K Nearest Neighbor

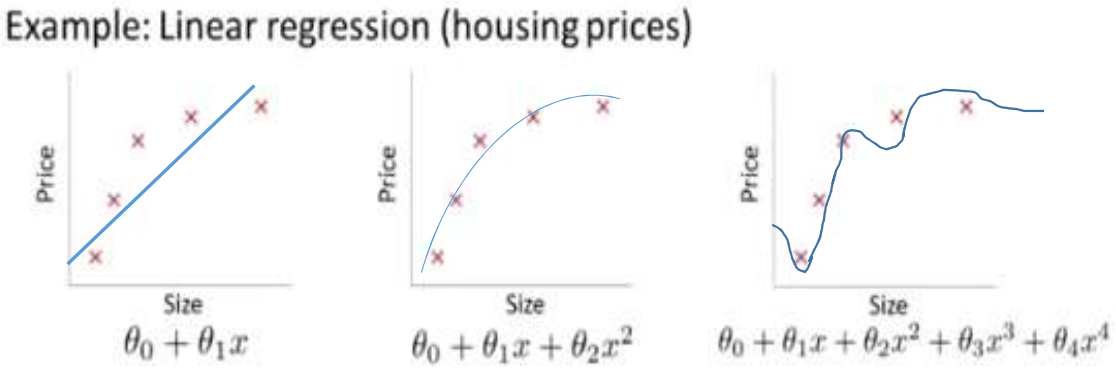
Supervised Learning

Some slides were adapted/taken from various sources, including Prof. Andrew Ng's Coursera Lectures, Stanford University, Prof. Kilian Q. Weinberger's lectures on Machine Learning, Cornell University, Prof. Sudeshna Sarkar's Lecture on Machine Learning, IIT Kharagpur, Prof. Bing Liu's lecture, University of Illinois at Chicago (UIC), CS231n: Convolutional Neural Networks for Visual Recognition lectures, Stanford University and many more. We thankfully acknowledge them. Students are requested to use this material for their study only and **NOT** to distribute it.

Recap

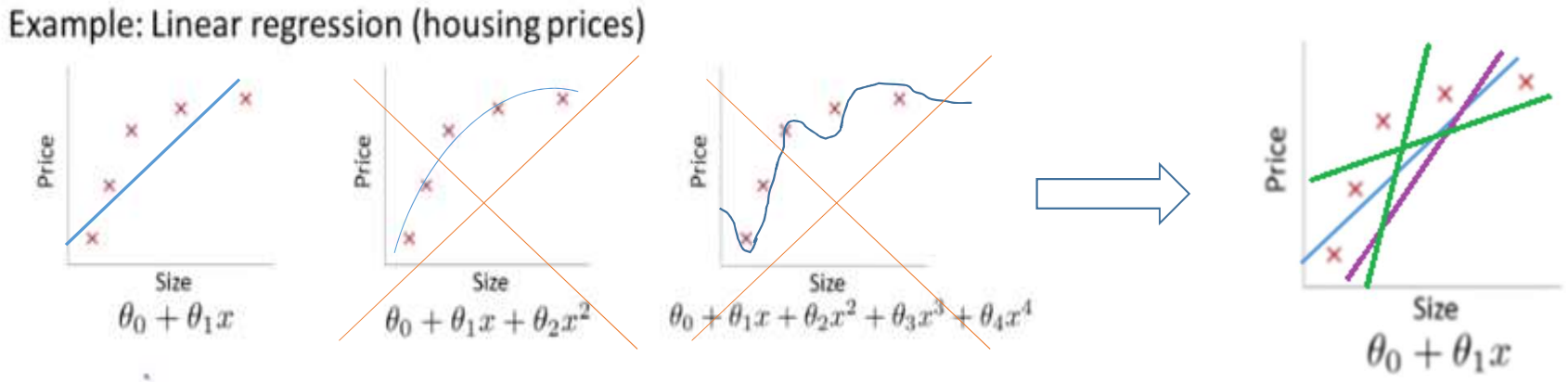
- For any supervised learning problem, given a set of examples (labeled data), we have to find a suitable hypothesis function h from a hypothesis class ($\mathcal{H}$) such that

$$h(\vec{x}_1) = y_1, \text{ where } (\mathbf{x}_1, y_1) \subseteq \mathcal{R}^d \times C \text{ and } h \epsilon \mathcal{H}$$



Hypothesis Class

- How to find a hypothesis class?
- Some assumption is required. Example: function is linear.



K- Nearest Neighbor

Assumption: Similar Inputs have similar outputs/labels
Classification rule: For a test input x, assign the most common label amongst its k most similar training inputs
Formal (and borderline incomprehensible) definition of k-NN:
Test point: X
Denote the set of the k nearest neighbors of x as S_x
Formally S_x is defined as $S_{\mathbf{x}} \subseteq D$ s.t. $|S_{\mathbf{x}}| = k$ and $\forall (\mathbf{x}', y') \in D \backslash S_{\mathbf{x}}$

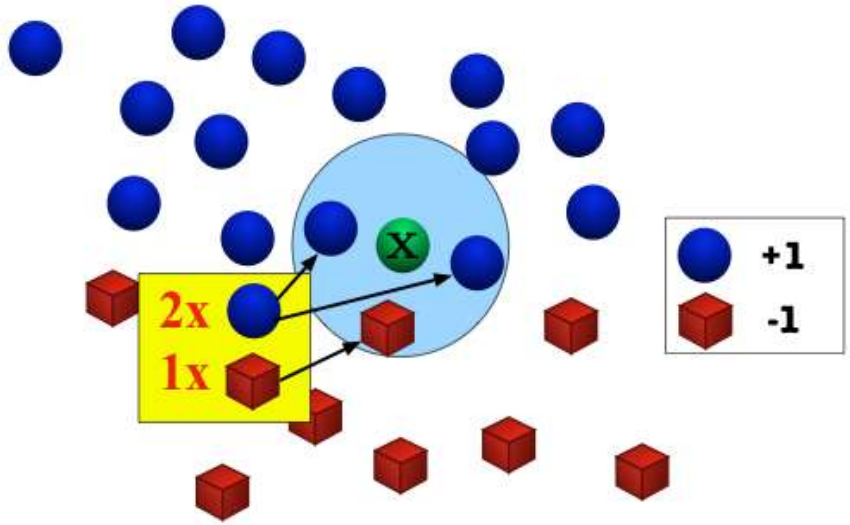
$$\text{dist}(\mathbf{x}, \mathbf{x}') \geq \max_{(\mathbf{x}'', y'') \in S_{\mathbf{x}}} \text{dist}(\mathbf{x}, \mathbf{x}''),$$

(i.e. every point in D but *not* in S_x is at least as far away from x as the furthest point in S_x). We can then define the classifier h(.) as a function returning the most common label in S_x :

$$h(\mathbf{x}) = \text{mode}(\{y'' : (\mathbf{x}'', y'') \in S_{\mathbf{x}}\}),$$

where mode(·) means to select the label of the highest occurrence

K- Nearest Neighbor



A binary classification example with k=3. The green point in the center is the test sample x. The labels of the 3 neighbors are 2×(+1) and 1×(-1) resulting in majority predicting (+1)

Distance Function

- The k-nearest neighbor classifier fundamentally relies on a distance metric. The better that metric reflects label similarity, the better the classification will be. The most common choice is the Minkowski distance

$$\text{dist}(\mathbf{x}, \mathbf{z}) = \left(\sum_{r=1}^{d} |x_r - z_r|^p \right)^{1/p}.$$

- p = 1, Manhattan Distance
- p = 2, Eucledian distance etc.

Curse of Dimensionality

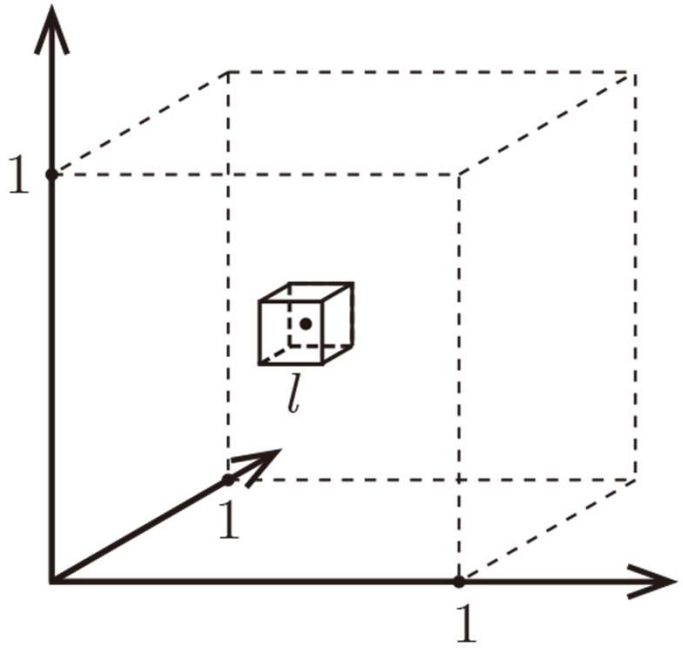
- The kNN classifier makes the assumption that similar points share similar labels.
- Unfortunately, in high dimensional spaces, points that are drawn from a probability distribution, tend to never be close together.
- We can illustrate this on a simple example. We will draw points uniformly at random within the unit cube (illustrated in the figure) and we will investigate how much space the k nearest neighbors of a test point inside this cube will take up.

Curse of Dimensionality

Formally, imagine the unit cube $[0,1]^d$. All training data is sampled uniformly within this cube, i.e. $\forall i, x_i \in [0,1]^d$, and 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 neighbor of a test point.

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



Curse of Dimensionality

Let ℓ be the edge length of the smallest hyper-cube that contains all k-nearest neighbor of a test point.

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

So as d≫0 almost the entire space is needed to find the 10-NN. This breaks down the k-NN assumptions, because the k-NN are not particularly closer (and therefore more similar) than any other data points in the training set. Why would the test points share the label with those k-nearest neighbors, if they are not actually similar to it?

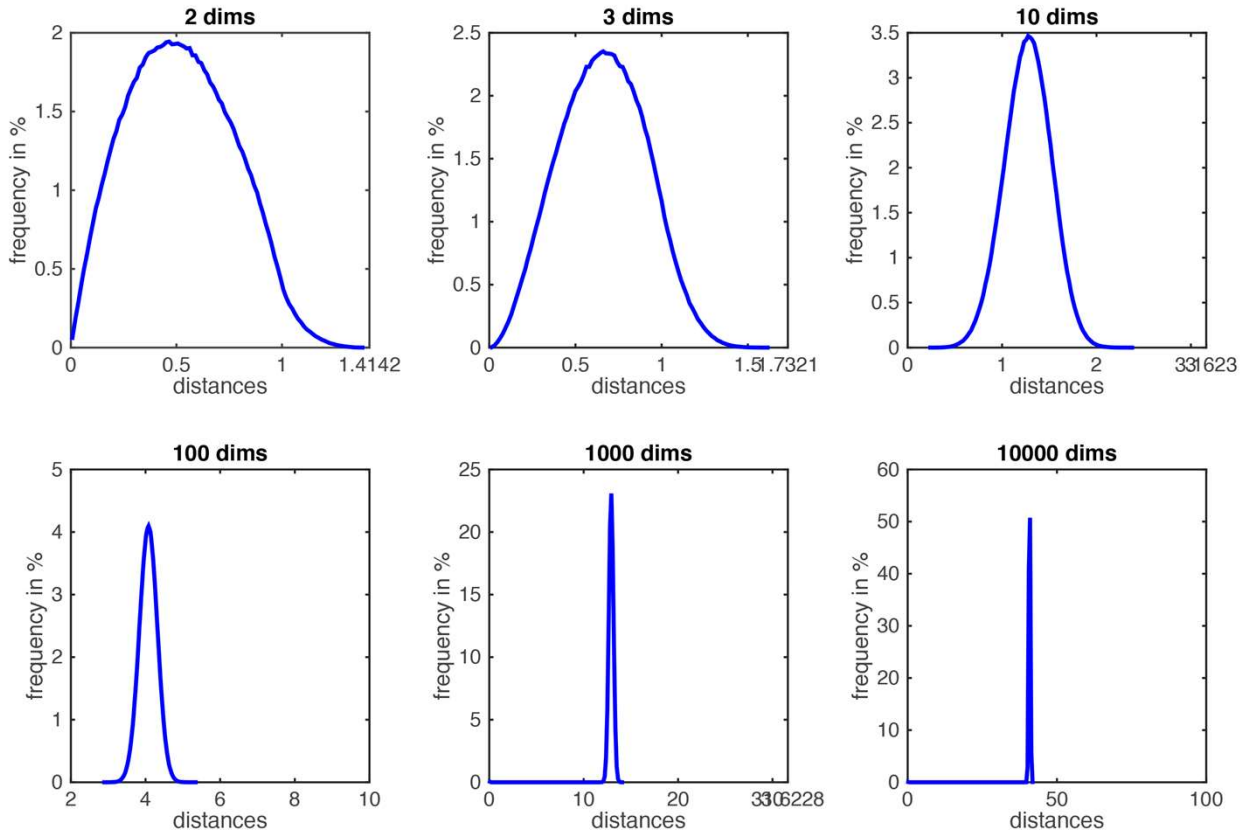
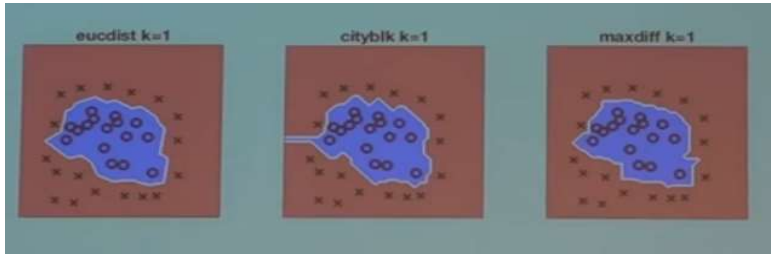
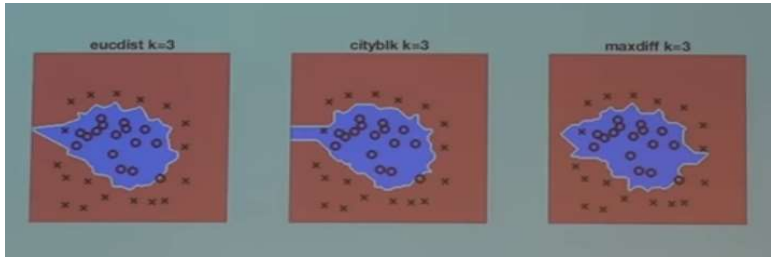
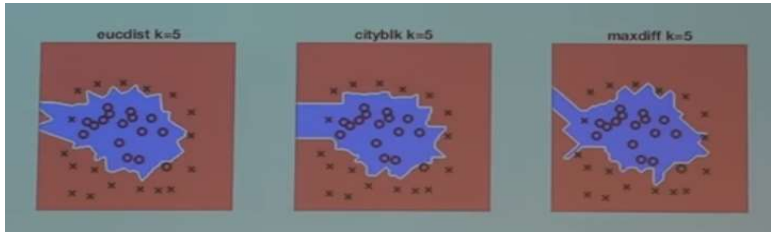
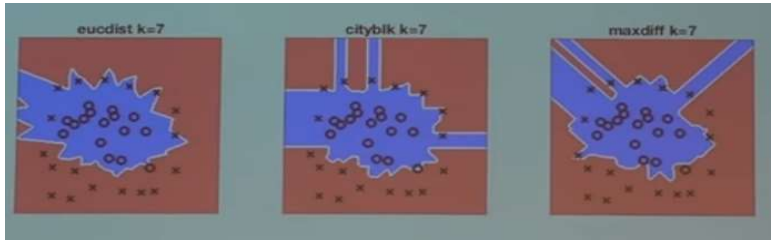


Figure demonstrating ``the curse of dimensionality''. The histogram plots show the distributions of all pairwise distances between randomly distributed points within d-dimensional unit squares. As the number of dimensions d grows, all distances concentrate within a very small range.

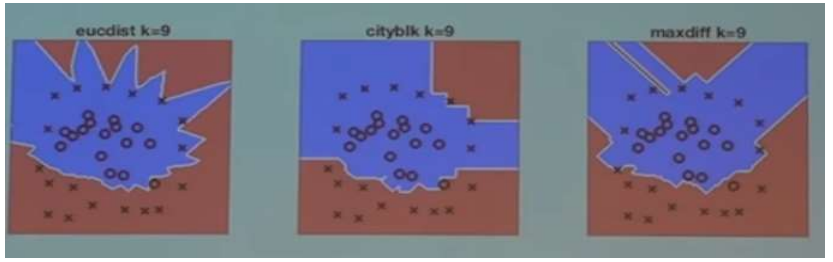


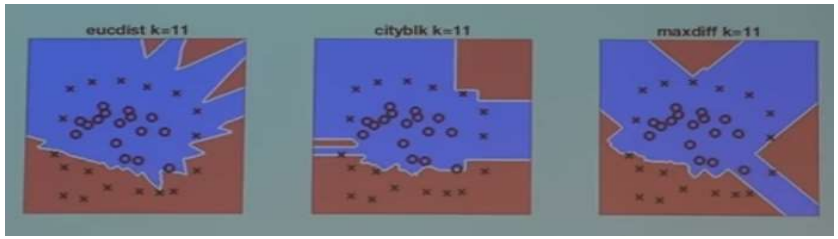


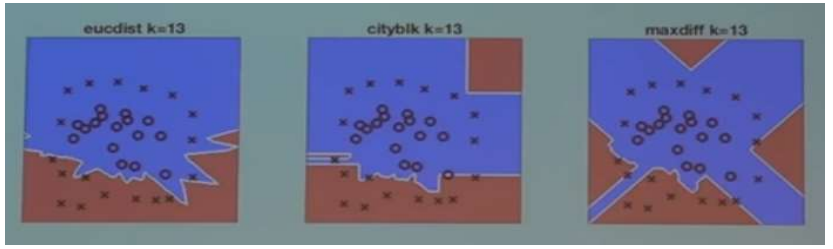


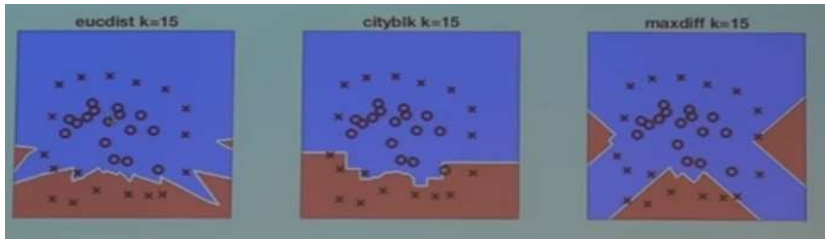


What happens if we increase k?









Curse of Dimensionality

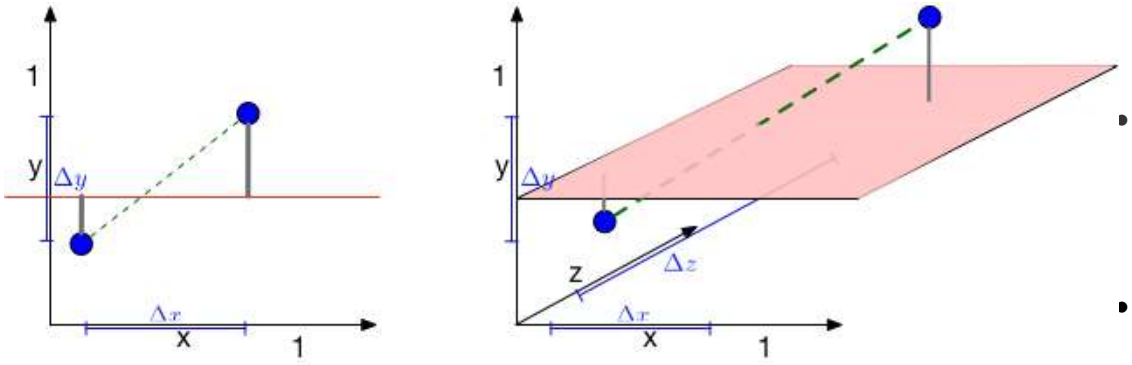
- One might think that one rescue could be to increase the number of training samples, n, until the nearest neighbors are truly close to the test point. How many data points would we need such that ℓ becomes truly small?

- Fix $\ell = \frac{1}{10} = 0.1 \Rightarrow n = \frac{k}{\ell^d} = k \cdot 10^d$ which grows exponentially!

- For d>100, we would need far more data points than there are electrons in the universe...

Distances to hyperplanes

- So the distance between two randomly drawn data points increases drastically with their dimensionality.
- How about the distance to a hyperplane?
- Consider the following figure.



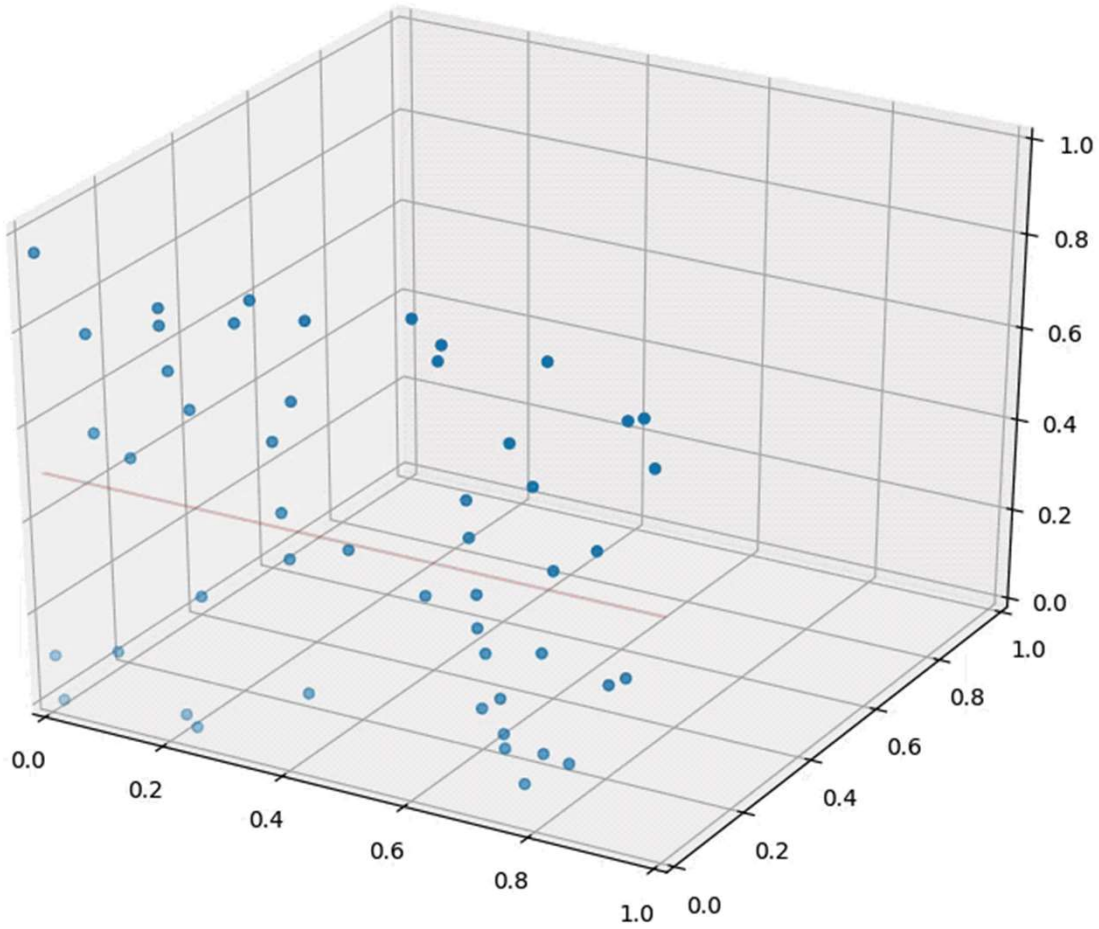
- There are two blue points and a red hyperplane. The left plot shows the scenario in 2d and the right plot in 3d.
- As long as d=2, the distance between the two points is $\sqrt{\Delta x^2 + \Delta y^2}$
- When a third dimension is added, this extends to $\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$, which must be at least as large (and is probably larger).

This confirms again that pairwise distances grow in high dimensions. On the other hand, the distance to the red hyperplane remains unchanged as the third dimension is added.

Distances to hyperplanes

- The reason is that the normal of the hyper-plane is orthogonal to the new dimension. This is a crucial observation.
- In d dimensions, d−1 dimensions will be orthogonal to the normal of any given hyper-plane. Movement in those dimensions cannot increase or decrease the distance to the hyperplane --- the points just shift around and remain at the same distance.
- As distances between pairwise points become very large in high dimensional spaces, distances to hyperplanes become comparatively tiny.
- For machine learning algorithms, this is highly relevant. As we will see later on, many classifiers (e.g. the Perceptron or SVMs) place hyper planes between concentrations of different classes.
- One consequence of the curse of dimensionality is that most data points tend to be very close to these hyperplanes and it is often possible to perturb input slightly (and often imperceptibly) in order to change a classification outcome. This practice has recently become known as the creation of adversarial samples, whose existents is often falsely attributed to the complexity of neural networks.

Distances to hyperplanes



An animation illustrating the effect on randomly sampled data points in 2D, as a 3rd dimension is added (with random coordinates).

As the points expand along the 3rd dimension they spread out and their pairwise distances increase. However, their distance to the hyper-plane (z=0.5) remains unchanged ---

so in relative terms the distance from the data points to the hyper-plane shrinks compared to their respective nearest neighbors.

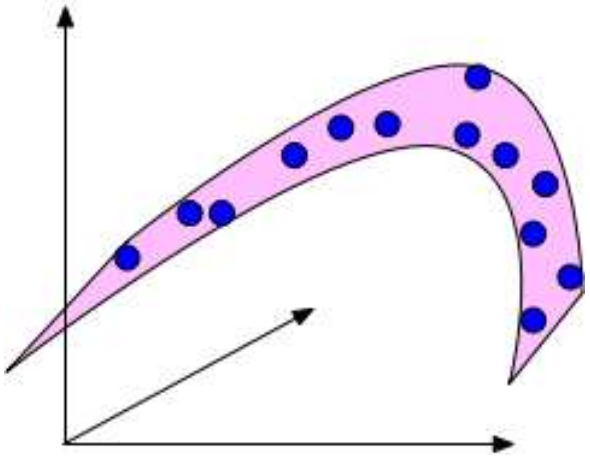
Data with low dimensional structure

However, not all is lost. Data may lie in low dimensional subspace or on sub-manifolds.

Example: natural images (digits, faces). Here, the true dimensionality of the data can be much lower than its ambient space.

The figure below shows an example of a data set sampled from a 2-dimensional manifold (i.e. a surface in space), that is embedded within 3d.

Human faces are a typical example of an intrinsically low dimensional data set. Although an image of a face may require 18M pixels, a person may be able to describe this person with less than 50 attributes (e.g. male/female, blond/dark hair, ...) along which faces vary.



An example of a data set in 3d that is drawn from an underlying 2-dimensional manifold. The blue points are confined to the pink surface area, which is embedded in a 3-dimensional ambient space.

k-NN summary

- k-NN is a simple and effective classifier if distances reliably reflect a semantically meaningful notion of the dissimilarity. (It becomes truly competitive through metric learning)

- As $n \rightarrow \infty$, k-NN becomes provably very accurate, but also very slow.

- As $d \gg 0$, points drawn from a probability distribution stop being similar to each other, and the kNN assumption breaks down.