

DIMENSIONALITY REDUCTION AND VISUALIZATION

Loose ends from HW2

- **Hyper-parameters**

- something you decide
- bin size = ? Parametric or non-parametric? number of parameter
- Tuned on validation set

- **Parameters** $\max_{\theta} L(\theta | X)$

- something that is learned by the algorithm \xrightarrow{GD}
- weights in linear regression, probability in histogram bins, Gaussian means $p(X|w_1)$ $p(w_1)$ $p(w_2)$

- A majority class baseline

- Powerful if one class dominates. Often happens in real life
- Recognizer becomes biased towards the majority class (the prior term)
- How to deal with this? ↗

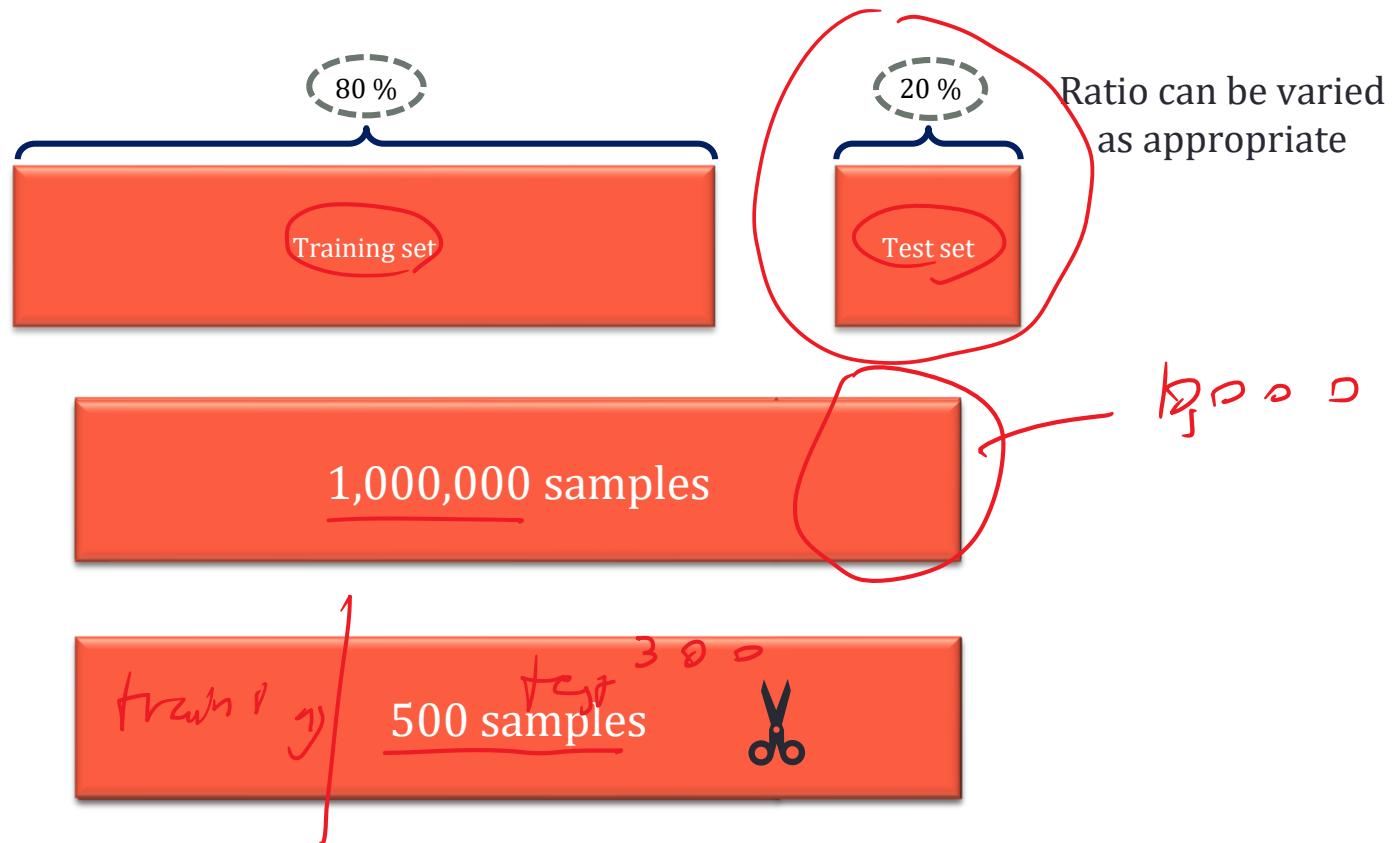


Splitting data

Data



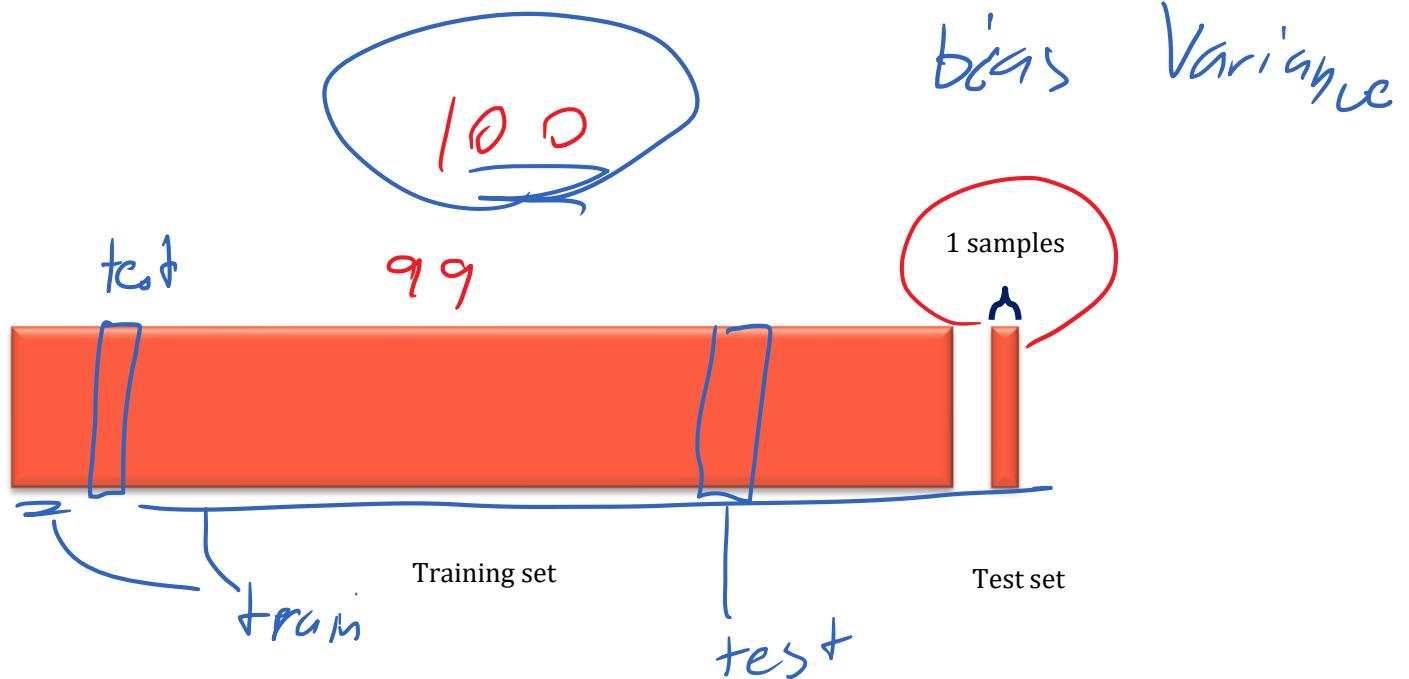
Simple train-test split



Stratified splitting – tries to keep the distribution in the training and test the same

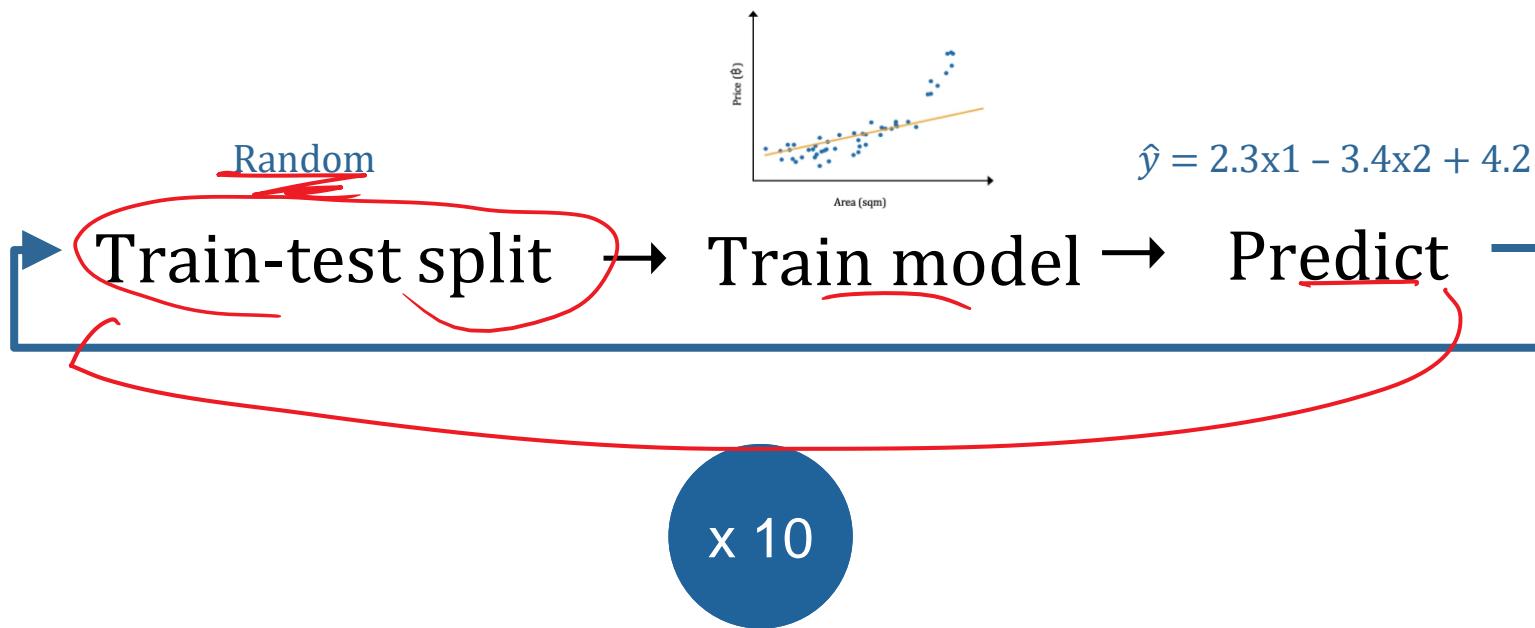
Leave-one-out

Expected error

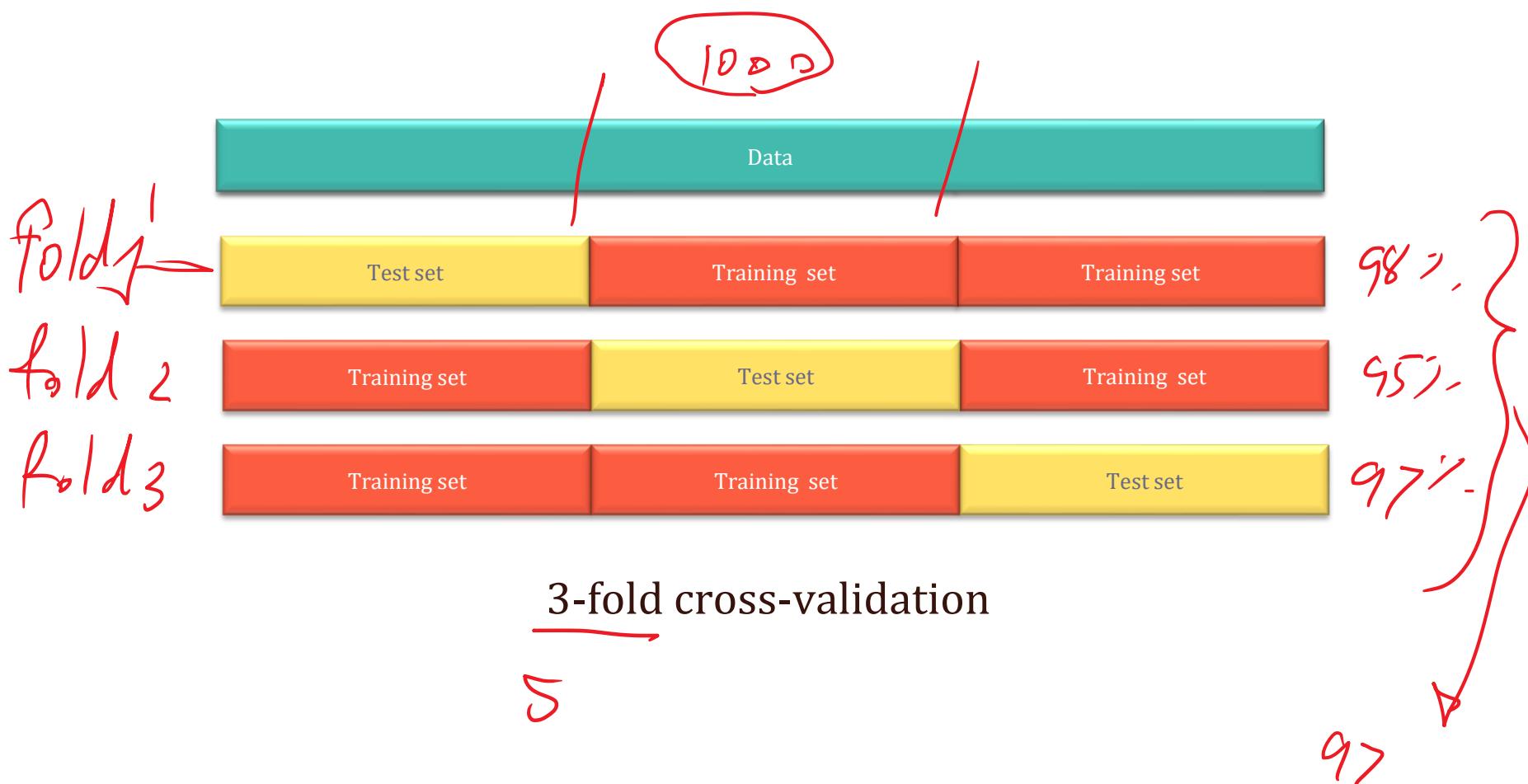


Multiple train-test split

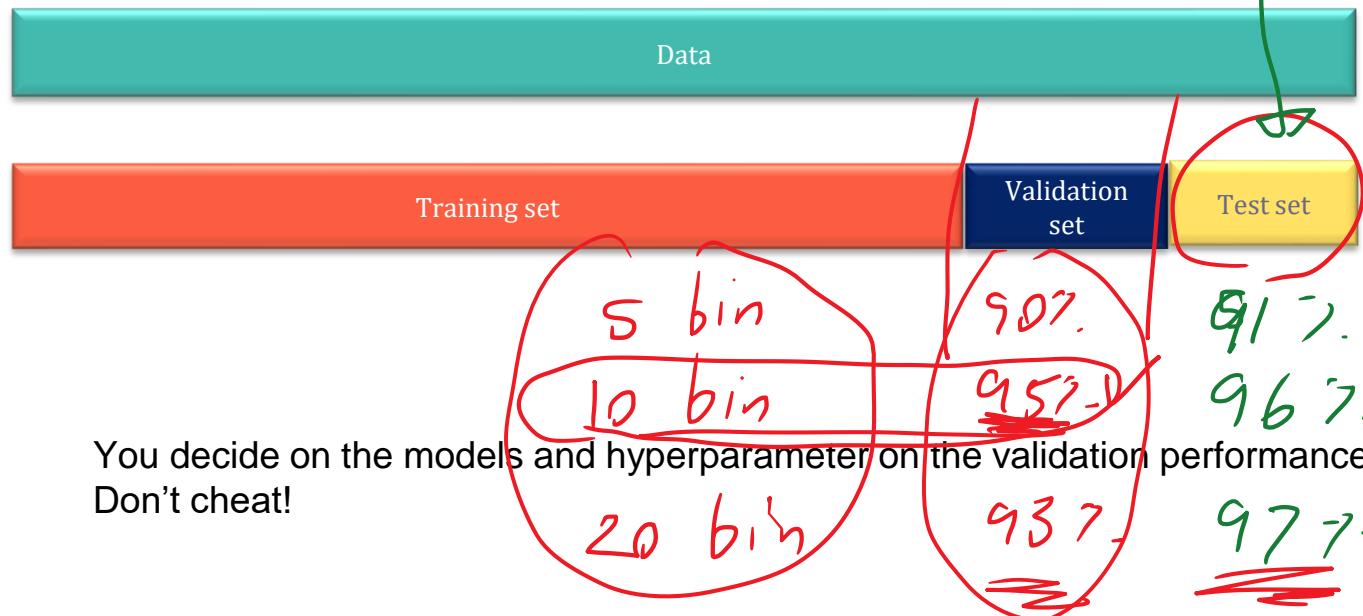
Estimate the true performance (expected performance)



Cross-validation (CV)

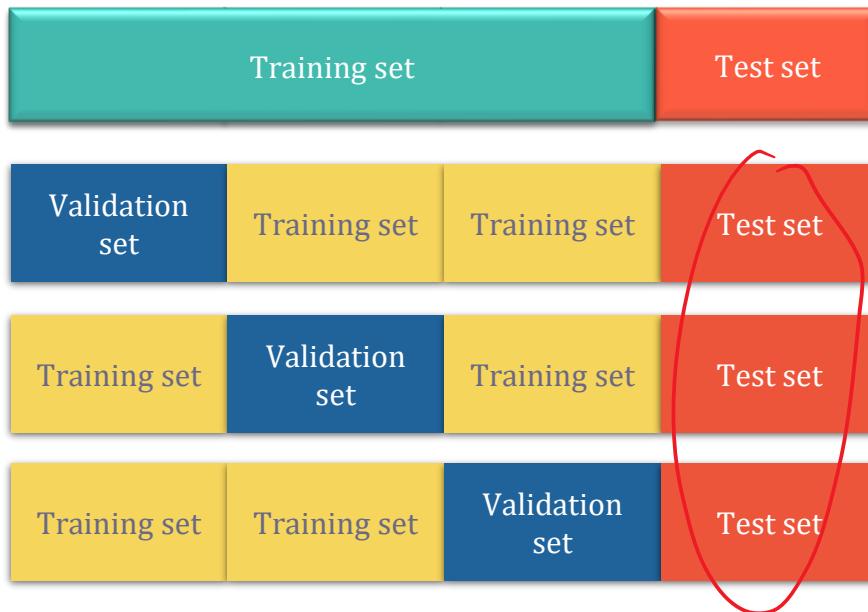


Estimating hyper-parameters

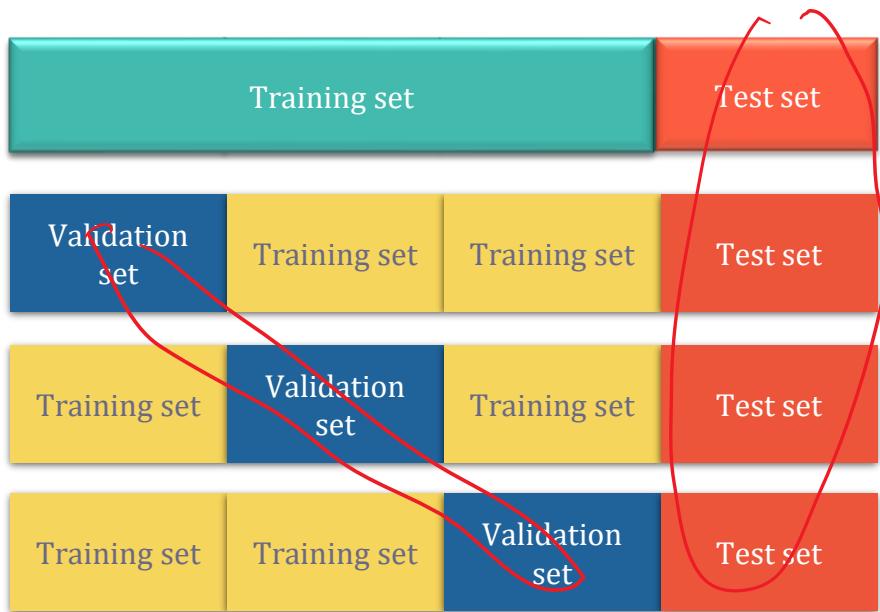


CV with validation set

$b_{in} \geq 80$



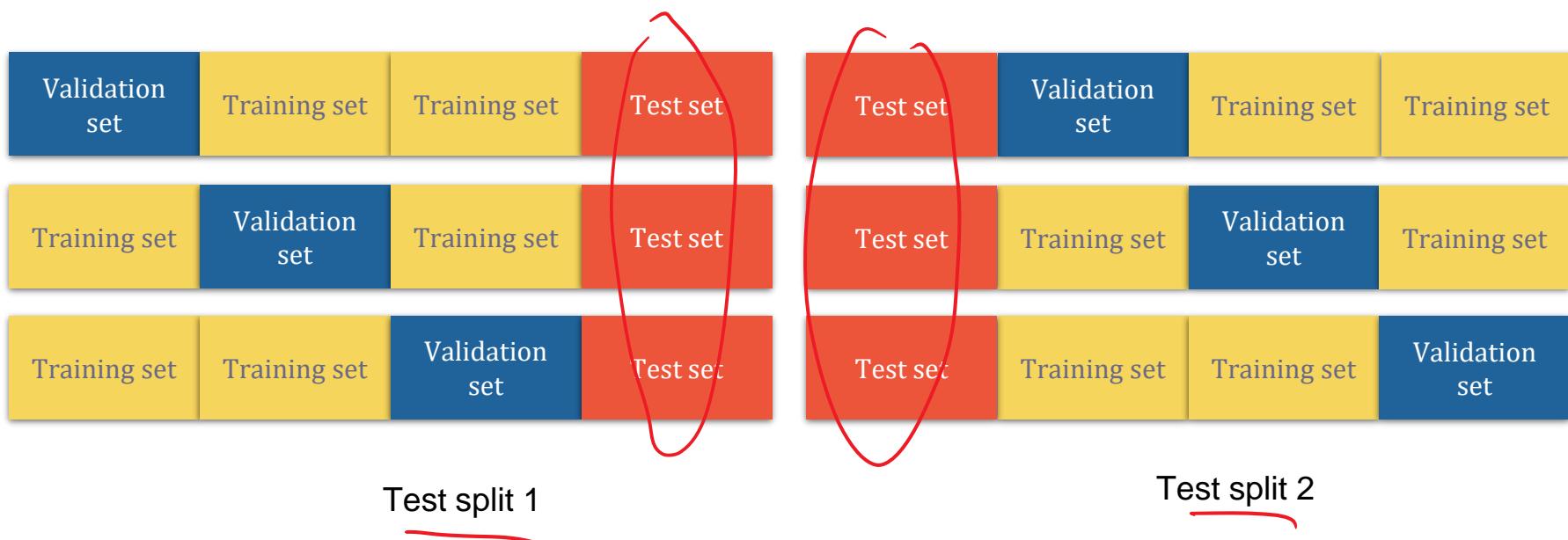
Estimating true performance of our pipeline



Freeze the test set
Touch the test set as less as possible

Nested CV

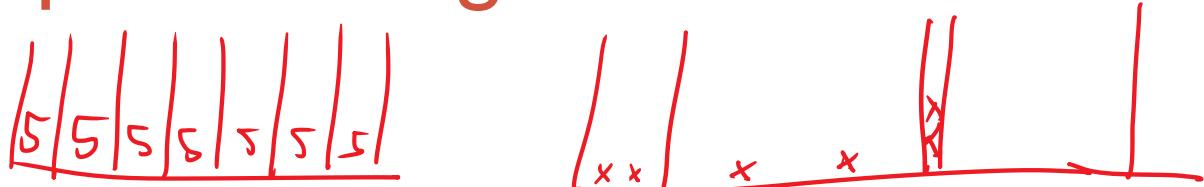
If the fixed test set is too small to give a reliable estimate, use nested CV.



Congratulations on your first attempt on (almost) re-implementing a research paper!

RESEARCH ARTICLE

OPEN ACCESS



Mining housekeeping genes with a Naive Bayes classifier

Luna De Ferrari and Stuart Aitken

BMC Genomics 2006 7:277 | <https://doi.org/10.1186/1471-2164-7-277> | © De Ferrari and Aitken; licensee BioMed Central Ltd. 2006

Received: 28 February 2006 | Accepted: 30 October 2006 | Published: 30 October 2006

The main advantage of Bayesian classifiers is that they are probabilistic models, robust to real data noise and missing values. ~~The Naive Bayes classifier assumes independence of the attributes used in classification but it has been tested on several artificial and real data sets, showing good performances even when strong attribute dependences are present. In addition, the Naive Bayes classifier can outperform other powerful classifiers when the sample size is small. Using the words of Domingos and Pazzani: "In summary, [...] the Bayesian classifier has much broader applicability than previously thought. Since it also has advantages in terms of simplicity, learning speed, classification speed, storage space and incrementality its use should perhaps be considered more often."~~ [18].

Another trick to reduce 0 bins in histograms

Algorithms

Discretisation

The Weka algorithm used for filtering with Unsupervised discretisation involves separating the data in ranges using equal-frequency binning (histogram equalization) so that the same number of training example fall into each bin. No class information is taken into consideration [29]. For

Mixture models

$$p(x) = \sum_k p(k) \underbrace{p_k(x)}_{\text{independent model}}$$

- A mixture of models from the same distributions (but with different parameters)
- Different mixtures can come from different sub-class
 - Cat class
 - Siamese cats
 - Persian cats
- $p(k)$ is usually categorical (discrete classes)
- Usually the exact class for a sample point is unknown.
 - Latent variable

EM on GMM

- E-step

- Set soft labels: $w_{n,j}$ = probability that nth sample comes from jth mixture p
- Using Bayes rule
 - $p(k|x ; \mu, \sigma, \phi) = p(x|k ; \mu, \sigma, \phi) p(k; \mu, \sigma, \phi) / p(x; \mu, \sigma, \phi)$
 - $p(k|x ; \mu, \sigma, \phi) \propto p(x|k ; \mu, \sigma, \phi) p(k; \phi)$

$$p(k_n = j | x_n; \phi, \mu, \Sigma) = \frac{p(x_n; \mu_j, \sigma_j) p(k_n = j; \phi)}{\sum_l p(x_n; \mu_l, \sigma_l) p(k_n = l; \phi)}$$

EM on GMM

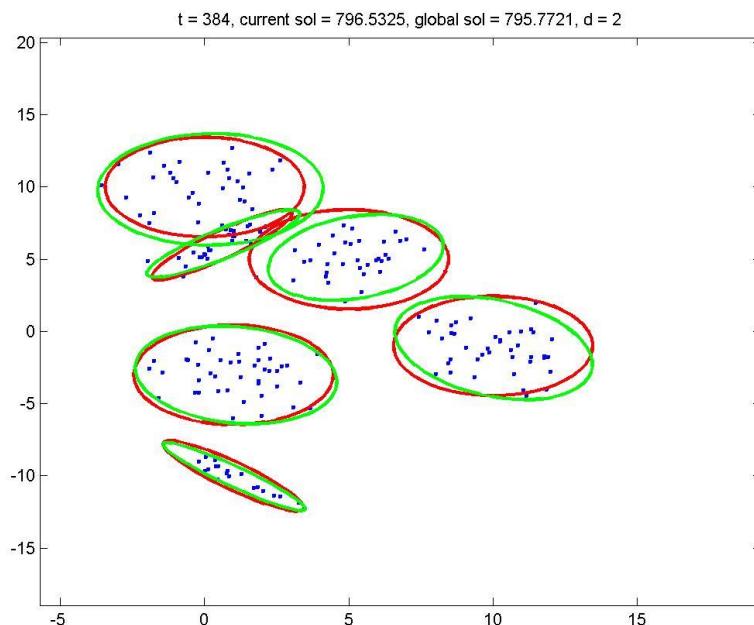
- M-step (soft labels)

$$\phi_j = \frac{1}{N} \sum_{n=1}^N w_{n,j}$$
$$\mu_j = \frac{\sum_{n=1}^N w_{n,j} x_n}{\sum_{n=1}^N w_{n,j}}$$

$$\sigma_j^2 = \frac{\sum_{n=1}^N w_{n,j} (x_n - \mu_j)^2}{\sum_{n=1}^N w_{n,j}}$$

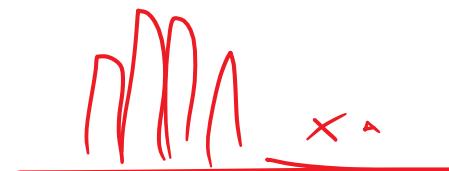
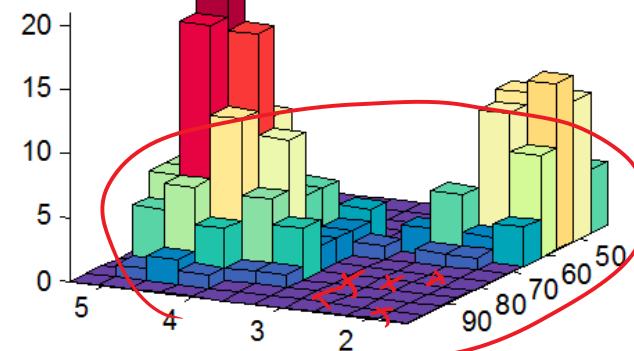
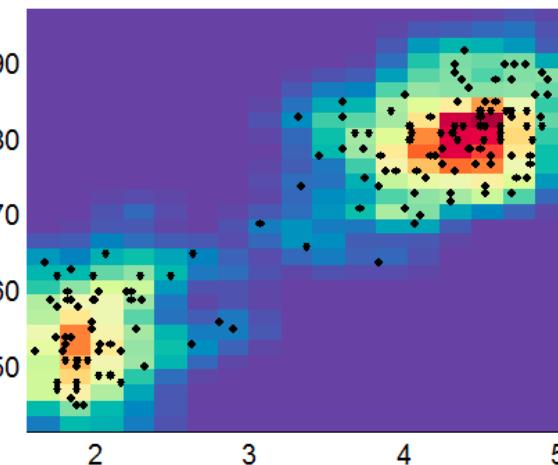
EM/GMM notes

- Converges to local maxima (maximizing likelihood)
 - Just like k-means, need to try different initialization points
- What if it's a multivariate Gaussian?
 - The grid search gets harder as the number of number of dimension grows



Histogram estimation in N-dimension

- Cut the space into N-dimensional cube
 - How many cubes are there?
 - Assume I want around 10 samples per cube to be able to estimate a nice distribution without overfitting. How many more samples do I need per one additional dimension?



The curse of dimensionality



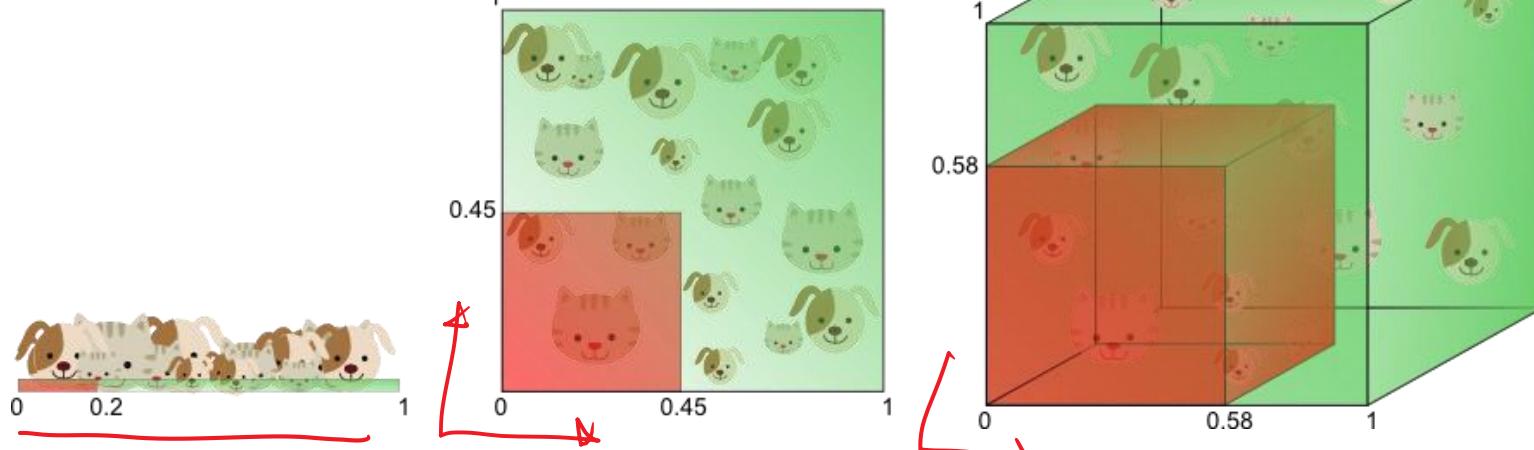
The Curse of Dimensionality

- 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 good estimation of the data

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

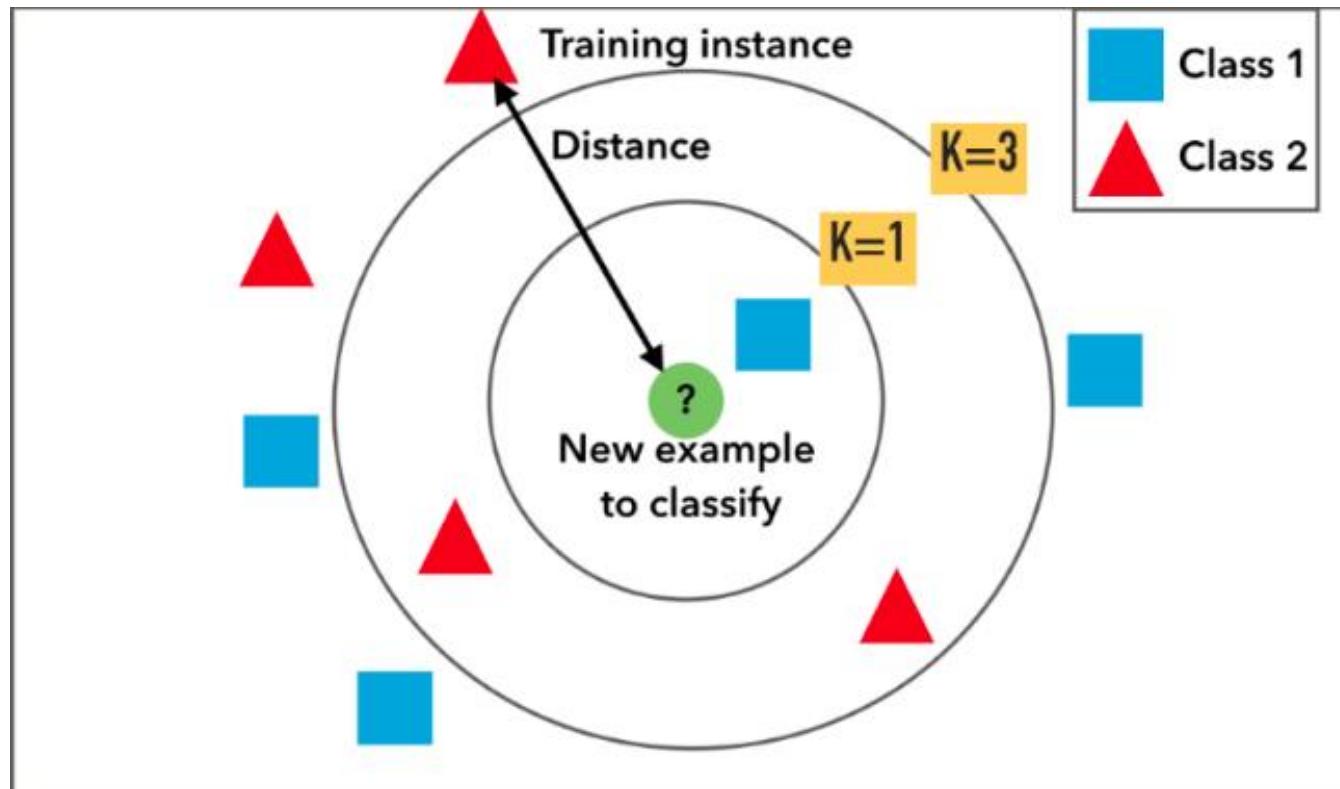
Day	X	Y	Z
1	W	L	M
2	C	M	M
3	W	H	M
4	W	M	H
5	C	M	L
6	W	M	L
7	C	L	H
8	W	H	L

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

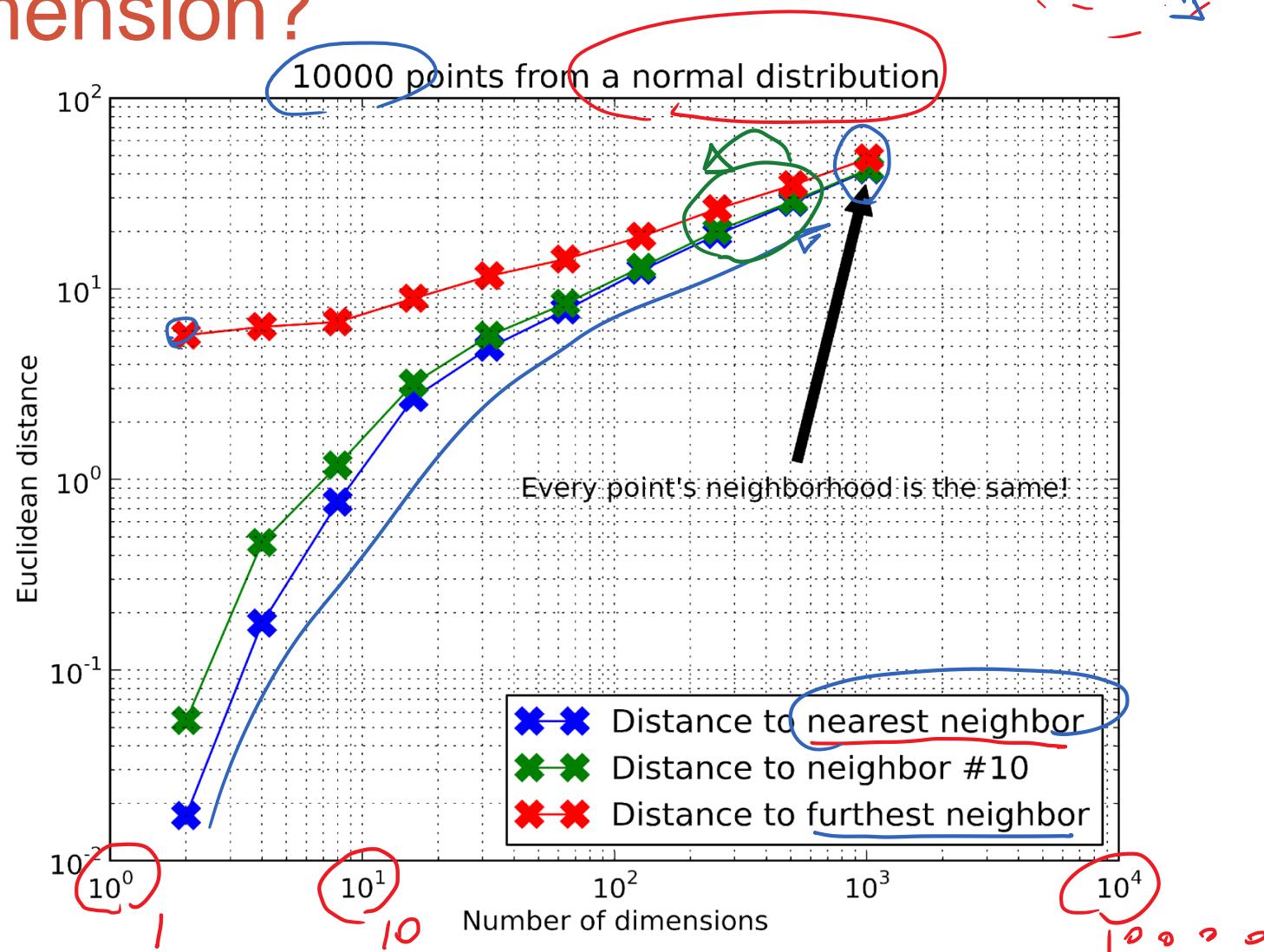
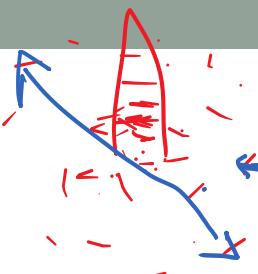


K-Nearest Neighbor Classifier

- Nearest neighbor is susceptible to label noise
- Use the k-nearest neighbors as the classification decision
 - Use majority vote



What's wrong with knn in high dimension?



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

$P(\text{特征} | \text{模型})$ ✓✓

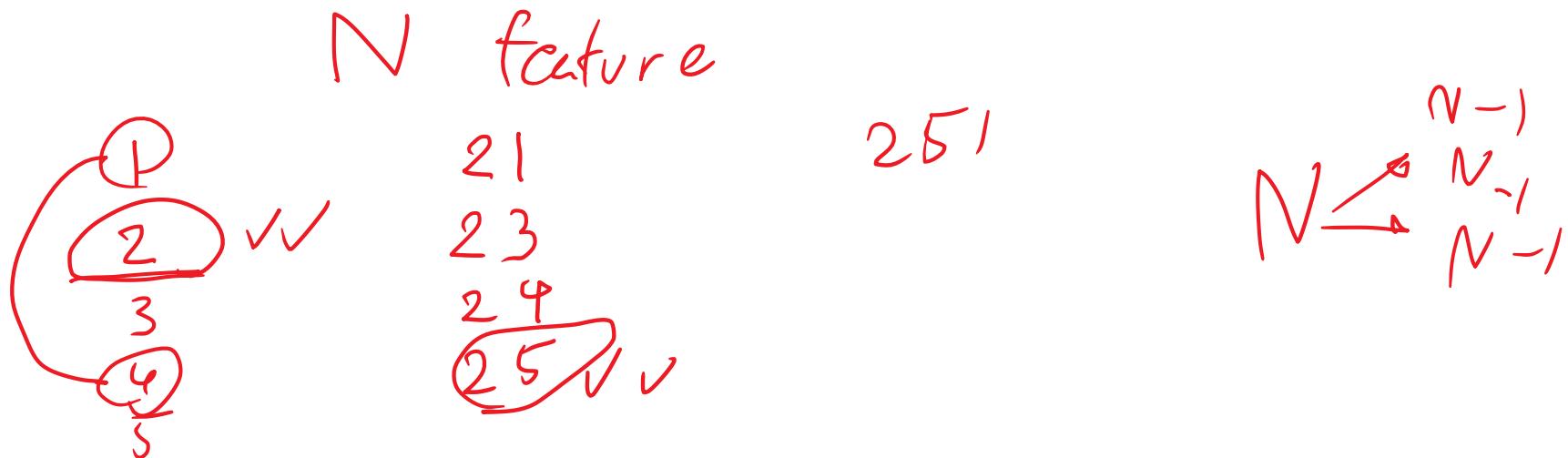
$P(\text{新特征} | \text{模型})$

- New features (a combination of old features)
- Usually more powerful
 - Harder to interpret the model

$P(\text{新特征} | \text{模型})$

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.

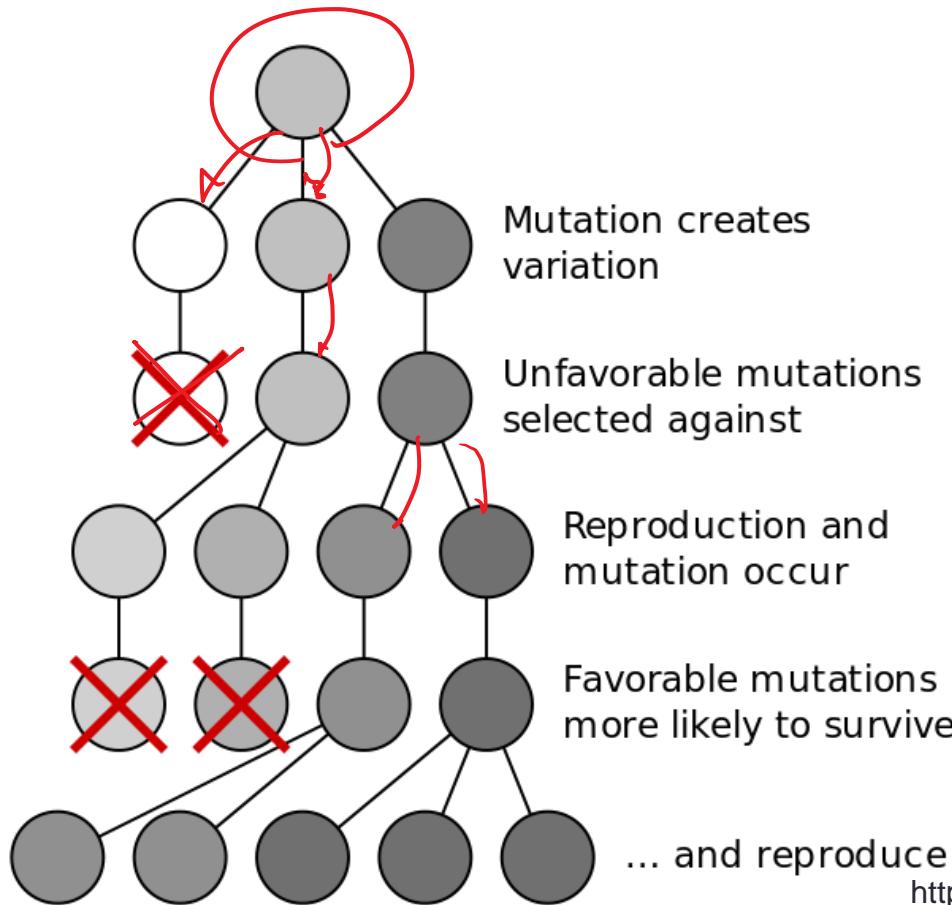


Feature selection

- Proper methods
 - Algorithm that handles high dimension well and do selection as a by product
 - Tree-based classifiers
 - Random forest
 - Adaboost
 - Genetic Algorithm

Genetic Algorithm

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

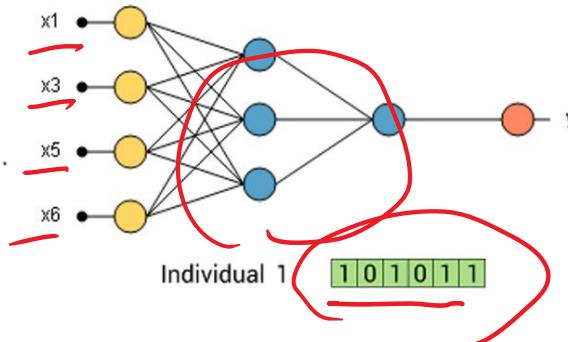


Genetic Algorithm

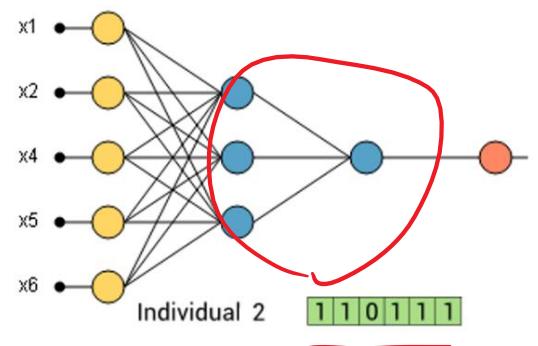
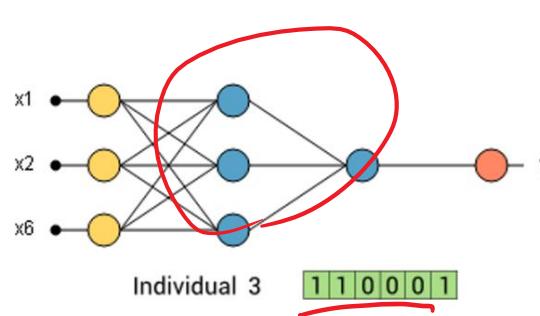
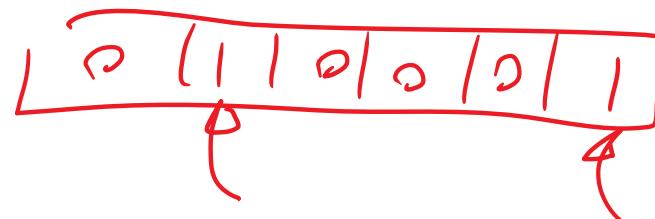
- Initialization
 - Create N classifiers, each using different subset of features
- Selection process
 - Rank the N classifiers according to some criterion, kill the lower half
- Crossover
 - The remaining classifier breeds offsprings by selecting traits from the parents
- Mutation
 - The offsprings 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



6 feature



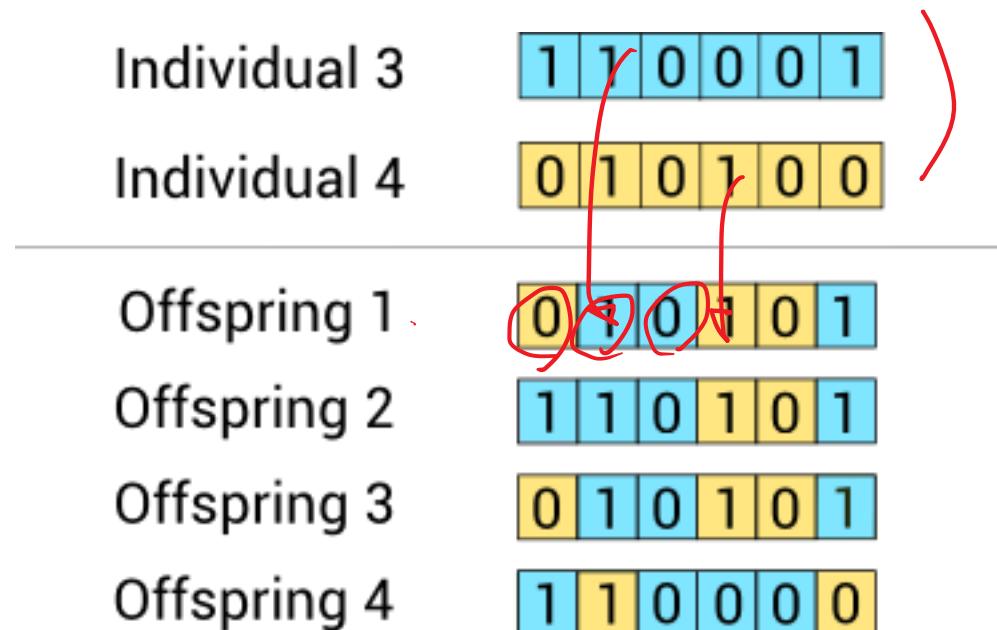
Selection process

- 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

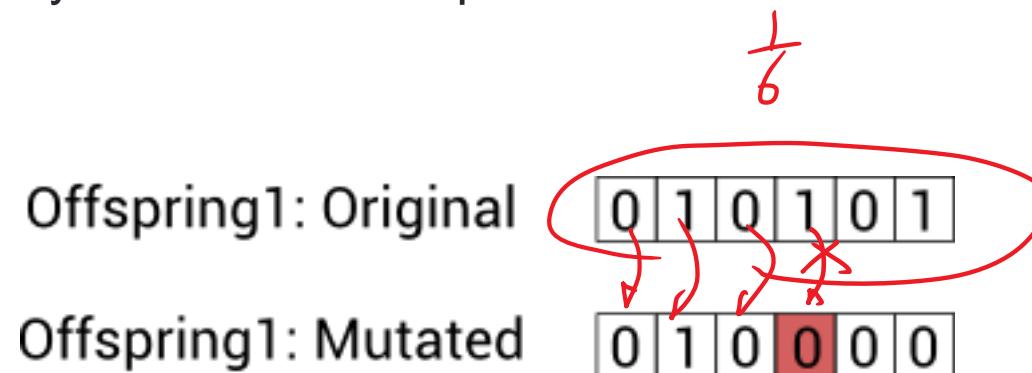
Crossover

- Breed offsprings by randomly select genes from parents



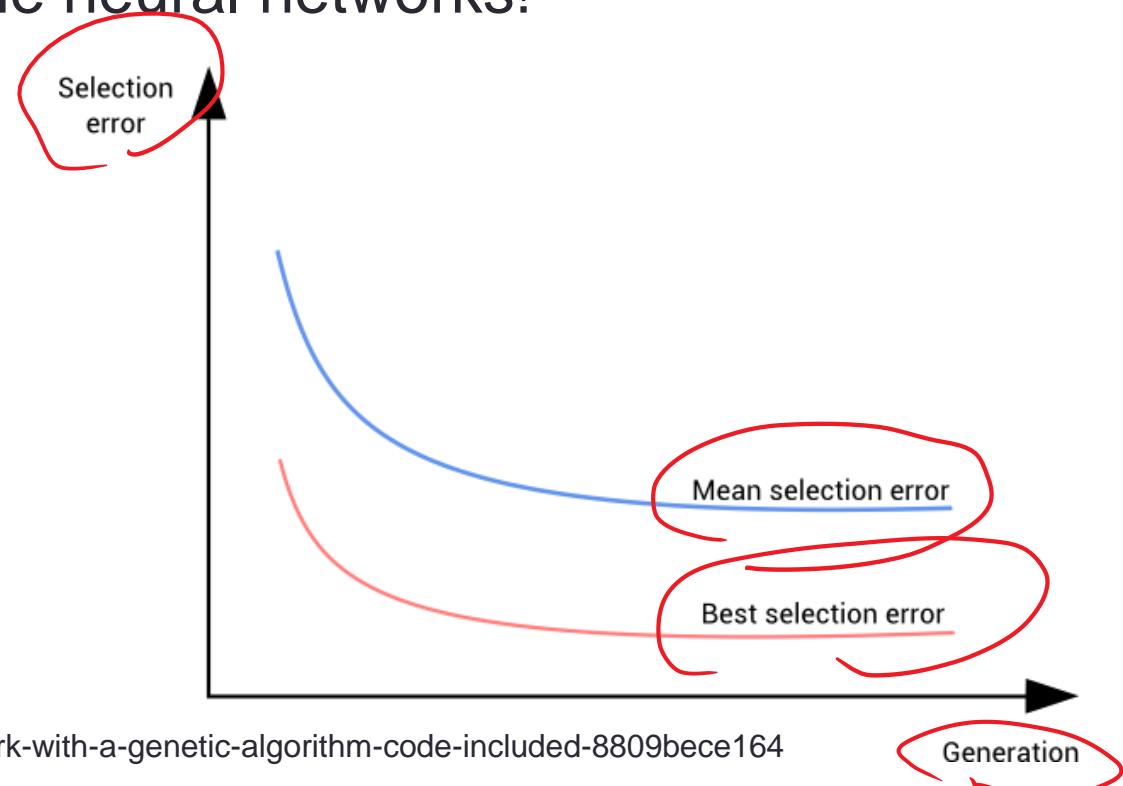
Mutation

- Offspring can mutate with some probability to introduce diversity
- Mutation rate is usually $\frac{1}{k}$ 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
- Can be used to tune neural networks!



Feature transformation

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

Linear Algebra Review

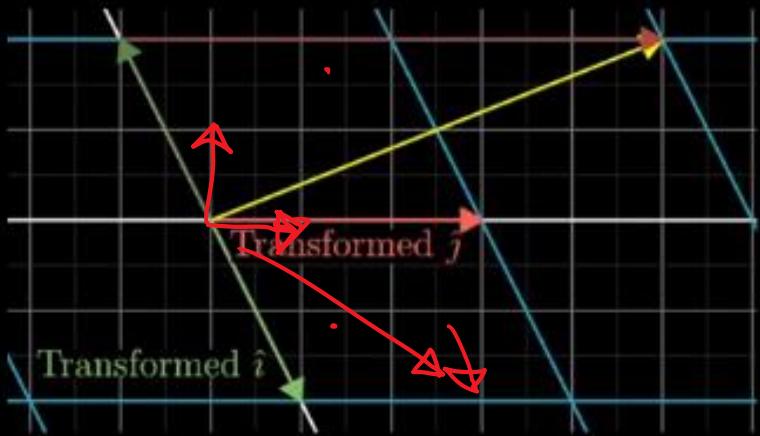
$$\begin{bmatrix} 3 \times 5 + 2 \times 7 \\ -2 \times 5 + 1 \times 7 \end{bmatrix}$$

- Matrix as a sequence of column vectors

“2x2 Matrix”

basis

$$\begin{bmatrix} 3 & 2 \\ -2 & 1 \end{bmatrix} \quad \begin{bmatrix} 5 \\ 7 \end{bmatrix}$$



$$5 \begin{bmatrix} 3 \\ -2 \end{bmatrix} + 7 \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Linear Algebra Review

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

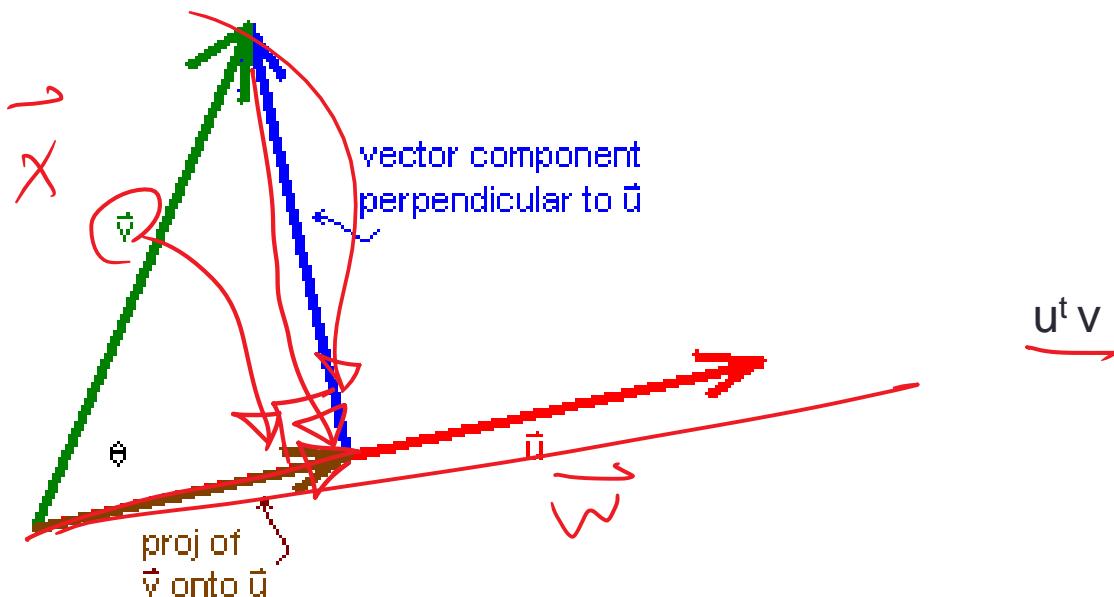
$$\sum A = Q D Q^{-1}$$

$$q_1 \quad q_2 \quad q_3$$

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



Covariance matrix

$$\Sigma \geq \Sigma^T$$

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

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

Annotations in red:

- Var X_1 circled in the top-left entry $E[(X_1 - \mu_1)(X_1 - \mu_1)]$.
- Var X_2 circled in the middle-right entry $E[(X_2 - \mu_2)(X_2 - \mu_2)]$.
- Var X_n circled in the bottom-right entry $E[(X_n - \mu_n)(X_n - \mu_n)]$.
- A red arrow points from the circled $E[(X_2 - \mu_2)(X_1 - \mu_1)]$ entry to the circled $E[(X_2 - \mu_2)(X_2 - \mu_2)]$ entry.

Positive semi-definite and eigen decomposition

$$\cancel{x^T \Sigma x \geq 0}$$

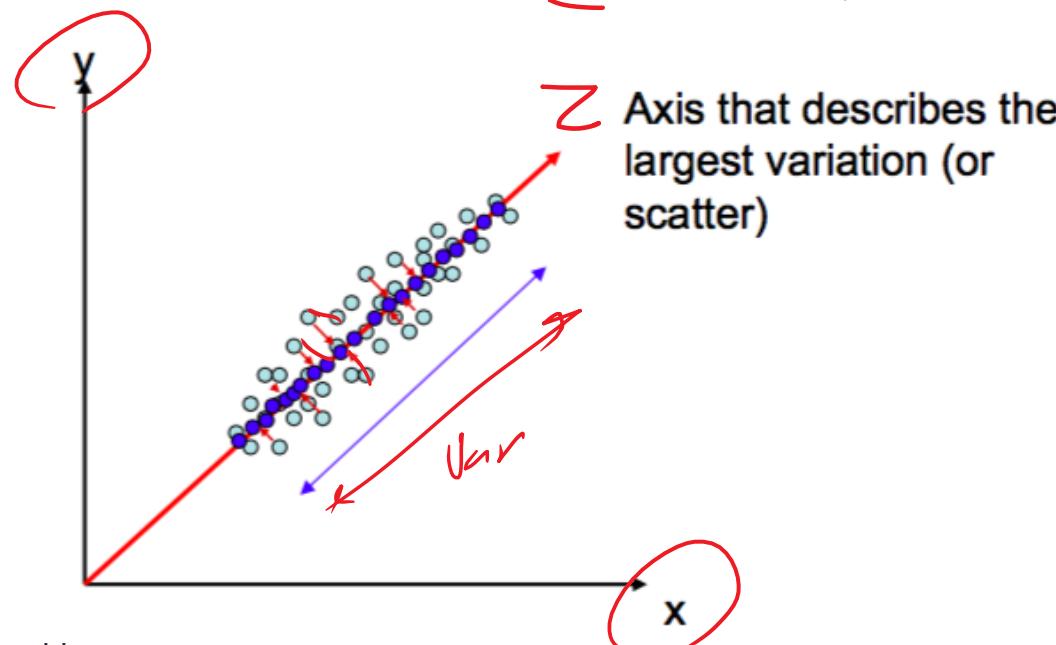
- Covariance matrix is positive semi-definite and symmetric.
- Eigenvalues are nonnegative, eigenvalues and vectors are real values, eigenvectors are mutually orthogonal.
- $q_i^T q_j = 0$ for $i \neq j$

Principal Component Analysis

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 describes maximum variation

$2 \rightarrow 1$



$$\mathbb{E}_{\overset{x}{\sim}}^{\varepsilon}[w^T x] = w^T \mathbb{E}_x[x] = w^T \mu_x$$

Formulation

- Maximize the variance after projection ie

$$\underset{w}{\operatorname{argmax}} \operatorname{Var}(w^T x) = \mathbb{E}[(w^T x - \mathbb{E}[w^T x])^2]$$

- Subject to w is a unit vector $\|w\| = 1$

$$= \mathbb{E}[(w^T x - w^T \mu_x)(w^T x - w^T \mu_x)]$$

$$= \mathbb{E}[(w^T x - w^T \mu_x)(x^T w - \mu_x^T w)]$$

$$= \mathbb{E}[w^T (x - \mu_x)(x^T - \mu_x^T) w]$$

$$= w^T \mathbb{E}[(x - \mu_x)(x - \mu_x)^T] w$$

$$= w^T \Sigma w$$

- $\Sigma w = \lambda w$ <- eigenvector

$$\begin{matrix} w - kx \\ \mu_x - kx \end{matrix}$$

$$\text{Var}(w^T x) = \lambda(\|w\| - 1)$$

$$\nabla_w w^T \Sigma w = \lambda(w^T w - 1)$$

$$\nabla_w \text{tr}[w^T \Sigma w] - \lambda \nabla_w \text{tr}[w^T w] + \lambda \nabla_w \text{tr}[1]$$

$$A = w^T \quad B = \Sigma \quad C = I \quad A^T = w$$

$$I w^T \Sigma + I w^T \Sigma = 2 w^T \Sigma$$

$$A = w^T \quad B = I \quad C = I \quad A^T = w$$

$$= 2 w^T$$

$$2 w^T \Sigma - 2 \lambda w^T = 0$$

$$w^T \Sigma = \lambda w^T$$

$$\boxed{\Sigma w} = \lambda w$$

vector w is eigenvector von Σ

Condition
 $n_o = n_{\max}$

Trace properties

$$1 \cdot \text{tr}(\mathbf{a}) = \mathbf{a}$$

$$2 \cdot \text{tr}A = \text{tr}A^T$$

$$3 \cdot \text{tr}(A+B) = \text{tr}A + \text{tr}B$$

$$4 \cdot \text{tr}(aA) = a\text{tr}(A)$$

$$5 \quad \nabla_A \text{tr}AB = B^T$$

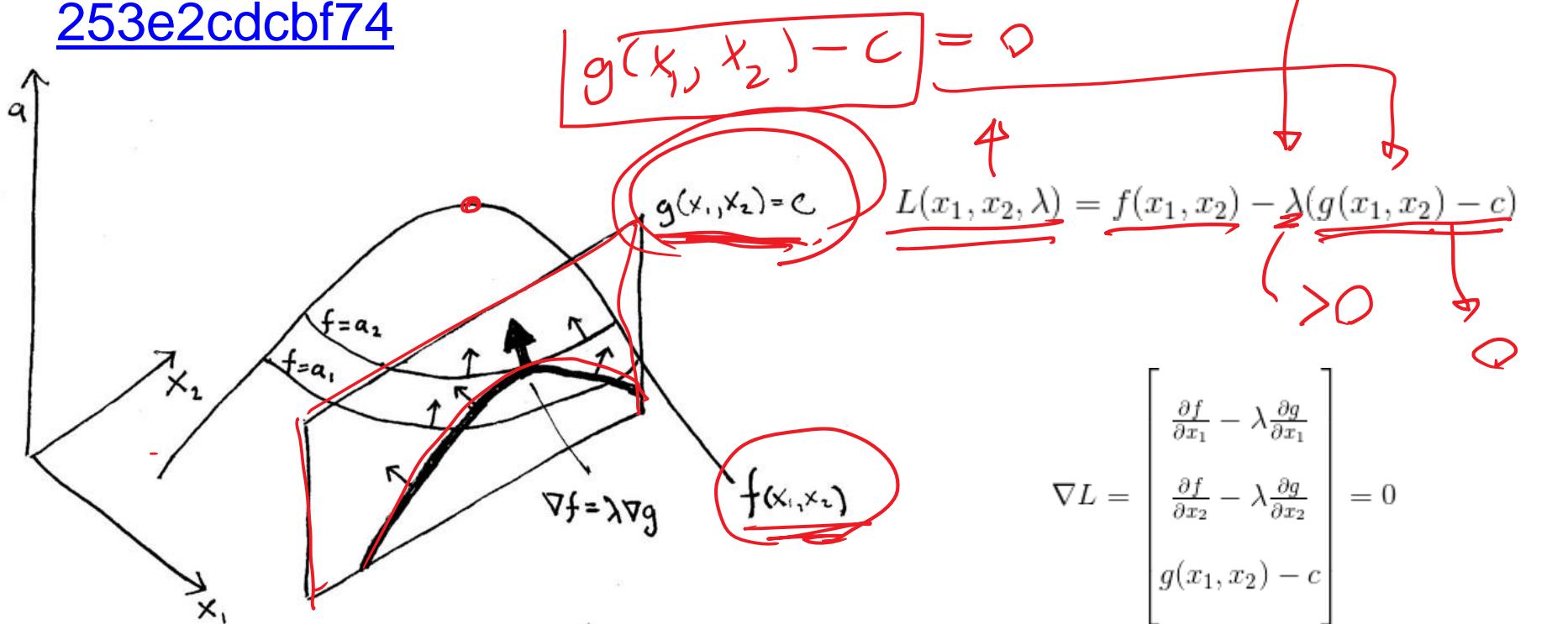
$$6 \quad \nabla_{A^T} f(A) = (\nabla_A f(A))^T$$

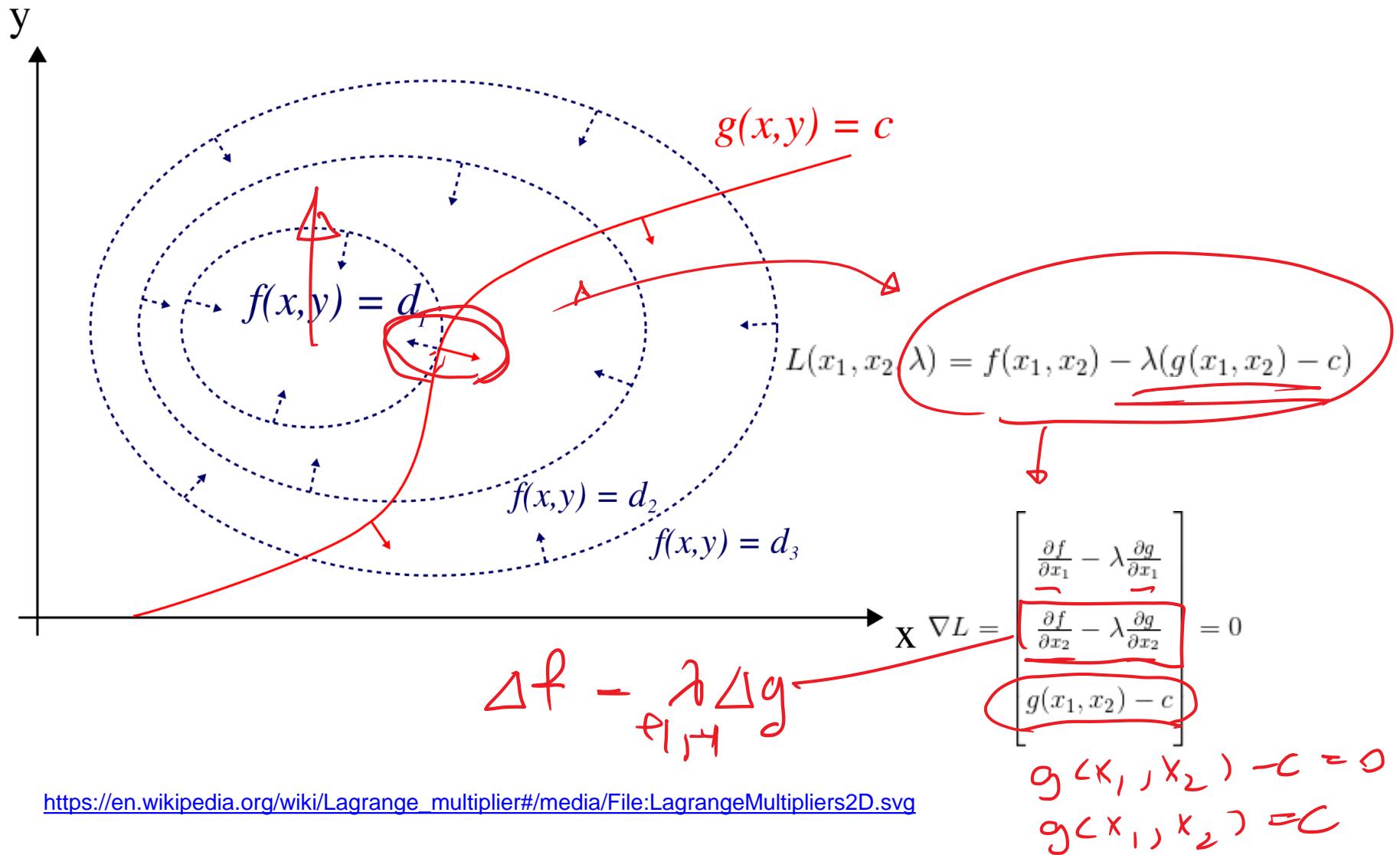
$$7 \quad \underbrace{\nabla_A}_{\text{red}} \text{tr}ABA^TC = CAB + C^T AB^T$$

$$8 \quad \underbrace{\nabla_{A^T}}_{\text{red}} \text{tr}ABA^TC = B^T A^T C^T + BA^T C$$

Lagrange Multiplier

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





So we got to eigenvectors

of dimension feature

- 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

$$\text{Var} = \underline{\omega^T \Sigma \omega} \quad (1)$$

- And our solution yielded

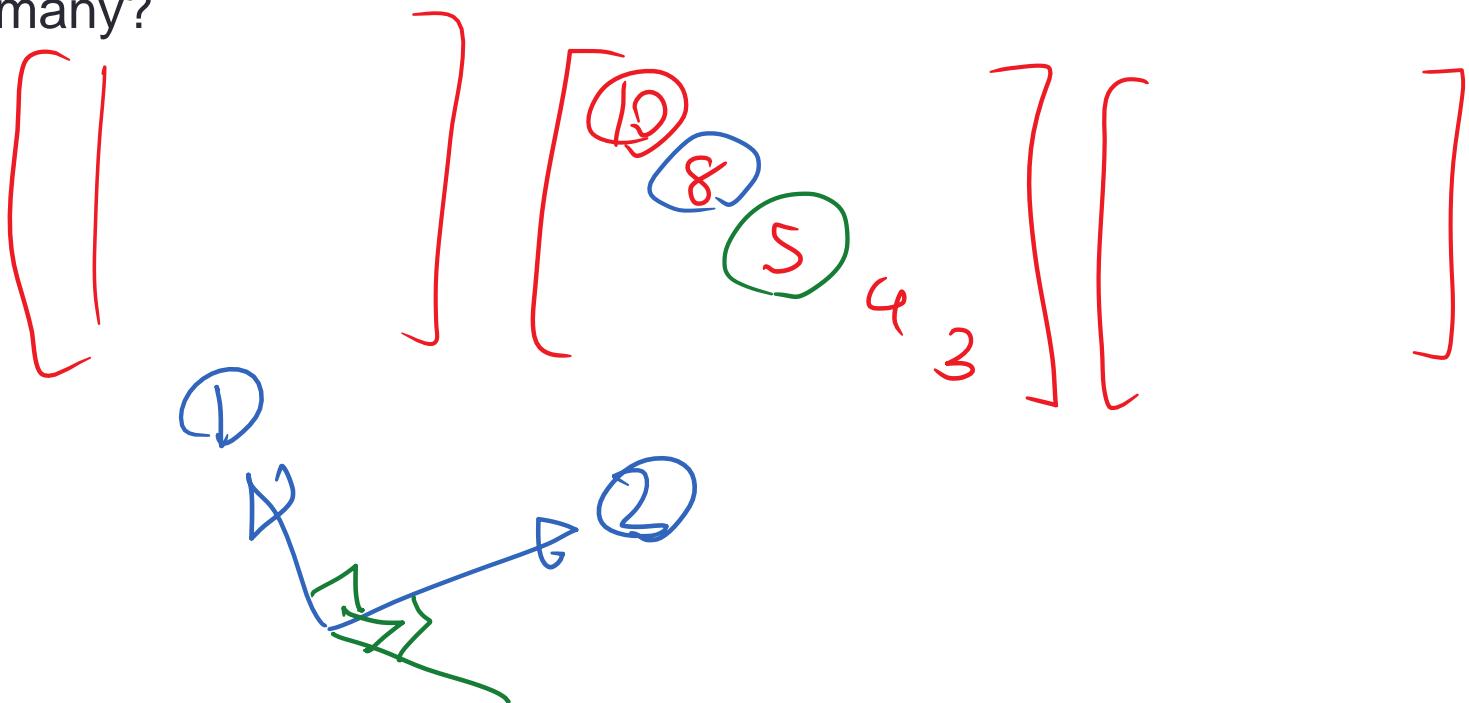
$$\underline{\Sigma \omega = \lambda \omega} \quad (2)$$

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

$$\begin{aligned}\text{projected variance} &= \underline{\omega^T \Sigma \omega} = \underline{\omega^T \lambda \omega} \\ &= \underline{\lambda} \underline{\omega^T \omega} \quad (\text{remember } \|\omega\|=1) \\ &= \underline{\lambda} \quad \text{eigen value}\end{aligned}$$

PCA

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



Matrix rank

$$\sum_{f \times f}$$

- A square $d \times d$ 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 $d \times d$ will 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 $\underline{999}$. The missing rank is because of the mean.

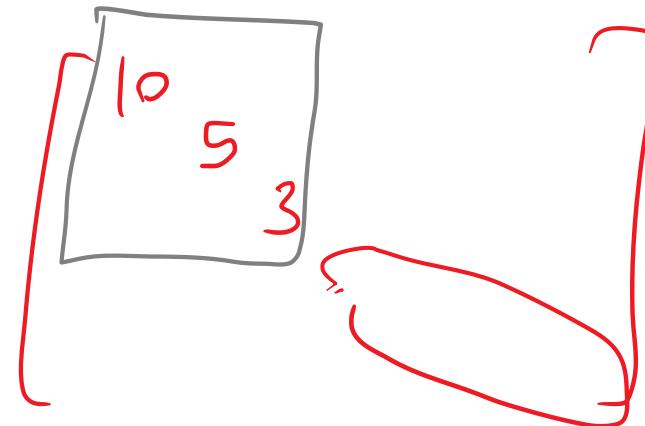
$$\sum [(\mathbf{x} - \mathbf{\mu})(\mathbf{x} - \mathbf{\mu})^T]$$

400,000 40,000

PCA

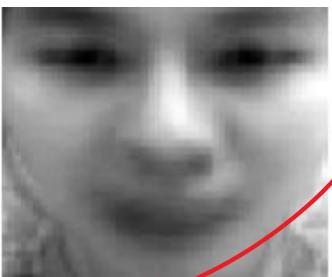
- The direction vector captures the variance corresponding to the eigenvalue
- So we want the higher eigenvalues
- Take the eigenvalues with non-zero eigenvalues (at most $N-1$ non-zero eigenvalues)

rank 3



Eigenfaces

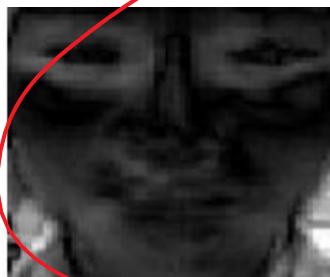
Meanface



V1



V2



V3



V4



V5



V6



V7



V8



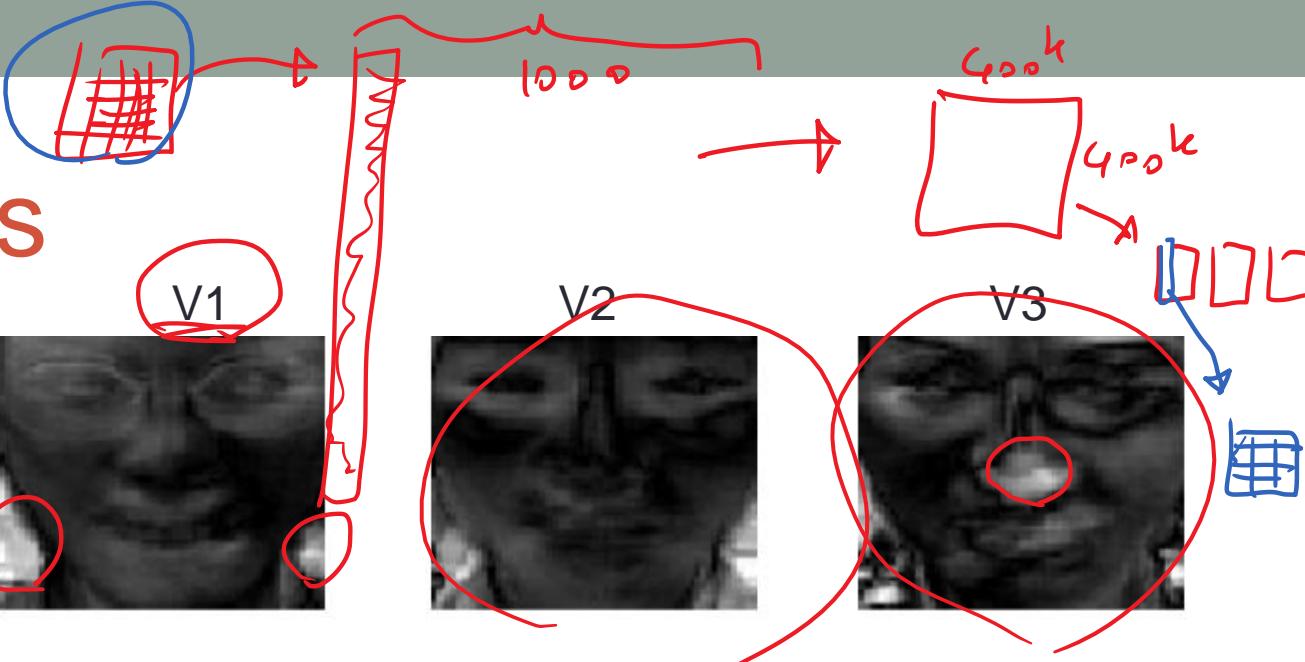
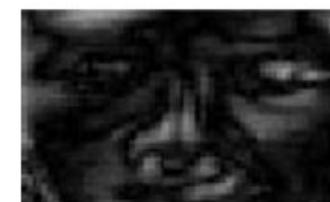
V9



V10

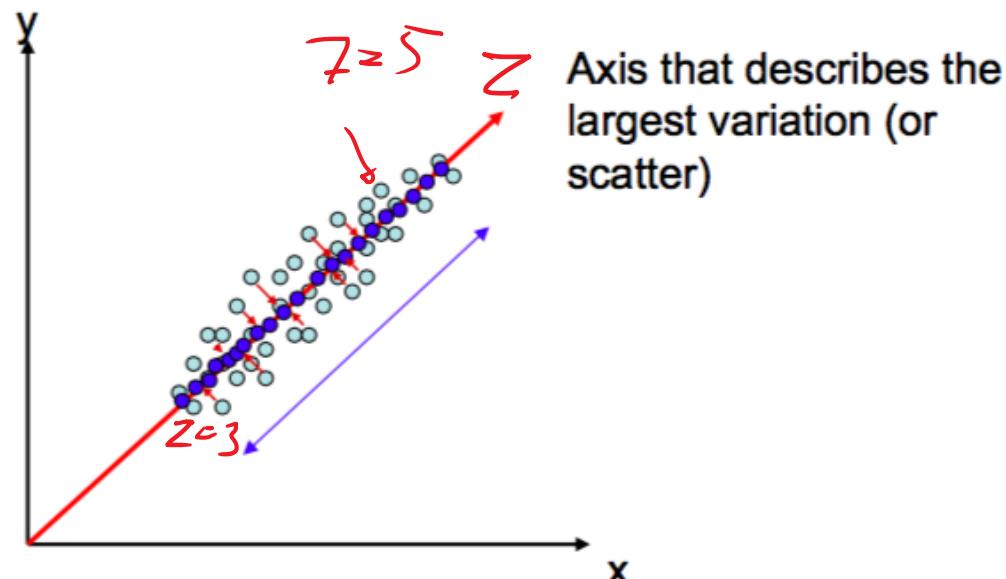


V11



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 describes maximum variation



Means

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

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

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

eigen decomposition

$$\mathbf{x} = \sum_{i=1}^N p_i \mathbf{v}_i + \mathbf{m}$$

proj X i i ✓

$$= p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + \dots + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} + \mathbf{m} = \mathbf{V}\mathbf{p} + \mathbf{m}$$

$\mathbf{x} \approx \sum_{i=1}^3 p_i \mathbf{v}_i$

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:

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

Finding the weights

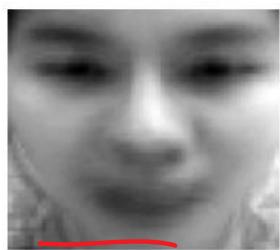
$$\mathbf{x} = \sum_{i=1}^N p_i \mathbf{v}_i = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + \dots + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} = \mathbf{V}\mathbf{p}$$

- If \mathbf{v}_i are orthogonal, the projection of \mathbf{x} onto \mathbf{v}_i gives p_i

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

Reconstruction with eigenfaces

Mean



v1



v2



v3

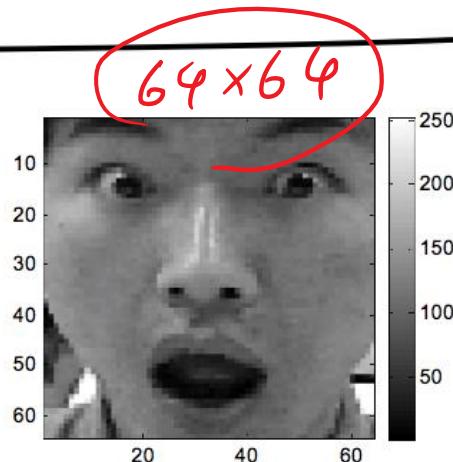


MSE=758.13

=



reconstructed
 $\tilde{\mathbf{X}}$



Original
 \mathbf{X}

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 $\mathcal{O}(n^3)$



Practical issues

1000 → \leq rank 999 → eigenvectors
nearly linearly 999

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

Gram Matrix

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

$X^T X$

Covariance matrix
is the **outer-product**
of the input matrix

$d \times N$

$d \times d$

Must solve $\Sigma v = \lambda v$

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

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

Solve eigenvalue problem

$$X^T X v' = \lambda v'$$

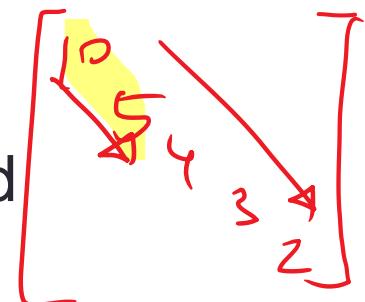
- $X^T X$ is a gram of **inner-product** matrix. Its size is $N \times N$ where N is the number of data samples. $\{ \circ \circ \circ \}$

But how to get v from v' ?

- From previous slide, equation (1) and (2)
 - $\underline{XX^T v} = \lambda v$ (1)
 - $\underline{v'} = X^T v$ (2)
- Substitute (2) into (1)
 - $\underline{Xv'} = \lambda v$
eigen in Gram-Schmidt
- 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

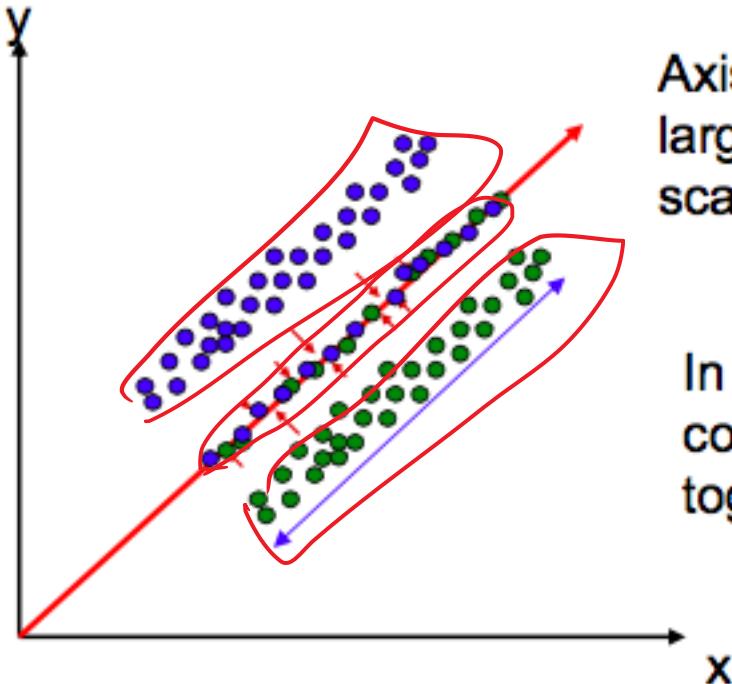


$$\mathbf{x} = \sum_{i=1}^N p_i \mathbf{v}_i = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + \dots + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} = \mathbf{Vp}$$

- Select enough v so that the difference between original x and reconstructed x is small

PCA for classification

- PCA does not care about the class labels

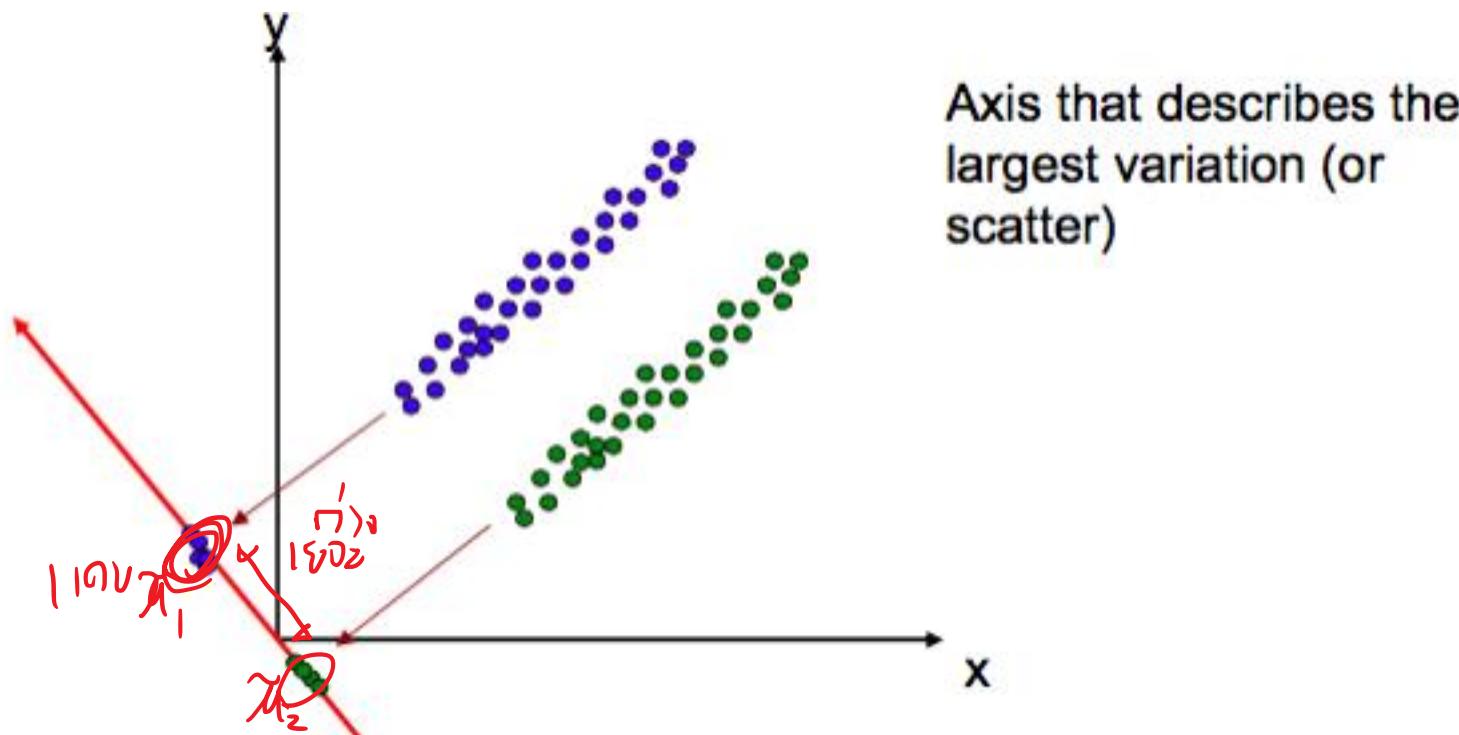


Axis that describes the largest variation (or scatter)...

In this case the projection vector completely smears the two classes together, making them inseparable

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 $|\underline{\tilde{\mu}_1 - \tilde{\mu}_2}|^2$

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

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

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

$$\begin{aligned} |\underline{(\tilde{\mu}_1 - \tilde{\mu}_2)}|^2 &= |(\mathbf{w}^T \underline{\mu}_1 - \mathbf{w}^T \underline{\mu}_2)|^2 \\ &= \mathbf{w}^T \boxed{(\underline{\mu}_1 - \underline{\mu}_2) (\underline{\mu}_1 - \underline{\mu}_2)^T} \mathbf{w} \\ &= \mathbf{w}^T \mathbf{S}_B \mathbf{w} \end{aligned}$$

Between class

Between class scatter matrix S_B

$$\begin{aligned}(\tilde{\mu}_1 - \tilde{\mu}_2)^2 &= (\mathbf{w}^T \boldsymbol{\mu}_1 - \mathbf{w}^T \boldsymbol{\mu}_2)^2 \\&= \mathbf{w}^T (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \mathbf{w} \\&= \boxed{\mathbf{w}^T S_B \mathbf{w}}\end{aligned}$$

We want to maximize $\mathbf{w}^T S_B \mathbf{w}$ where S_B is the between class scatter matrix defined as:

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

We also want to minimize within class scatter

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

Variance of within project

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

Minimize the total scatter

$$\tilde{s}_1^2 + \tilde{s}_2^2$$

Within class scatter

- Lets expand on scatter s_1, s_2 .

$$\begin{aligned}\tilde{s}_1^2 &= \sum_{i=1}^{N_1} (\tilde{x}_i - \tilde{\mu}_1)^2 \\ &\geq \\ &= \sum_{i=1}^{N_1} (\mathbf{w}^T \mathbf{x}_i - \mathbf{w}^T \boldsymbol{\mu}_1)^2 \\ &= \sum_{i=1}^{N_1} \mathbf{w}^T (\mathbf{x}_i - \boldsymbol{\mu}_1) (\mathbf{x}_i - \boldsymbol{\mu}_1)^T \mathbf{w} \\ &\quad \text{---} \quad \text{---} \\ &= \mathbf{w}^T \mathbf{S}_1 \mathbf{w}\end{aligned}$$

Total within class scatter

- We want to minimize

$$\tilde{S}_1^2 + \tilde{S}_2^2 \rightarrow S_3 + S_4 + \dots$$

- This is the same as

$$\mathbf{w}^T \mathbf{S}_w \mathbf{w}$$

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

within class

C number of classes, N_i 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

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

$$\sum_k w_k = \lambda w$$

eigen

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

If \mathbf{S}_W is non-singular and invertible.

- Generalized eigenvalue problem. The number of solutions is $\min(\text{rank } \mathbf{S}_B, \text{rank } \mathbf{S}_W) = C-1$ or $N-C$
- For 2 class this simplifies to
 - Note this is only one projection direction

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

15:47

LDA+PCA

400 p o o

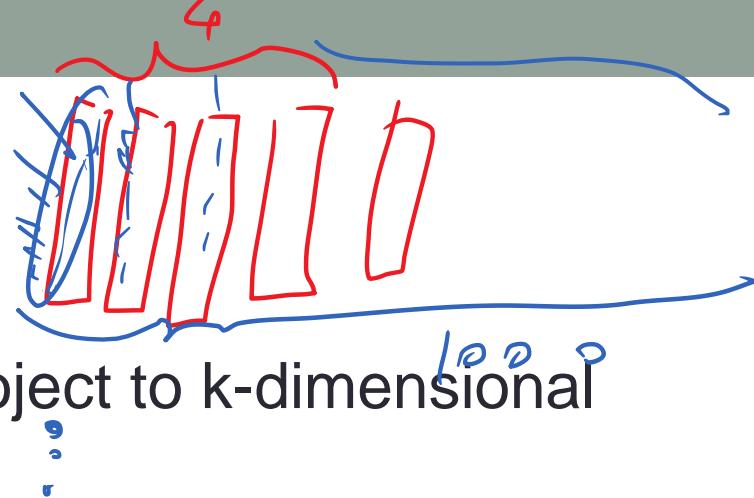
- Original features dimension L
- S_B, S_w are LxL matrixes
- $\text{rank } S_B = C-1$, $\text{rank } S_w = \underline{N-C}$
- First do PCA to reduce dimension $\rightarrow \underline{997} \xrightarrow{\text{LDA}} \underline{997}$
- Then do LDA to maximize classification ability $S_w \ 997 \times 997$
- How many dimensions to PCA?
 - Do PCA to keep $N-C$ eigenvectors \rightarrow Makes S_w full rank and invertible
 - Then, do LDA and compute $C-1$ projections in this $N-C$ subspace
- PCA+LDA = Fisher projection

~~Cov~~ 10×10
full rank $10 \rightarrow 10$

~~N-C~~
99

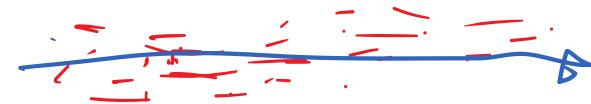
1000

Random projection



- Original d -dimensional data is project to k -dimensional subspace
- Using a random $k \times d$ 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~~ no data
- Faster to train than PCA if data has high dimension
 - $O(k^2 \times n + k^3)$ (covariance computation + eigen decomposition) 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

~~PCA LDA~~

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

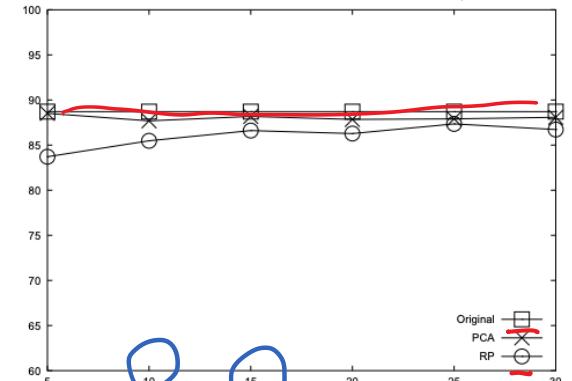
Table 1: Description of Datasets

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



C4.5 DT

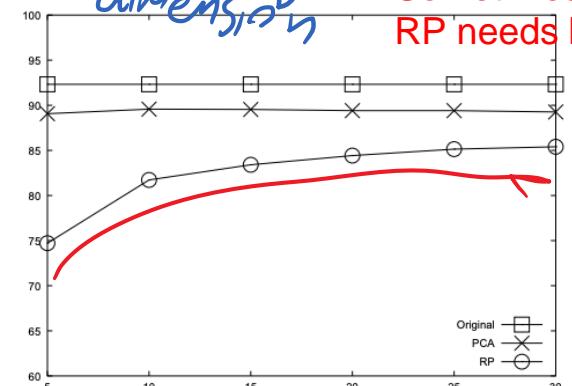
N Ion



dimension

1NN

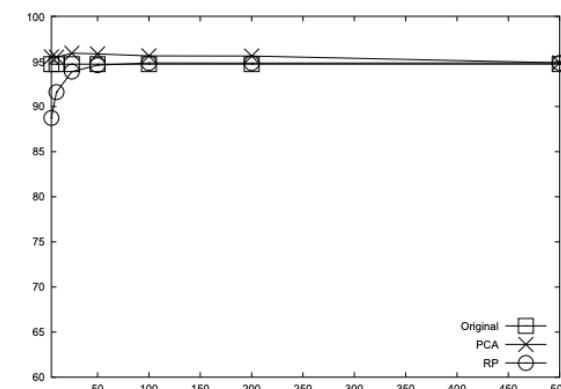
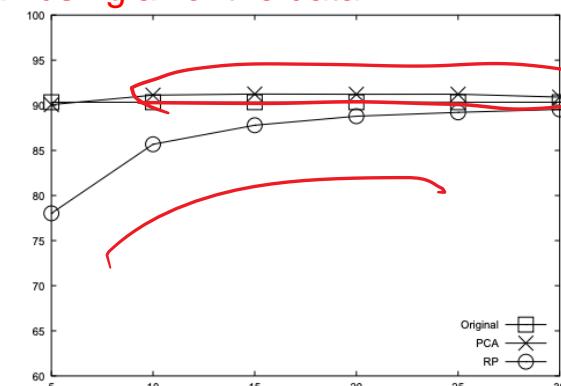
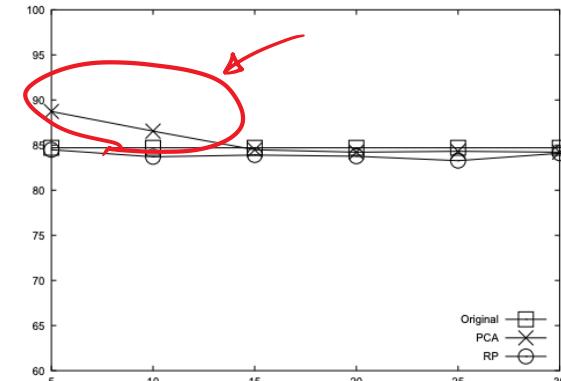
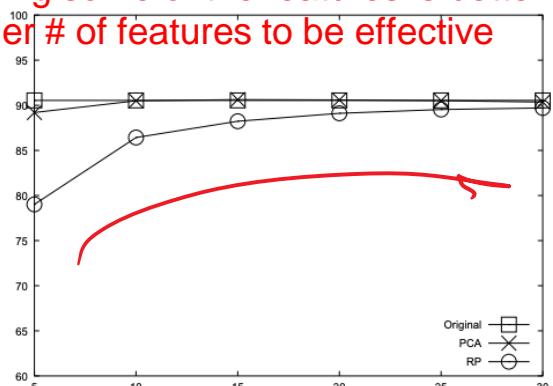
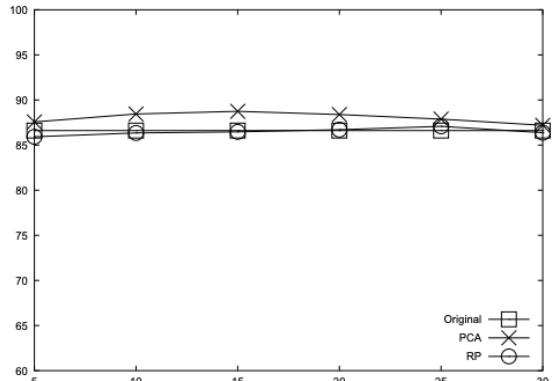
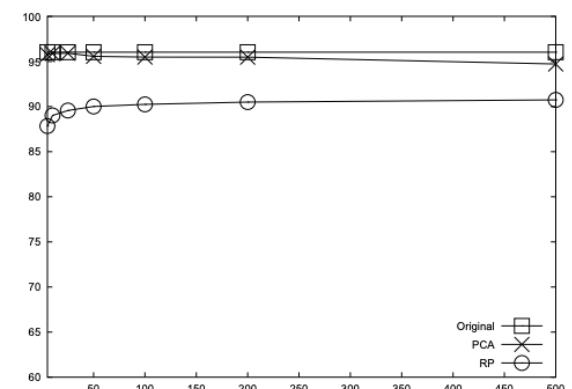
Spam



Sometimes using some of the features is better than using all of the data
RP needs larger # of features to be effective

5NN

Ads

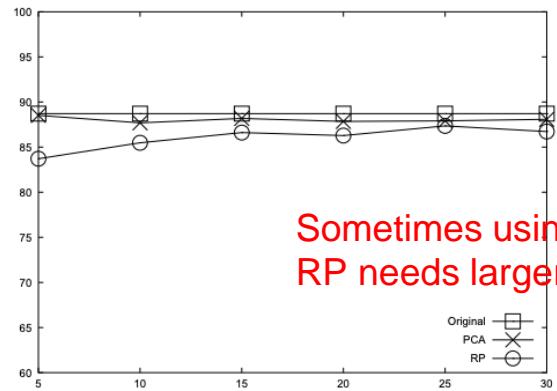


C4.5

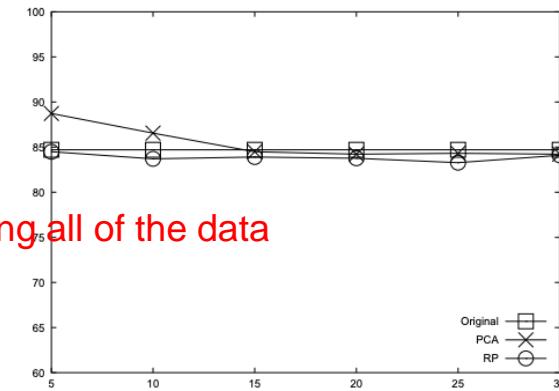
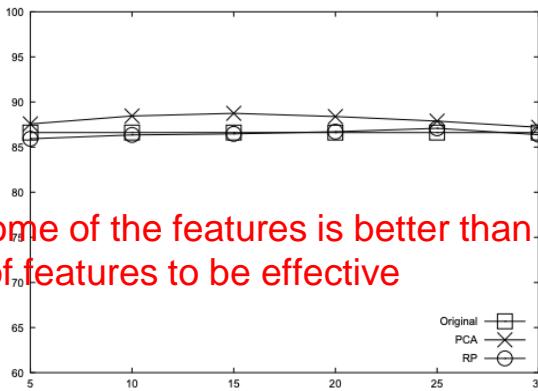
1NN

5NN

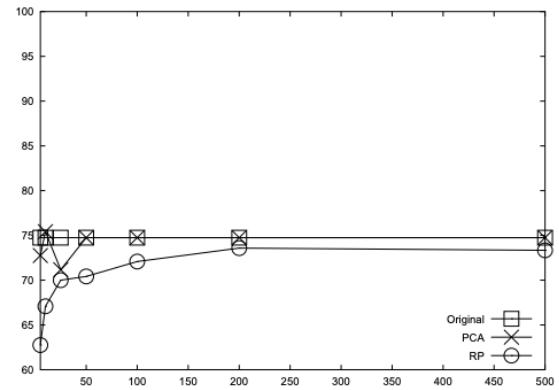
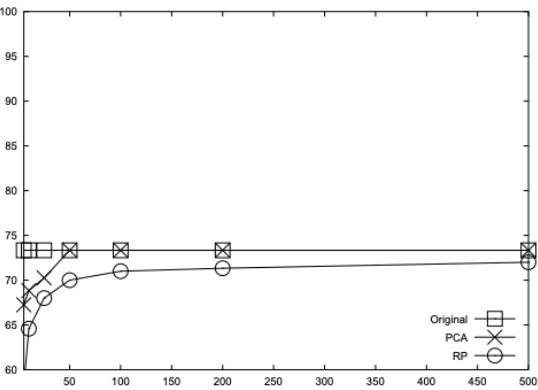
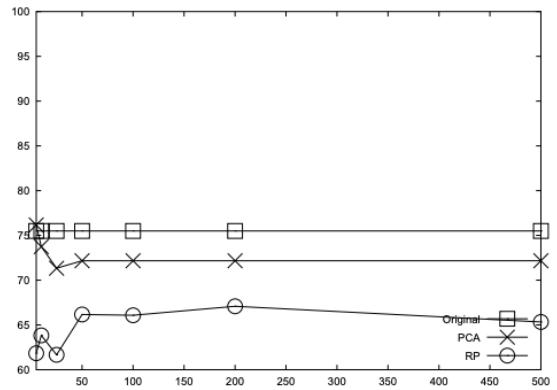
Ion



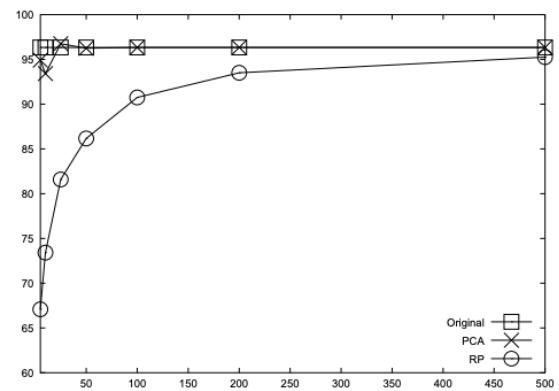
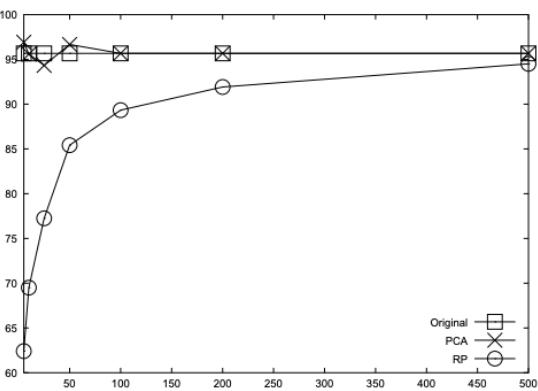
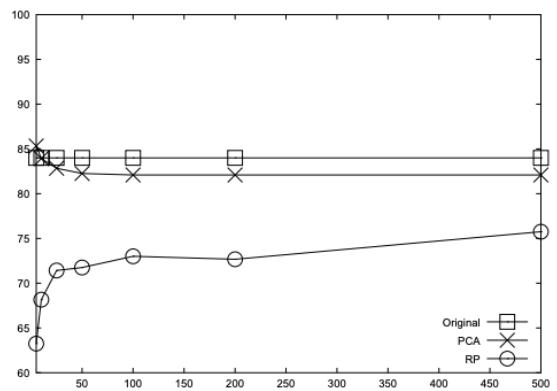
Sometimes using some of the features is better than using all of the data
RP needs larger # of features to be effective



Colon



Leukemia



Visualization

Methods covered

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

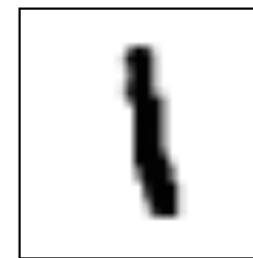
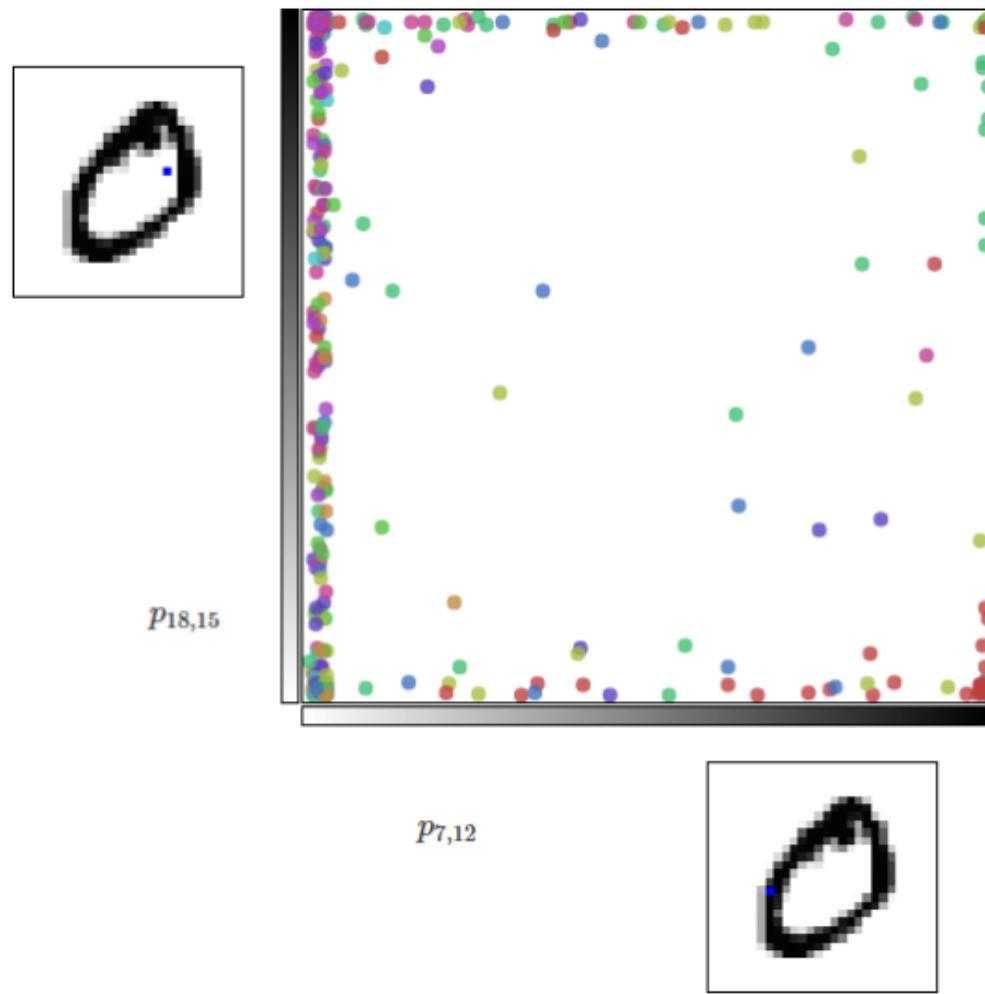
These are usually useful for downstream machine learning methods. (Classification/Regression/Clustering)

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

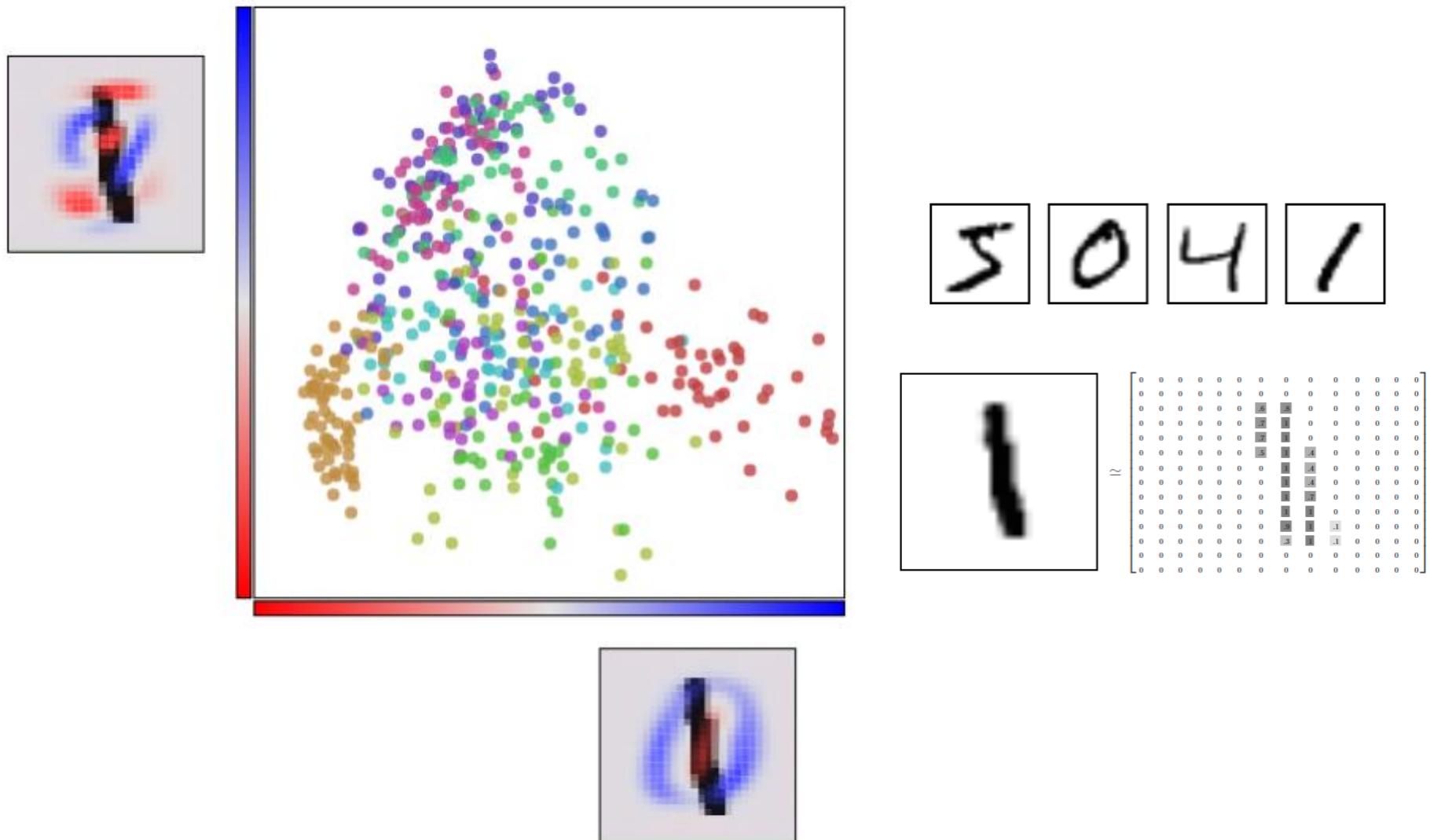
1. Interpretability (in some sense): ???

- Check if our dimensionality reduction is good
- Analyze clusters from k-mean, GMM, etc.

Visualizing MNIST



PCA with MNIST



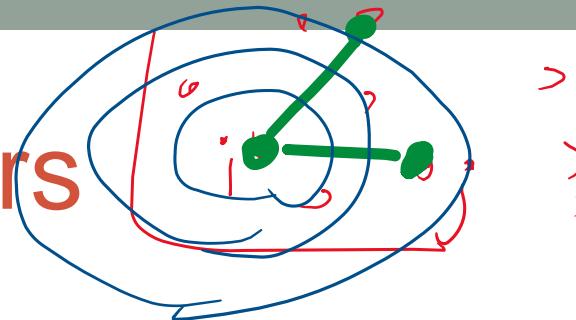
t-distributed Stochastic Neighbor Embedding (t-SNE)

probabilistic

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

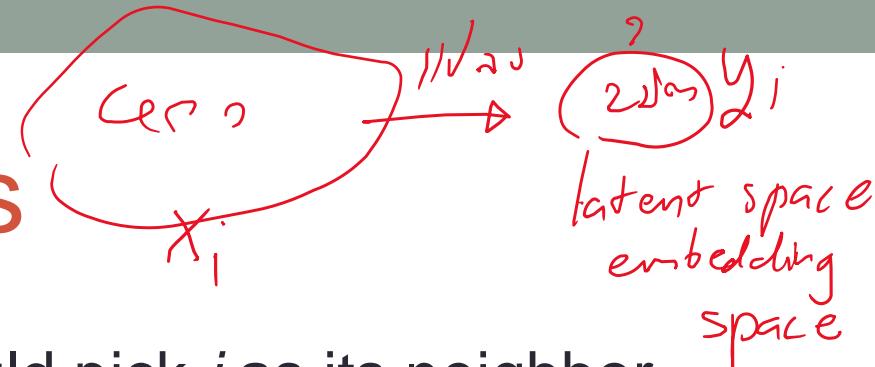
$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2)/2\sigma_i^2}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2)/2\sigma_i^2}$$

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

The variance is fixed to some value.

is this sum $\Sigma_{j \neq i}$

Defining neighbors



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

Assume i picks proportional to Gaussian centered at i

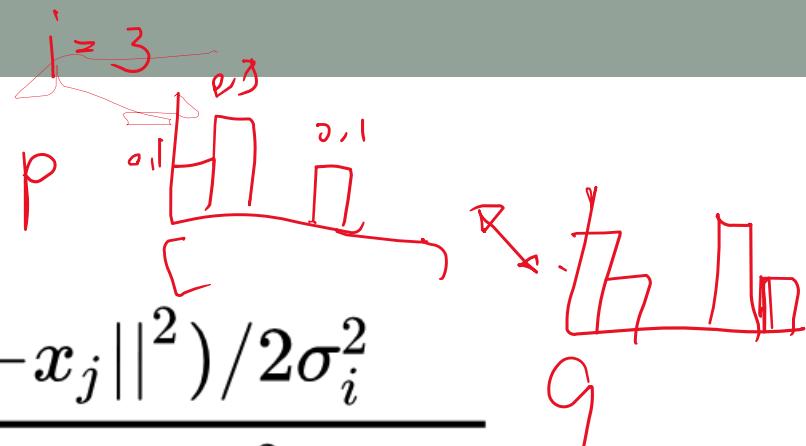
$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2)/2\sigma_i^2}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2)/2\sigma_i^2}$$

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



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

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

Euclidean

We expect p and q to be the same \rightarrow 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.

$$P(\text{head}) = 0.5 \quad P(\text{tail}) = 0.5$$

$$Q(\text{head}) = 0.7 \quad Q(\text{tail}) = 0.3$$

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

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

Loss function

$$(Q^T X - y)$$

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

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

We expect p and q to be the same \rightarrow small distance

Loss function

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

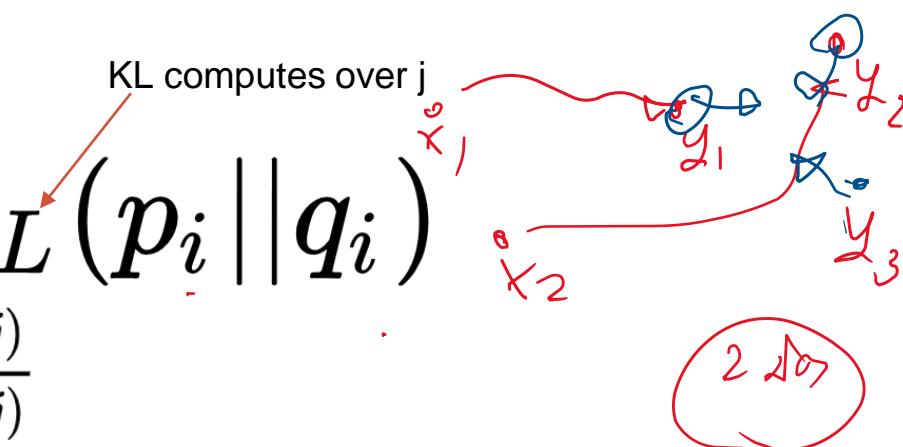
$$D_{KL}(P||Q) = \sum_j P(j) \log \frac{P(j)}{Q(j)}$$

Note P can be considered as the weight for the distance

Where p is large but q is small \rightarrow large penalty

p is small but q is large \rightarrow 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 $p_{j|i}$

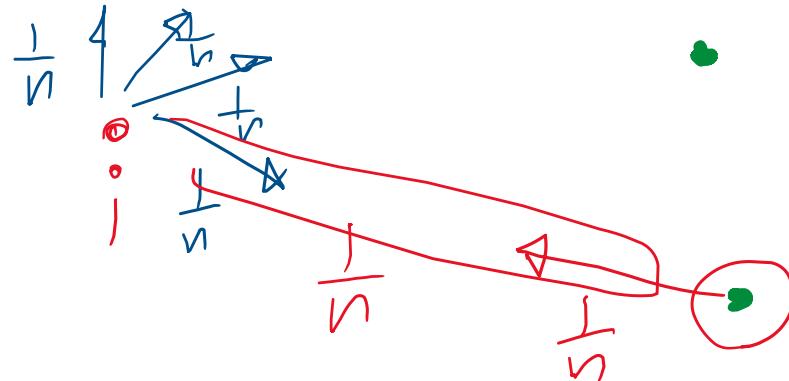
- Location of points in q no longer matter

$0, 0, 1$

Create “joint probability” $p_{ij} = p_{ji} = (\underline{p_{ij}} + \underline{p_{ji}})/2$

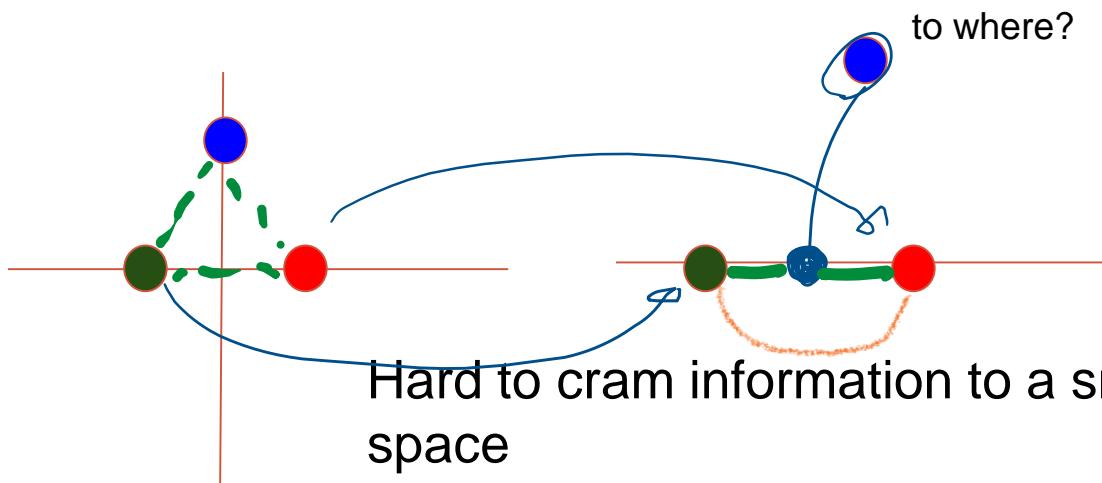
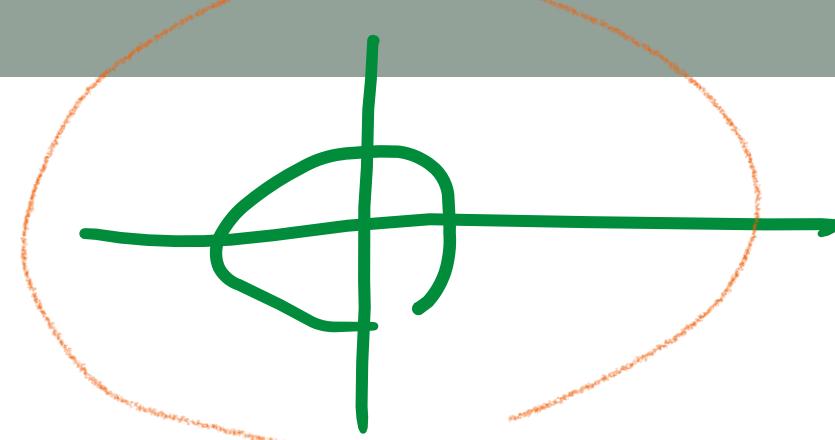
- Each data point will contribute to the loss

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



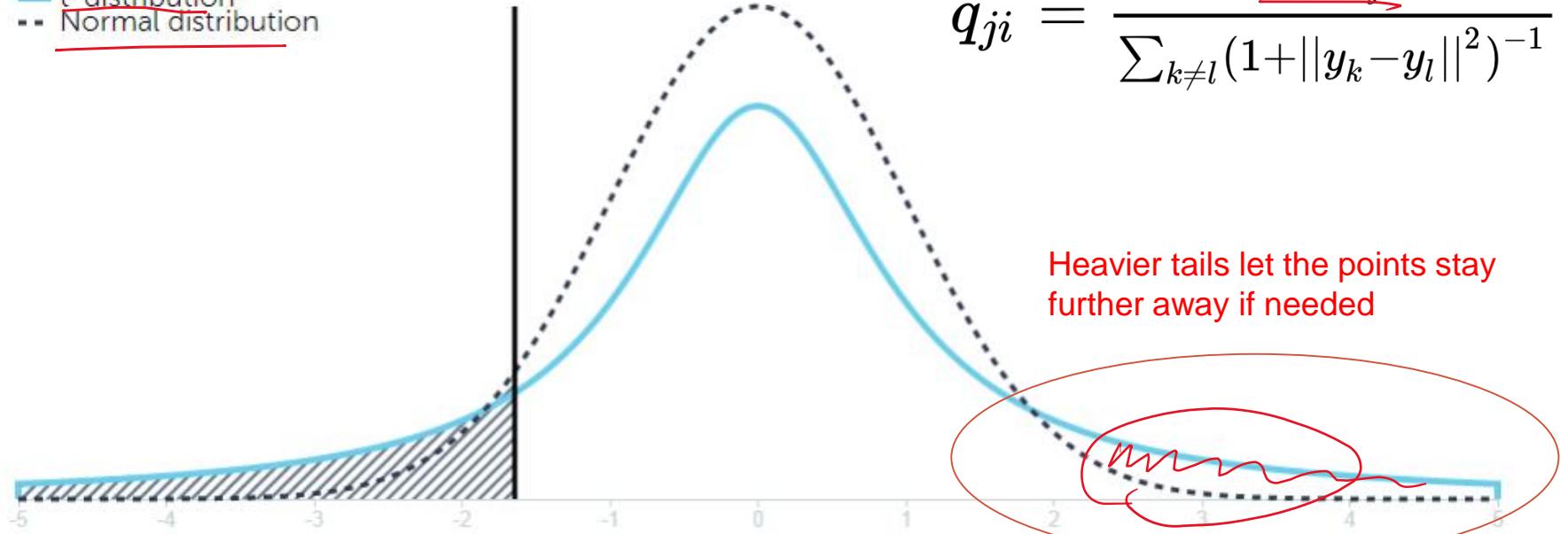
Hard to cram information to a smaller dimension space
Points tend to cram in the embedding space (no gap)

t-distributed

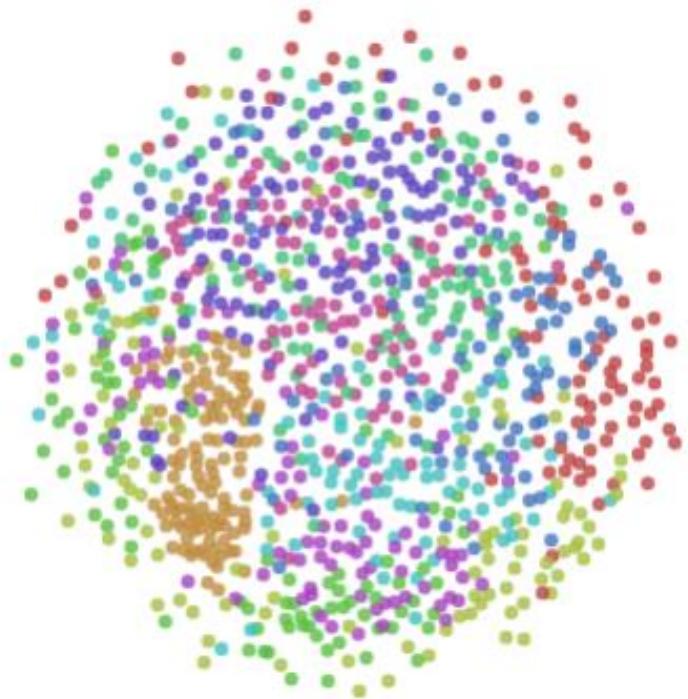
- Instead of Gaussian for q we use student's t distribution

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

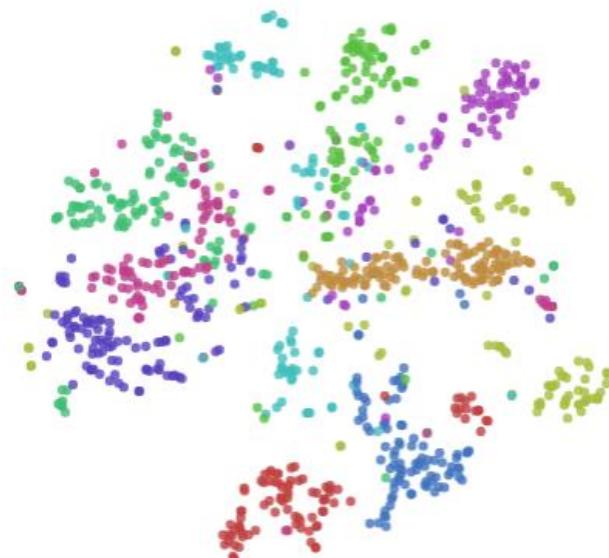
— t-distribution
-- Normal distribution



Crowding and t-SNE



Sammon's mapping
(crowding problem)



t-SNE

Variance

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

hyper recta

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

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

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

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

= 3

Entropy

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

Set $\text{Perp}(P_i)$ 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

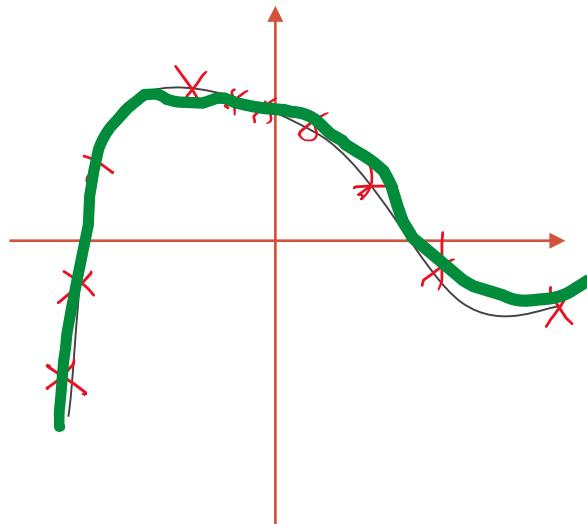
UMAP big picture

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

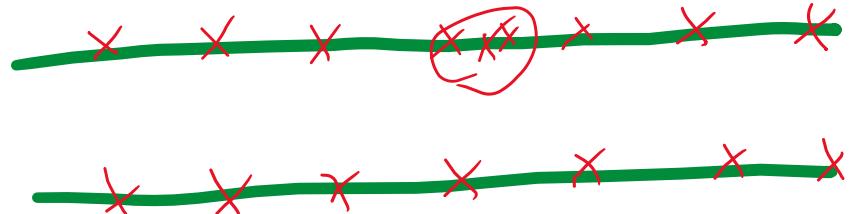
Manifold

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)
³

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

The distance between X_i and X_j is

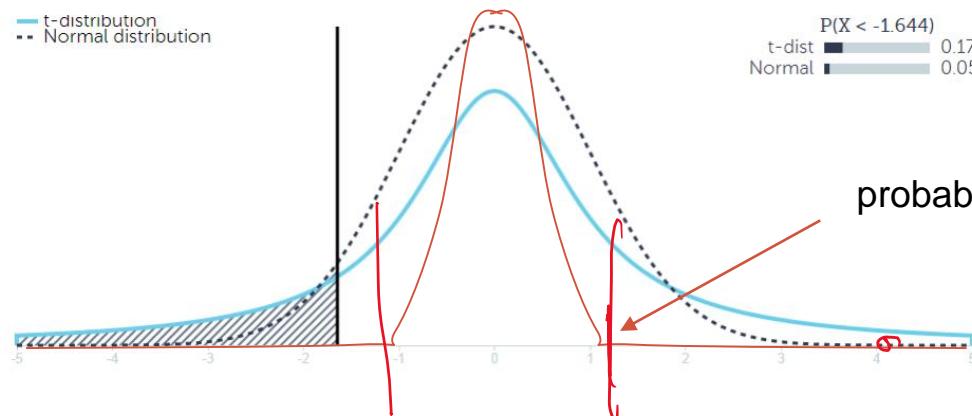
$$w_i(X_i, X_j) = \exp(-(d(X_i, X_j) - \rho_i)/\sigma_i)$$

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

L2 distance in
original space

diameter of the
neighbors so it
maxes at 1

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, \dots, X_k be the k -nearest neighbors of X

The distance between X_i and X_j is

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

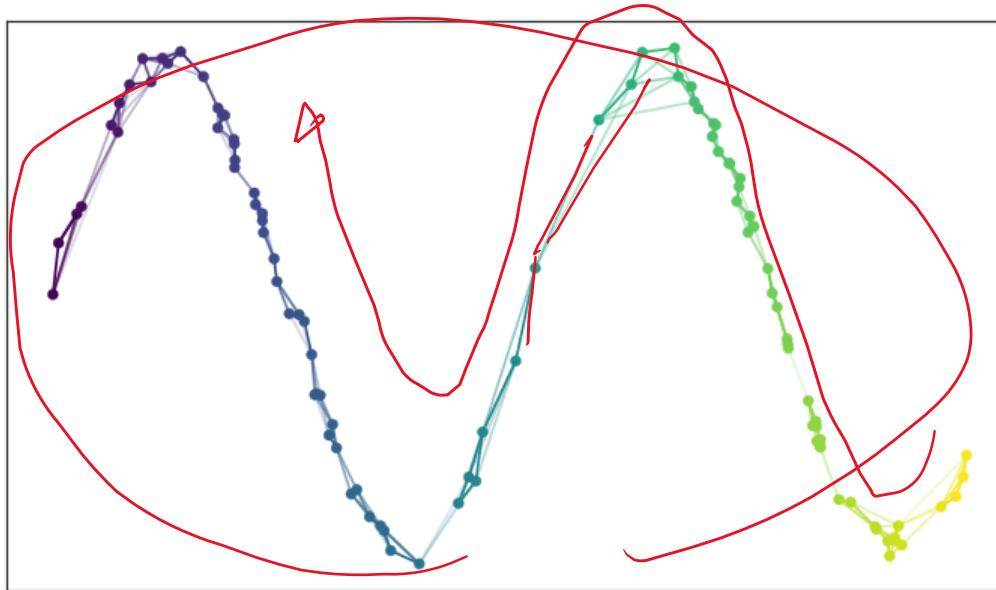
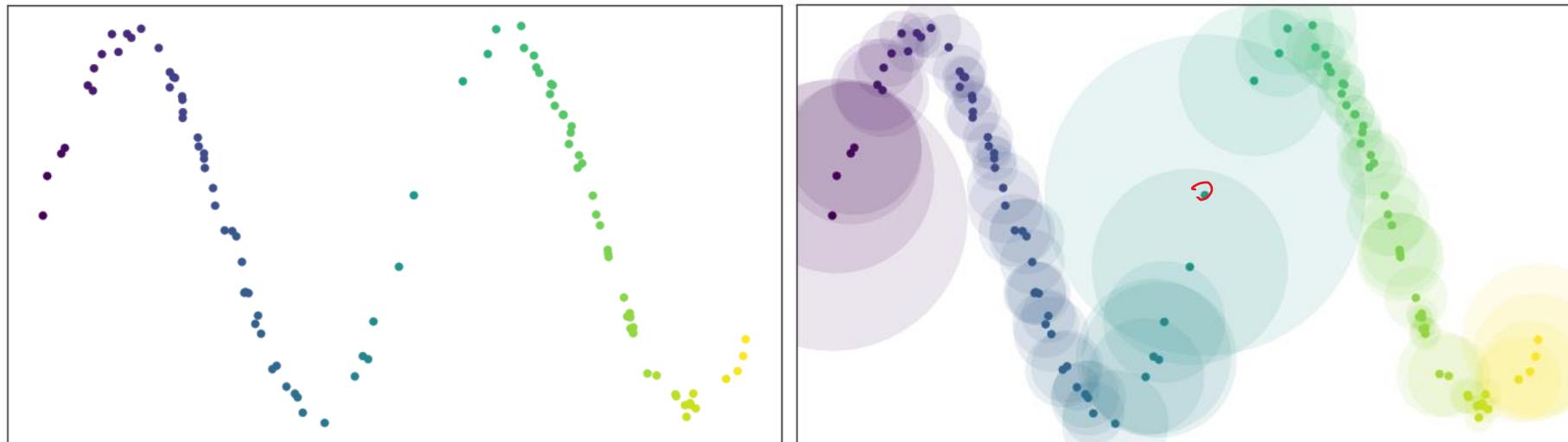
L2 distance in
original space

distance to nearest
neighbor so it starts
from 0

diameter of the
neighbors so it
maxes at 1

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)

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

All neighbors

Distance in original space
Distance in new space

Similar to KL metric in t-SNE

Loss function

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

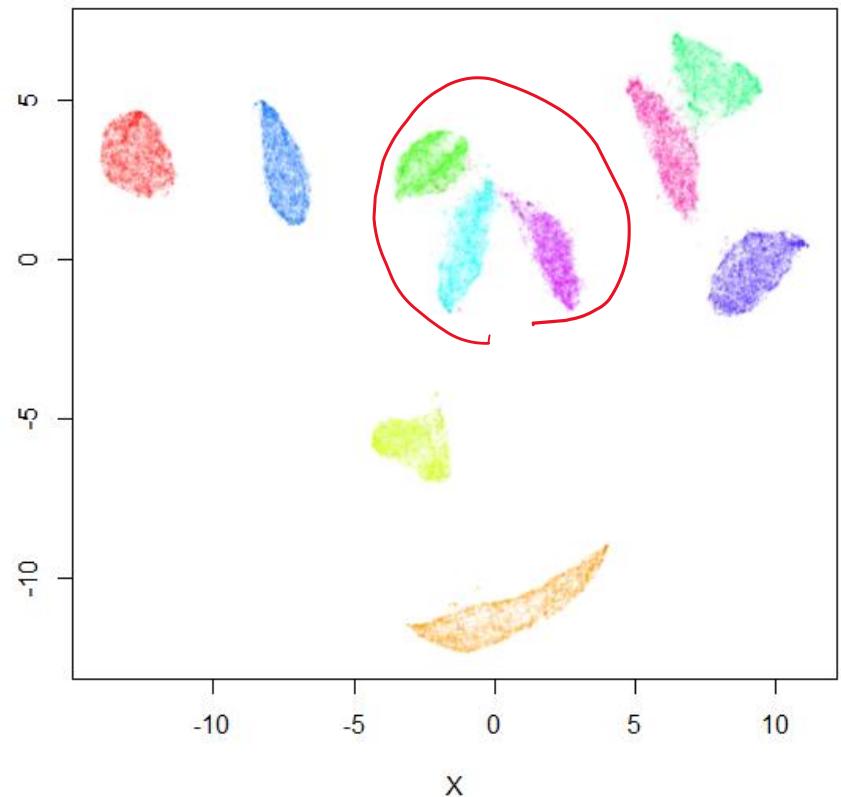
All points i KL computes over j

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

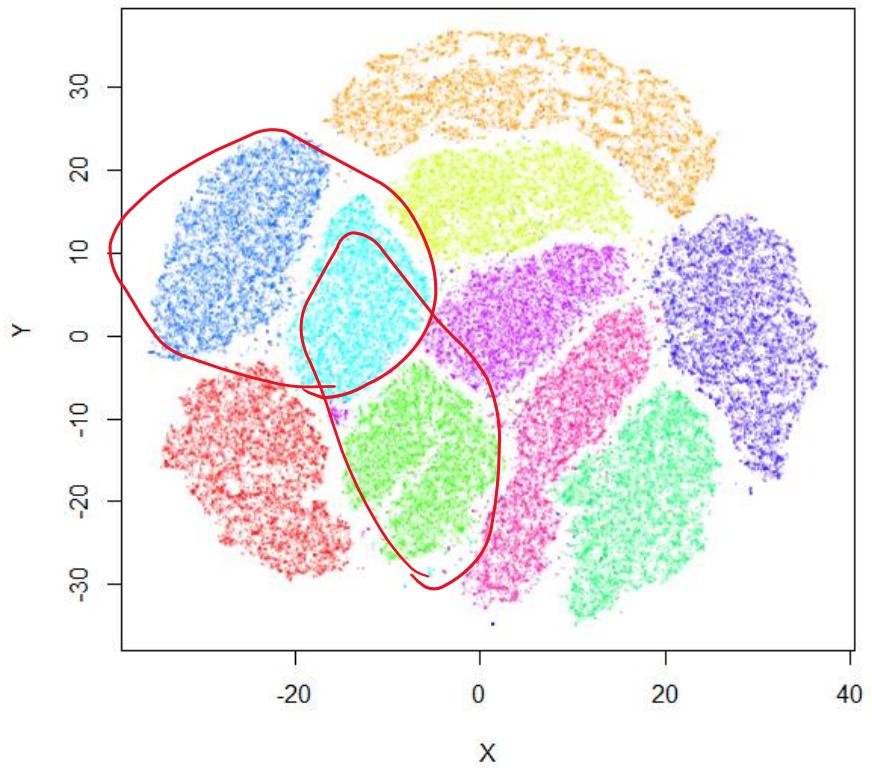
Use gradient descent to optimize

MNIST

MNIST UMAP



MNIST t-SNE

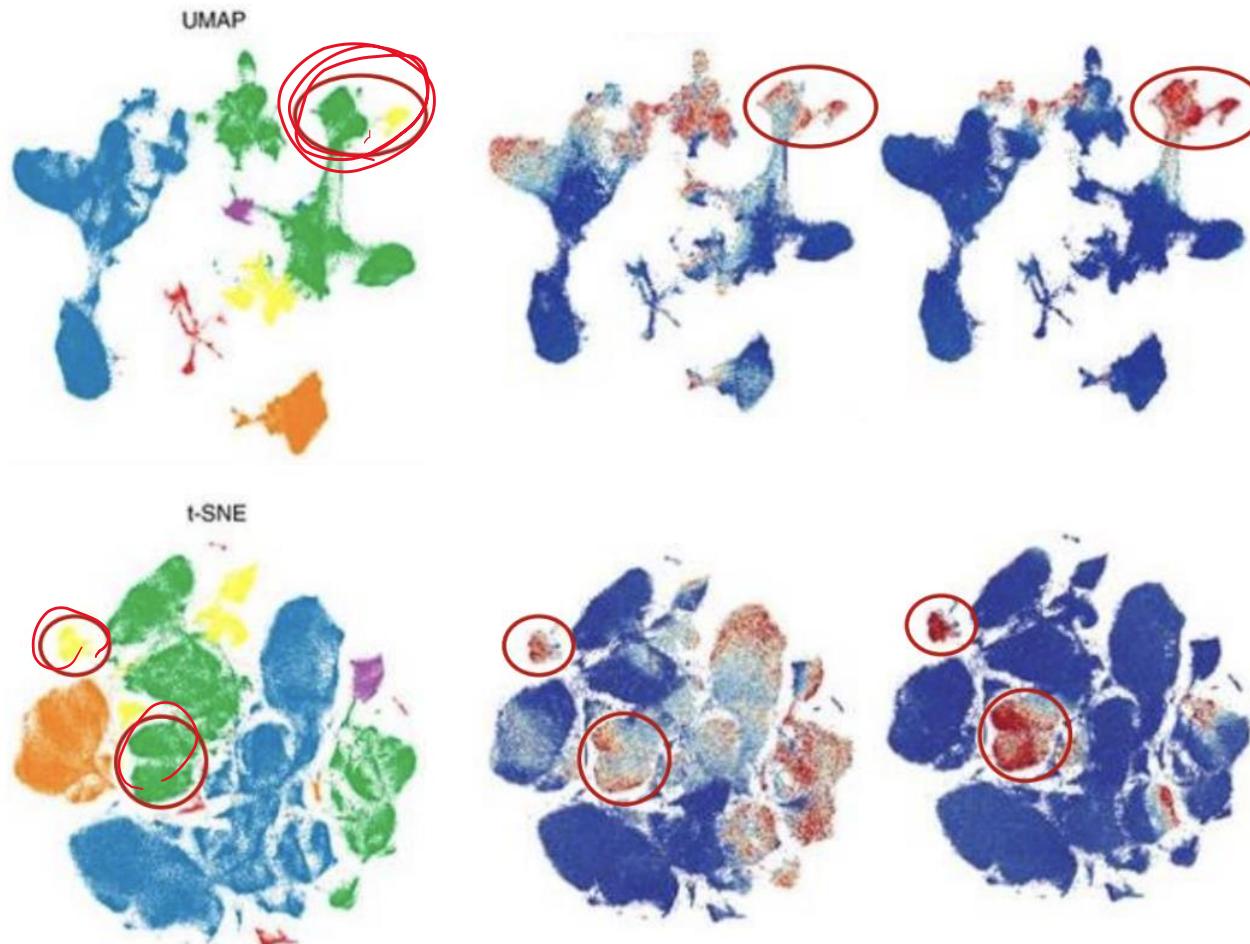


<https://jlmelville.github.io/uwot/umap-examples.html>

Dimensionality reduction for visualizing single-cell data using UMAP

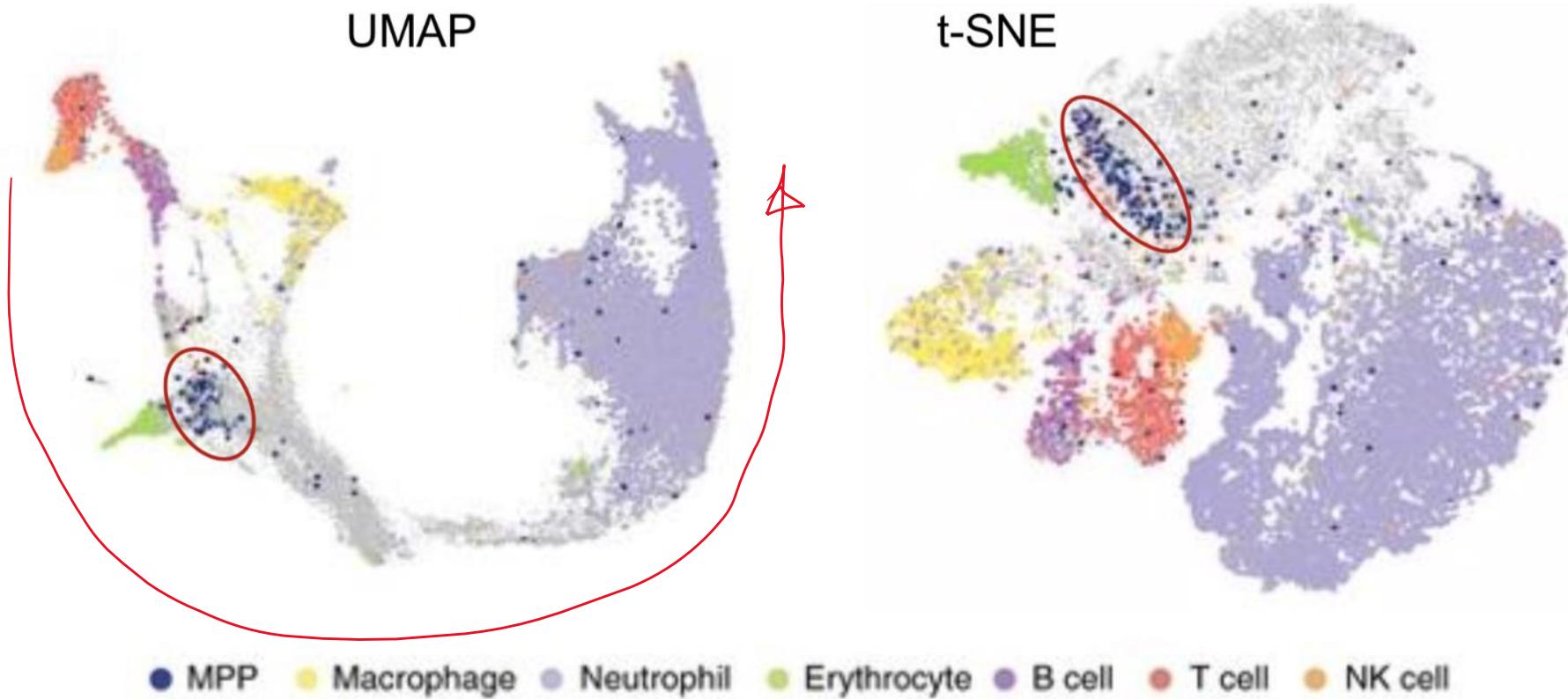
Analysis | Published: 03 December 2018

nature
biotechnology



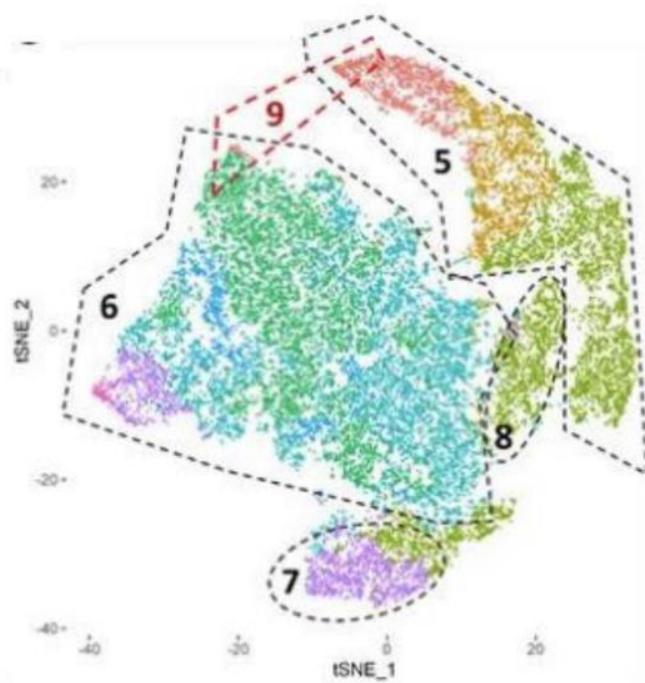
- UMAP can capture cross-class relationship

UMAP vs t-SNE on continuous classes

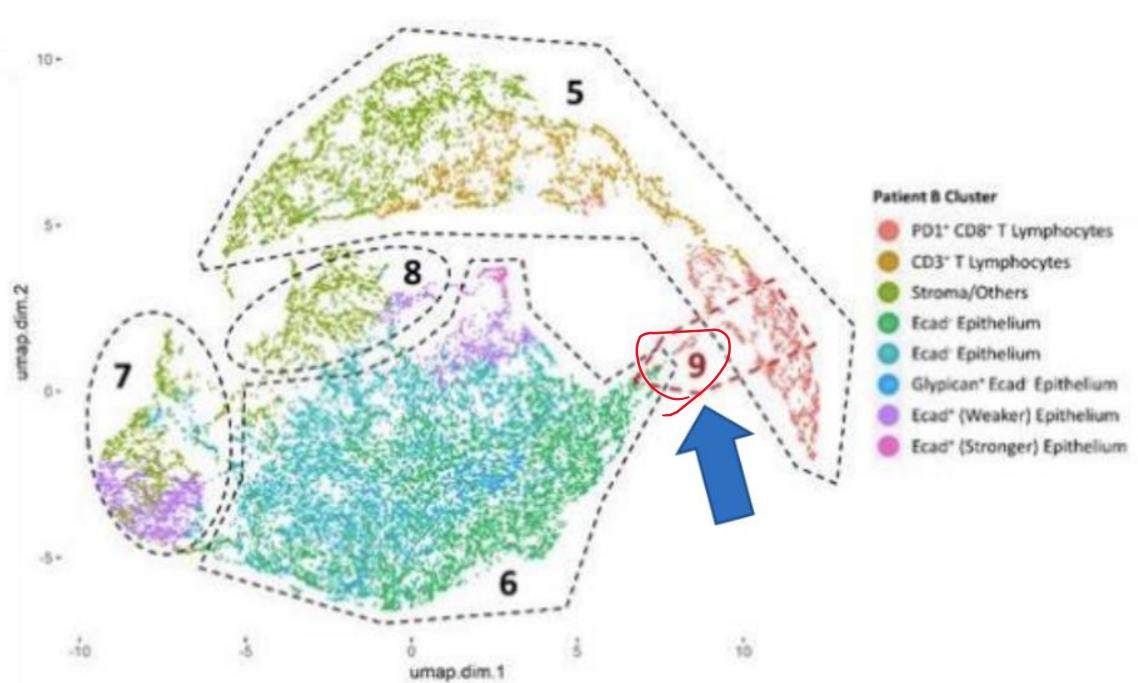


UMAP find continuous transition across classes

t-SNE

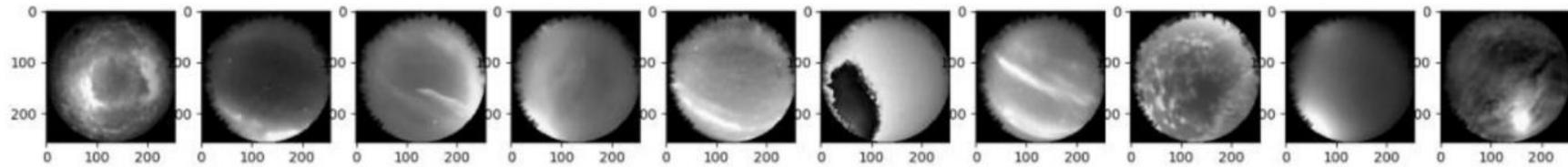


UMAP

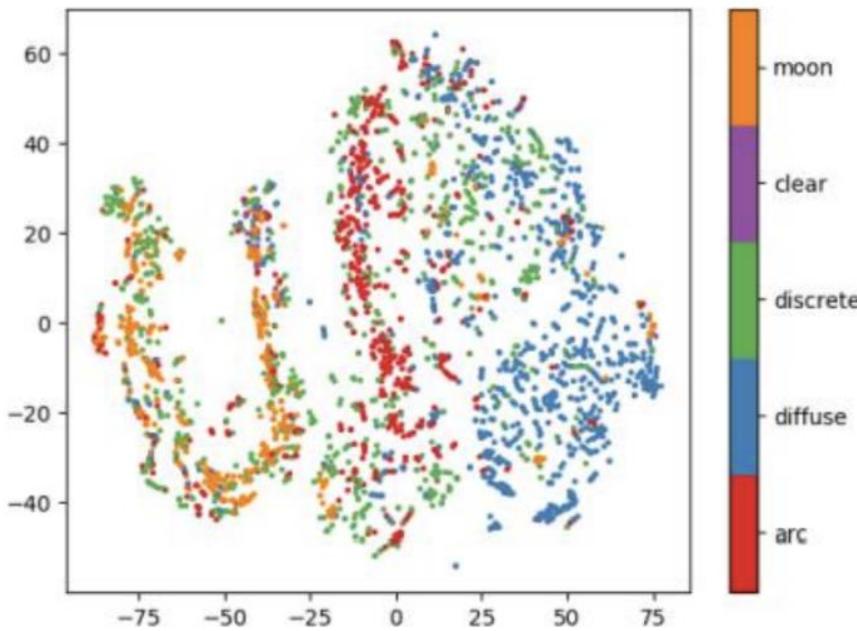


Correlation of Auroral Dynamics and GNSS Scintillation with an Autoencoder

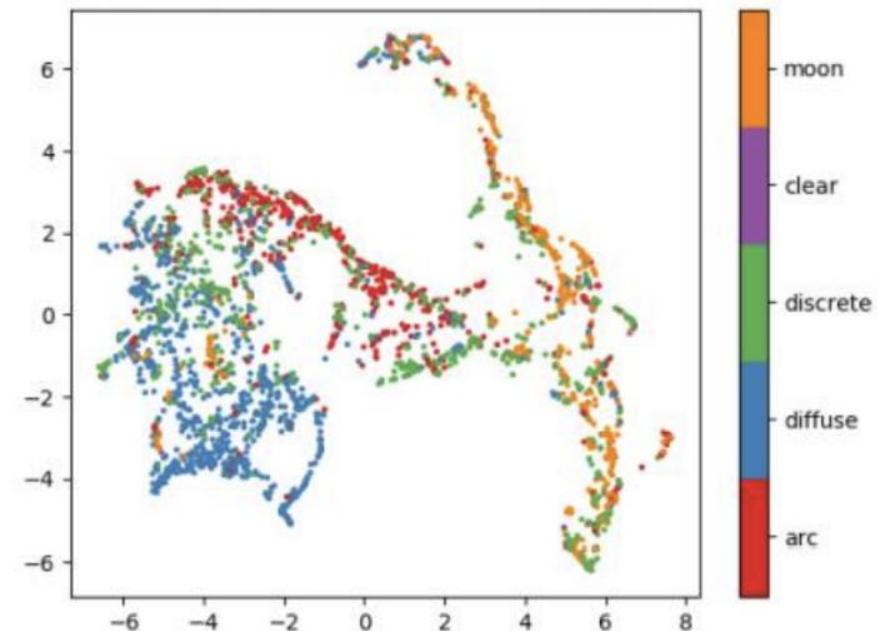
33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.



t-SNE

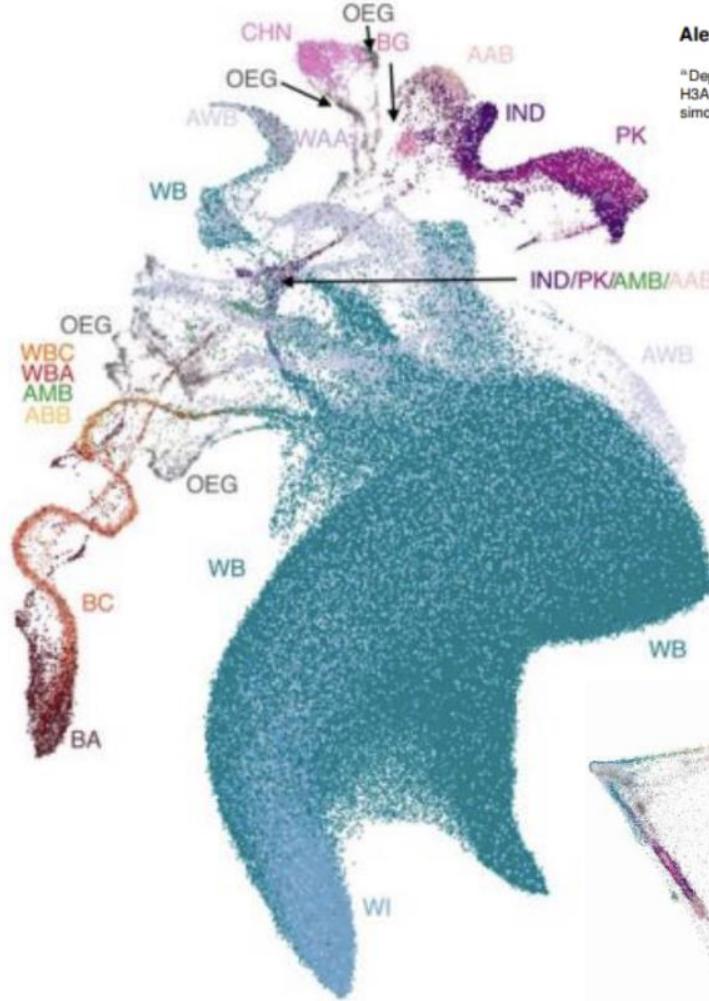


UMAP



UMAP vs t-SNE on population genomics

UMAP

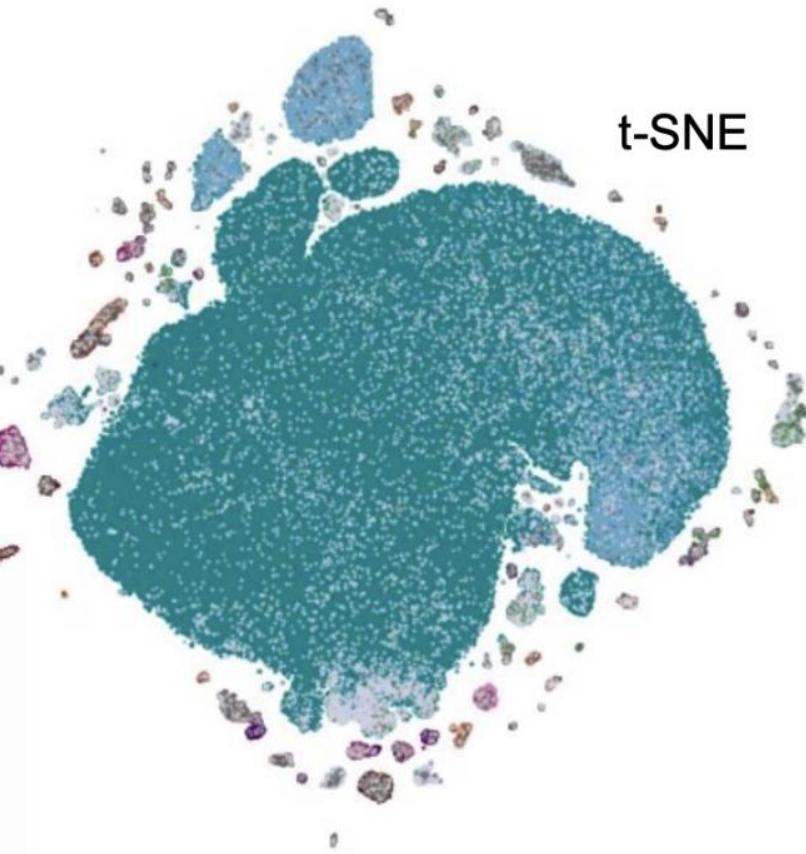


Revealing multi-scale population structure in large cohorts

Alex Diaz-Papkovich^{a,b}, Luke Anderson-Trocme^{b,c}, and Simon Gravel^{b,c,1}

^aDepartment of Quantitative Life Sciences, McGill University, Montreal, QC, H3A 0G1 Canada; ^bMcGill University and Genome Quebec Innovation Centre, Montreal, QC, H3A 0G1, Canada; ^cDepartment of Human Genetics, McGill University, Montreal, QC, H3A 0G1, Canada. ¹To whom correspondence should be addressed. E-mail: simon.gravel@mcgill.ca

t-SNE



PCA

Embedding Projector

DATA

5 tensors found

Word2Vec 10K

Label by

word

Color by

No color map

Edit by

word

Tag selection as

Load

Publish

Download

Label

Sphereize data 

Checkpoint: Demo datasets

Metadata: oss_data/word2vec_10000_200d

UMAP

T-SNE

PCA

CUSTOM

Dimension

2D



3D

Neighbors 



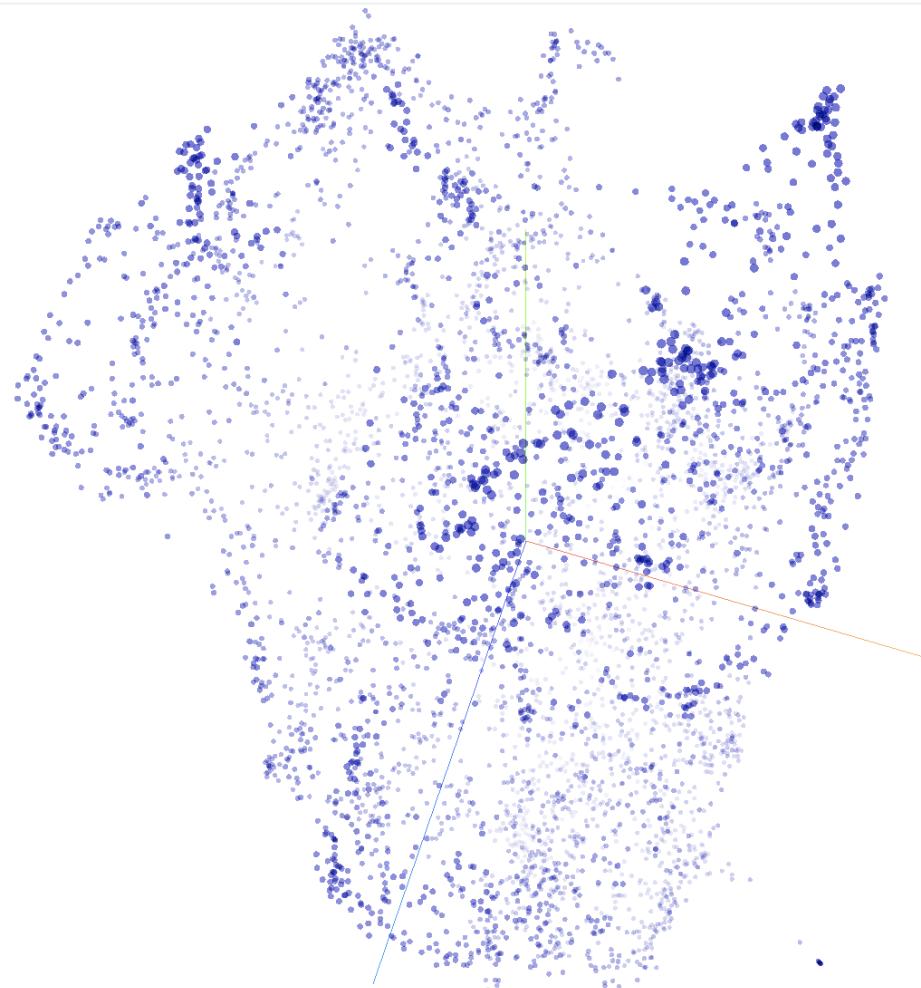
15

Run

For faster results, the data will be sampled down to 5,000 points.

 [Learn more about UMAP.](#)

    Points: 10000 | Dimension: 200

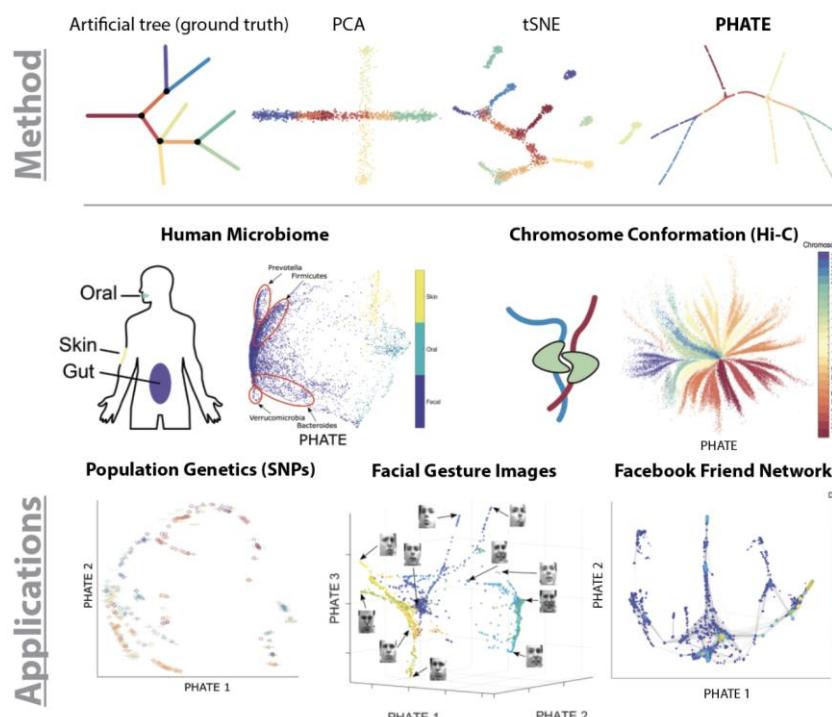


<https://projector.tensorflow.org/>

Other improvements

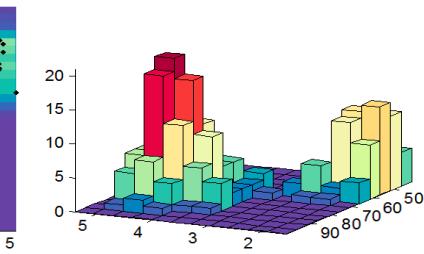
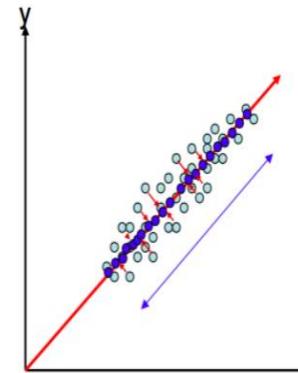
PHATE (<https://www.krishnaswamylab.org/projects/phate>)

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



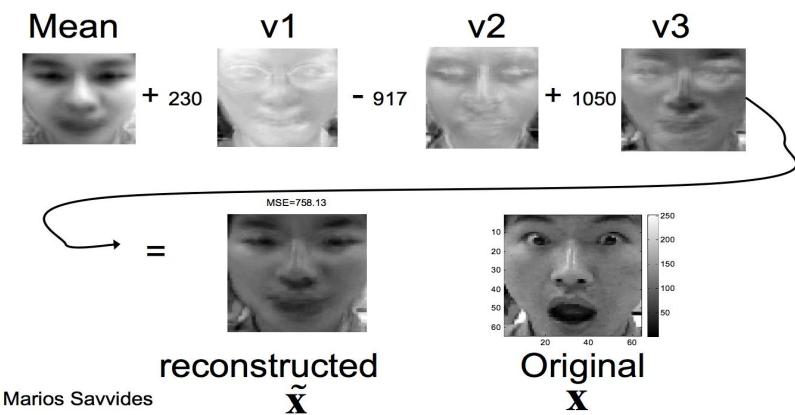
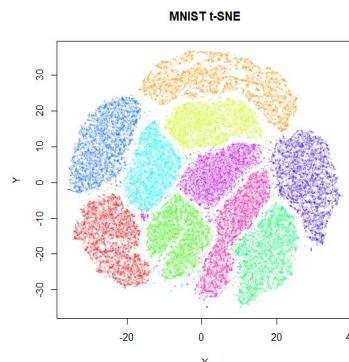
Summary

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



- Next time SVM

14:22



Marios Savvides

reconstructed
 $\tilde{\mathbf{X}}$

Original
 \mathbf{X}