Problem set 2
Usvyatsov Mikhail

Problem 1

1. We can represent two training objects as two vectors x and y in the space.

   Then for object to classify can be represented as vector z in the same space.

   The decision boundry is the following:

   $||x - z||_2^2 = ||y - z||_2^2$
   $x^2 - 2 < x, z > +z^2 = y^2 - 2 < y, z > +z^2$
   $x^2 - 2 < x, z >= y^2 - 2 < y, z >$

   That is linear subject to z.

2. All the vectors belonging to the class A have the following definition:

   For every training point a belonging to A and test point x:

   $\Big\{ ||a - ||_2^2 \leq ||any\_other\_point\_not\_in\_A - x||_2^2$

   The same for other classes. And we know, that all that inequalities are linear s.t. x. QED.

Problem 2

1. $L(\omega_i) = \sum_{j=1}^{k} \lambda_i p(\omega_j|x) = \sum_{j=1}^{k} \lambda_i \dfrac{p(x|\omega_j)p(\omega_j)}{p(x)}$

   Bayes minimum cost decision rule:

   $argmin_\omega L(\omega)$

2. $\hat{\omega} = argmax_\omega p(\omega|x)$ We could see that if all λ are equal then all $p(\omega|x)$ are equal. So, it reduces to predicting most probable class.

Problem 3

1. The first step is on level 0 we have to sort all the data according to every feature that is $O(nd\ log_2(n))$ (criterion calculation is O(n) + (n - 1) O(d). In the worst case we have N nodes and each node remove only one example. Than for the next node i algorithm would require $O\left((n-i)d\ log_2\left(n-i\right)\right)$. Summing the complexity for every N nodes we will have exactly $O(n^2 d\ log_2(n))$.

2. If we have that each node removes approximately the half of our examples, than we have almost balanced tree. The height of this tree if $O(log_2(n))$. For every next level of nodes in this tree the coast of sorting would be $O\left(\frac{n}{2}dlog_2\left(\frac{n}{2}\right)\right)$. Thus the complexity for all the level would be $O\left(ndlog_2\left(\frac{n}{2}\right)\right)$. Summing this $log_2(N)$ times we have exactly $O(nd\ log_2^2(n))$ in the average.

Problem 4

1. Considering classification problem where we have two features and decision boundaries lies on the line $x_1 = x_2$ we can understand that in order to classify objects correctly we have to build the tree with ifinite height in order to be able to divide our space into two parts (border by feature x_1 depends on the values of x_2). But during building the tree we can build at most N nodes. Thus if the test set will have data out from the region of train data the prediction would be very much incorrect.

2. For each node t we can apply linear classifer (e.g. perceptron). Then to obtain weights and threshold we should train this perceptron on all the train data. Every false classification will punish perceptron and change the weights. Treshold is also easy it is always 0 for perceptron. Stopping criteria is the amount of weight changes per node. Once it becomes to small we can stop our process.