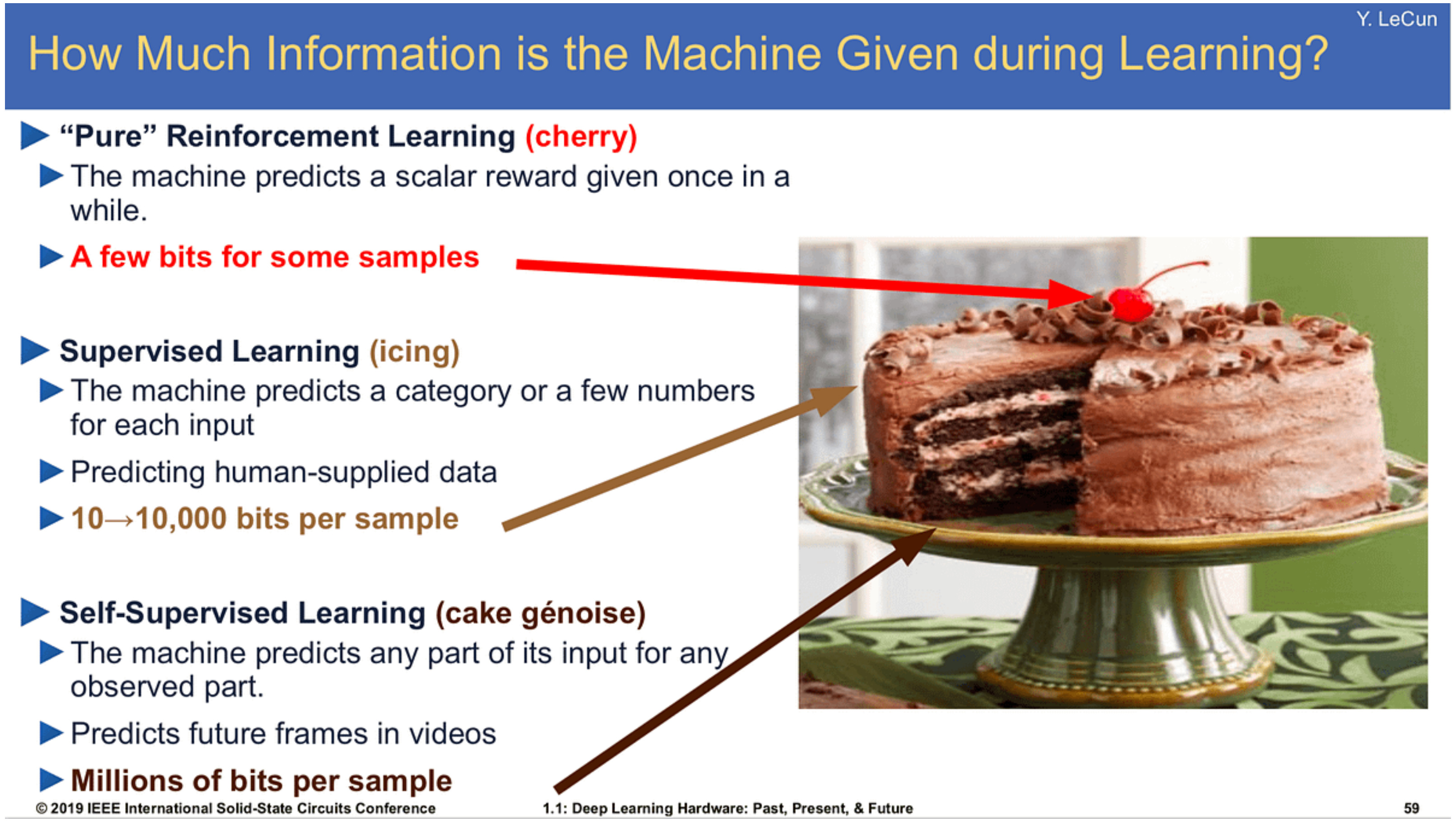
Unsupervised learning - the big picture

- Supervised learning: $(\boldsymbol{x}_i, y_i)$
- Unsupervised learning: $(\boldsymbol{x}_i)$
- Reinforcement learning: $s_i, a_i \rightarrow r_i$

Yann LeCun's cake



Y. LeCun

How Much Information is the Machine Given during Learning?

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

© 2019 IEEE International Solid-State Circuits Conference 1.1: Deep Learning Hardware: Past, Present, & Future 59

The unsupervised learning dilemma

- We can acquire a lot of data $(\boldsymbol{x}_i)$
- But what can we do with it?
- Basically, two main directions
  - Find a way to acquire labels automatically
  - Learn about the distribution from which the data was sampled $\boldsymbol{x} \sim P(X)$

Self-supervision

- Creating supervised data from unsupervised one by **inferring** labels
- Then, supervised learning algorithms can be used.

Example 1: Learning sequences

- Let us assume that the unsupervised data comes in form of a sequence $\boldsymbol{a}_0, \boldsymbol{a}_1 \ldots \boldsymbol{a}_i, \boldsymbol{a}_{i+1} \ldots$
  - Example: text, voice, video, stock prices
- Creating a supervised learning task:
- Prediction $\boldsymbol{x_i} = \{a_{i-3}, a_{i-2}, a_{i-1}, a_i\}$, $y = a_{i+1}$
  - Application: predicting next word, financial predictions...
- Cloze task $\boldsymbol{x_i} = \{a_{i-1}, a_{i-2}, a_i + 1\}$, $y = a_i$
  - Application: spell checking, anomaly detection

Example 2: Contrastive learning

- Transform the unsupervised data into classification data
- Assume all the data in your unsupervised dataset are positive examples

$$\{\boldsymbol{x}_0, \boldsymbol{x}_1 \ldots\} \rightarrow \{(\boldsymbol{x}_0, T), (\boldsymbol{x}_1, T) \ldots\}$$

- Now, generate a collection of data (e.g. randomly) and call those the negative ones

$$\{(\boldsymbol{x'}_0, F), (\boldsymbol{x'}_1, F) \ldots\}$$

- Can be used to teach an anomaly detector
- As a side effect, the learner often develops a model of the real data (an embedding)

Contrastive learning cont'd

- Further idea: the negative examples generated randomly are too obviously negative
    - Eg. positive examples: pictures of white horses
    - Negative examples: random noise
- Maybe we can simultaneously improve the way we are generating the negative examples, such that they are different from positive ones only in subtle ways
    - this leads to **generative adversarial networks (GANs)**

Example 3: Autoencoders

- Transform the unsupervised data into supervised ones by making the label and the input the same: $y_i = \boldsymbol{x}_i$
  - This obviously can be solved by a learner that copies the input to output
- It gets interesting, if we impose a bottleneck in the architecture of the learner
  - $\boldsymbol{z} = enc(\boldsymbol{x})$, $\hat{y} = dec(\boldsymbol{z})$, with $\boldsymbol{z}$ being the **latent encoding**
  - We can make $||z|| << ||x||$, achieving a sort of compression
- A famous example: **variational autoencoders**

Finding structure in data

- Having an unsupervised dataset we can make the assumption that it represents samples from an underlying distribution

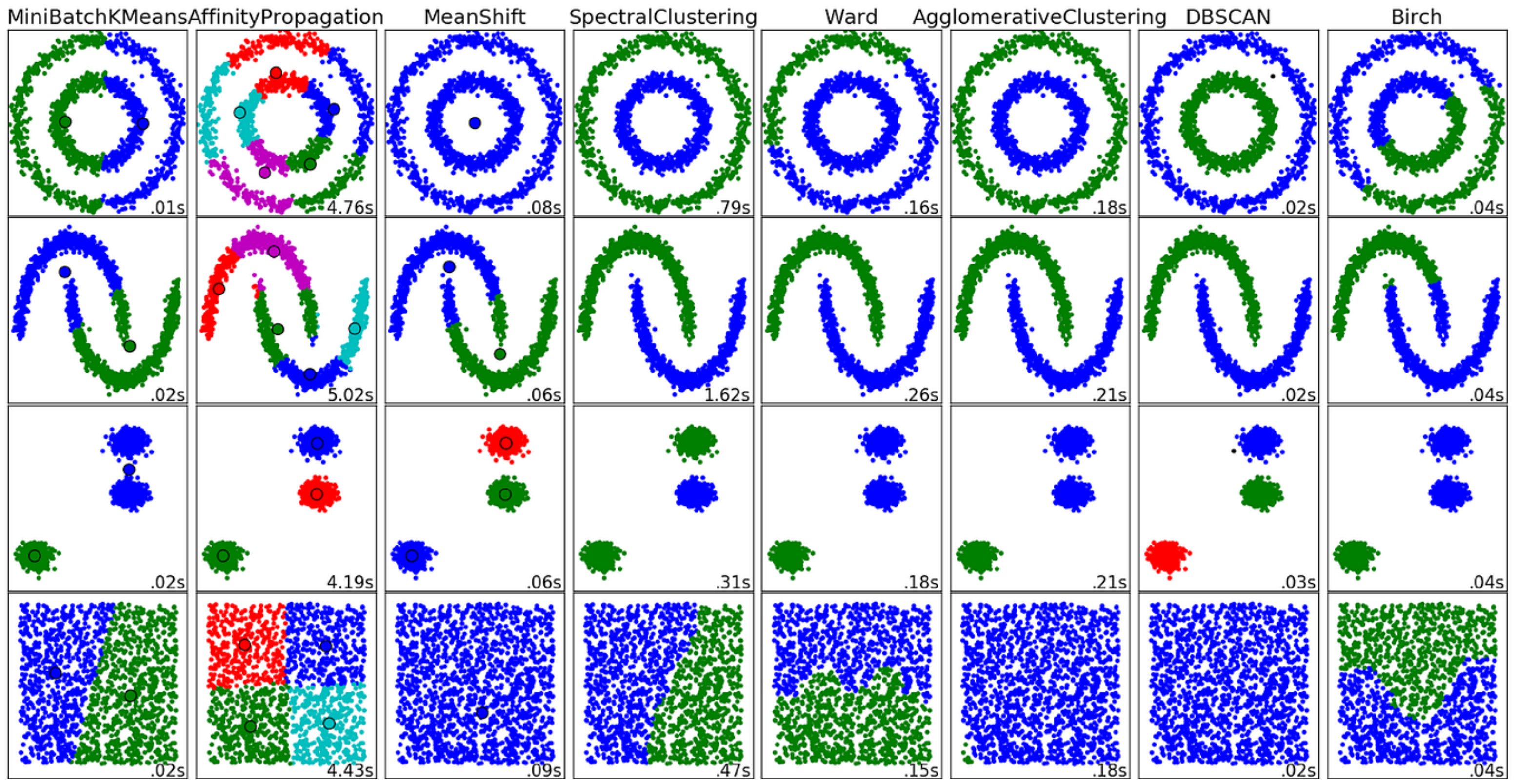
$$\boldsymbol{x}_i \sim P(X)$$

- We can learn things about this distribution
- Clustering
- Efficient representations
- Compression
- Manifold learning
- Disentangled representations
- ... etc

Clustering

- We assume that the data is pulled from a multinomial distribution
    - i.e. multiple, distinct peaks of probability
- We should be able to identify these peaks even if we don't have labels!
    - Group the data into **clusters**
    - Later, we might simply assign a label to each of these clusters
- Famous algorithms: k-means clustering, DBSCAN etc.

Clustering example



Latent representations with desirable properties

- We learn a latent representation and associated encoder $\boldsymbol{z} = enc(\boldsymbol{x})$
  - Such that, for some reason, we like $\boldsymbol{z}$ better than $\boldsymbol{x}$
- Examples:
  - $\boldsymbol{z}$ much smaller that $\boldsymbol{x}$ (compression)
  - $\boldsymbol{z}$ contains all the information needed for a task T and discards the rest
  - It is easier to learn in $\boldsymbol{z}$-space than in $\boldsymbol{x}$-space - for instance, things of interest are linearly separatable in $\boldsymbol{z}$
  - There are interesing mathematical properties in this representation eg. Paris - France + England = London (word2vec)

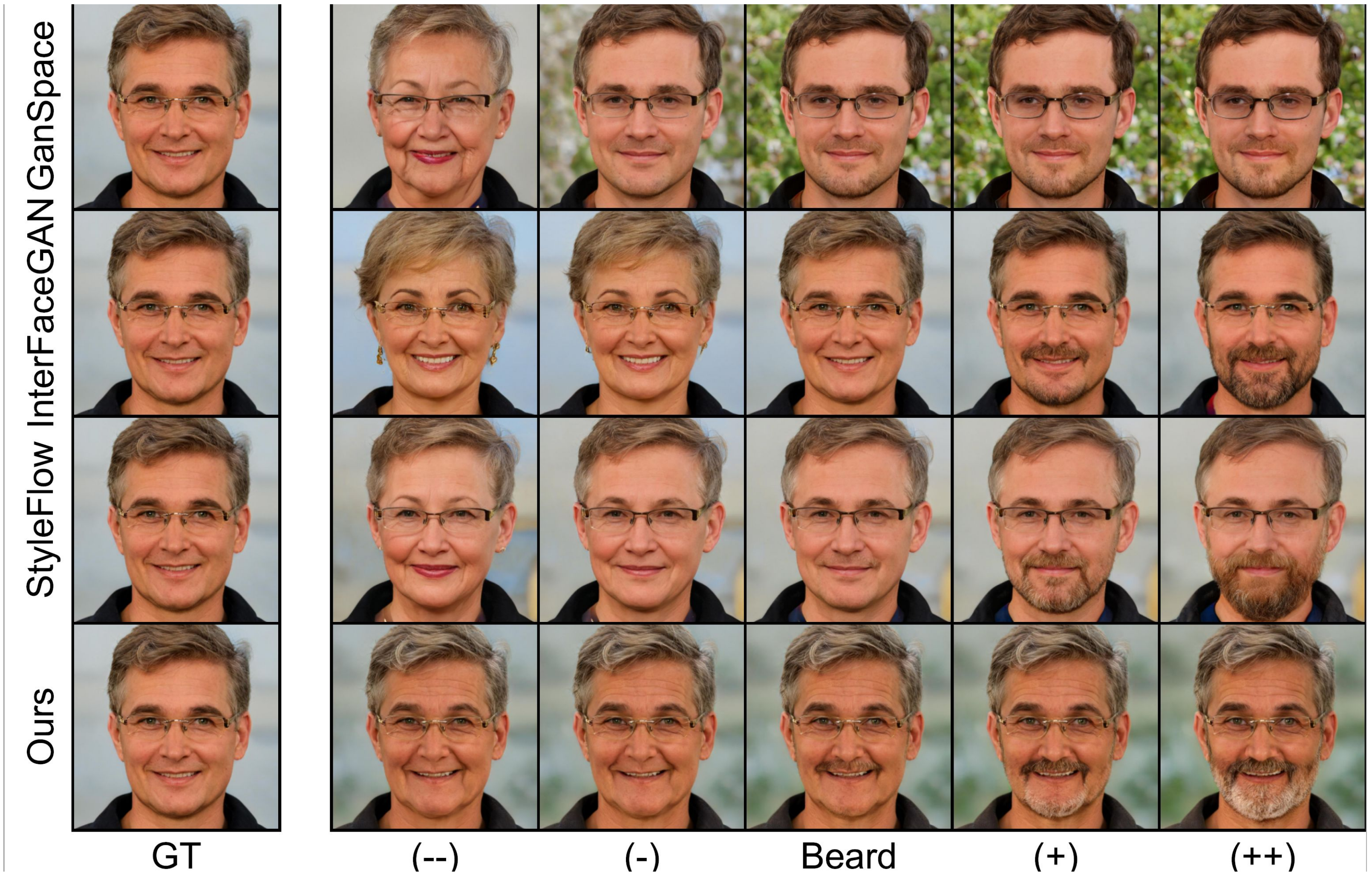
Manifold learning

- Often $\boldsymbol{x}$ might have a very high dimensionality
    - Eg. phone photos: 36 million dimensions
- However: if we generate a random picture it will almost surely be white noise!
- We assume that the data **we are interested in might** have a much smaller dimensionality
    - Sits in a lower dimensional **manifold** embedded in the high dimensional space
- If we can identify it and we can learn how to move around while staying in the manifold, it would unlock many interesting applications

Disentangled representations

- We want to find a representation z such that
  - Nearby z encode objects that are perceived as having the same identity
  - Moving in certain directions in z is perceived as changing certain identifiable **attributes**, and the farther we move, the more the attribute changes
- What "identity" means and what an "attribute" means is domain dependent, and often has to reflect human preference
- Example domain: human faces
  - Identity: human identity as perceived by humans
  - Attributes: age, facial hair, smile, hair color etc.

Example: Disentangled representation in StyleGAN



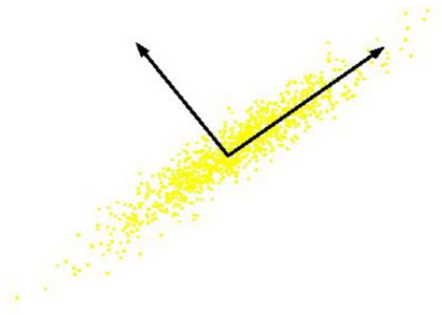
A historical perspective - principal component analysis

- A technique invented in 1909 by Karl Pearson
- Re-invented a number of times under a number of names in various disciplines
  - signal processing
  - mechanical engineering
  - population genetics
  - meteorology
- The IQ is the principal component of a number of related intelligence measures

PCA

Principal Component Analysis (PCA)

What is PCA: Unsupervised technique for extracting variance structure from high dimensional datasets.



- PCA is an orthogonal projection or transformation of the data into a (possibly lower dimensional) subspace so that the variance of the projected data is maximized.

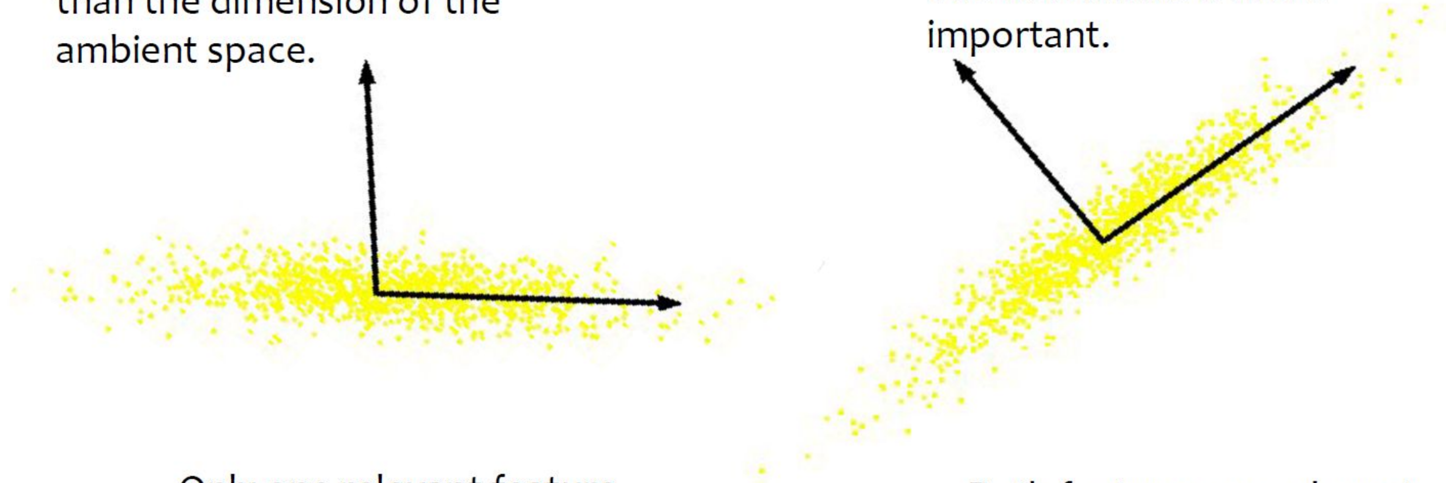
Slide from Nina Balcan

PCA

Principal Component Analysis (PCA)

Intrinsically lower dimensional than the dimension of the ambient space.

If we rotate data, again only one coordinate is more important.



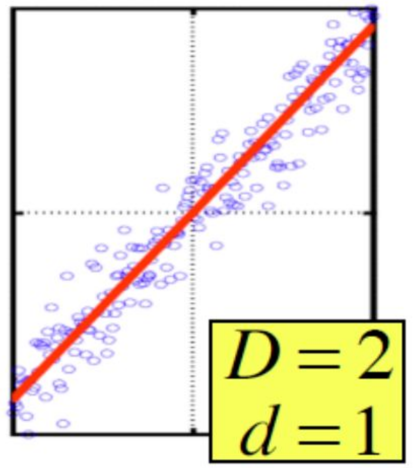
Only one relevant feature

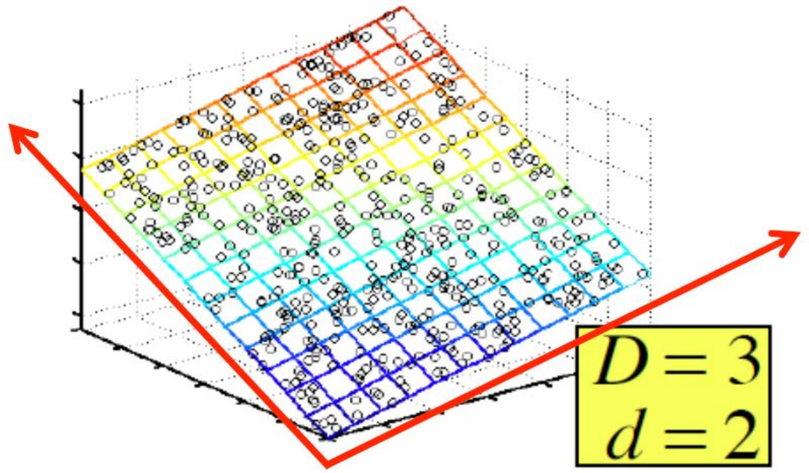
Both features are relevant

Question: Can we transform the features so that we only need to preserve one latent feature?

PCA

Principal Component Analysis (PCA)





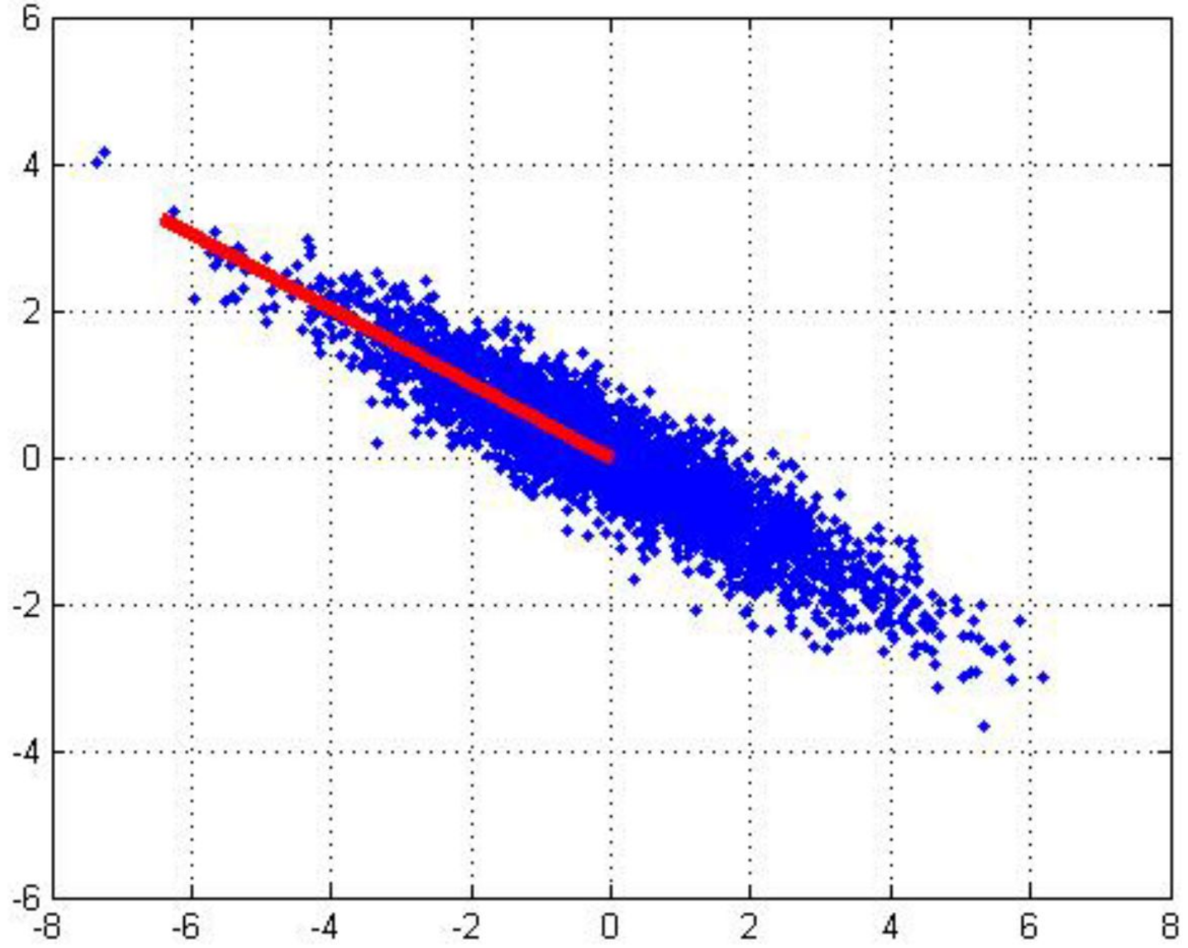
In case where data lies on or near a low d-dimensional linear subspace, axes of this subspace are an effective representation of the data.

Identifying the axes is known as Principal Components Analysis, and can be obtained by using classic matrix computation tools (Eigen or Singular Value Decomposition).

Slide from Nina Balcan

PCA

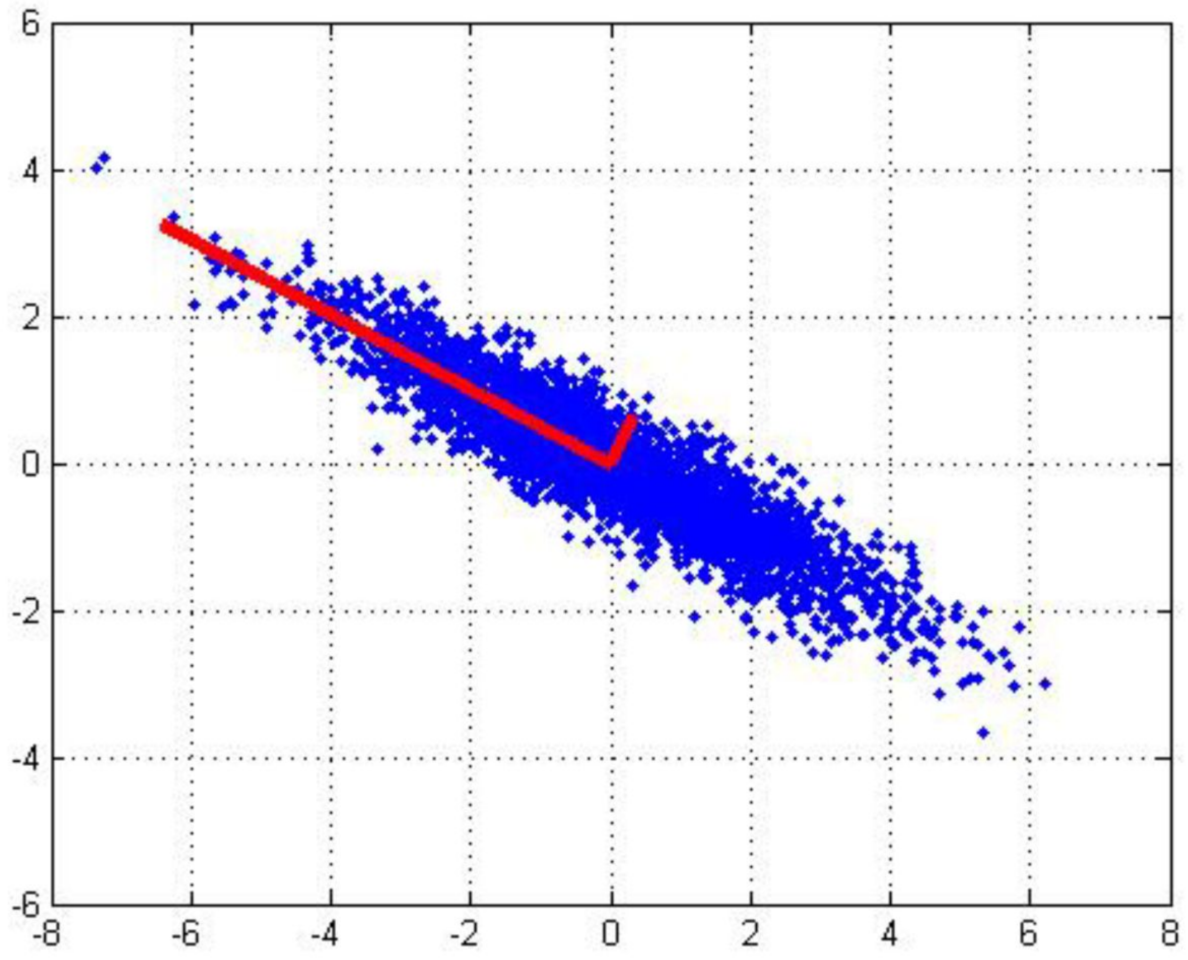
1st PCA axis



Slide from Barnabas Poczos

PCA

2nd PCA axis

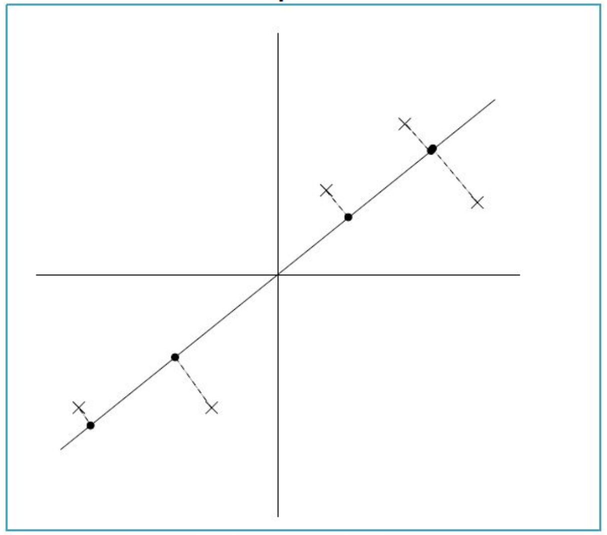


Slide from Barnabas Poczos

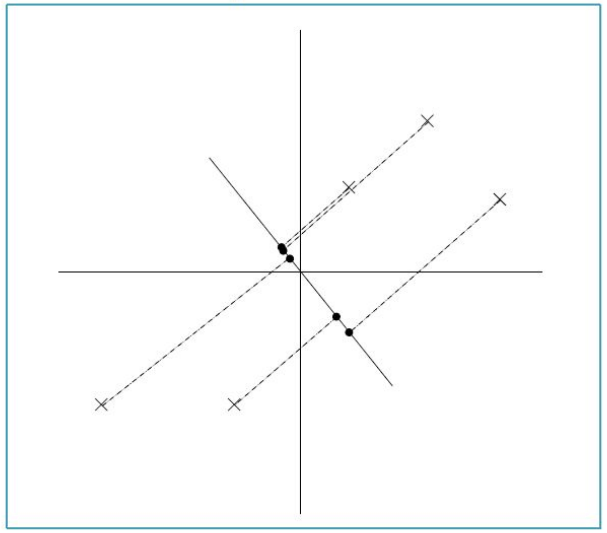
Maximizing the Variance

- Consider the two projections below
- Which maximizes the variance?

Option A

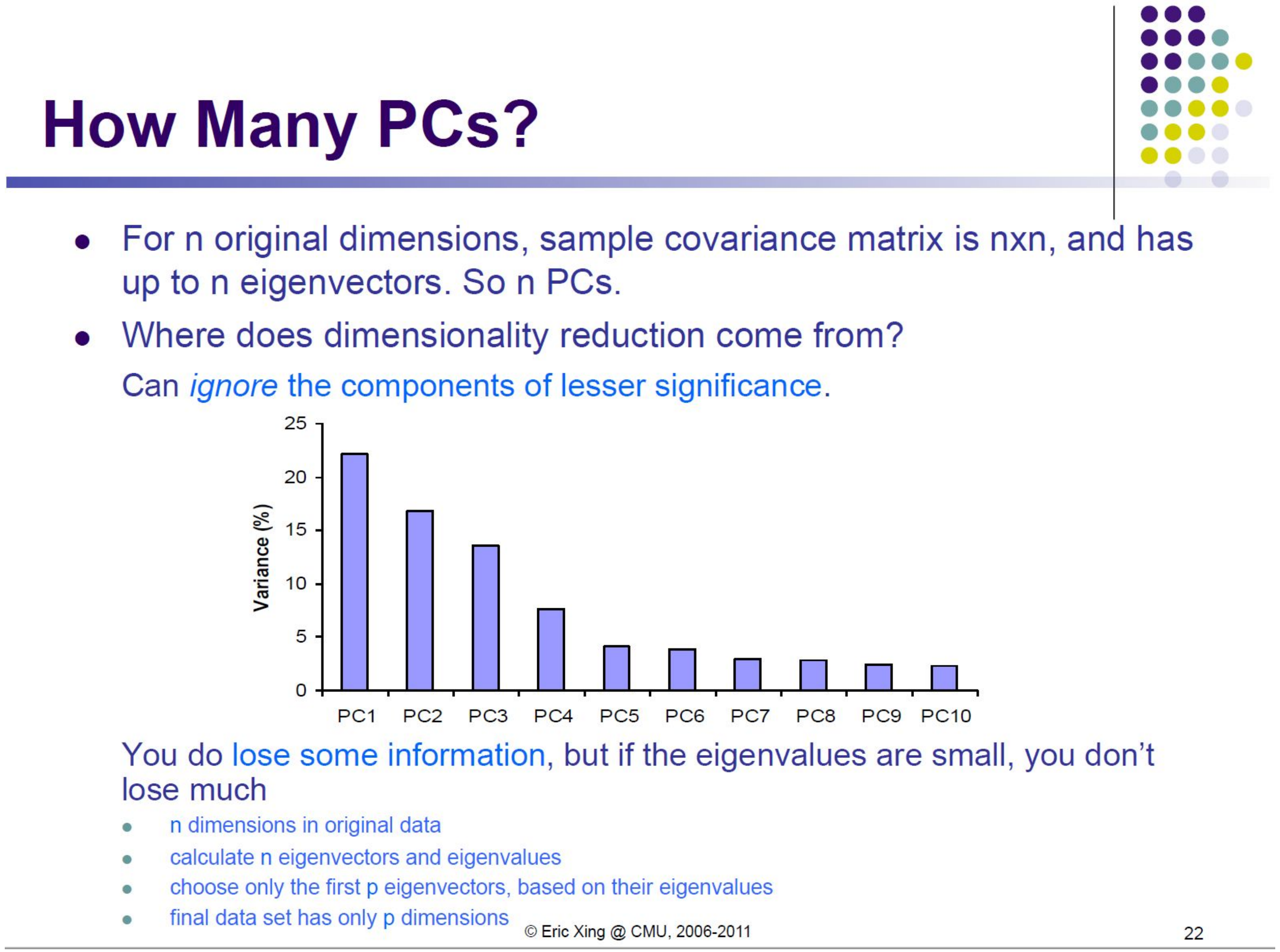


Option B



Figures from Andrew Ng (CS229 Lecture Notes)

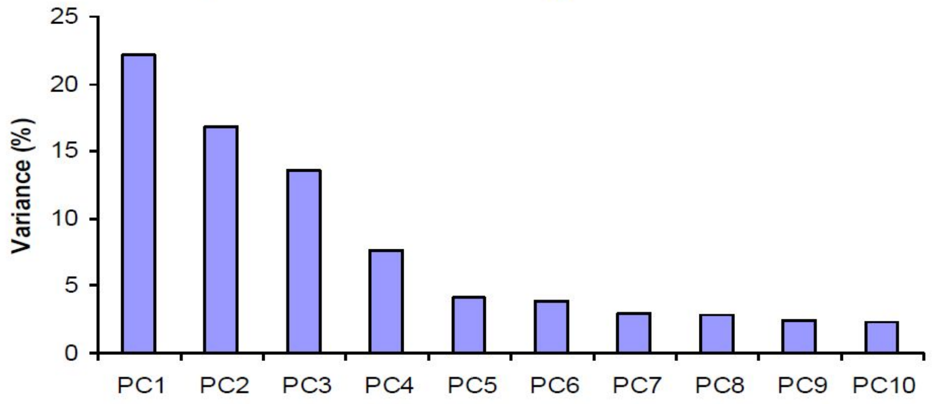
PCA



How Many PCs?

- For n original dimensions, sample covariance matrix is nxn, and has up to n eigenvectors. So n PCs.
- Where does dimensionality reduction come from?

Can *ignore* the components of lesser significance.



You do lose some information, but if the eigenvalues are small, you don't lose much

- n dimensions in original data
- calculate n eigenvectors and eigenvalues
- choose only the first p eigenvectors, based on their eigenvalues
- final data set has only p dimensions

PCA

PCA: Two Interpretations

E.g., for the first component.

Maximum Variance Direction: 1st PC a vector v such that projection on to this vector capture maximum variance in the data (out of all possible one dimensional projections)

$$\frac{1}{n}\sum_{i=1}^{n}(\mathbf{v}^T\mathbf{x}_i)^2 = \mathbf{v}^T\mathbf{X}\mathbf{X}^T\mathbf{v}$$

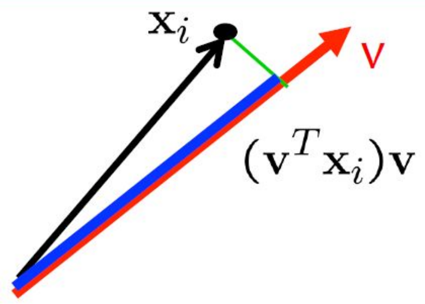
Minimum Reconstruction Error: 1st PC a vector v such that projection on to this vector yields minimum MSE reconstruction

$$\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i - (\mathbf{v}^T\mathbf{x}_i)\mathbf{v}\|^2$$

blue^2 + green^2 = black^2

black^2 is fixed (it's just the data)

So, maximizing blue^2 is equivalent to minimizing green^2



Slide from Nina Balcan

Applications of PCA in machine learning

- Dimensionality reduction
  - Reducing the number of dimensions by discarding the components of lesser significance
- Visualization / explanation
- More efficient use of resources
  - Perform computations on smaller dimensions
- Statistical improvements
  - Fewer dimensions might have better generalization
- Noise removal
  - An image reconstructed from PCA might be less noisy.

Perspective from 2020s

- Comparatively, PCA has a lesser importance in our current techniques
- Assumption of linearity is a big problem
    - Even if the data sits on a lower dimensional manifold, it is unlikely to be linear
- Does not scale very well (cubic complexity in samples or dimensions)
- The PCA transformation creates new features that are often less explainable than the original ones.