https://www.cs.ubc.ca/~fwood/CS340/

Lecture XXV - Boosting

https://www.cs.ubc.ca/~fwood/CS340/lectures/L25.pdf

Lecture XXVI - MLE and MAP

`https://www.cs.ubc.ca/~fwood/CS340/lectures/L26.pdf`

Lecture XXVII - Principal Component Analysis

https://www.cs.ubc.ca/~fwood/CS340/lectures/L27.pdf

Part 4: Latent-Factor Models

"Part weights" are a change of basis from x_i to some z_i. But in high dimensions it may be hard to find a basis....

Part 4 is about learning the basis from the data.

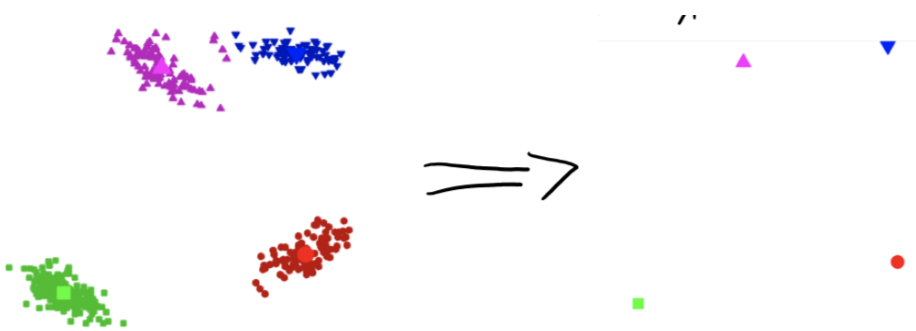
$$X \rightarrow \boxed{\text{Latent-factor model}} \rightarrow Z$$

Why?

- Supervised learning: We could use the part weights as our features
- Outlier detection: it may be an outlier if it is not a combination of the usual parts
- Dimension reduction: compress the data into a limited number of part weights
- Visualization: if we have only 2 part weights, we can view the data as a scatter plot
- Interpretation: we can try and figure out what the "parts" represent.

Recall using k-means for vector quantization

- Run k-means to find a set of "means" w_c.
- This gives a cluster $\hat{y}_i$ for each object i.
- Replace features x_i by mean of cluster: $\hat{x}_i \approx w_{\hat{y}_i}$



This can be viewed as a (very bad) latent-factor model.

Vector Quantization (VQ) as Latent-Factor Model

When $d = 3$, we could write x_i exactly as:

$$x_i = z_{i1} \underbrace{\begin{bmatrix}1\\0\\0\end{bmatrix}}_{\text{"part 1"}} + z_{i2} \underbrace{\begin{bmatrix}0\\1\\0\end{bmatrix}}_{\text{"part 2"}} + z_{i3} \underbrace{\begin{bmatrix}0\\0\\1\end{bmatrix}}_{\text{"part 3"}}$$

In this "pointless" latent-factor model we have $z_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \end{bmatrix}$. If x_i is in cluster 2, VQ approximates x_i by mean w_2 of cluster 2:

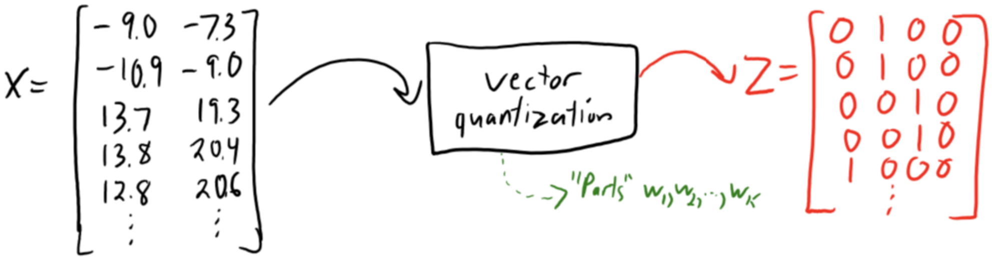
$$x_i \approx w_2 = 0w_1 + 1w_2 + 0w_3 + 0w_4$$

So in this example we would have $z_i = [0 \quad 1 \quad 0 \quad 0]$.

- The "parts" are the means from k-means
- VQ only uses one part (the "part" from the cluster)

Vector Quantization (VQ) vs PCA

Viewing vector quantization as a latent-factor model



Suppose we're doing supervised learning, and the colors are the true labels y: Then classification would be easy using this k-means basis Z.

But it only uses 1 part, it's just memorizing k points in x_i space. What we really want is a combination of parts.

PCA is a generalization that allows continuous z_i:

- It can have more than one non-zero
- It can use fractional weights and negative weights

$$Z = \begin{bmatrix} 0.2 & 1.6 \\ 0.3 & 1.5 \\ 0.1 & -2.7 \\ 0.2 & -2.7 \\ \vdots & \end{bmatrix}$$

PCA Notation (MEMORIZE)

PCA takes in a matrix X and an input k, and outputs two matrices:

$$Z = \underbrace{\begin{bmatrix} — z_1^T — \\ — z_2^T — \\ \vdots \\ — z_n^T — \end{bmatrix}}_{k} \Big\} n \qquad W = \underbrace{\begin{bmatrix} — w_1^T — \\ — w_2^T — \\ \vdots \\ — w_k^T — \end{bmatrix}}_{d} \Big\} k = \underbrace{\begin{bmatrix} | & | & & | \\ w^1 & w^2 & \cdots & w^d \\ | & | & & | \end{bmatrix}}_{d} \Big\} k$$

- For row c in W we use the notation w_c

  - Each w_c is a "part" (also called a "factor" or "principal component")
- For row i of Z, we use notation z_i
  - Each z_i is a set of "part weights" (or "factor loadings" or "features")
- For column j of W we use the notation w^j
  - Index j of all the k parts (value of pixel j in all the different parts)

With this notation, we can write our approximation of one x_{ij} as:

$$\begin{aligned}\hat{x}_{ij} &= z_{i1}w_{1j} + z_{i2}w_{2j} + \cdots + z_{ik}w_{kj} \\ &= \sum_{c=1}^{k} z_{ic}w_{cj} \\ &= (w^j)^T z_i \\ &= < w^j, z_i >\end{aligned}$$