UCMERCED

CSE 176 Introduction to Machine Learning

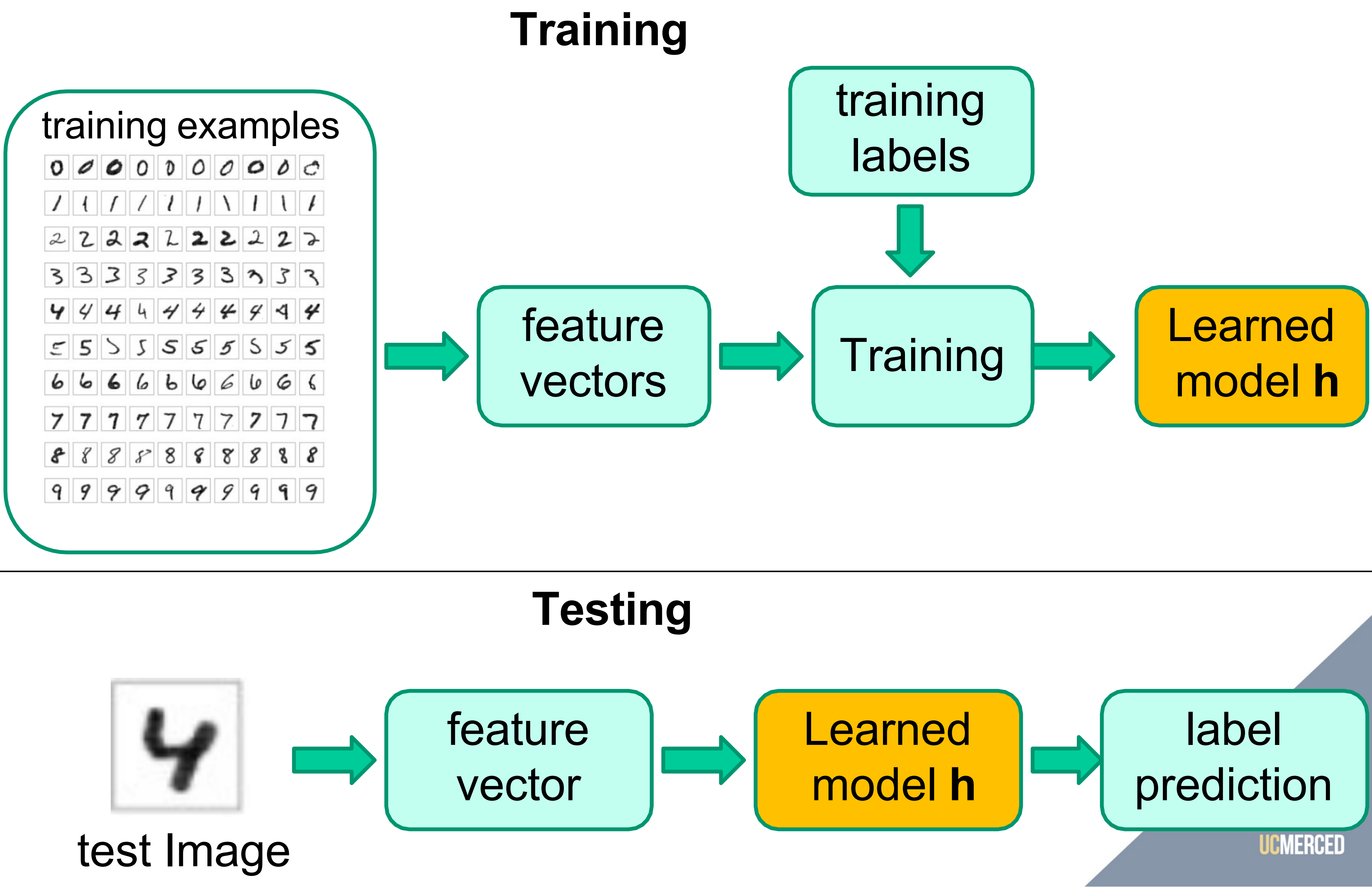
Lecture 4: KNN Classifier and Curse of Dimensionality

Some materials from Pascal Poupart and Kilian Weinberger

Recap: Supervised ML algorithm

1. A model $h(\mathbf{x}; \mathbf{\Theta})$ (hypothesis class) with parameters $\mathbf{\Theta}$. A particular value of $\mathbf{\Theta}$ determines a particular hypothesis in the class.
   Ex: for linear models, $\mathbf{\Theta}$ = slope w_1 and intercept w_0.
2. A *loss function* $L(\cdot, \cdot)$ to compute the difference between the desired output (label) y_n and our prediction to it $h(\mathbf{x}_n; \mathbf{\Theta})$. *Approximation error (loss)*:
   $$E(\mathbf{\Theta}; \mathcal{X}) = \sum_{n=1}^{N} L(y_n, h(\mathbf{x}_n; \mathbf{\Theta})) = \text{sum of errors over instances}$$
   Ex: 0/1 loss for classification, squared error for regression.
3. An optimization procedure (learning algorithm) to find parameters $\mathbf{\Theta}^*$ that minimize the error:
   $$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta}} E(\mathbf{\Theta}; \mathcal{X})$$

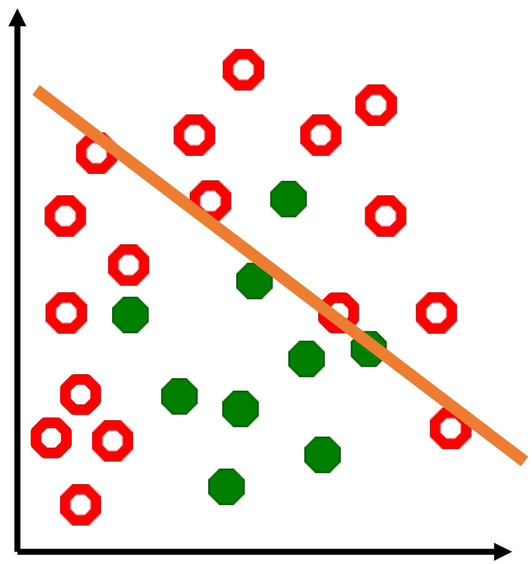
Recap: Training/Testing Phases Illustrated



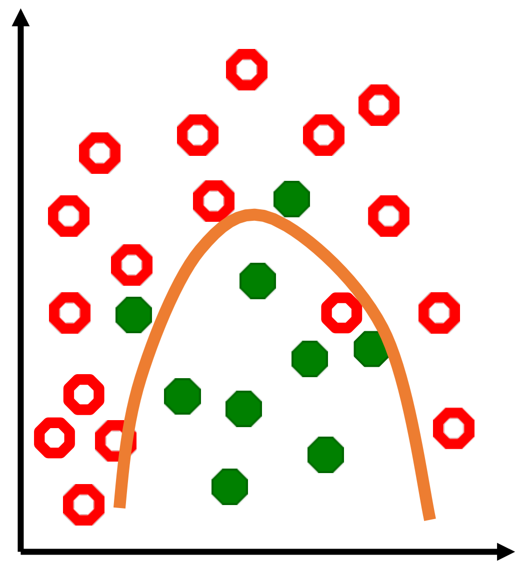
Quiz

Which of the following leads to low training error but high testing error?

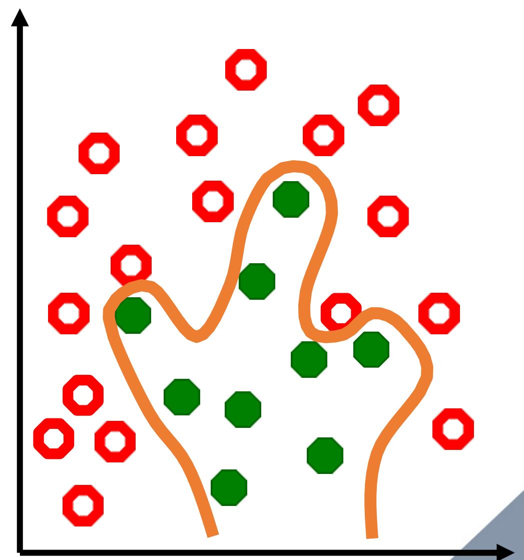
underfitting



"just right"



overfitting



Recap: Cross Validation

- Training set:
  - Used to train, i.e., to fit a hypothesis $h \in H_i$.
  - Optimize parameters of h given the model structure and hyperparameters.
  - Usually done with an optimization algorithm (the learning algorithm).
- Validation set:
  - Used to minimize the generalization error.
  - Optimize hyperparameters or model structure.
  - Usually done with a "grid search". Ex: try all values of $H \in \{10, 50, 100\}$ and $\lambda \in \{10−5, 10−3, 10−1\}$.
- Test set:
  - Used to report the generalization error.
  - We optimize nothing on it, we just evaluate the final model on it

Today’s topoic

- K Nearest Neighbor Classifier
- Curse of Dimensionality

K Nearest Neighbor Classifier

Nearest neighbor classification

- Classification function: $h(x) = y_{\text{nn}}$

  where y_{nn} is the label associated with the nearest neighbor

$$x_{\text{nn}} = argmin_{x'}\, d(x,x')$$

How to measure distances?

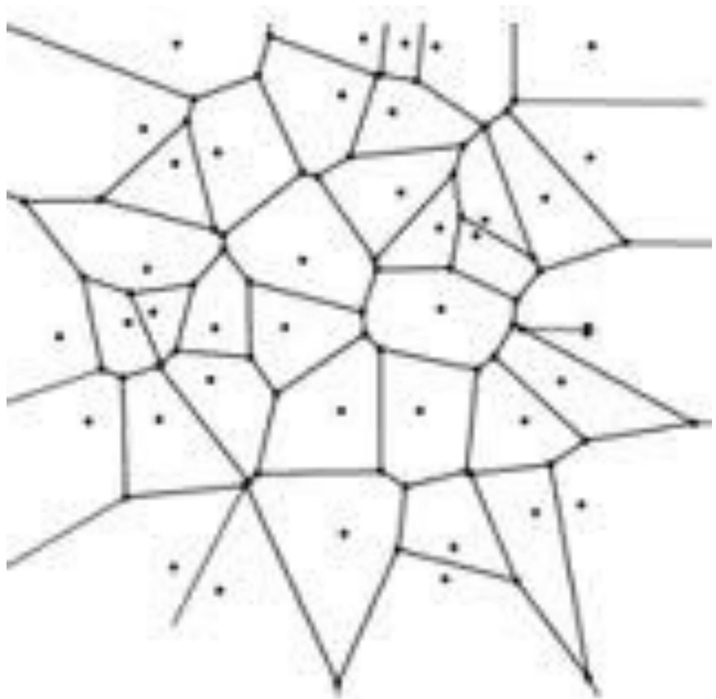
- L^p norm

$$\left\|\boldsymbol{x}\right\|_p = \left(\sum_i \left|x_i\right|^p\right)^{\frac{1}{p}}$$

- Quiz: What if p=1, 2, or +∞?
- Metric learning (more to come)

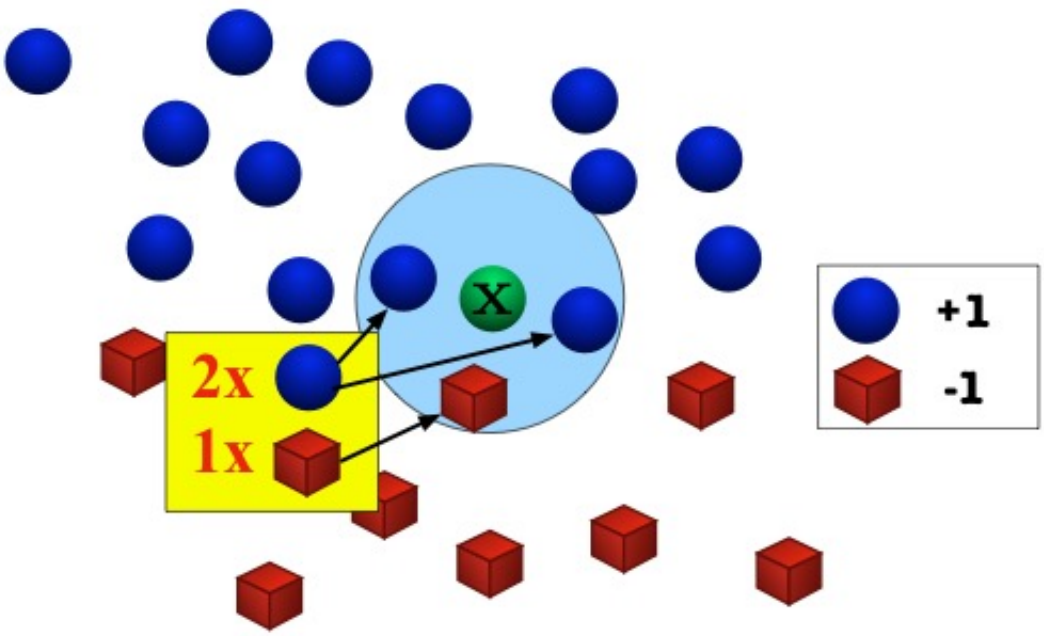
Voronoi Diagram

- Partition implied by nearest neighbor
- Assuming Euclidian distance



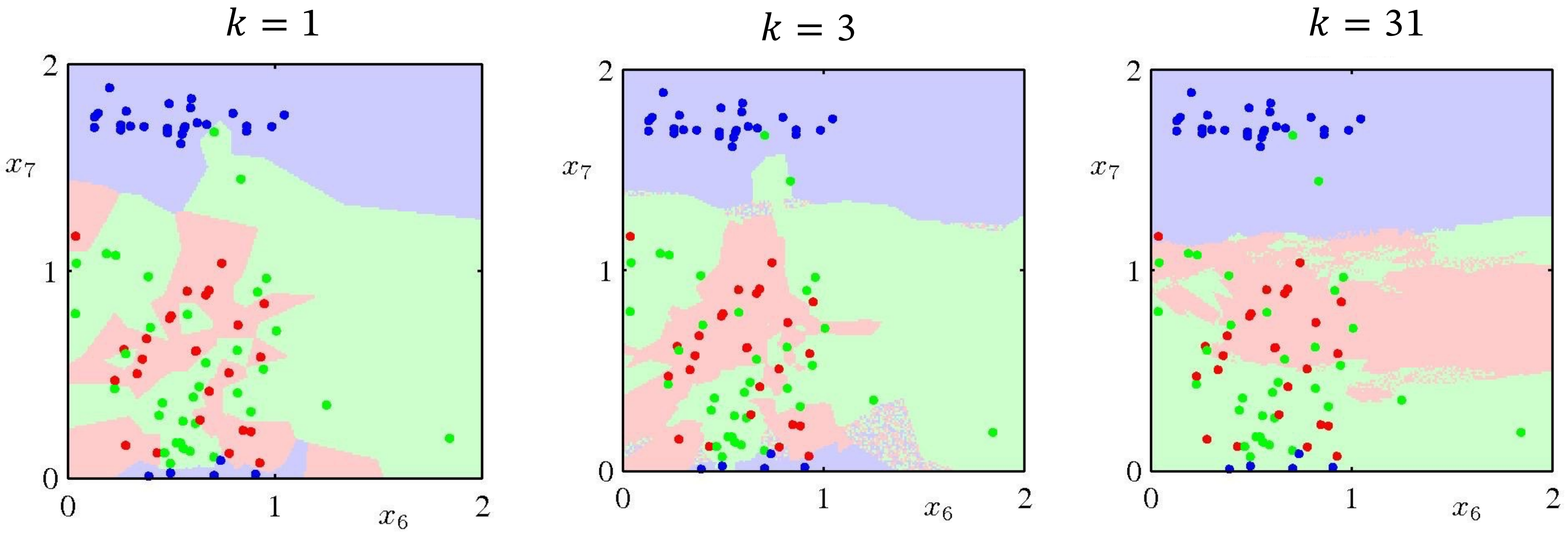
K nearest neighbor algorithm

- Nearest neighbor often instable (noise)
- For a test input ***x***, assign the most common label amongst its k most similar training inputs



Effect of K

- ❑Which partition do you prefer? Why?



- ❑K controls the degree of smoothing.
- ❑What if K=N(number of data points)?

Choosing K

- How should we choose K?
  - Select K with highest test accuracy
- Can we simply split to training and testing set?
- Solution: split data into training, validation and test sets
  - Training set: compute nearest neighbour
  - Validation set: optimize hyperparameters such as K
  - Test set: measure performance

Choosing K based on validation set

Let k be the number of neighbours
For $k = 1$ to max # of neighbours
$\quad accuracy_k \leftarrow eval(k, trainingData, validationData)$
$k^* \leftarrow argmax_k\ accuracy_k$
$accuracy \leftarrow eval(k^*, trainingData \cup validationData, testData)$
Return $k^*, accuracy$

$eval(k, trainingData, dataset)$
$\quad error \leftarrow 0$
$\quad$ For each $(x, y) \in dataset$
$\quad\quad$ Find $\{x_1, x_2, \dots, x_k | (x_i, y_i) \in trainingData\}$ closest to x
$\quad\quad y^* \leftarrow mode(\{y_1, y_2, \dots, y_k\})$
$\quad\quad$ if $y \neq y^*$ then $error_k \leftarrow error_k + 1$
$\quad accuracy \leftarrow \frac{|dataset| - error}{|dataset|}$
$\quad$ return $accuracy$

Robust Validation

- How can we ensure that validation accuracy is representative of future accuracy?
- Validation accuracy becomes more reliable as we increase the size of the validation set
- However, this reduces the amount of data left for training
- Popular solution: cross-validation

Cross Validation

- Repeatedly split training data in two parts, one for training and one for validation. Report the average validation accuracy.
- $\boldsymbol{k}$-fold cross validation

Weighted K Nearest Neighbor

- We can often improve K-nearest neighbours by weighting each neighbour based on some distance measure

$$w(x, x') \propto \frac{1}{distance(x, x')}$$

- Label $y_x \leftarrow argmax_y \sum_{\{x' | x' \in knn(x) \wedge y = y_{x'}\}} w(x, x')$

  where $knn(x)$ is the set of K nearest neighbours of x

K Nearest Neighbor for Regression

- We can also use KNN for regression
- Let y_x be a real value instead of a categorical label

- K-nearest neighbour regression:

$$y_x \leftarrow average(\{y_{x'} | x' \in knn(x)\})$$

- Weighted K-nearest neighbour regression:

$$y_x \leftarrow \frac{\sum_{x' \in knn(x)} w(x, x') y_{x'}}{\sum_{x' \in knn(x)} w(x, x')}$$

UCMERCED

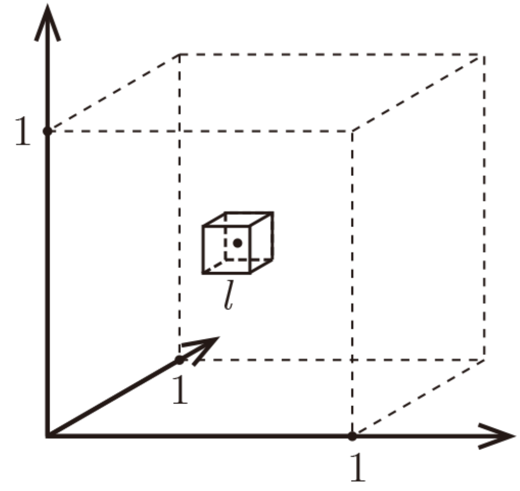
Curse of Dimensionality

KNN for high-dimensional data

- ❑Can we use KNN classifier for high-dimensional data?
- ❑Assumption for KNN classifier to work:
  - ❑K nearest neighbors are ***nearby***
- ❑Are K nearest neighbors nearby for d>>0?

KNN for high-dimensional data

- Formally, imagine the unit cube. All training data is sampled *uniformly* within this cube
- We are considering the k=10 nearest neighbors of such a test point.



- Let ℓ be the edge length of the smallest hyper-cube that contains all k-nearest neighbors

KNN for high-dimensional data

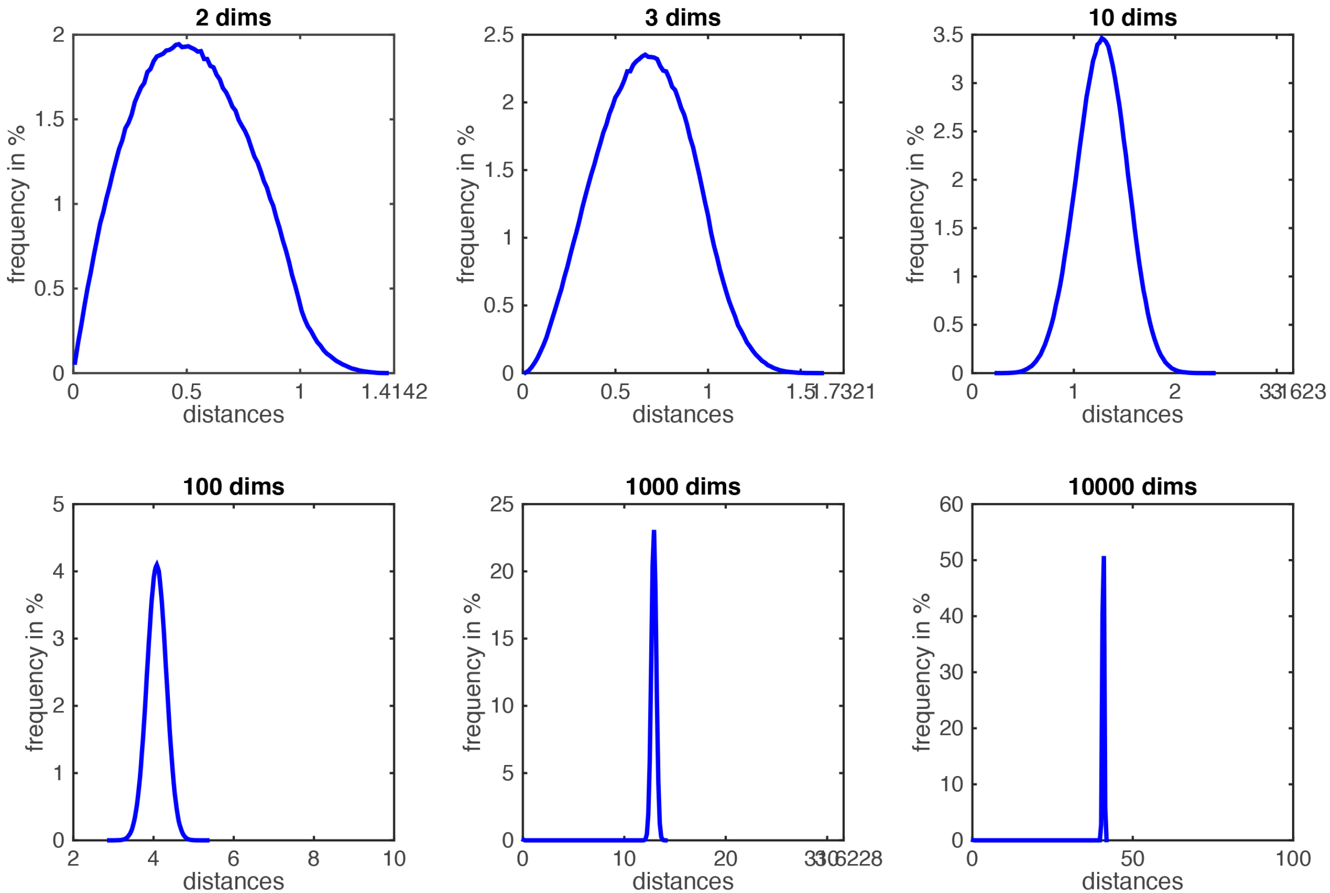
Then $\ell^d \approx \frac{k}{n}$ and $\ell \approx \left(\frac{k}{n}\right)^{1/d}$. If $n = 1000$, how big is ℓ?

d	ℓ
2	0.1
10	0.63
100	0.955
1000	0.9954

UCMERCED

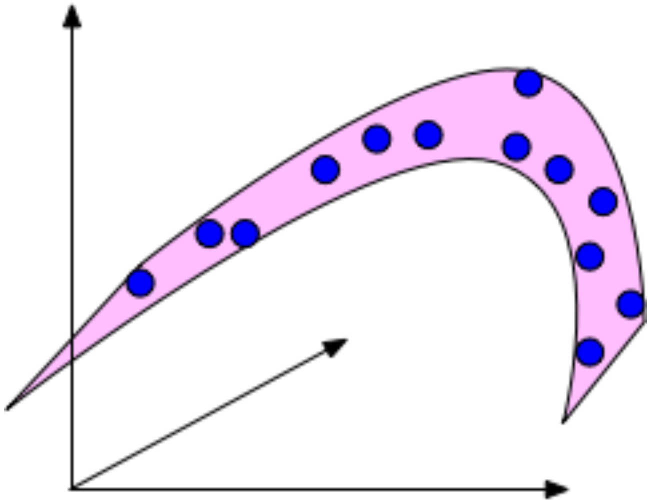
Curse of Dimensionality

- For large dimension, all distances concentrate within a very small range



Data with low dimensional structure

- Data often lie in sub-space or sub-manifold



Low-dimensional structure of face images