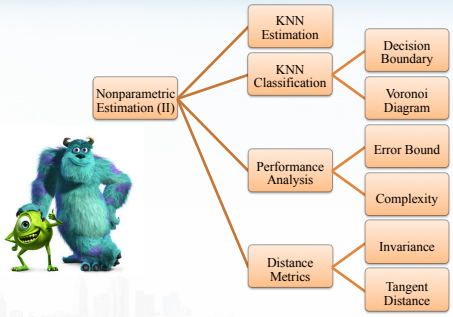
Lecture Notes on

Pattern Recognition

Session 05(B): Non-Parametric Estimation (II)

Gang Li
School of Information Technology
Deakin University, VIC 3125, Australia

Road map



KNN Estimation

- Approximations
- KNN Estimation

Density Estimation

- Combining these two approximations we have:

$$p(\mathbf{x}) \approx \frac{k_R / n}{V_R}$$

- The above approximation is based on ***contradictory*** assumptions:
  - R is relatively ***small*** so that $p(\mathbf{x})$ is approximately constant inside the integration region – **Approximation 1**
  - R is relatively ***large*** (i.e., it contains many samples so that P_k is sharply peaked) – **Approximation 2**
- We need to choose an optimum R in practice …

Notation

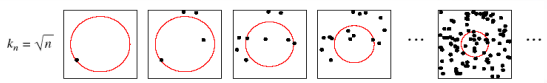
- Suppose we form regions R_1, R_2, ... containing $\mathbf{x}$.
  - R_1 contains 1 sample, R_2 contains 2 samples, etc.
  - R_i has volume V_i and contains k_i samples.
- The n-th estimate $p_n(\mathbf{x})$ of $p(\mathbf{x})$ is given by:

$$p_n(\mathbf{x}) \cong \frac{k_n / n}{V_n}$$

k_n-nearest-neighbor estimation

$$p_n(\mathbf{x}) = \frac{k_n / n}{V_n}$$

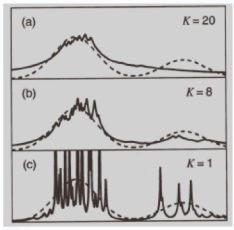
- Fix k_n and allow V_n to vary:
  - Consider a hypersphere around $\mathbf{x}$.
  - Allow the radius of the hypersphere to grow until it contains k_n data points.
  - V_n is determined by the volume of the hypersphere.



- size depends on density

k_n-nearest-neighbor estimation

- The parameter k_n acts as a smoothing parameter and needs to be optimized.



Main conditions for convergence

- The following conditions must be satisfied in order for $p_n(\mathbf{x})$ to converge to $p(\mathbf{x})$:

$$p_n(\mathbf{x}) = \frac{k_n / n}{V_n}$$

$\lim_{n\to\infty} V_n = 0$ • Approximation 1

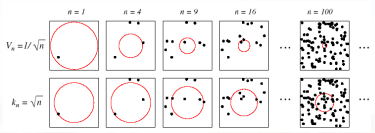
$\lim_{n\to\infty} k_n = \infty$ • Approximation 2

$\lim_{n\to\infty} k_n / n = 0$ • to allow $p_n(\mathbf{x})$ to converge

Density Estimation

- Control k_n: KNN
  - If takes $k_n = sqrt(n)$, $p_n(x)$ will be a reasonable good approximation to $p(x)$.
    - In this case, $V_n = 1/(sqrt(n)\, p(x))$, which is still a version of $V_n = V_1/(sqrt(n))$ but now V_1 is determined by the data's nature.

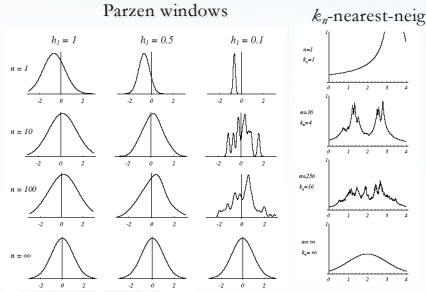
$$k_n = \sqrt{n}$$



Density Estimation

Parzen windows

k_n-nearest-neighbor

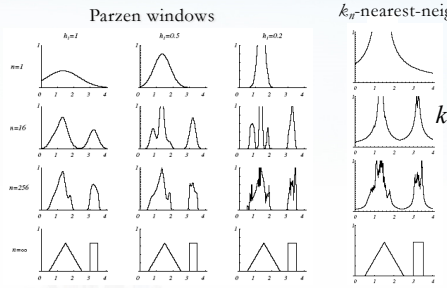


$$k_n = k_1\sqrt{n}$$

Density Estimation

Parzen windows

k_n-nearest-neighbor



$$k_n = k_1\sqrt{n}$$

KNN Classification

- KNN Classification
- 1NN Classification

k_n-nearest-neighbor Classification

- Suppose that we have c classes and that class ω_i contains n_i points with $n_1+n_2+...+n_c=n$

$$P(\omega_i / \mathbf{x}) = \frac{p_n(\mathbf{x}/\omega_i)P(\omega_i)}{p_n(\mathbf{x})}$$

- Given a point x, we find the k_n nearest neighbors. Suppose that k_i points from k_n belong to class ω_i, then:

$$p_n(\mathbf{x}/\omega_i) = \frac{k_i}{n_i V_n}$$

k_n-nearest-neighbor Classification

- The prior probabilities can be computed as:

$$P(\omega_i) = \frac{n_i}{n}$$

- Using the Bayes' rule, the posterior probabilities can be computed as follows:

$$P_n(\omega_i \mid \mathbf{x}) = \frac{p_n(x,\omega_i)}{\sum_{j=1}^{c} p_n(x,\omega_j)} = \frac{k_i}{k_n}$$

$$p_n(x_n,\omega_i) = \frac{k_i/n}{V_n} \qquad \sum_{j=1}^{c} p_n(x_n,\omega_j) = \frac{k_n/n}{V_n}$$

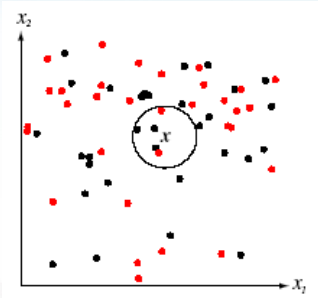
k_n-nearest-neighbor Rule

- Given a data point x, find a hypersphere around it that contains k points and assign x to the class having the largest number of representatives inside the hypersphere.

$$P(\omega_i / \mathbf{x}) = \frac{p_n(\mathbf{x}/\omega_i)P(\omega_i)}{p_n(\mathbf{x})} = \frac{k_i}{k_n}$$

- When k=1, we get the nearest-neighbor rule.
  - 1NN Classification

k_n-nearest-neighbor Example



k_n-nearest-neighbor Example

- k = 3 (odd value)
- x = $(0.10, 0.25)^t$

Prototypes	Labels
(0.15, 0.35)	ω_1
(0.10, 0.28)	ω_2
(0.09, 0.30)	ω_5
(0.12, 0.20)	ω_2

- Closest vectors to x with their labels are:

  {(0.10, 0.28, ω_2); (0.12, 0.20, ω_2); (0.15, 0.35, ω_1)}

  - Assign the label ω_2 to x since ω_2 is the most frequently represented.

1-nearest-neighbor Example

- k = 3 (odd value)
- x = $(0.10, 0.25)^t$

Prototypes	Labels
(0.15, 0.35)	ω_1
(0.10, 0.28)	ω_2
(0.09, 0.30)	ω_5
(0.12, 0.20)	ω_2

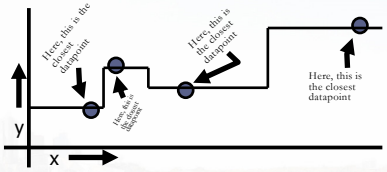
- Closest vectors to x with their labels are:

  {(0.10, 0.28, ω_2)

  - Assign the label ω_2 to x

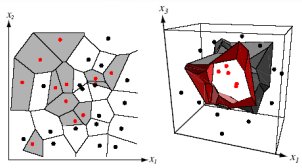
Decision boundary (nearest-neighbor rule)

- The nearest neighbor rule leads to a ***Voronoi tessellation*** of the feature space.
  - Each cell contains all the points that are closer to a given training point x than to any other training points.
  - All the points in a cell are labeled by the category of the training point in that cell.



Decision boundary (nearest-neighbor rule)

- The nearest neighbor rule leads to a ***Voronoi tessellation*** of the feature space.
  - Each cell contains all the points that are closer to a given training point x than to any other training points.
  - All the points in a cell are labeled by the category of the training point in that cell.



- Knowledge of this boundary is sufficient to classify new points.
- The boundary itself is rarely computed…
- Many algorithms seek to retain only those points necessary to generate an identical boundary.

The Zen of Voronoi Diagrams

CNN Article

Mystery of renowned Zen garden revealed

Thursday, September 26, 2002 Posted: 10:11 AM EDT (1411 GMT)

LONDON (Reuters) -- For centuries visitors to the renowned Ryoanji Temple garden in Kyoto, Japan have been entranced and mystified by the simple arrangement of rocks.

The five sparse clusters on a rectangle of raked gravel are said to be pleasing to the eyes of the hundreds of thousands of tourists who visit the garden each year.

Scientists in Japan said on Wednesday they now believe they have discovered its mysterious appeal.

"We have uncovered the implicit structure of the Ryoanji garden's visual ground and have shown that it includes an abstract, minimalist depiction of natural scenery," said Gert Van Tonder of Kyoto University.

The researchers discovered that the empty space of the garden evokes a hidden image of a branching tree that is sensed by the unconscious mind.

"We believe that the unconscious perception of this pattern contributes to the enigmatic appeal of the garden," Van Tonder added.

He and his colleagues believe that whoever created the garden during the Muromachi era between 1333-1573 knew exactly what they were doing and placed the rocks around the tree image.

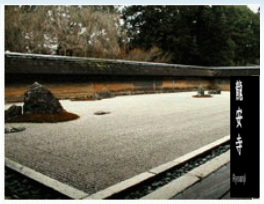
By using a concept called medial-axis transformation, the scientists showed that the hidden branched tree converges on the main area from which the garden is viewed.

The trunk leads to the prime viewing site in the ancient temple that once overlooked the garden.

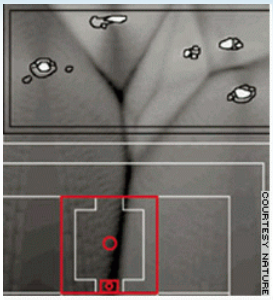
It is thought that abstract art may have a similar impact.

"There is a growing realisation that scientific analysis can reveal unexpected structural features hidden in controversial abstract paintings," Van Tonder said

The Zen of Voronoi Diagrams



Ryoanji Temple garden in Kyoto



Layout shows the rock clusters (top) and the preferred viewing spot of the garden from the main hall (the circle in the middle of the square).

Performance Analysis

- Error Bound
- Complexity
- Improvement Measures

Error Rate

- **Baysian (optimum):**

$$\omega_m = \arg\max_i P(\omega_m \mid \mathbf{x})$$

$$P(error \mid \mathbf{x}) = 1 - P(\omega_m \mid \mathbf{x})$$

$$P(error) = \int p(error, \mathbf{x}) d\mathbf{x}$$

$$= \int P(error \mid \mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

$$\mathcal{D} = \left\{ \begin{pmatrix} \mathbf{x}_1 \\ \theta_1 \end{pmatrix}, \begin{pmatrix} \mathbf{x}_2 \\ \theta_2 \end{pmatrix}, \ldots, \begin{pmatrix} \mathbf{x}_n \\ \theta_n \end{pmatrix} \right\}$$

$$\theta_i \in \{\omega_1, \omega_2, \ldots, \omega_c\}$$

Error Rate

- 1-NN
  - Suppose the true class for x is **θ**

$$P(error \mid \mathbf{x}, \mathbf{x}')$$
$$= 1 - \sum_{i=1}^{c} P(\theta = \omega_i, \theta' = \omega_i \mid \mathbf{x}, \mathbf{x}')$$
$$= 1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x}) P(\omega_i \mid \mathbf{x}')$$

$$\mathcal{D} = \left\{ \begin{pmatrix} \mathbf{x}_1 \\ \theta_1 \end{pmatrix}, \begin{pmatrix} \mathbf{x}_2 \\ \theta_2 \end{pmatrix}, \ldots, \begin{pmatrix} \mathbf{x}_n \\ \theta_n \end{pmatrix} \right\}$$
$$\theta_i \in \{\omega_1, \omega_2, \ldots, \omega_c\}$$

$$P(error \mid \mathbf{x}) = \int P(error \mid \mathbf{x}, \mathbf{x}') p(\mathbf{x}' \mid \mathbf{x}) d\mathbf{x}'$$

Error Rate

- 1-NN
  - As n$\longrightarrow\infty$, $\mathbf{x} \approx \mathbf{x}'$ $\quad p(\mathbf{x}' \mid \mathbf{x}) \approx \delta(\mathbf{x}' - \mathbf{x})$

$$P(error \mid \mathbf{x}, \mathbf{x}') = 1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x}) P(\omega_i \mid \mathbf{x}')$$
$$= 1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2$$

$$P(error \mid \mathbf{x}) = \int P(error \mid \mathbf{x}, \mathbf{x}') p(\mathbf{x}' \mid \mathbf{x}) d\mathbf{x}'$$
$$= 1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2$$

$$P(error) = \int P(error \mid \mathbf{x}) p(\mathbf{x}) d\mathbf{x} = \int \left[1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2 \right] p(\mathbf{x}) d\mathbf{x}$$

$$\mathcal{D} = \left\{ \begin{pmatrix} \mathbf{x}_1 \\ \theta_1 \end{pmatrix}, \begin{pmatrix} \mathbf{x}_2 \\ \theta_2 \end{pmatrix}, \ldots, \begin{pmatrix} \mathbf{x}_n \\ \theta_n \end{pmatrix} \right\}$$
$$\theta_i \in \{\omega_1, \omega_2, \ldots, \omega_c\}$$

Error Rate

(The most complex case: uniform classes)

Bayesian

$$P^*(error) = \int [1 - P(\omega_m \mid \mathbf{x})] p(\mathbf{x}) d\mathbf{x}$$
$$= \int [1 - 1/c)] p(\mathbf{x}) d\mathbf{x}$$
$$= 1 - 1/c$$

1NN

$$P(error) = \int \left[1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2 \right] p(\mathbf{x}) d\mathbf{x}$$
$$= \int [1 - 1/c^2)] p(\mathbf{x}) d\mathbf{x}$$
$$= 1 - 1/c^2$$

$$P^*(error) \le P(error)$$

1NN Error Bound

- 1-NN

Minimized this term

$$P(error) = \int \underbrace{\left[1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2 \right]}_{\text{Maximized this term to find the upper bound}} p(\mathbf{x}) d\mathbf{x}$$

$$\sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2 = P(\omega_m \mid \mathbf{x})^2 + \underbrace{\sum_{i \ne m} P(\omega_i \mid \mathbf{x})^2}_{\text{Minimum when all elements are same}} \quad i.e., \quad P(\omega_i \mid \mathbf{x}) = \frac{1 - P(\omega_m \mid \mathbf{x})}{c-1} = \frac{P^*(error)}{c-1}$$

$$P(\omega_m \mid \mathbf{x})^2 = [1 - P^*(error \mid \mathbf{x})]^2 = 1 - 2P^*(error \mid \mathbf{x}) + P^*(error \mid \mathbf{x})^2$$

$$\sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2 = P(\omega_m \mid \mathbf{x})^2 + \sum_{i \ne m} P(\omega_i \mid \mathbf{x})^2 \ge 1 - 2P^*(error \mid \mathbf{x}) + \frac{c}{c-1} P^*(error \mid \mathbf{x})^2$$

$$1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2 \le 2P^*(error \mid \mathbf{x}) - \frac{c}{c-1} P^*(error \mid \mathbf{x})^2 \le 2P^*(error \mid \mathbf{x})$$

1NN Error Bound

- 1-NN

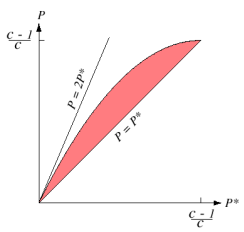
$$P(error) = \int \left[1 - \sum_{i=1}^{c} P(\omega_i \mid \mathbf{x})^2 \right] p(\mathbf{x}) d\mathbf{x}$$
$$\le \int 2P^*(error \mid \mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

$$\boxed{P^*(error) \le P(error) \le 2P^*(error)}$$

  - The nearest-neighbor rule is a suboptimal procedure.
  - The error rate is never worse than twice the Bayes rate.

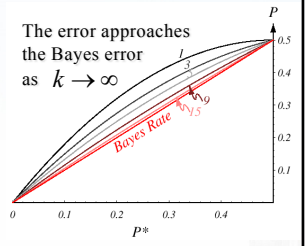
Error Bound

1NN



KNN

The error approaches the Bayes error as $k \to \infty$

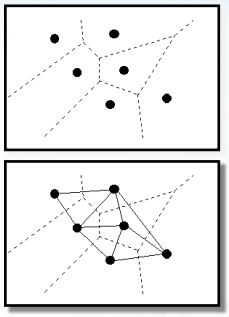


Computation Complexity

- The computation complexity of the nearest-neighbor algorithm (both in ***time*** and ***space***) has received a great deal of analysis.
- Require **O(dn)** space to store n prototypes in a training set.
  1. **Editing, pruning or condensing**
- To search the nearest neighbor for a d-dimensional test point x, the time complexity is **O(dn)**.
  2. **Partial distance**
  3. **Search tree**

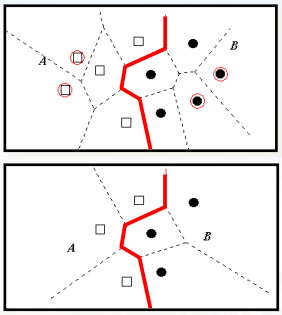
Editing Nearest Neighbor

- ***Voronoi diagram*** is a partition of space into ***regions***,
  - within which all points are closer to some particular node than to any other node.
- If two Voronoi regions *share a boundary*, the nodes of these regions are connected with an edge.
  - Such nodes are called the ***Voronoi neighbors*** (or ***Delaunay neighbors***).



Editing Nearest Neighbor

- Compute the *Delaunay triangulation* for the training set.
- Visit each node, *marking* it if all its *Delaunay neighbors* are of the same class as the current node.
- *Delete all marked nodes*, exiting with the remaining ones as the edited training set.
- Based on this, the circled prototypes are redundant.



Partial Distance

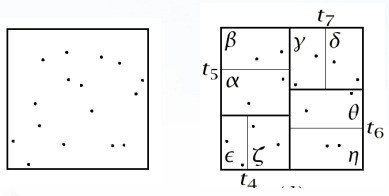
- Using the following fact to ***early throw far-away*** prototypes: compute distance using first r dimensions only

$$D_r(\mathbf{x},\mathbf{x}') = (\sum_{k=1}^{r}(x_k - x'_k)^2)^{1/2}$$

- If the partial distance is too great (i.e., greater than the distance of x to current closest prototype), there is no reason to compute additional terms.

KD-Tree and Cover Tree

- Leave this as the topic for this session's seminar talk



Distance Metrics

- Metrics
- Invariance
- Tangent Distance

Summary of KNN

- The nearest neighbor classifier provides a powerful tool.
  - Its error is bounded to be at most twice of the Bayes error (in the limiting case).
  - It is easy to implement and understand.
  - It can be implemented efficiently.
  - Its performance, however, relies on the metric used to compute distances!

Properties of a Distance Metric

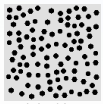
- Nonnegativity
$$D(\mathbf{a},\mathbf{b}) \geq 0$$
- Reflexivity
$$D(\mathbf{a},\mathbf{b}) = 0 \text{ iff } \mathbf{a} = \mathbf{b}$$
- Symmetry
$$D(\mathbf{a},\mathbf{b}) = D(\mathbf{b},\mathbf{a})$$
- Triangle Inequality
$$D(\mathbf{a},\mathbf{b}) + D(\mathbf{b},\mathbf{c}) \geq D(\mathbf{a},\mathbf{c})$$

Distance metrics

- Euclidean distance:
$$D(a,b) = \left(\sum_{k=1}^{d} (a_k - b_k)^2 \right)^{1/2}$$
  - Although we can always compute the Euclidean distance between any two vectors, the result may or may not be meaningful

Distance metrics

- Which image is more similar to the original image, image 1 or 2?


Original image

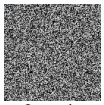


Image 1

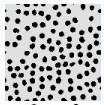
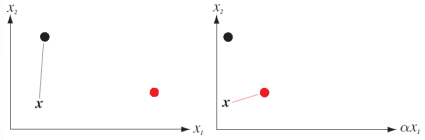


Image 2

Input image	Image 1	Image 2
Euclidean distance per pixel	133.3568	137.3505

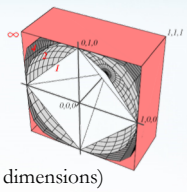
Distance metrics

- Distance relations may change by scaling (or other) transformations (e.g., choose different units).
  - Hint: normalize data in each dimension if there is a large disparity in the ranges of values.

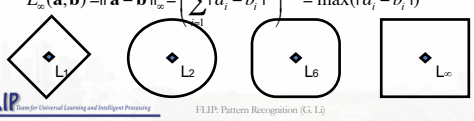


Distance metrics

- Minkowski metric (L_k norm) :
$$L_k(a,b) = \left(\sum_{i=1}^{d} |a_i - b_i|^k \right)^{1/k}$$
  - L_1 (Manhattan or city block distance)
  - L_2 (Euclidean distance)
  - L_∞ (Chessboard distance, max among dimensions)



$$L_\infty(\mathbf{a},\mathbf{b}) = \|\mathbf{a}-\mathbf{b}\|_\infty = \left(\sum_{i=1}^{d} |a_i - b_i|^\infty \right)^{1/\infty} = \max(|a_i - b_i|)$$



Distance metrics

- Jaccard's coefficient: similarity for sets A and B

$$d(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

- Tanimoto similarity for vectors x and y

$$d_{TS}(\vec{x},\vec{y}) = 1 - \frac{\vec{x}\cdot\vec{y}}{\|\vec{x}\|^2 + \|\vec{y}\|^2 - \vec{x}\cdot\vec{y}}$$

Distance metrics

- Hausdorff distance measure for sets: compares elements by a distance d_e

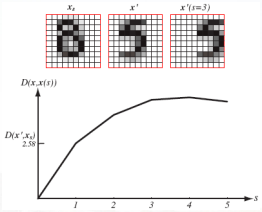
Measures the extent to which each point of the "model" set A lies near some point of the "image" set B and vice versa.

Two sets are within Hausdorff distance r from each other if and only if any point of one set is within the distance r from some point of the other set.

$$d_p(x,B) = \inf_{y \in B} d_e(x,y),$$
$$d_p(A,y) = \inf_{x \in A} d_e(x,y),$$
$$d_s(A,B) = \sup_{x \in A} d_p(x,B),$$
$$d_s(B,A) = \sup_{y \in B} d_p(A,y).$$
$$d(A,B) = \max\{d_s(A,B), d_s(B,A)\}.$$

Distance metrics (Invariance)

- Distance Measurement is an importance factor for nearest-neighbor classifier, e.g.,
  - To achieve ***invariant pattern recognition***



Distance metrics (Invariance)

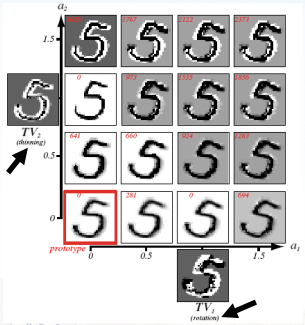
- How to deal with transformations?
  - Normalize data
    - e.g., shift center to a fixed location
  - More difficult to normalize with respect to rotation and scaling ...
  - How to find the rotation/scaling factors?

Distance metrics (Tangent Distance)

- Suppose there are *r* transformations applicable to our problem
  - (e.g., translation, shear, rotation, scale, line thinning).
- Take each prototype **x**' and apply each of the transformations $F_i(\mathbf{x}';a_i)$ on it.
- Construct tangent vectors TV_i for each transformation:

$$TV_i = F_i(\mathbf{x}';a_i) - \mathbf{x}'$$

Distance metrics (Tangent Distance)



$F_i(\mathbf{x}';a_i)$

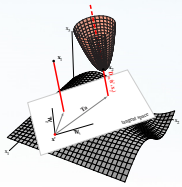
Distance metrics (Tangent Distance)

- Each prototype **x'** is represented by a *r* x *d* matrix *T* of tangent vectors.
- All possible transformed versions of **x'** are then approximated using a ***linear combination*** of tangent vectors.

$$\hat{x}' = x' + Ta = x' + a_1 TV_1 + a_2 TV_2$$

Distance metrics (Tangent Distance)

- The tangent distance from a test point **x** to a particular prototype **x'** is given by:



$$D_{tan}(\mathbf{x}', \mathbf{x}) = \min_{\mathbf{a}}[\|(\mathbf{x}' + \mathbf{Ta}) - \mathbf{x}\|]$$

Seminar S08

- **Topics**
  - Present the method of **KD-Tree** and **Cover Tree** and demonstrate their package usage
- **Requirements**
  - Prepare a **15 minutes** talk on your chosen topic
  - Make **ppt** to assist your talk
  - Prepare **at least 3 questions** to ask the audience after your talk
  - Get ready to **take questions** from the audience
- **Hints**
  - You can search for research articles from Google Scholar

Questions?