CN550: Neural and Computational Models of Recognition, Memory, and Attention

Supervised Learning, Geometrical Representation, k-Nearest Neighbor

Lecture 2

Instructors: Anatoli Gorchetchnikov <anatoli@bu.edu>
Heather Ames <starfly@cns.bu.edu>
Teaching Fellow: Karthik Srinivasan <skarthik@bu.edu>

Supervised Learning

The system receives a teaching signal from a “supervisor”

In pattern classification case it comes in the form of a class label

Given the training set consisting on pairs of (feature vector, class value) a system is supposed to learn the input/output relationship in such a way that it can predict class value for any valid feature vector

Classifiers can range in their flexibility – bias/variance threshold

High bias classifiers are more “sure” but also more straightforward in their decisions

High variance classifiers are more flexible, but also less “sure”

Considerations

Simple relationships do not require many data points and are better recognized with high bias classifiers

Complex relationships require many more data points and a more flexible high variance system

Ideally, the system will adjust bios/variance depending on the amount of data and the emergent complexity of the relationship

High dimensionality of the input space as well as noise in the training data might call for “Gordian knot cut” with a high bias low variance classifier

Another way is to preprocess the data to reduce dimensionality, find hidden dependencies, etc.

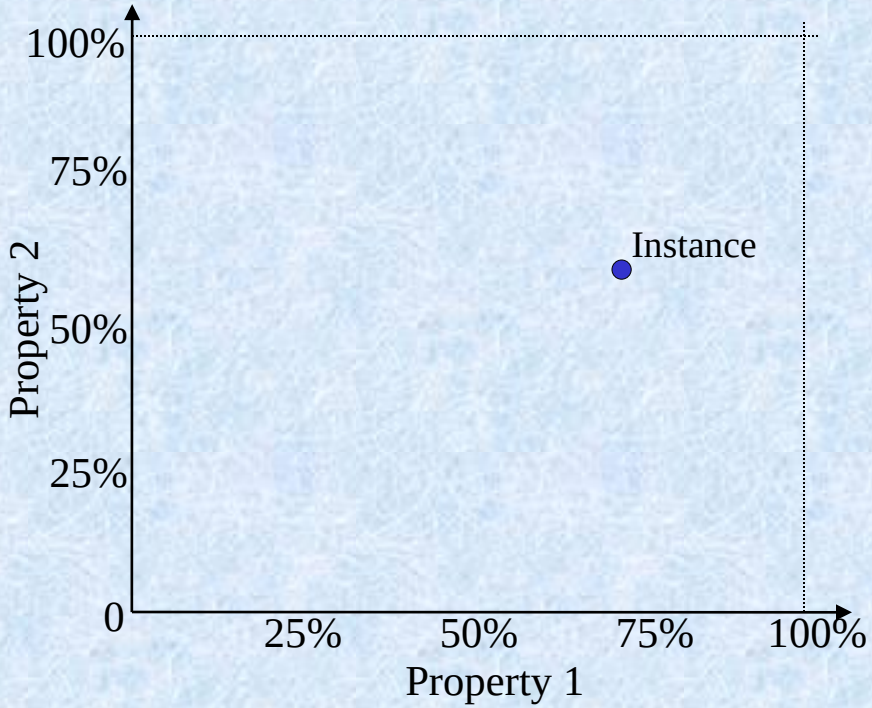
Geometrical Representation of Properties

Numerical dimensions (continuous properties) are more powerful than binary features

We will look at these dimensions and for simplicity use two

In this case we can plot them in 2D

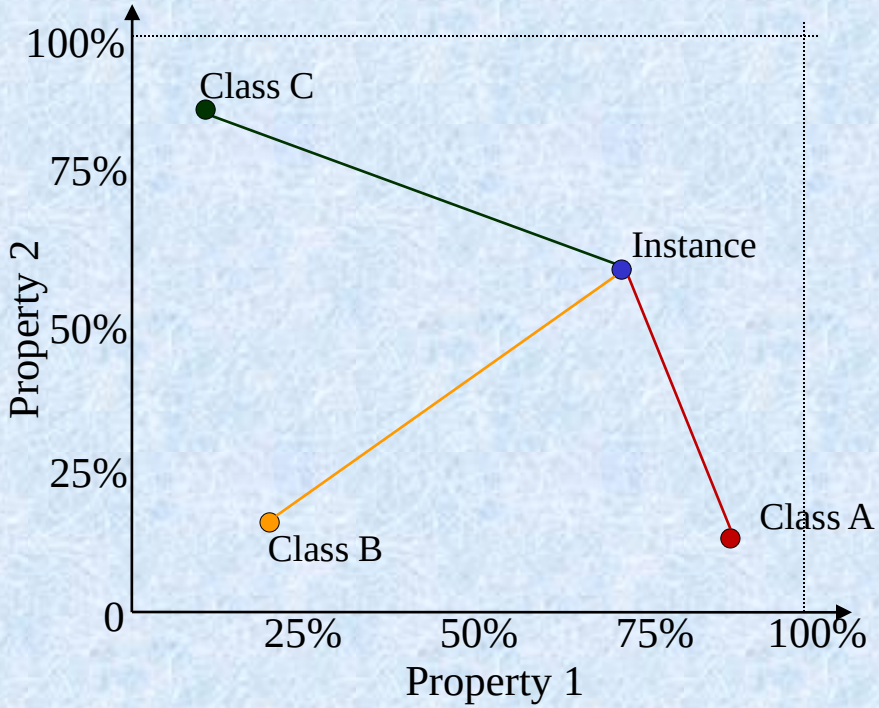
If some instance has its Property 1 at 70% of maximal and Property 2 at 60% we can plot it as shown



This way we can plot all instances of the training or testing set, as well as prototypes

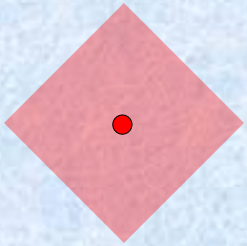
Simplest Nearest Neighbor Classifier

- Simple case: class definitions (prototypes or property values) are known beforehand
- Measure the distances between the test instance and all prototypes, the closest determines the output class
- Metrics can be Cartesian, city block, or some more complicated ones

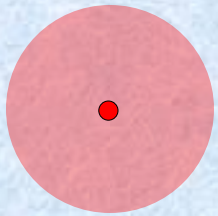


Distance, Norm, Metric

L_1 – City Block (Manhattan) $L_1 = \sum_{i=1}^{N} |x_i - y_i|$



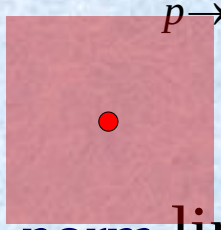
L_2 – Euclidean $L_2 = \sqrt{\sum_{i=1}^{N} |x_i - y_i|^2}$



L_p – for $p \geq 1$ (Minkowski) $L_p = \left(\sum_{i=1}^{N} |x_i - y_i|^p \right)^{\frac{1}{p}}$

L_0 – sometimes used in machine learning, not a true norm; $\lim_{p \to 0} L_p$

count of nonzero elements x_i-$y_i \neq 0$



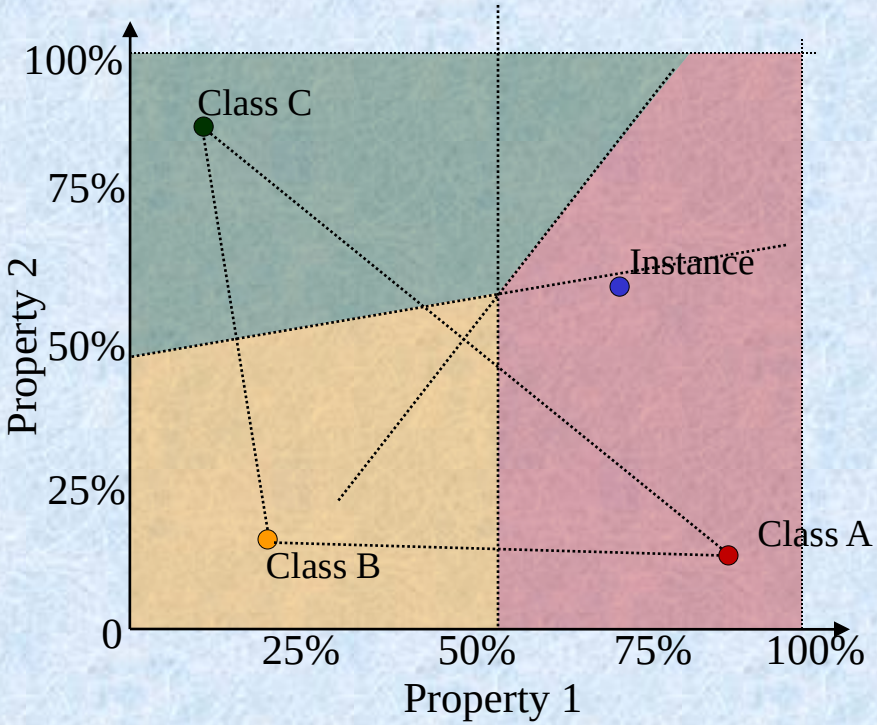
L_{inf} – sometimes is called infinity norm or supremum norm $\lim_{p \to \infty} L_p$

Simplest Nearest Neighbor Classifier

Can be optimized in geometric sense if we preprocess the space into class regions as shown

This is called Voronoi tesselation of space

Might be not as easy computationally especially for many classes and complex regions



What if we do not have predefined classes?

Supervised Learning

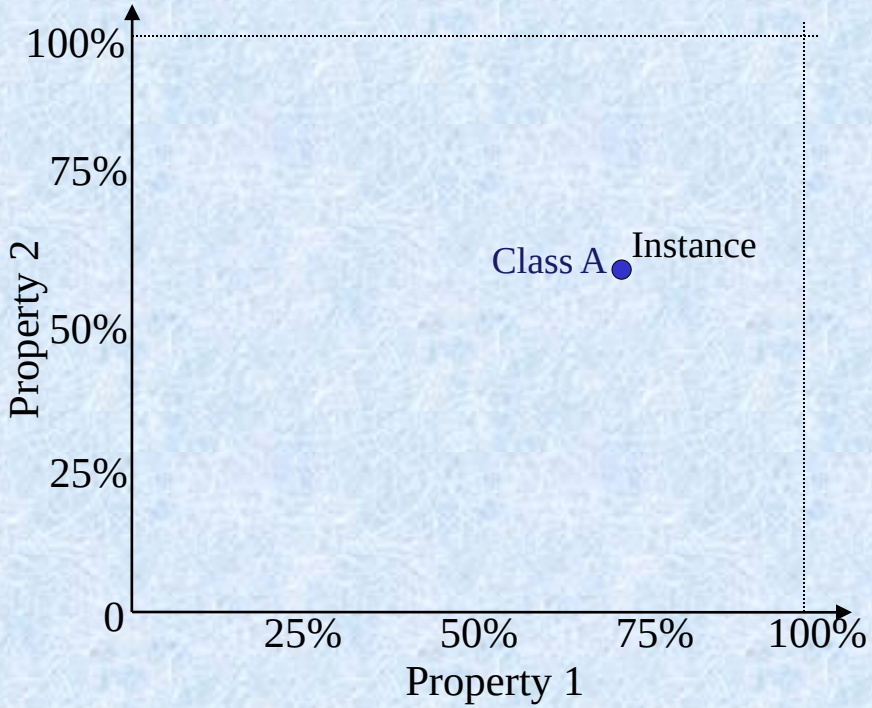
Provide a set of instances for training with a corresponding class label – training set

Provide a set of instances for testing and compare the prediction of the system to a correct class value for each instance – testing set

Depending how you do the training learning can be

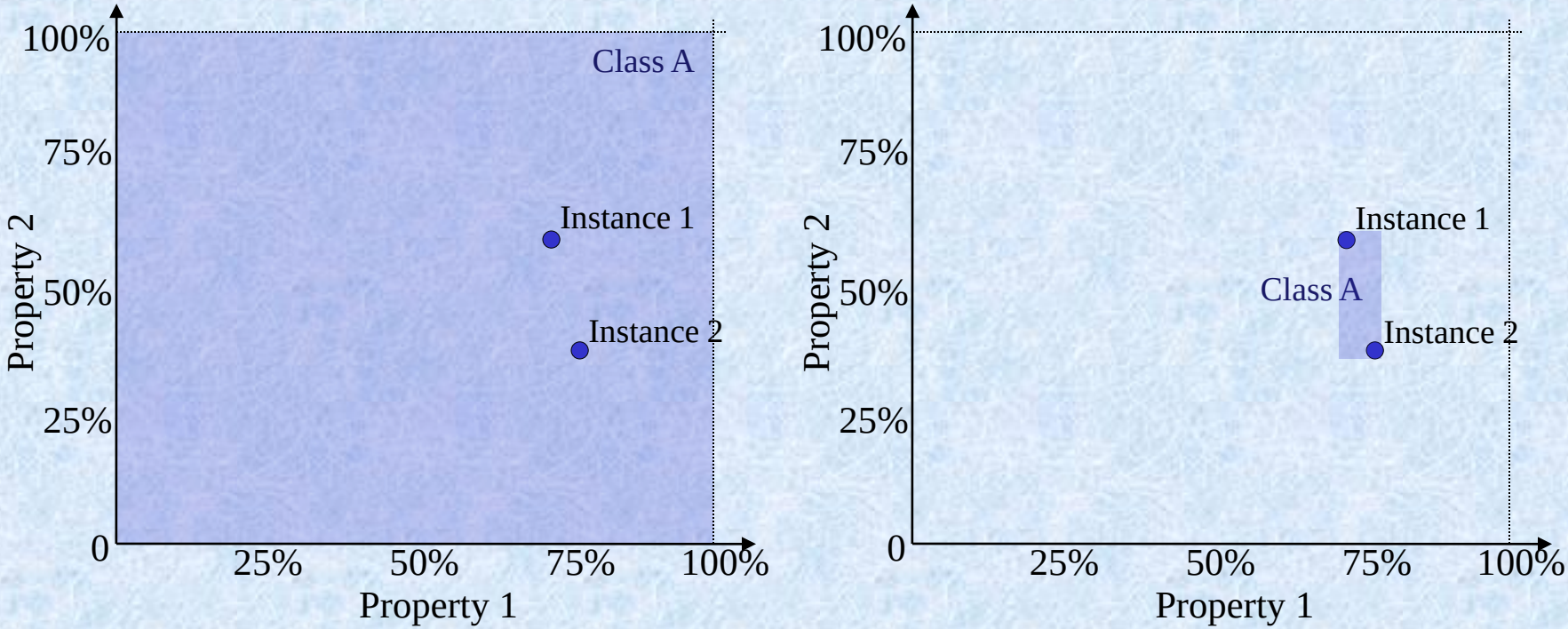
- Incremental – training instances are provided one by one, testing can be done after any of these instances
- Batch – training instances are provided all at once, testing is done after the whole training set is learnt

Incremental Learning: Step 1



Can assign the whole space to class A, only the instance point, or some neighborhood of the instance point

Incremental Learning: Step 2

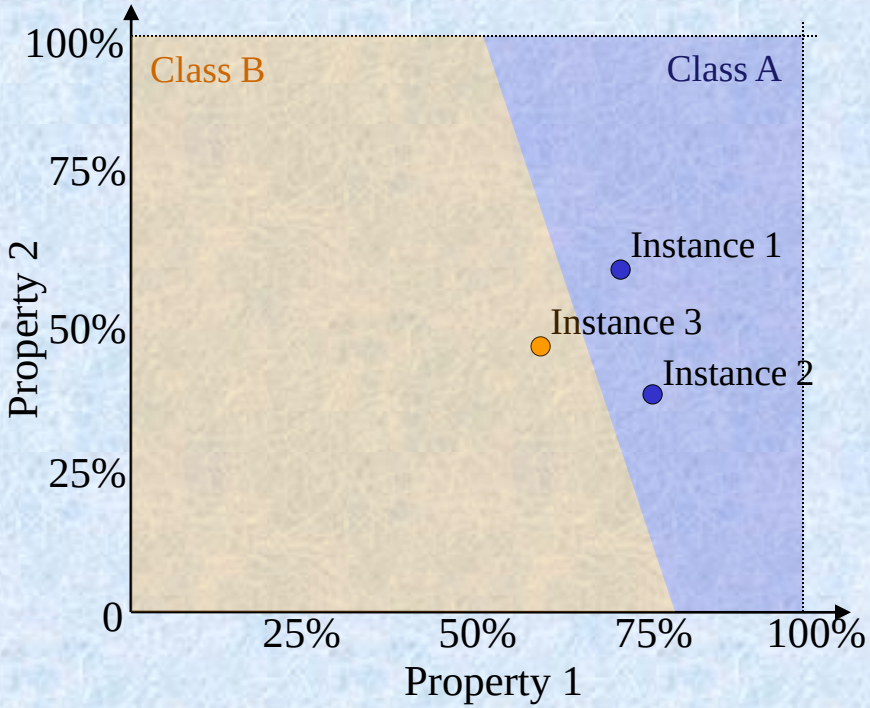


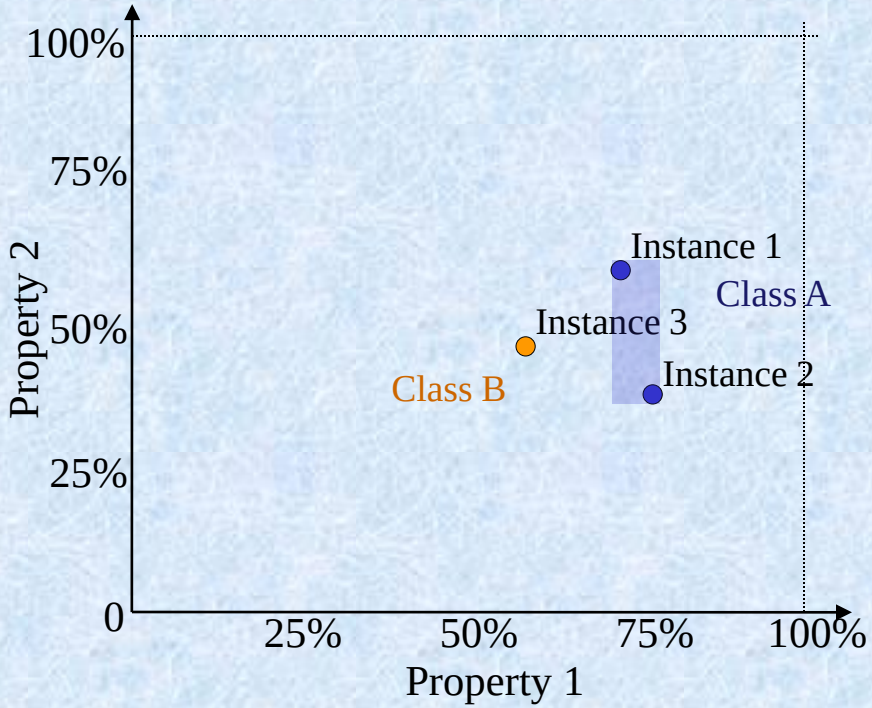
A second instance of Class A does not change the class representation in the first case

In the second case we can either just keep points

Or extend them to some shape that includes both

Incremental Learning: Step 3

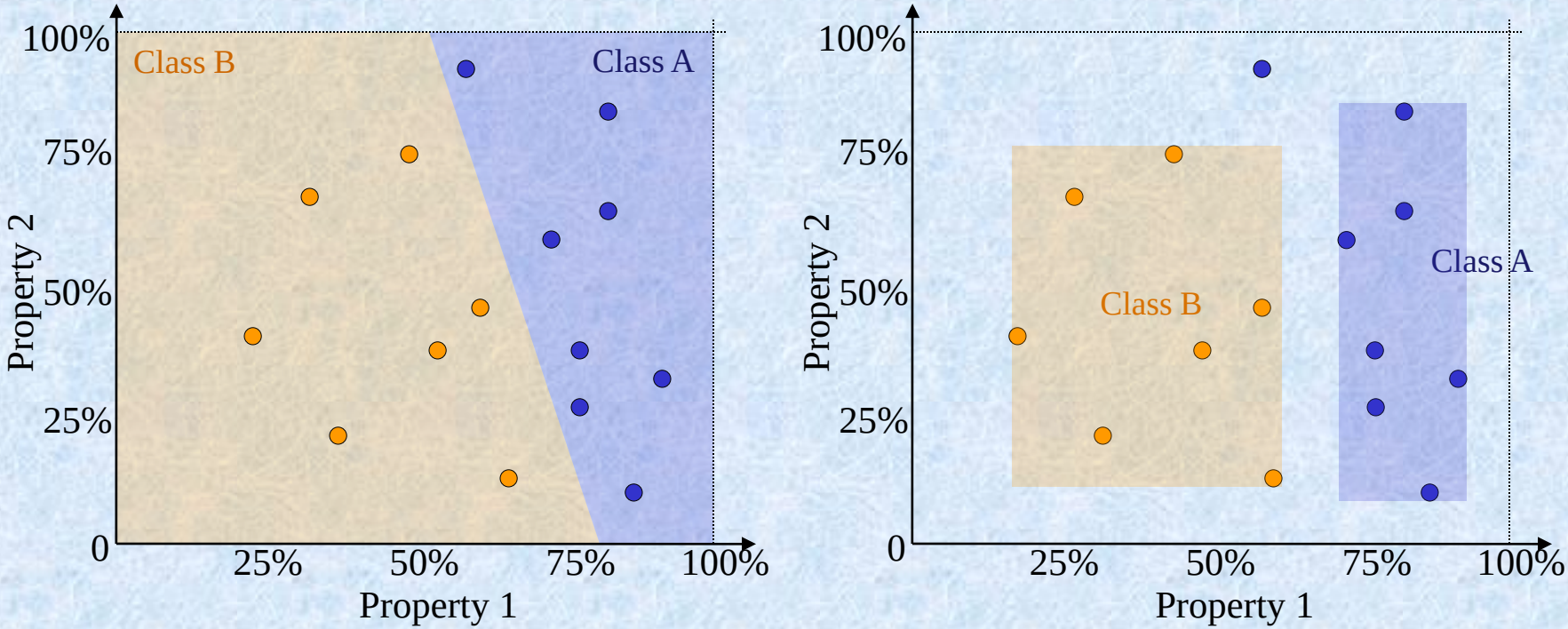




An instance of Class B requires reevaluation of the class representation in the first case

If the boundary between classes can be drawn as a single straight line, then the classes A and B are linearly separable

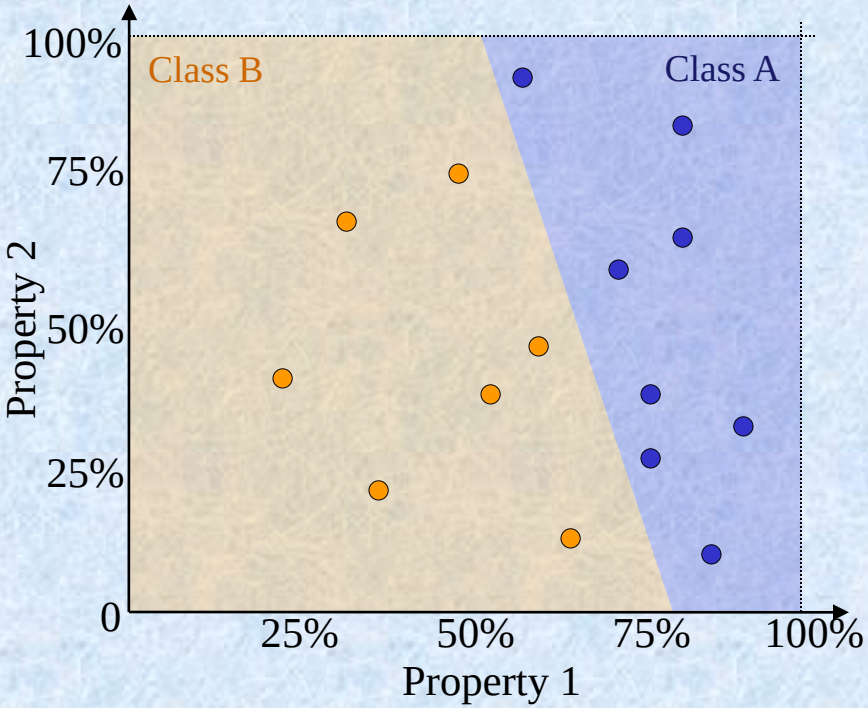
Incremental Learning: Final Step

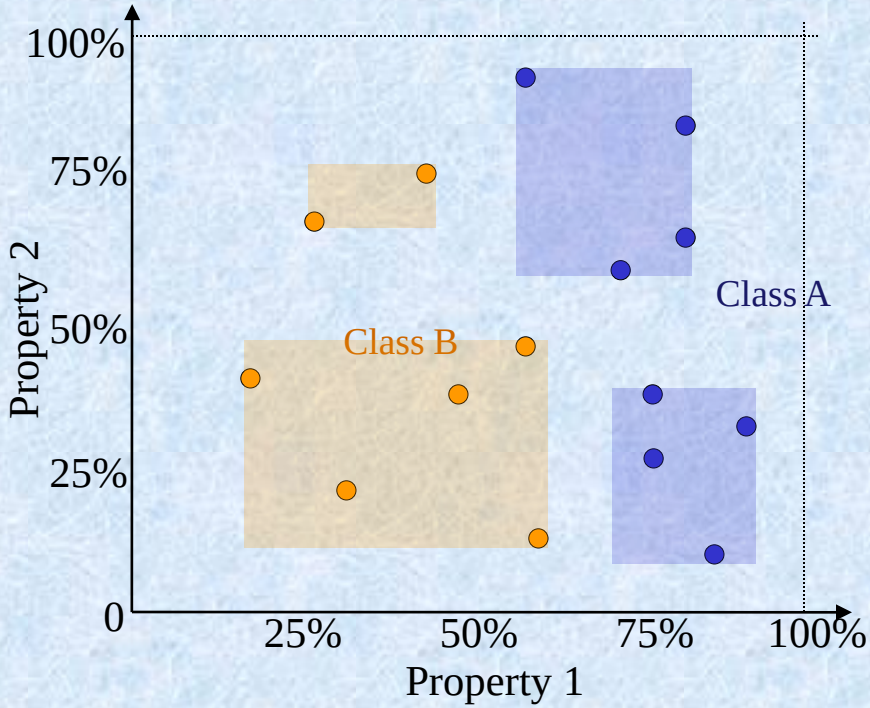


If we use category boxes sometimes it is necessary to have more than one per class even in the case of linearly separable problem

Also note that in case of category boxes order of instances matters a lot for resulting structure

Incremental Learning: Final Step

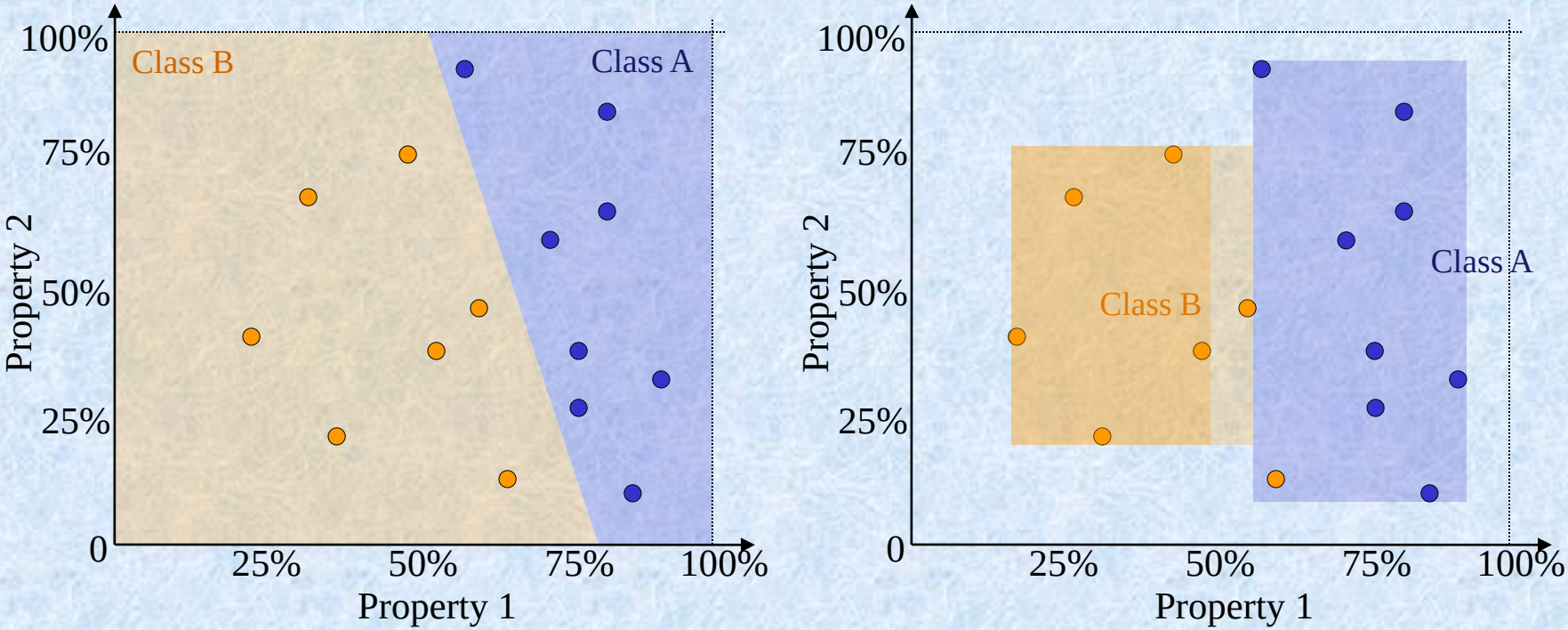




If we use category boxes sometimes it is necessary to have more than one per class even in the case of linearly separable problem

Also note that in case of category boxes order of instances matters a lot for resulting structure

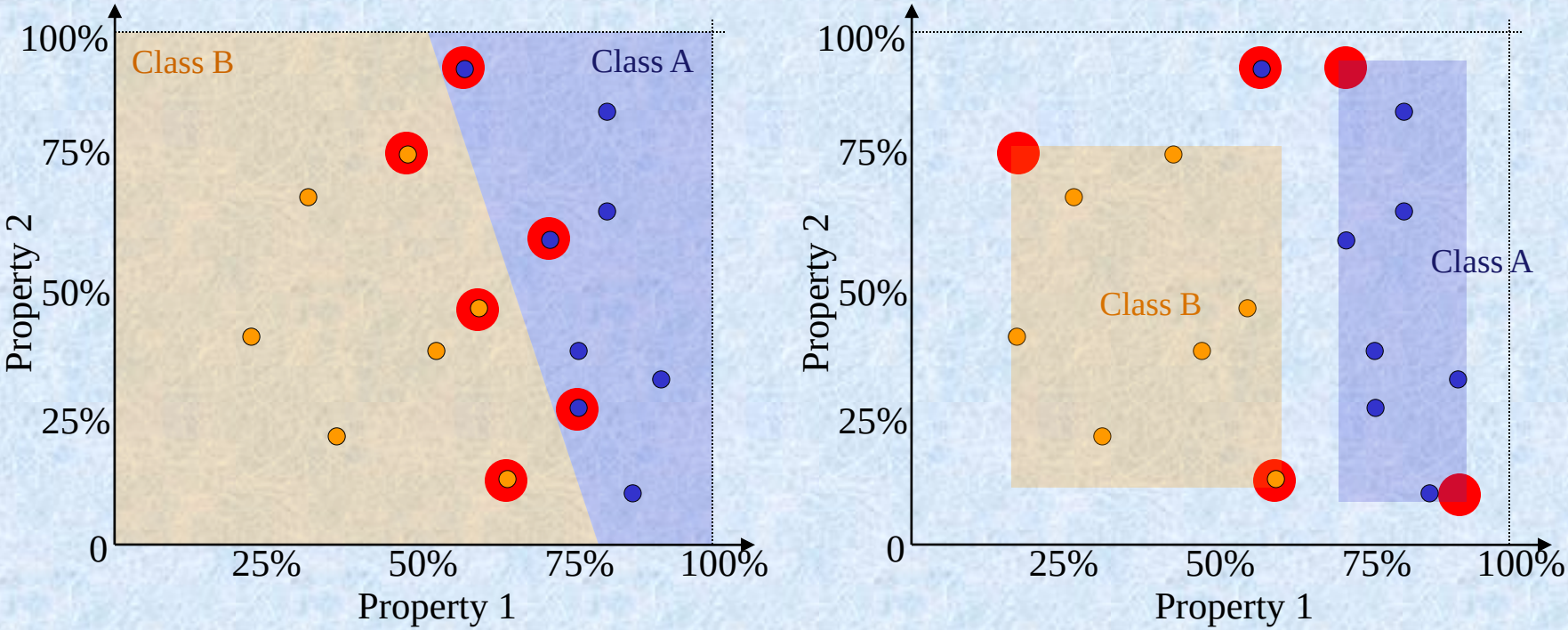
Incremental Learning: Final Step



Another caveat with using category boxes is the case of an instance from one class falling into the already existing box for the other class

ART will create overlapping boxes and use the smallest box to determine the class

Incremental Learning: Generalization

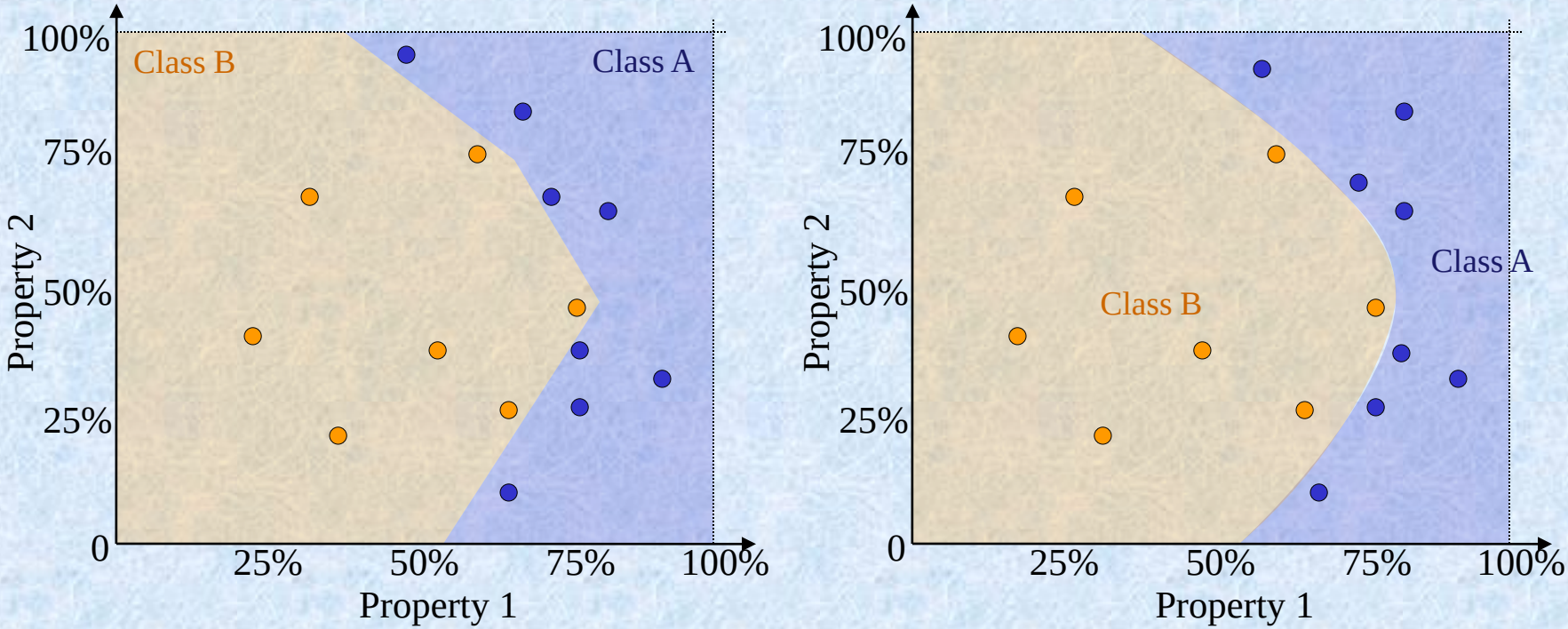


Algorithm does not have to keep all instances in memory

We can only keep the points that define the boundary

Or the coordinates of the boxes

Incremental Learning: Linearly Inseparable



Linearly inseparable cases can be classified using piece-wise linear class boundaries or continuous class boundaries

In general: for incremental learning the category box approach is more deterministic, but it is order dependent and less flexible

Batch Learning

Advantage:

- Can separate space into categories in the optimal way: lesser number of boundaries or category boxes

Disadvantage:

- Requires a complete training set to be present at the moment of learning

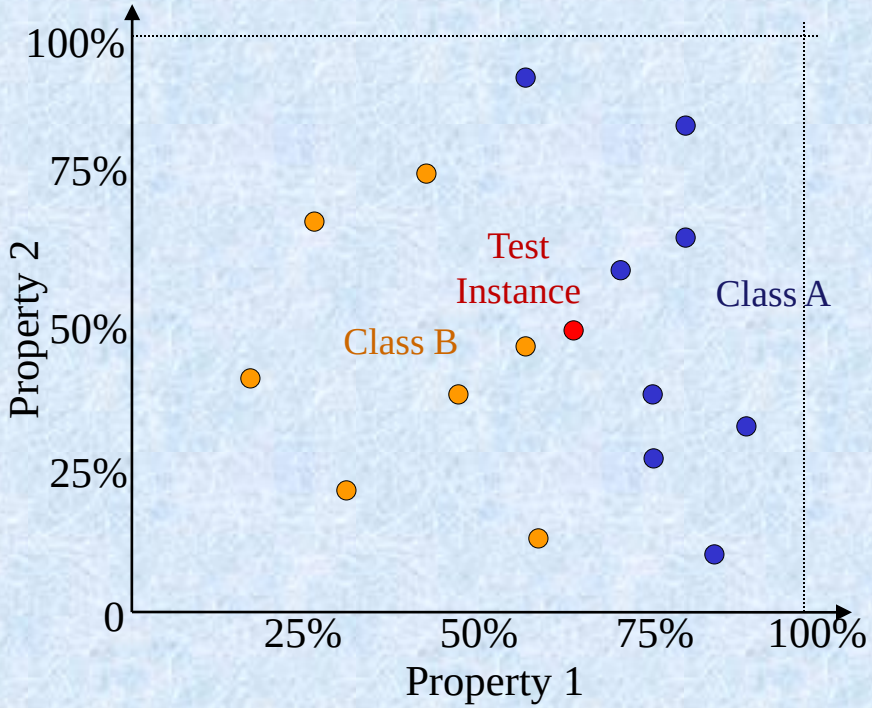
Back to Nearest Neighbor

Nearest neighbor classifier does not need to parcellate the space

For each instance in the test set a nearest neighbor is found according to some metric:

- L1 norm – taxicab, Manhattan, or city block distance
- L2 norm – Euclidean distance

Class of this neighbor is used as an output



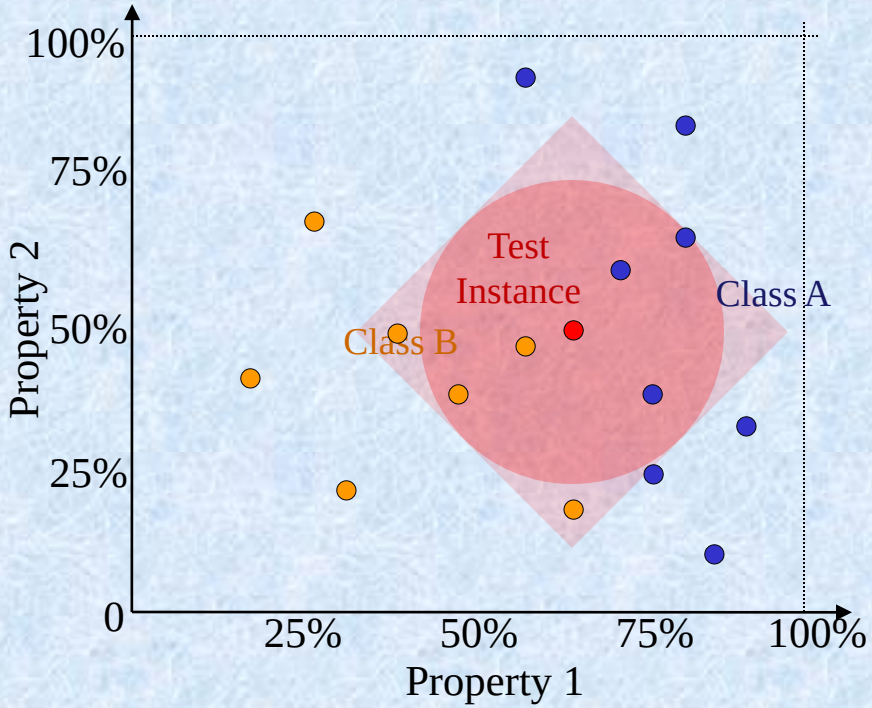
Can become very dependent on the outliers, to alleviate that the extension to use multiple neighbors was proposed

K Nearest Neighbor (KNN)

Train the classifier

Choose K (empirically or using some heuristic based on training instance distribution)

Choose norm (distance metric)



For each test instance

Pick the K nearest neighbors of the test instance according to the norm

Assign the output

Neighbors can vote, provide average, or use some other selection procedure to assign output

Mathematical Foundations

Probability that vector x will fall within region R

$$P = \int_R p(x')dx'$$

Expected value for k

$$e(k) = nP$$

If we assume that R is small

$$\int_R p(x')dx' \approx p(x)V$$

Combining all three

$$p(x) = \frac{k/n}{V}$$

KNN fixes k and changes the volume so that each test case has exactly k neighbors

Note that this formula gives $p(x)$ averaged over V

Mathematical Foundations

$$p(x) = \frac{k/n}{V}$$

Since k/n gives our probability estimate we want

$$\lim_{n\to\infty} k = \infty$$

At the same time we do not want the window to grow too fast, so k shall grow slower than n

$$\lim_{n\to\infty} \frac{k}{n} = 0$$

In fact, these are necessary and sufficient conditions for

$$\frac{k/n}{V}$$

to converge to $p(x)$

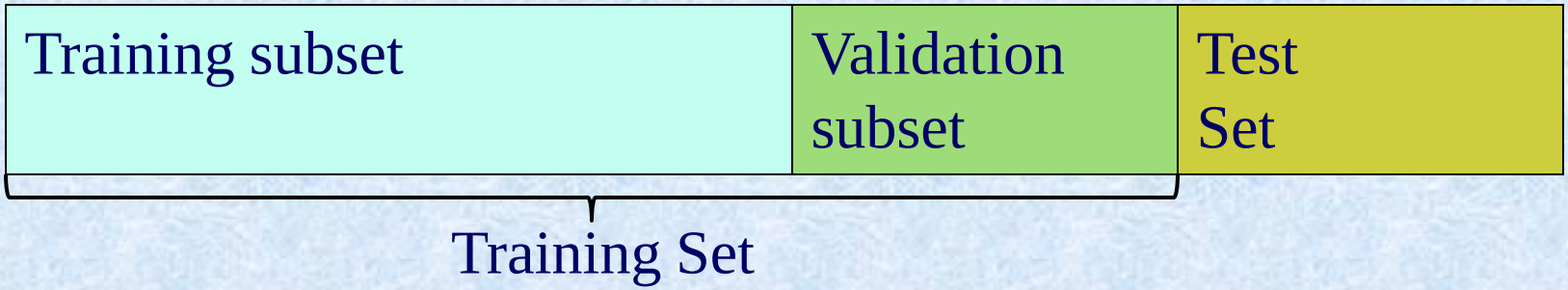
A good choice is

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

where k_1 is a parameter

Unfortunately, without any additional information there are no guidelines for the choice of k_1

Choosing k_1



For empirical choice:

- Split the training set into two – for training and for validation
- Train on the first subset, test on the second subset with different k_1
- Pick the k_1 with least error

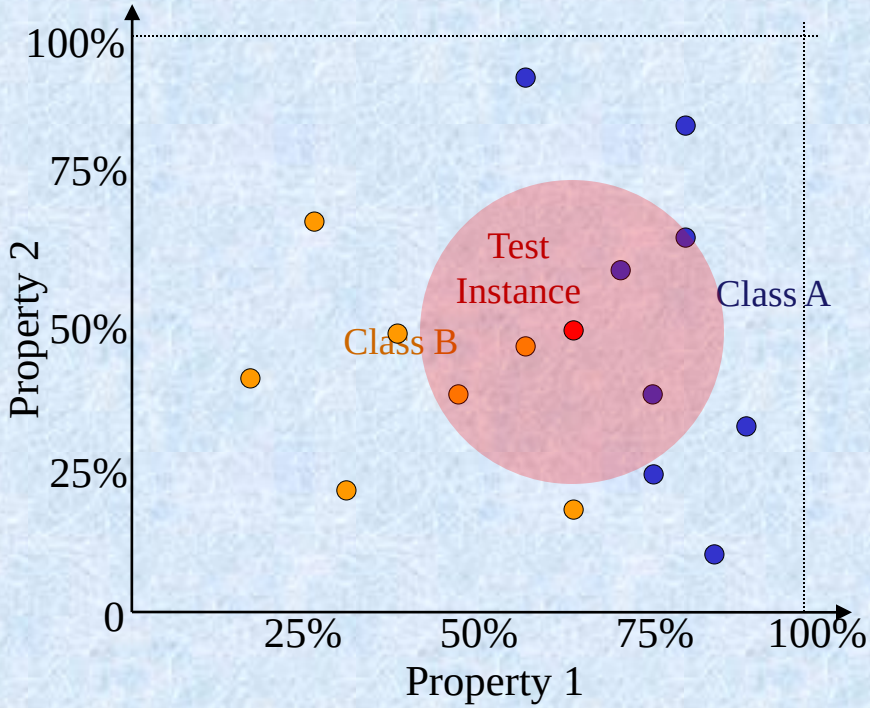
For better results do it multiple times with different splits

After k_1 is chosen retrain the system on complete training set before testing

K Nearest Neighbor

We can use a fraction of training instances that belong to a certain class as a measure of probability that the test case falls into the same class

This way we can theoretically split the whole space into decision regions with probabilistic boundaries



Another way is to use vote (preferably with odd *k* allowing to avoid ties in two-class case)

Computational Complexity

Straightforward approach is $O(dn^2)$

Partial distance approach: use a subset of d and if the distance is too large drop the point

- Based on non-decreasing nature of distance

Search tree approach: structure the training cases into a tree, drop the branches that are too far

- Not guaranteed that closest points will be found

Pruning: after training eliminate all the points that are surrounded by same class points

- Cannot add more points later (non-incremental)

K Nearest Neighbor Summary

Advantages:

- Simple, no adaptation just memorization of the training set

Disadvantages:

- Does not generalize, large memory requirements, brute force testing is slow