2)/2\sigma_i^2}{\sum_{k \neq i} exp(-||x_i - x_k||^2)/2\sigma_i^2}$$

$$Perp(P_i) = 2^{H(P_i)}$$

$$H(P_i) = -\sum_j p_{j|i} log_2 p_{j|i}$$ Entropy

Perplexity of P_i represents effective amount of neighbors for the point i

Set Perp(P_i) then t-SNE algorithm searches for the corresponding variance

Typical values for perplexity 5 to 50

t-SNE summary

Goal: preserves local neighbors

Gradient-based -> need multiple runs to see the best

Two parameters: #iteration, perplexity

Does not learn a projection (unlike PCA, LDA, RP)

- If you have a new sample, you have to re-run the whole thing
- Used for visualization only

UMAP: Uniform Manifold Approximation and Projection

t-SNE loses structure of important feature dimension or structure (global vs local structure)

Draw upon “inspiration” from Manifold learning theory

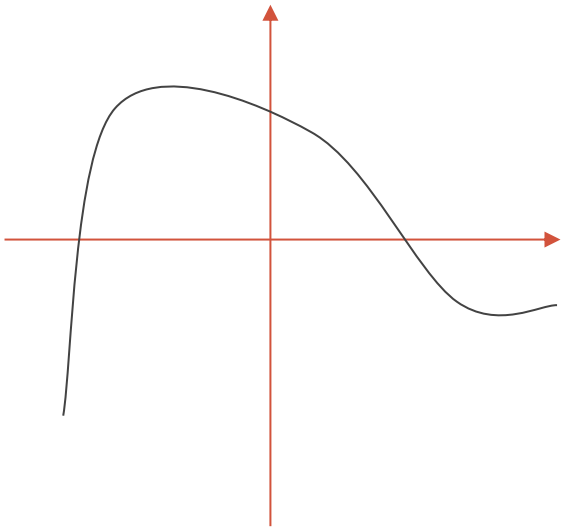
UMAP big picture

1. Define a new kind of probability for being neighbor
    a. Based on distance to neighbors
2. Do SGD just like t-SNE

Manifold

Manifold: a non-linear space but locally linear

Data can be represented in 2-d but lies in 1-d manifold



UMAP defines a manifold based on the neighbors of points

UMAP's distance function

Given fixed k (number of neighbors)

Let $X_1, \ldots, X_k$ be the k-nearest neighbors of X

The distance between X_i and X_j is

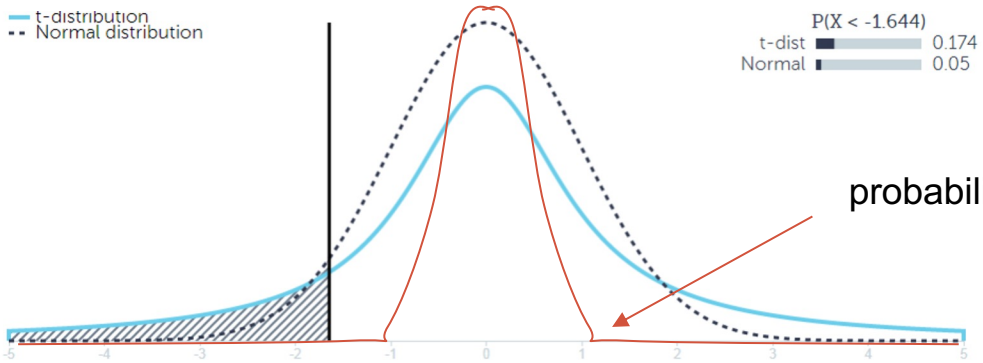
$$w_i(X_i, X_j) = \exp(-(d(X_i, X_j) - \rho_i)/\sigma_i)$$

diameter of the neighbors so it maxes at 1

Think of this w as giving "probability" that two points are connected

L2 distance in original space

distance to nearest neighbor so it starts from 0



probability becomes 0 after the k_{th} neighbor

UMAP's distance function

Given fixed k (number of neighbors)

Let $X_1, \ldots, X_k$ be the k-nearest neighbors of X

The distance between X_i and X_j is

$$w_i(X_i, X_j) = \exp(-(d(X_i, X_j) - \rho_i)/\sigma_i)$$

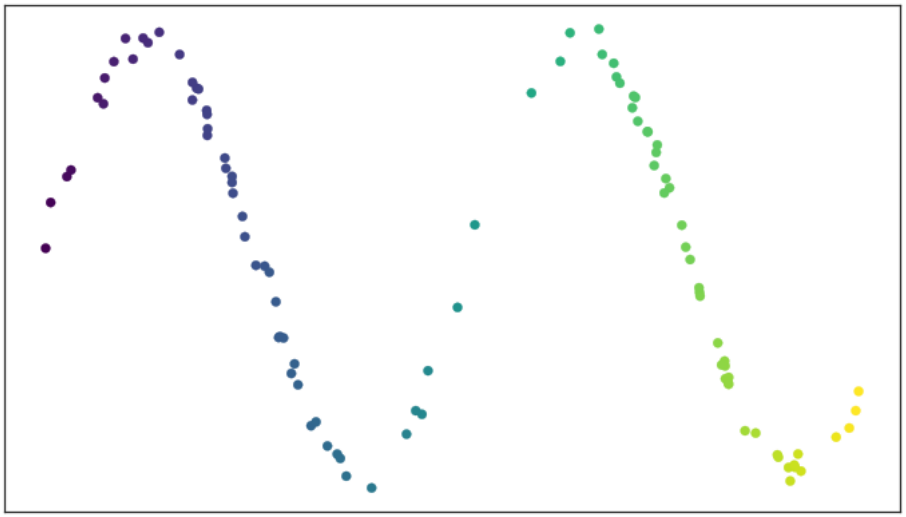
diameter of the neighbors so it maxes at 1

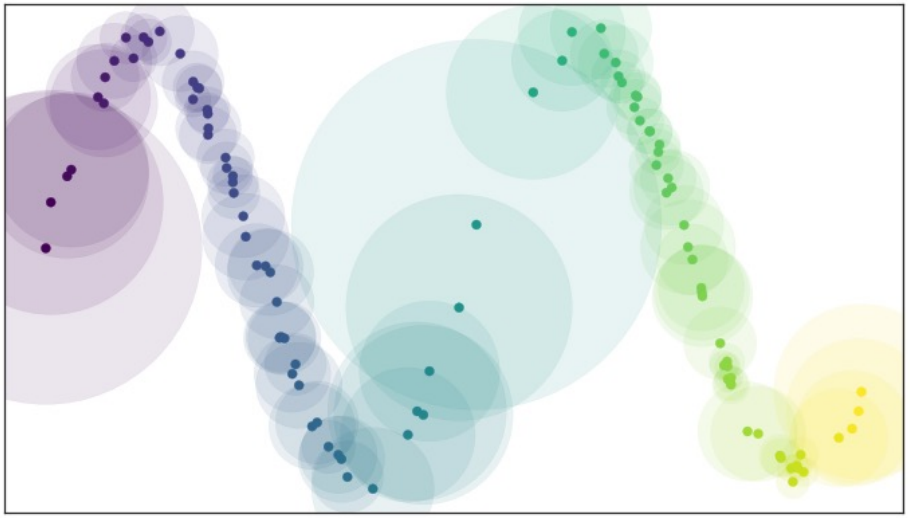
L2 distance in original space

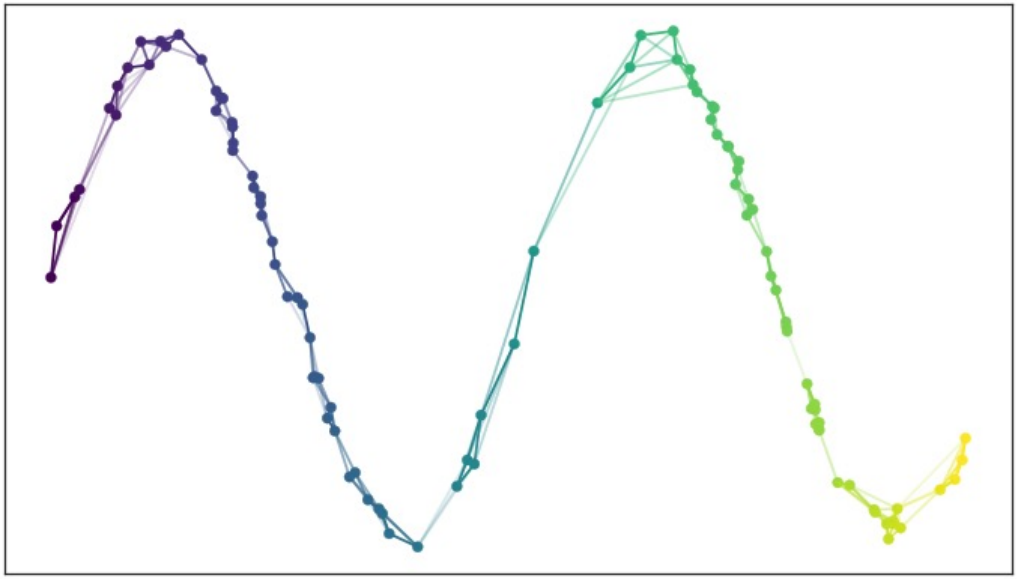
distance to nearest neighbor so it starts from 0

Make the distance symmetric by

$$w(X_i, X_j) = w_i(X_i, X_j) + w_j(X_j, X_i) - w_i(X_i, X_j)w_j(X_j, X_i)$$







https://umap-learn.readthedocs.io/en/latest/how_umap_works.html

UMAP optimization

From the weights minimize the cross-entropy between original and projected points (in terms of w)

$$\sum_{e \in E} w_h(e) \log\left(\frac{w_h(e)}{w_l(e)}\right) + (1 - w_h(e)) \log\left(\frac{1 - w_h(e)}{1 - w_l(e)}\right)$$

Distance in original space

All neighbors

Distance in new space

Similar to KL metric in t-SNE

Loss function

All points i

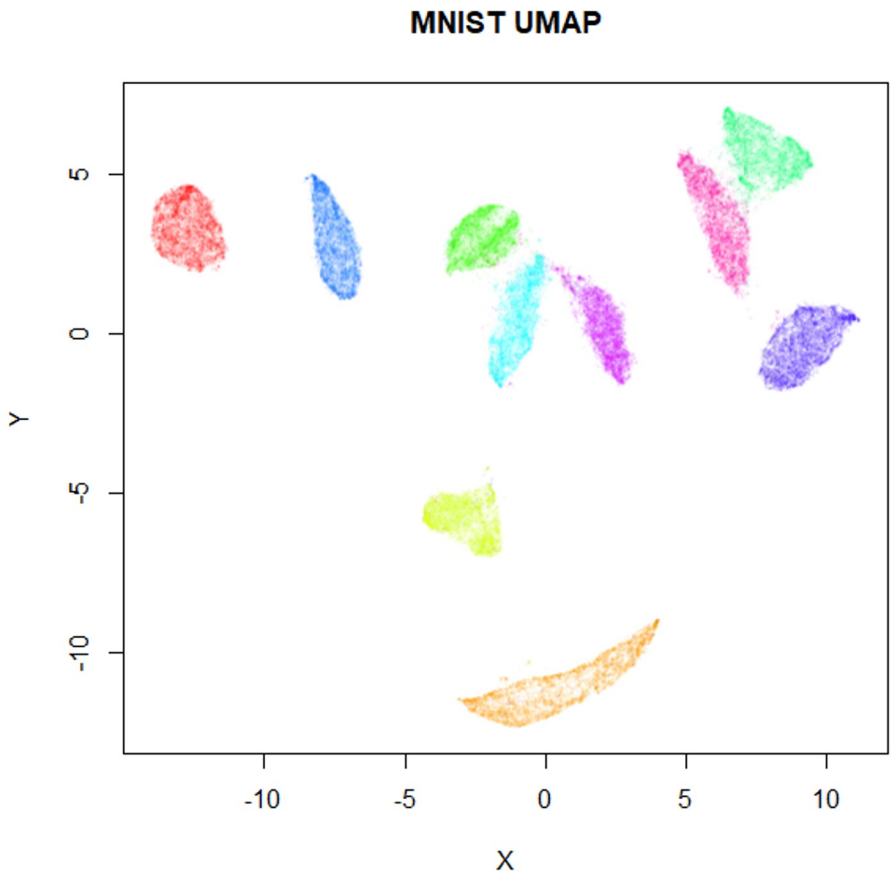
KL computes over j

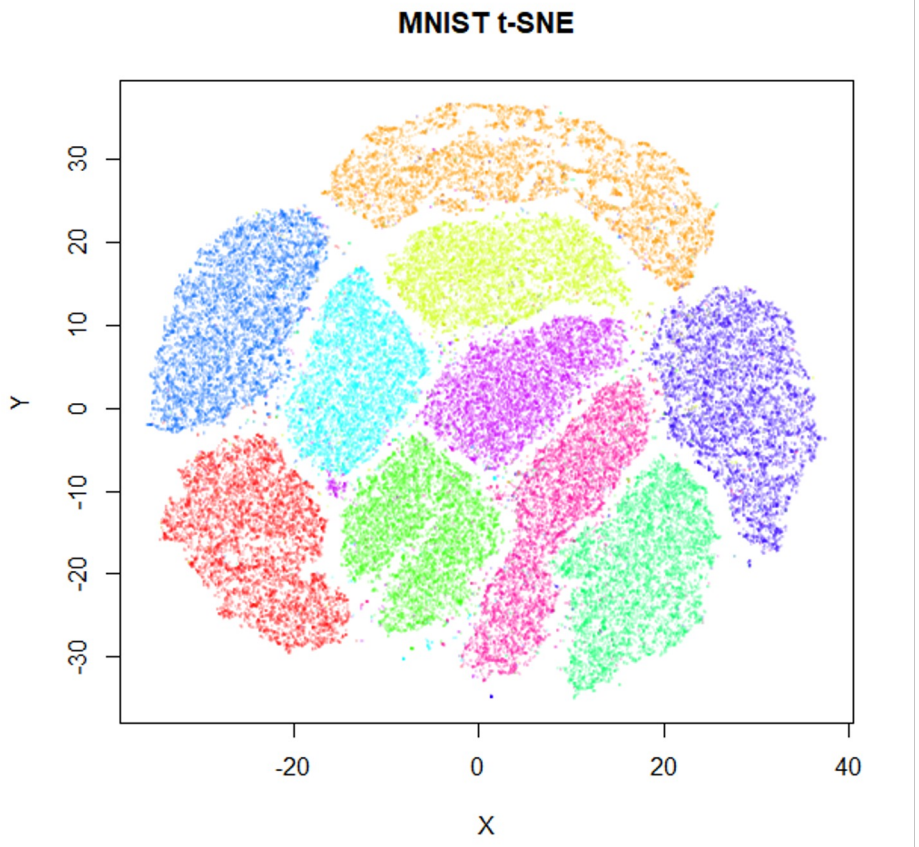
$$\sum_i D_{KL}(p_i || q_i)$$

$$D_{KL}(P||Q) = \sum_j P(j) log \frac{P(j)}{Q(j)}$$

Use gradient descent to optimize

MNIST



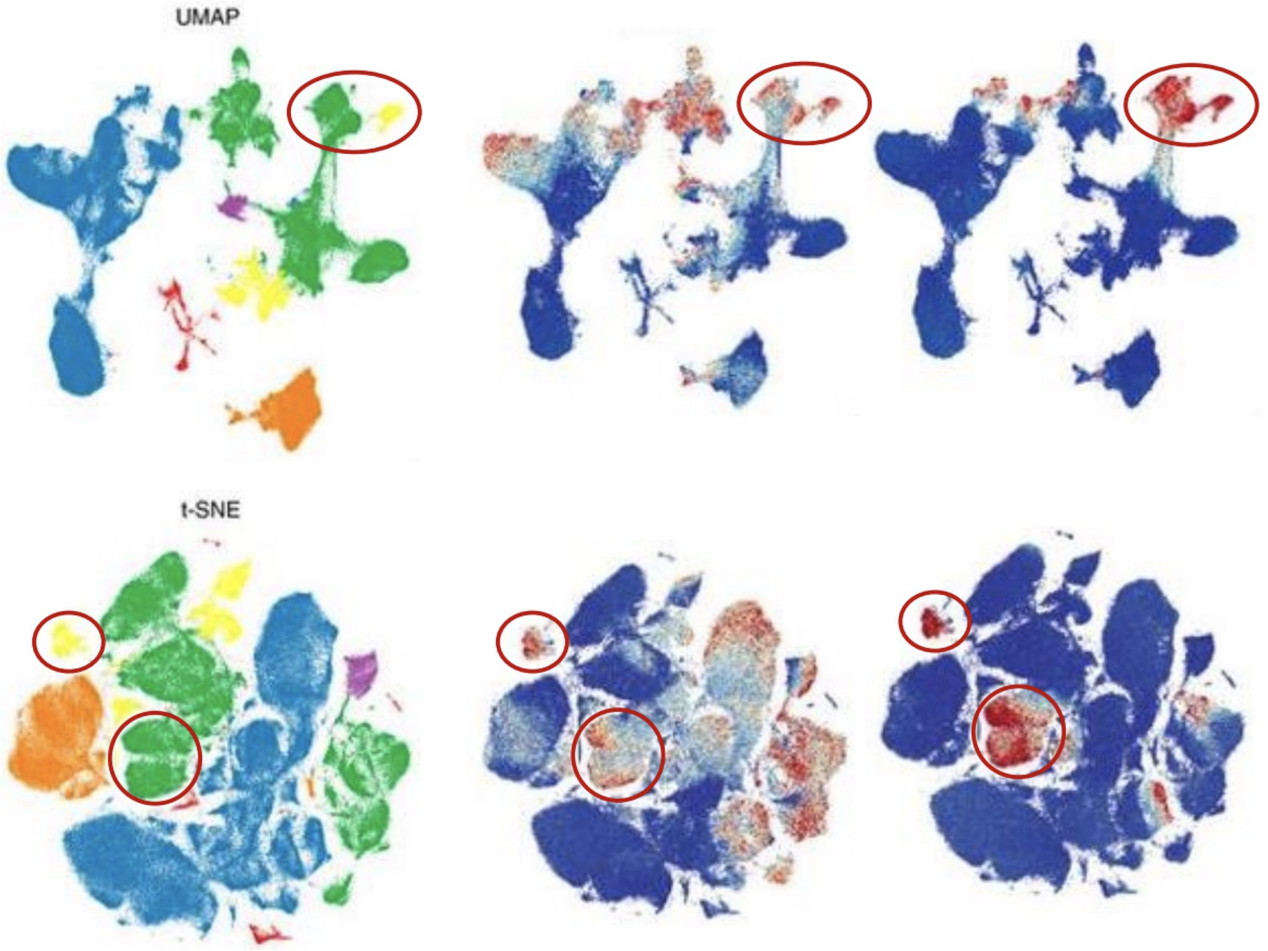


https://jlmelville.github.io/uwot/umap-examples.html

Dimensionality reduction for visualizing single-cell data using UMAP

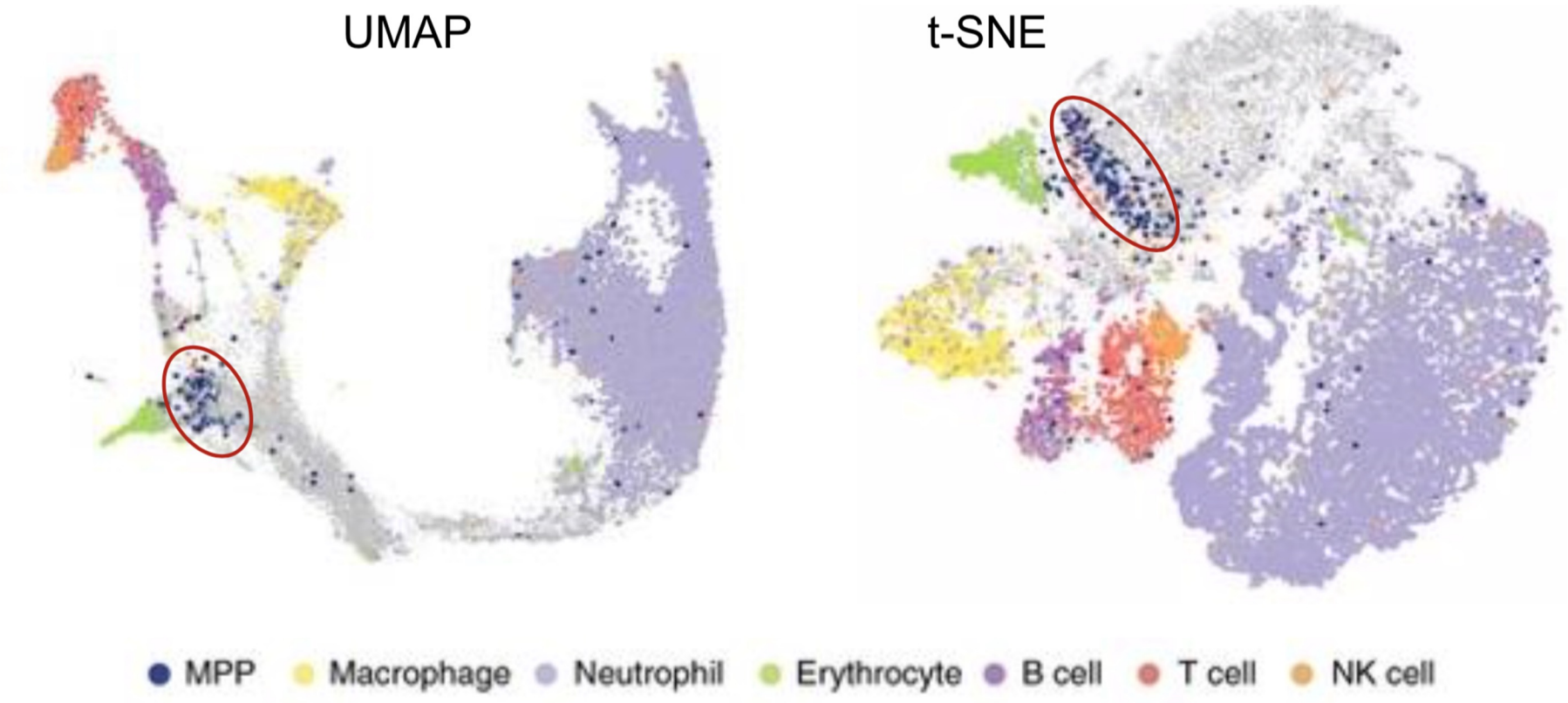
Analysis | Published: 03 December 2018

nature biotechnology

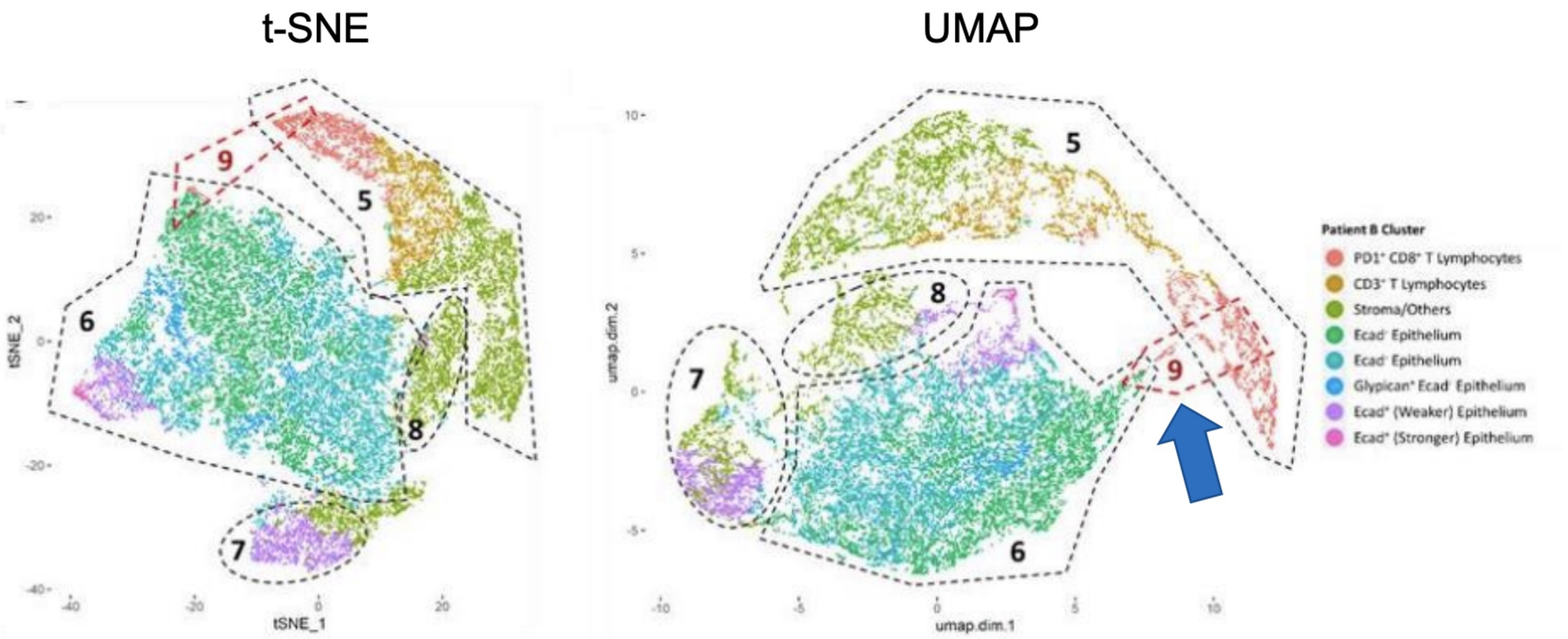


- UMAP can capture cross-class relationship

UMAP vs t-SNE on continuous classes

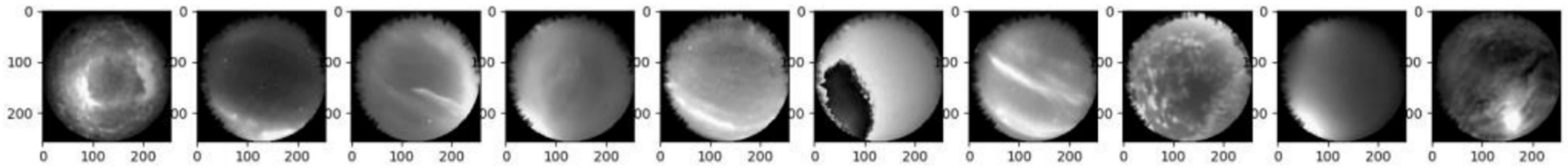


UMAP find continuous transition across classes

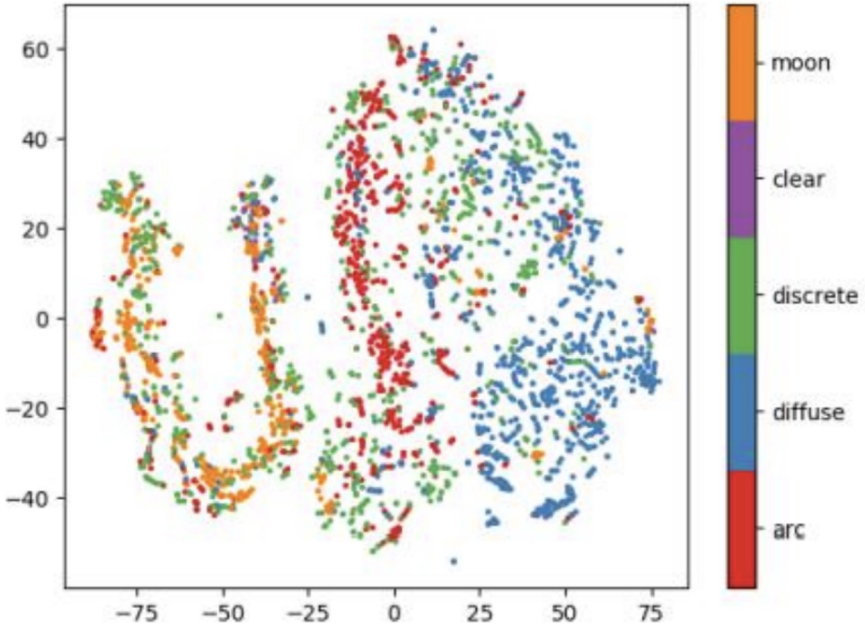


Correlation of Auroral Dynamics and GNSS Scintillation with an Autoencoder

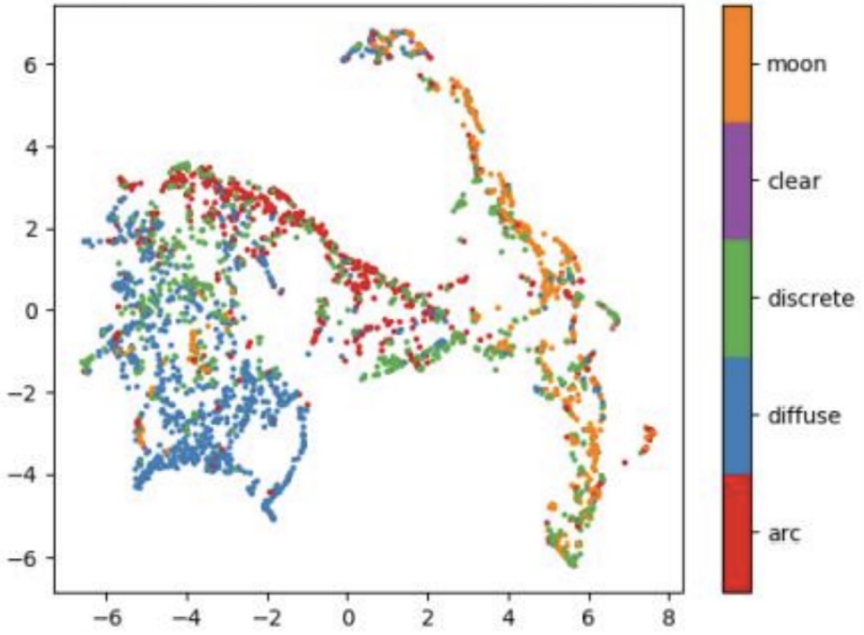
33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.



t-SNE

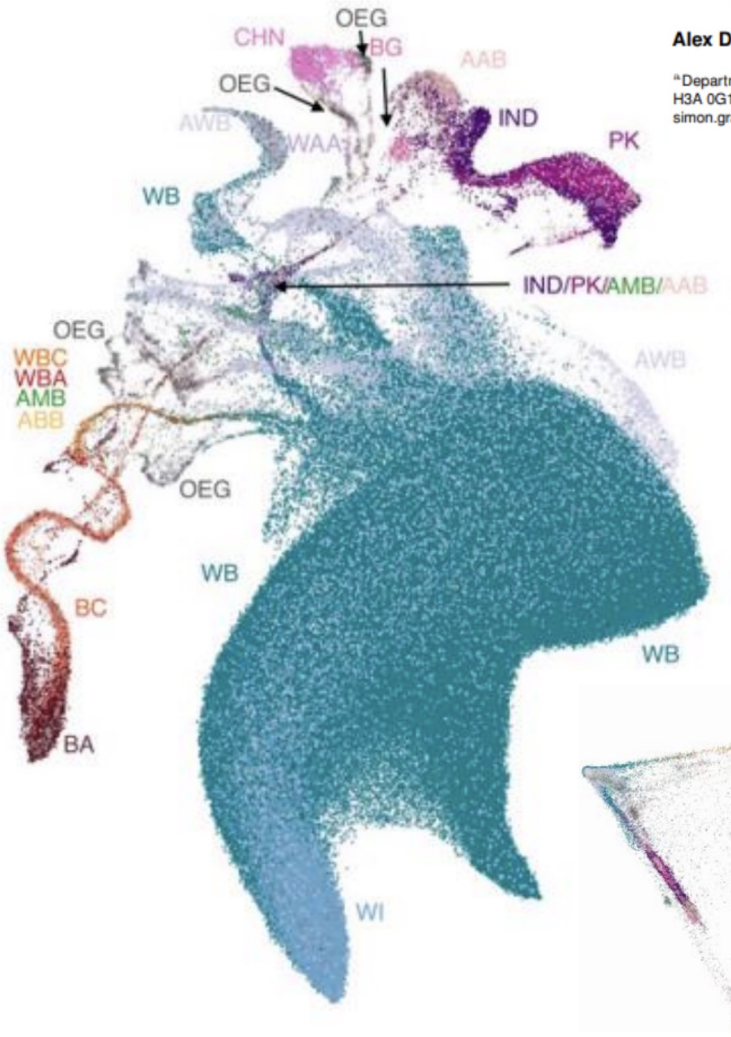


UMAP



UMAP vs t-SNE on population genomics

UMAP

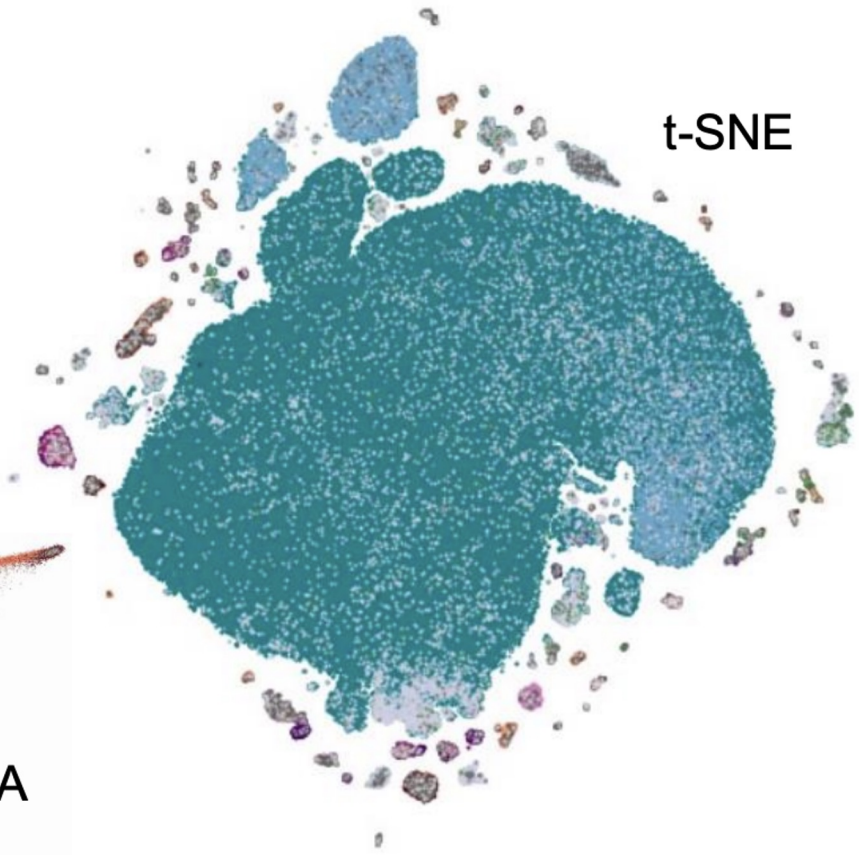


PCA

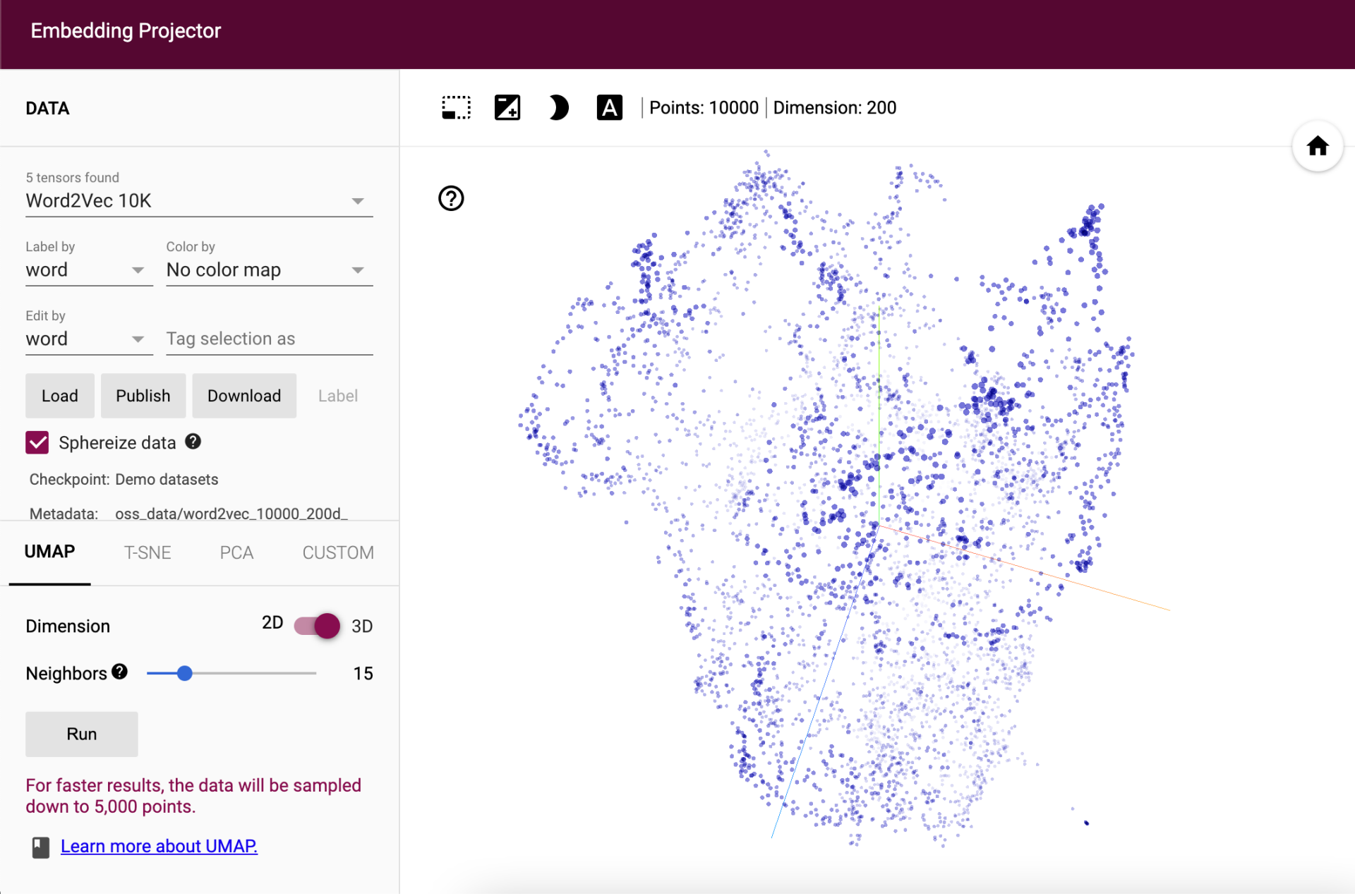
Revealing multi-scale population structure in large cohorts

Alex Diaz-Papkovich[a,b], **Luke Anderson-Trocmé**[b,c], **and Simon Gravel**[b,c,1]

[a]Department of Quantitative Life Sciences, McGill University, Montreal, QC, H3A 0G1 Canada; [b]McGill University and Genome Quebec Innovation Centre, Montreal, QC, H3A 0G1, Canada; [c]Department of Human Genetics, McGill University, Montreal, QC, H3A 0G1, Canada. [1]To whom correspondence should be addressed. E-mail: simon.gravel@mcgill.ca



t-SNE

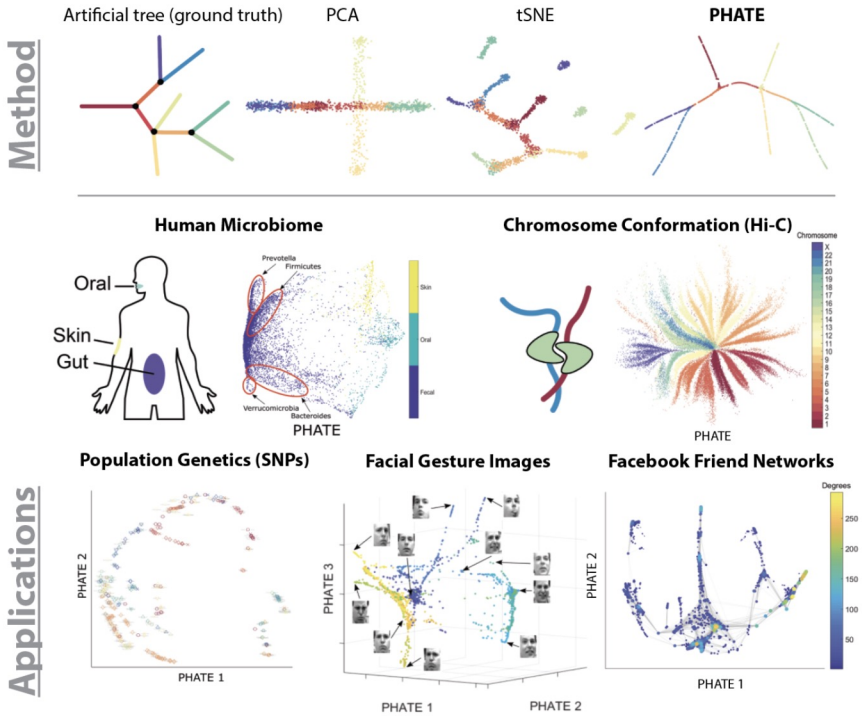


https://projector.tensorflow.org/

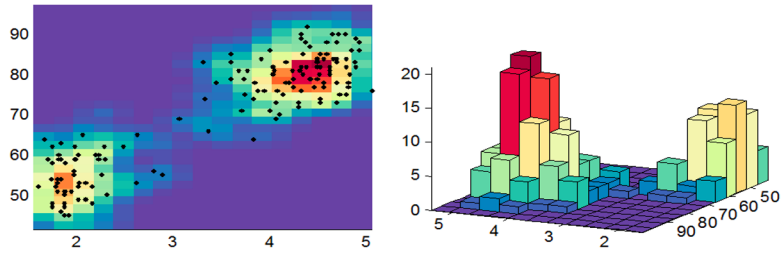
Other improvements

PHATE (https://www.krishnaswamylab.org/projects/phate)

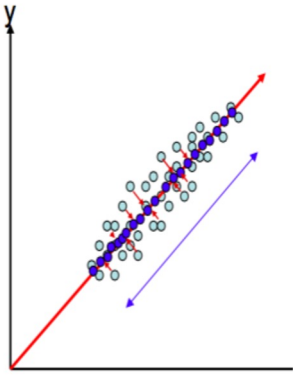
Learn local structure/distance (just like UMAP) and global structures (using diffusion maps to traverse the graph). Probably better, but not popular yet.

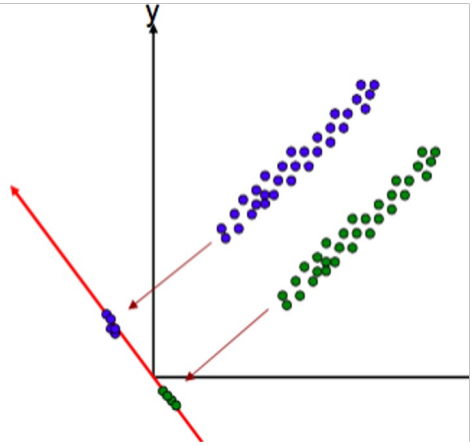


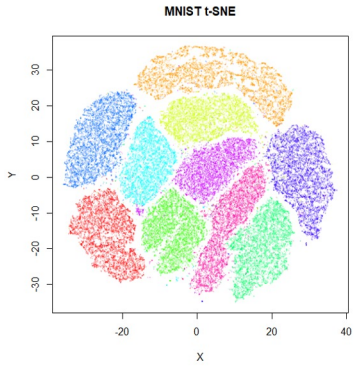
Summary



- Curse of dimensionality
- PCA
- LDA
  - PCA+LDA
- Random projection
- tSNE
- Homework







- Next time SVM