DIMENSIONALITY REDUCTION AND VISUALIZATION

The curse of dimensionality

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

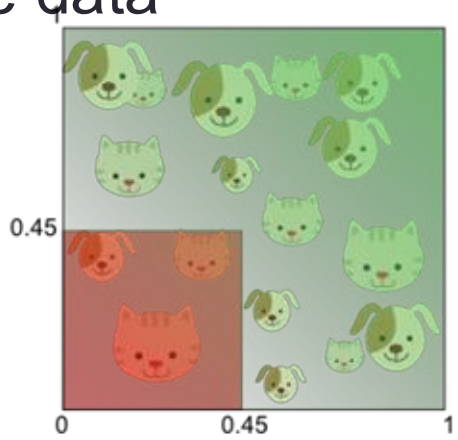
The Curse of Dimensionality

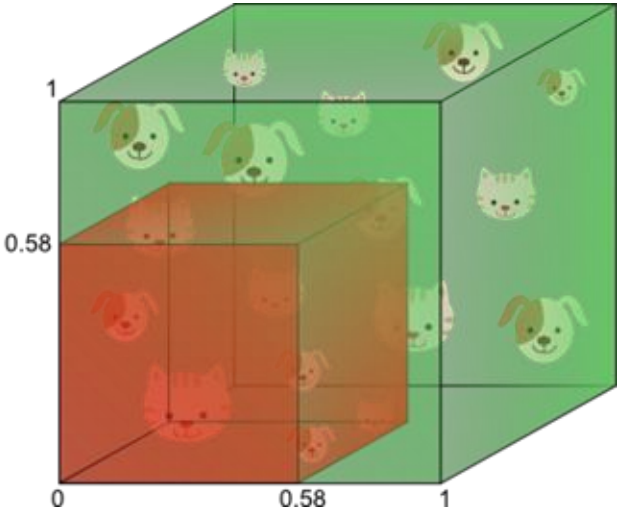
- Harder to visualize or see structure of
  - Verifying that data come from a straight line/plane needs n+1 data points
- Hard to search in high dimension – More runtime
- Need more data to get a get a good estimation of the data

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

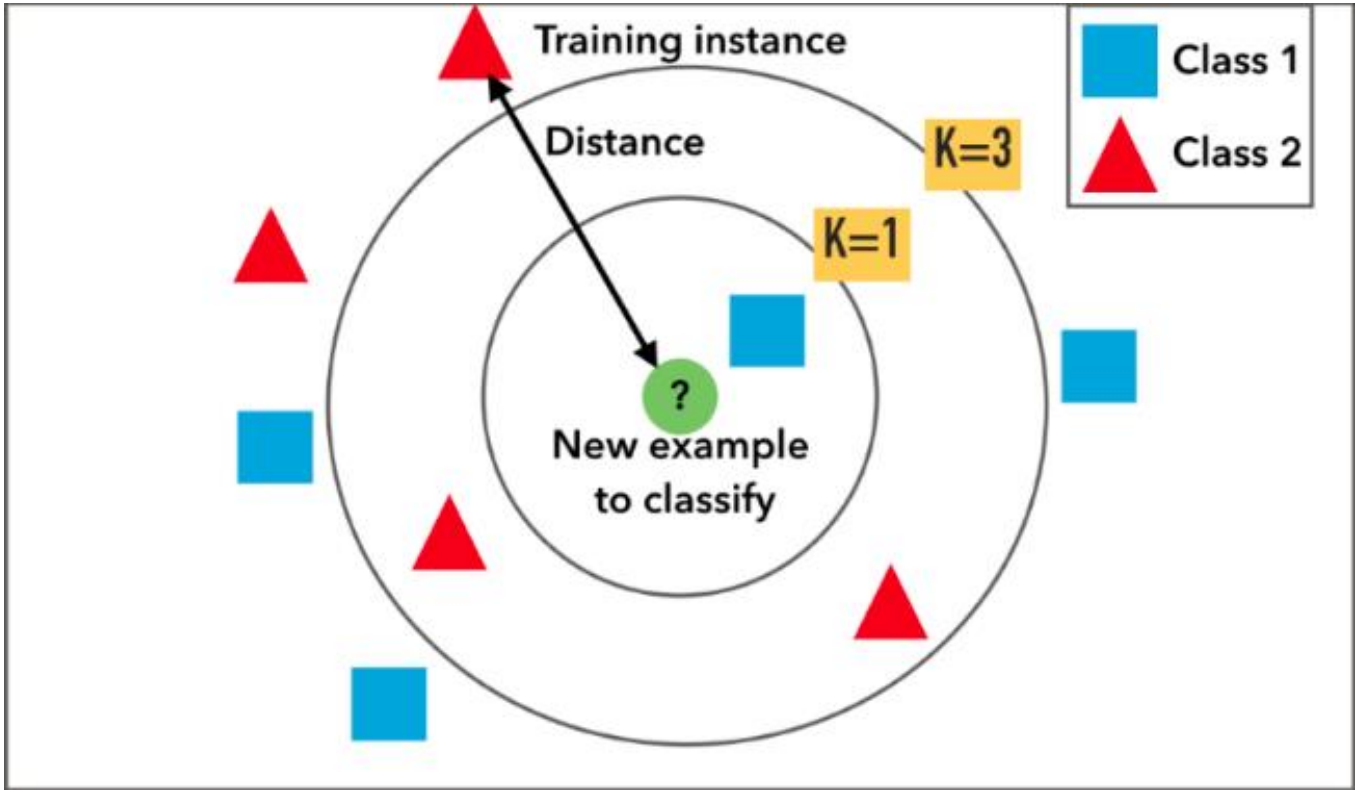




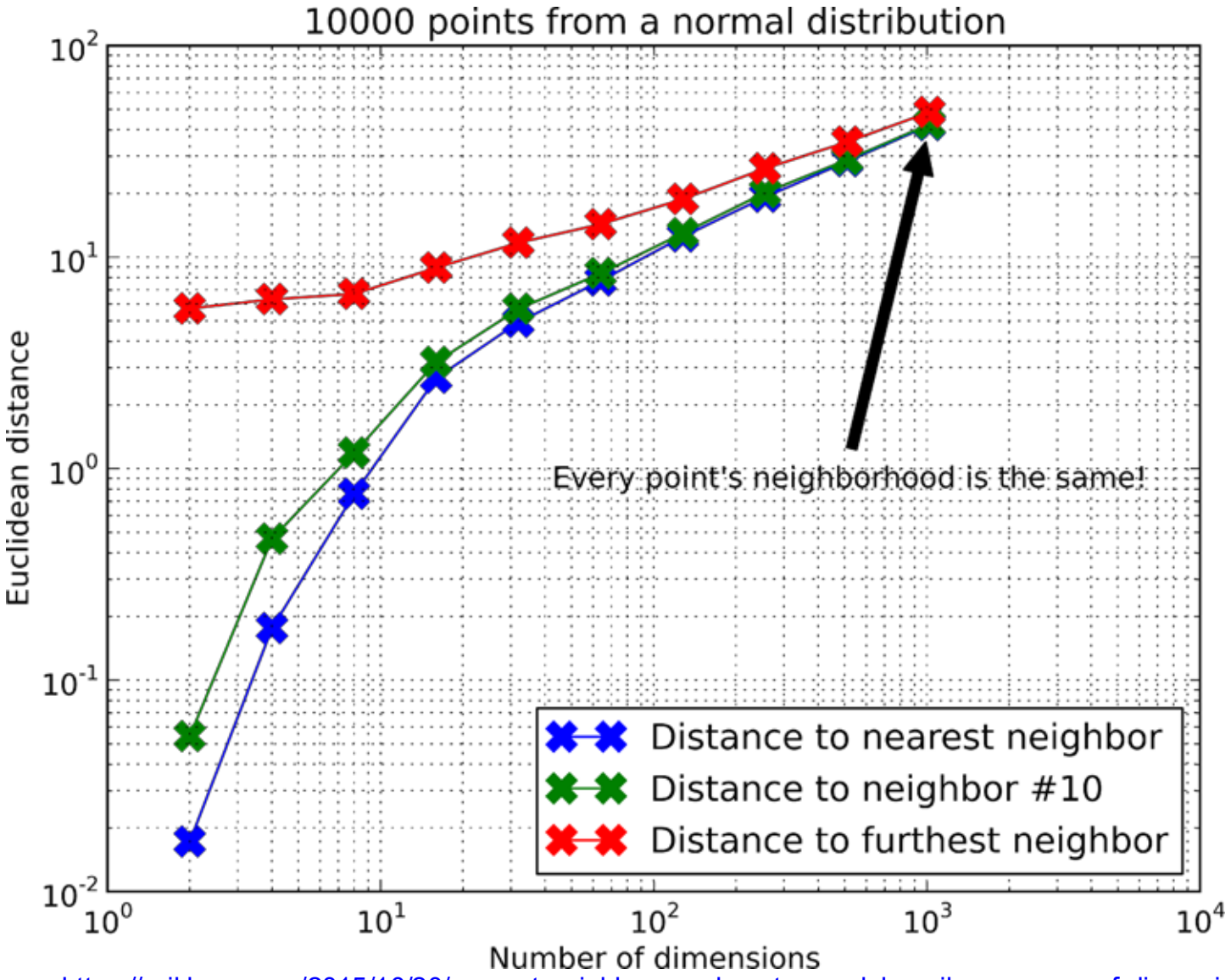
http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

K-Nearest Neighbor Classifier

- 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?



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

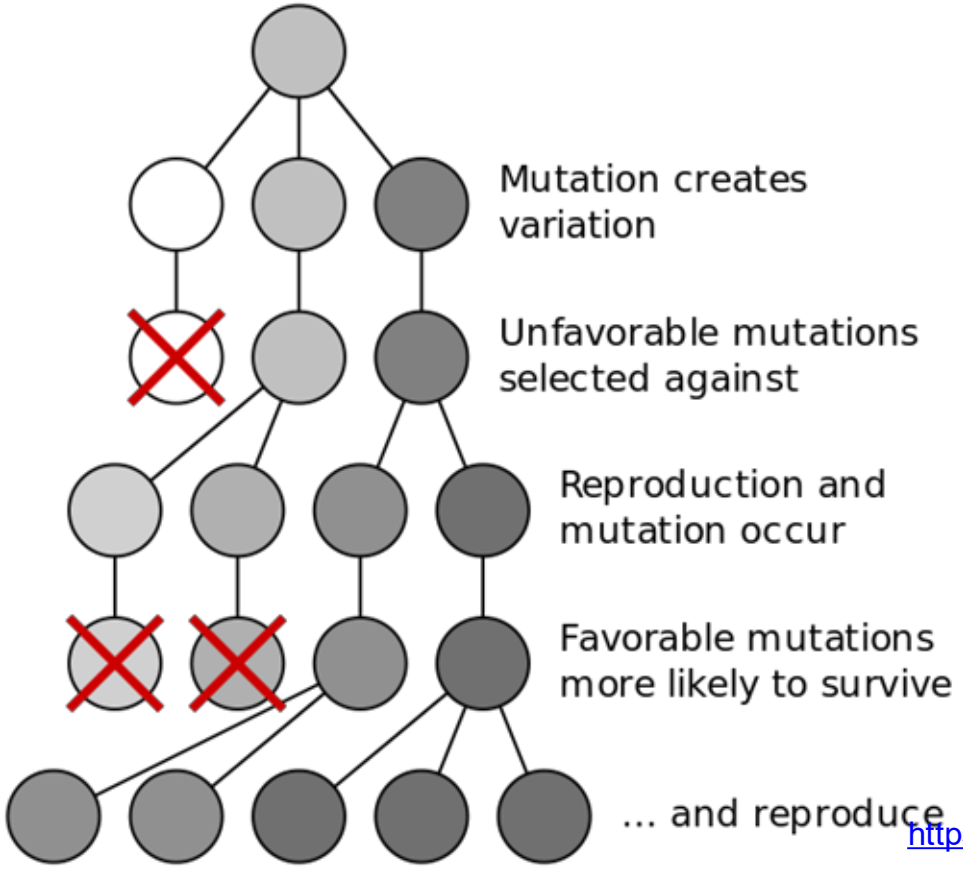
- Hackathon level (time limit days-a week)
  - Drop missing features
  - Low variance column
    - A feature that is a constant is useless. Tricky in practice
  - Forward or backward feature elimination
    - Greedy algorithm: create a simple classifier with n-1 features, n times. Find which one has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

Feature selection

- Proper methods
  - Algorithm that handles high dimension well and do selection as a by product
    - 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



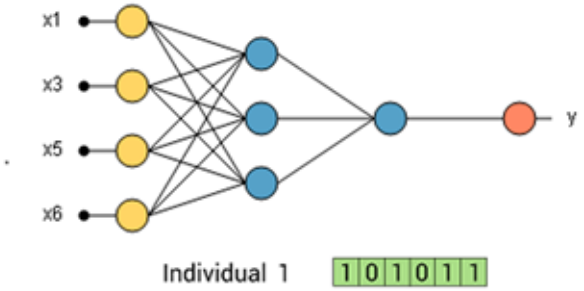
https://elitedatascience.com/dimensionality-reduction-algorithms

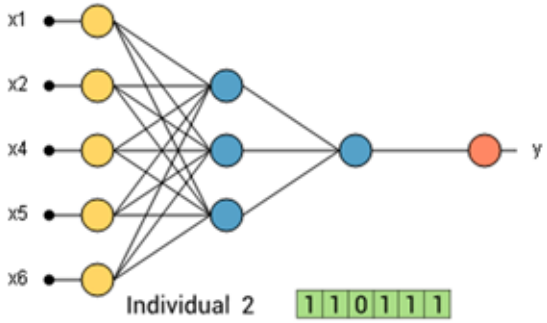
Genetic Algorithm

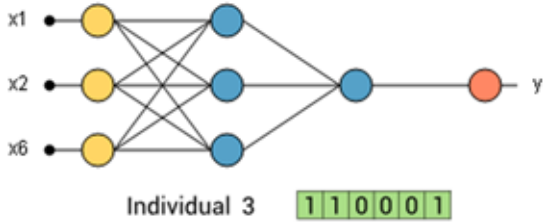
- Initialization
  - Create N classifiers, each using different subset of features
- Selection process
  - Rank the N classifiers according to some criterion, kill the lower half
- Crossover
  - The remaining classifier breeds offsprings by selecting traits from the parents
- Mutation
  - The offsprings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

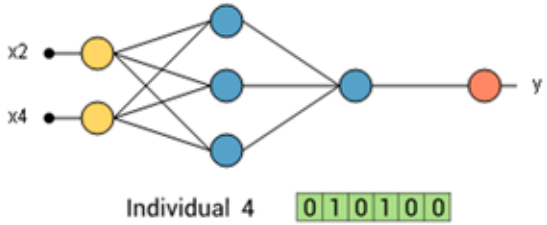
Initialization

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









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

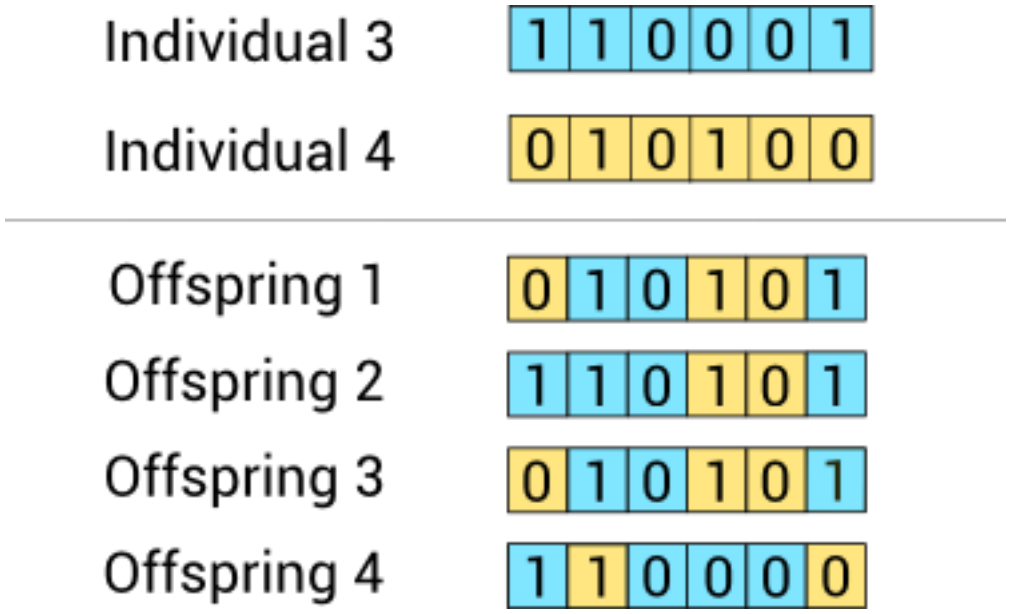
Selection process

- 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

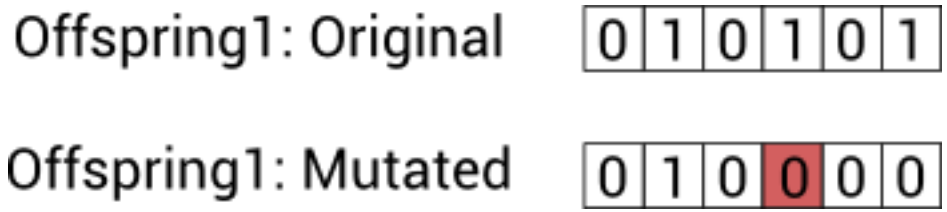
Crossover

- Breed offsprings by randomly select genes from parents



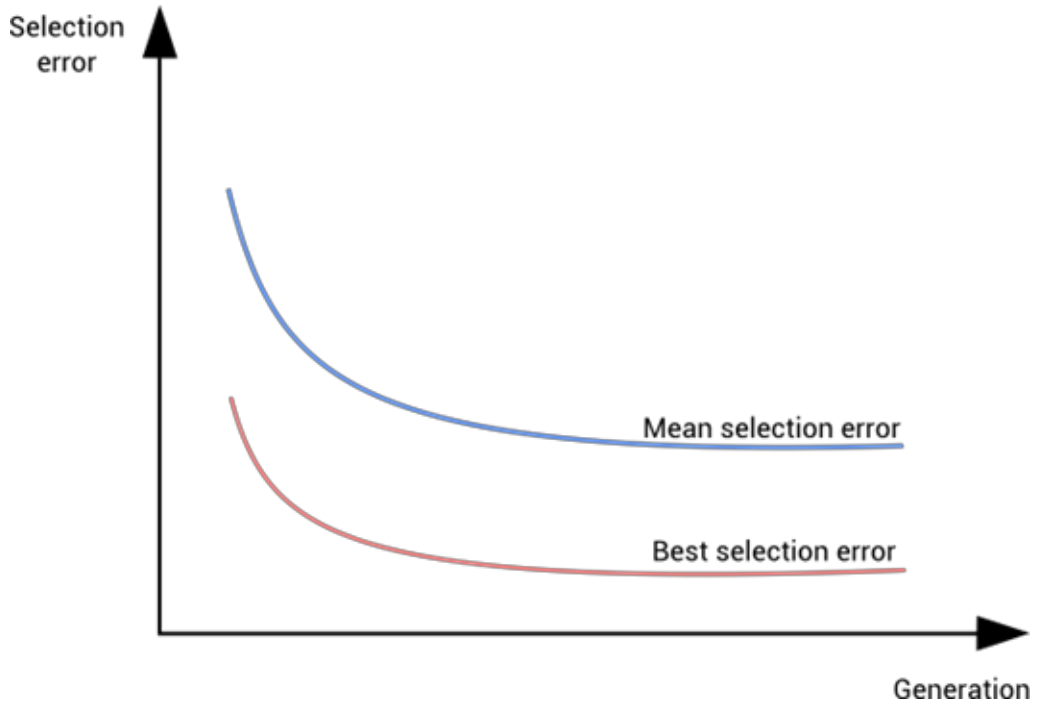
Mutation

- 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



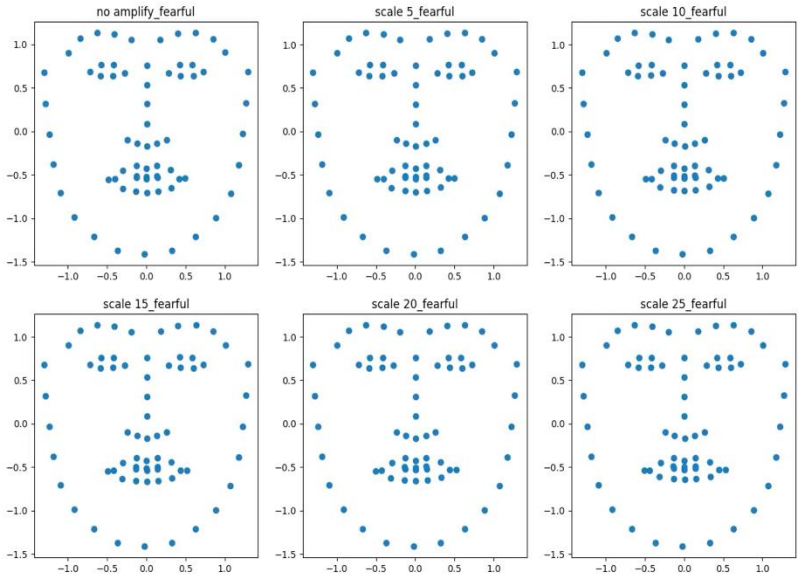
Performance

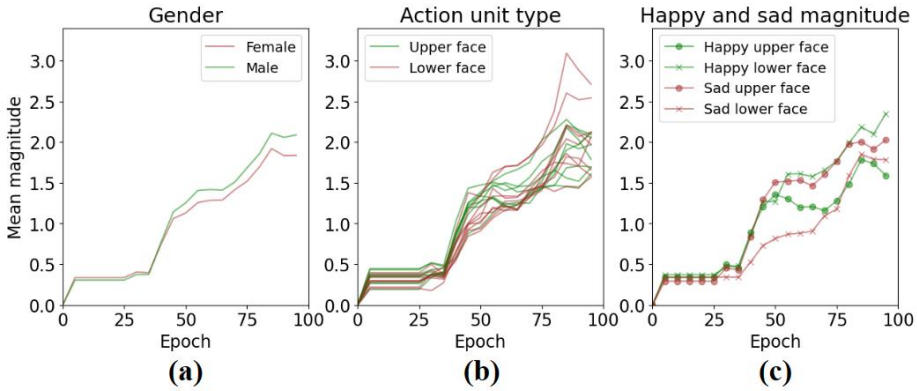
- Usually performs well. The general population usually gets better (mean). The best performing (individual) also gets better after each generation



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, ICASSP 2024

Feature transformation

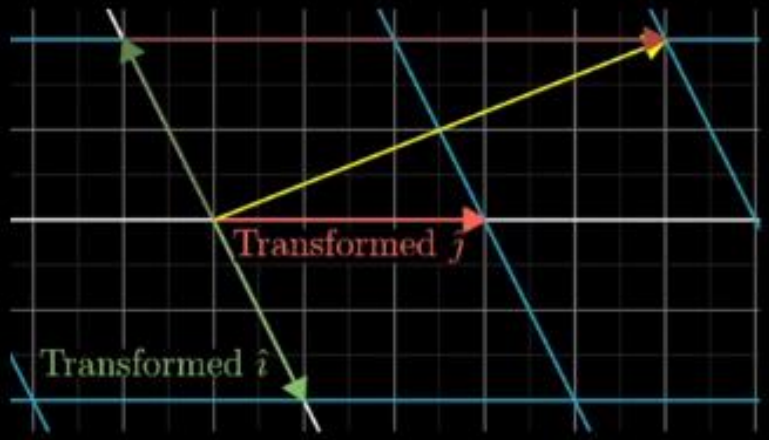
- Principal Component Analysis
- Linear Discriminant Analysis (NOT Latent Dirichlet Allocation)
- Random Projections

Linear Algebra Review

- Matrix as a sequence of column vectors

"2x2 Matrix"

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



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

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

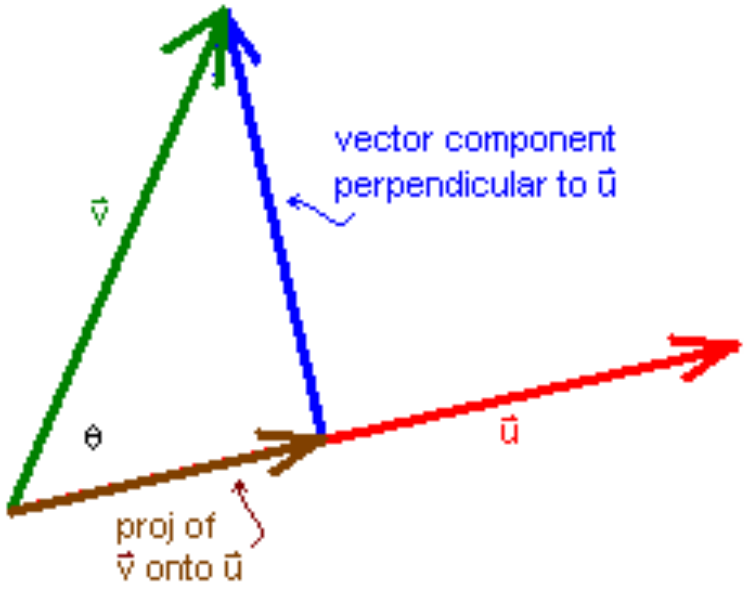
Linear Algebra Review

- 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, 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)

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

Covariance matrix (vector view)

- Given a set of datapoints $x_1\ x_2\ \ldots\ x_n$
- The covariance matrix based on the data can be written as

$E[\ x\ x^T\ ] = [\ \square + \square + \ldots. + \square\ ]/\ n$

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

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,

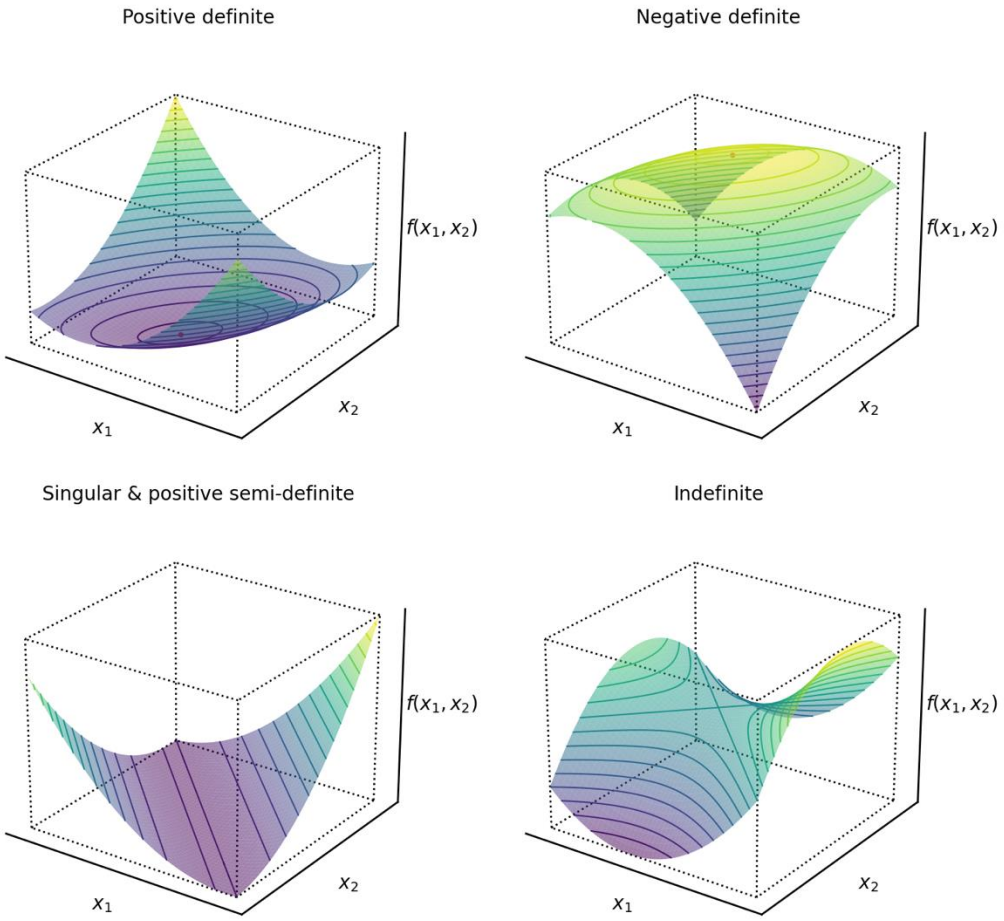
$$M \text{ positive semi-definite} \iff \mathbf{x}^{\mathrm{T}} M\mathbf{x} \geq 0 \text{ for all } \mathbf{x} \in \mathbb{R}^n$$

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

- Covariance matrix is positive semi-definite and symmetric.
  - Semi-definite because 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 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,

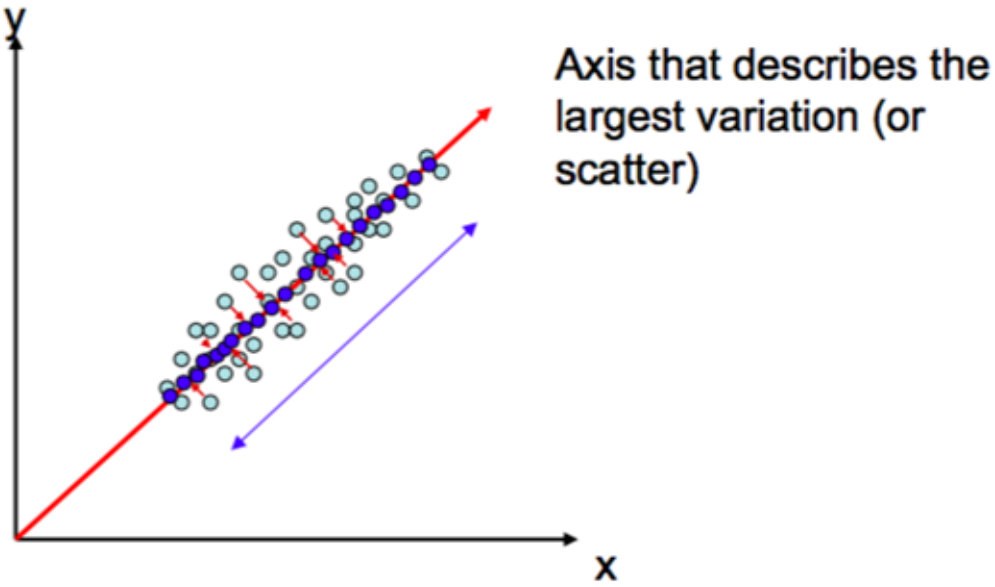
$$M \text{ positive semi-definite} \iff \mathbf{x}^{\mathrm{T}} M\mathbf{x} \geq 0 \text{ for all } \mathbf{x} \in \mathbb{R}^n$$

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

- Covariance matrix is positive semi-definite and symmetric.
  - 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

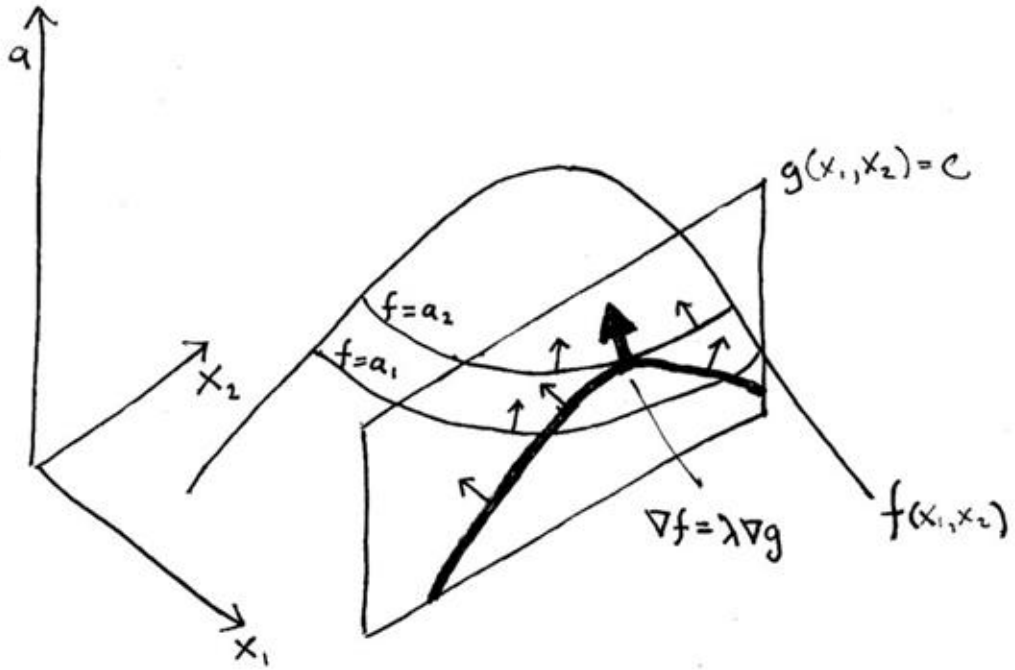
Trace properties

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. $$\nabla_{A^T} trABA^TC = B^TA^TC^T + BA^TC$$

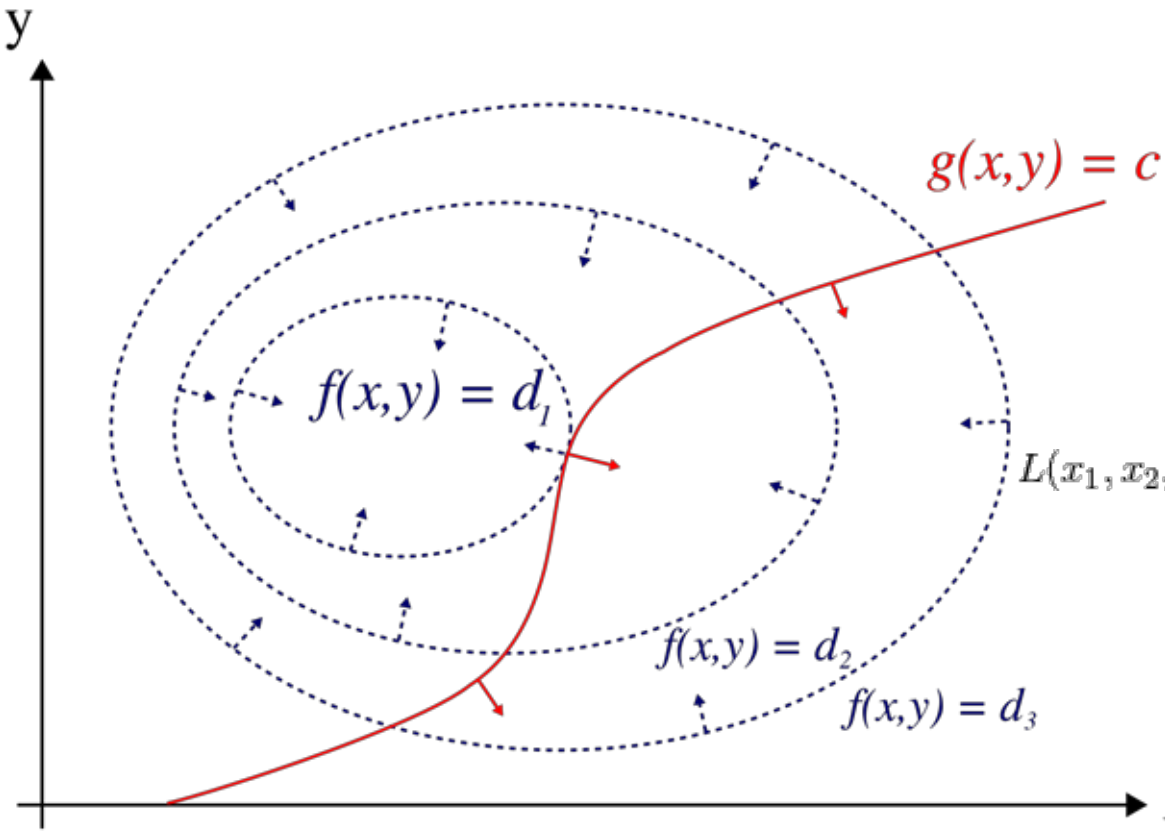
Lagrange Multiplier

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.

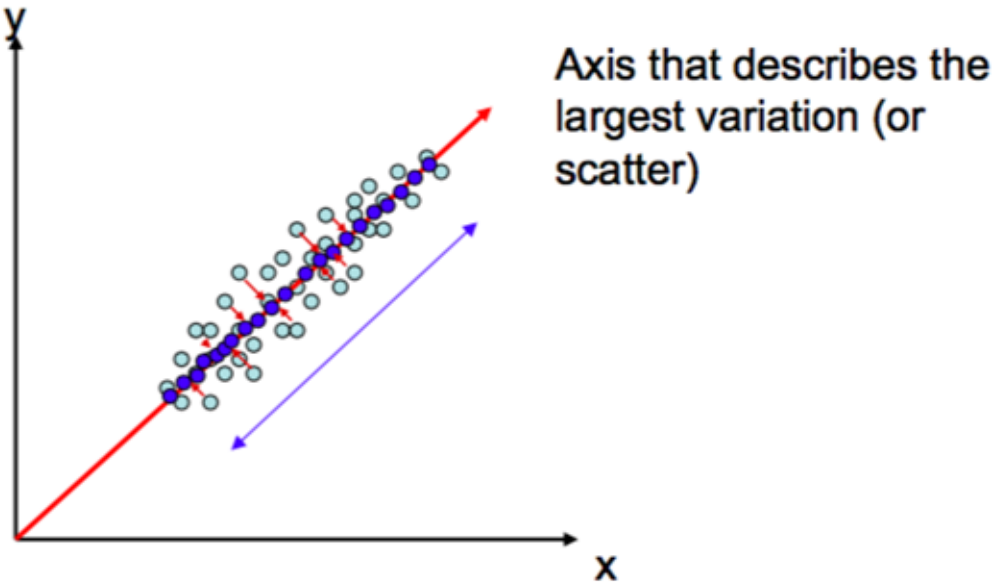
PCA

- 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:

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

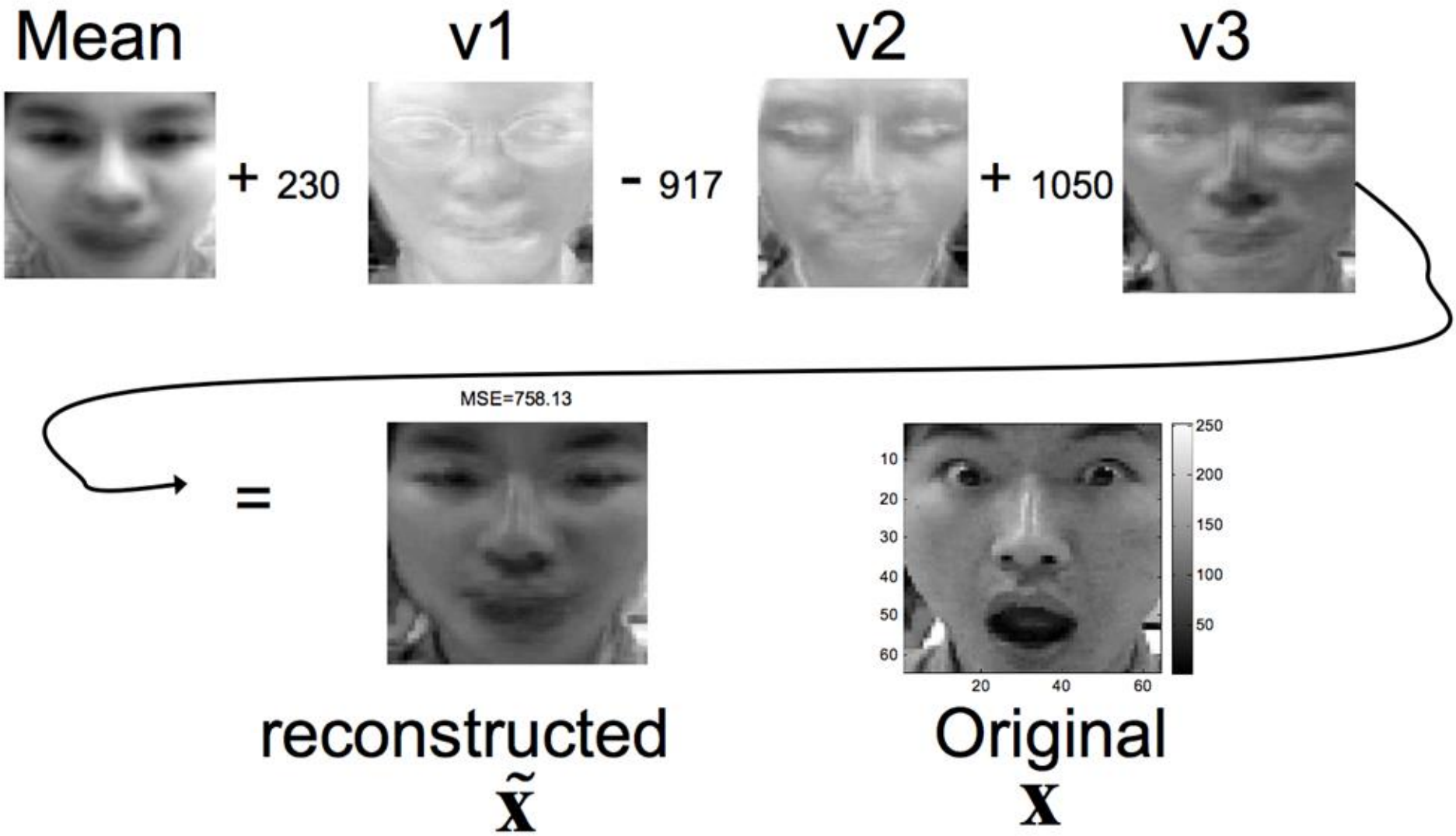
Finding the weights

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

- 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 $\Sigma v = \lambda v$

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

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

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

- $X^T X$ is a gram of inner-product matrix. Its size is 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

Summary

- Feature selection and dimensionality reduction techniques help combats curse of dimensionality.
- PCA tries to maximize reconstruction of the data while using the least amount of dimension.