AI and Machine Learning

Zhiyun Lin

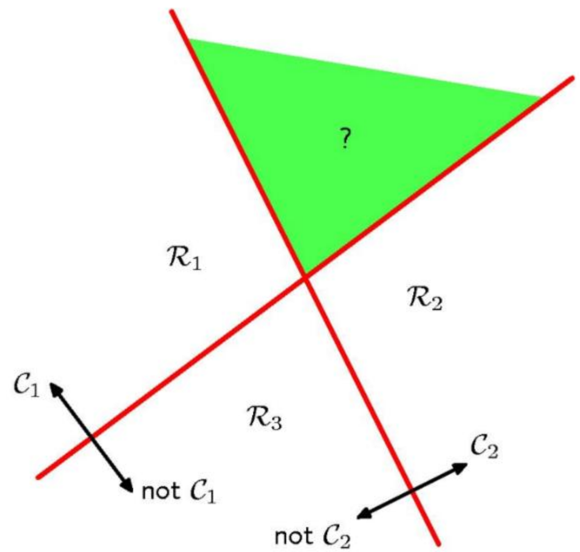
南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Multi-class Classification

- Least Squares Regression
- Perceptron and Logistic Regression
- MLP
- k-NN
- Decision Trees

Discriminant Functions for K > 2 classes

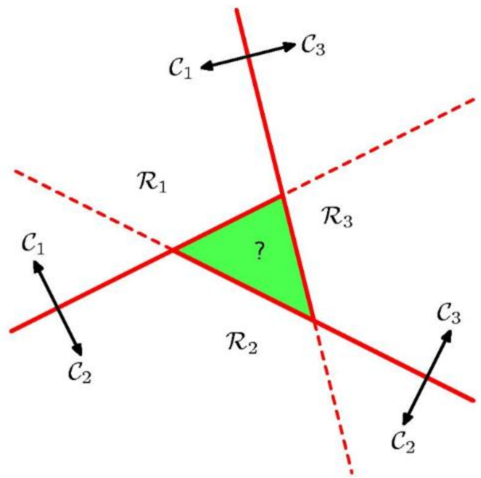
- **First idea:** Use K classifiers, each solving a two class problem of separating point in a class C_k from points not in the class.
- Known as **1 vs all** or **1 vs the rest** classifier



> PROBLEM: More than one good answer for green region!

Discriminant Functions for K > 2 classes

- **Another simple idea:** Introduce $K(K-1)/2$ two-way classifiers, one for each possible pair of classes
- Each point is classified according to majority vote amongst the disc. func.
- Known as the **1 vs 1** classifier



> PROBLEM: Two-way preferences need not be transitive

K-Class Discriminant

We can avoid these problems by considering a single K-class discriminant comprising K functions of the form

$$y_k(x) = w_k^T x + b_k, \quad k = 1, \dots, K$$

and then assigning a point x to class C_k if

$$y_k(x) > y_j(x), \quad \forall j \neq k$$

> Note that w_k is now a vector, not the k-th coordinate

K-Class Discriminant

- The **decision boundary** between class C_j and class C_k is given by

$$y_j(x) = y_k(x)$$

- and thus it's a $(D-1)$ dimensional hyperplane defined as

$$(w_k - w_j)^T x + (b_k - b_j) = 0$$

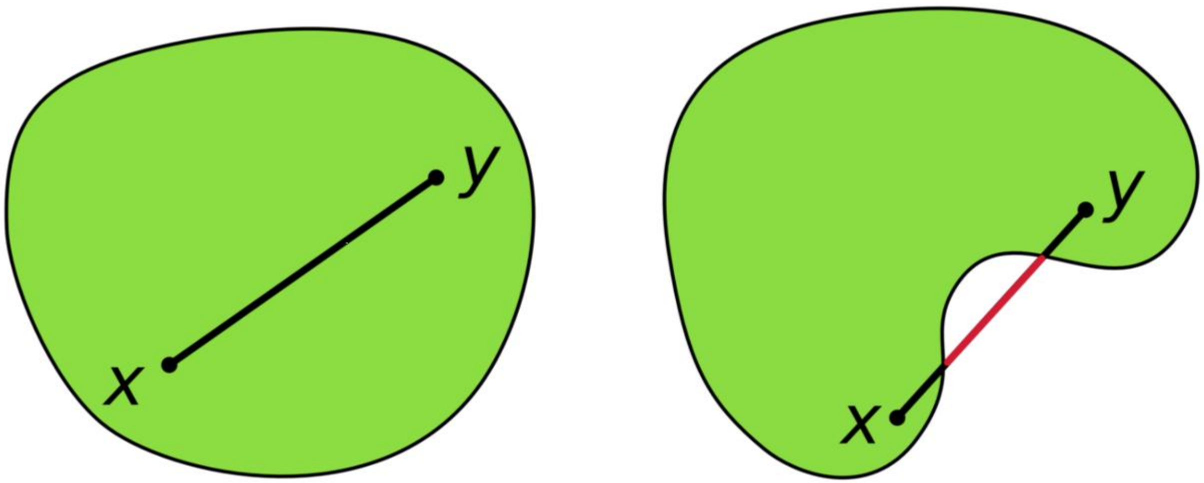
> What about the binary case? Is this different?

> What is the shape of the overall decision boundary?

- The decision regions of such a discriminant are always **singly connected** and **convex**

Convex Object

- In Euclidean space, an object is **convex** if for every pair of points within the object, every point on the straight line segment that joins the pair of points is also within the object

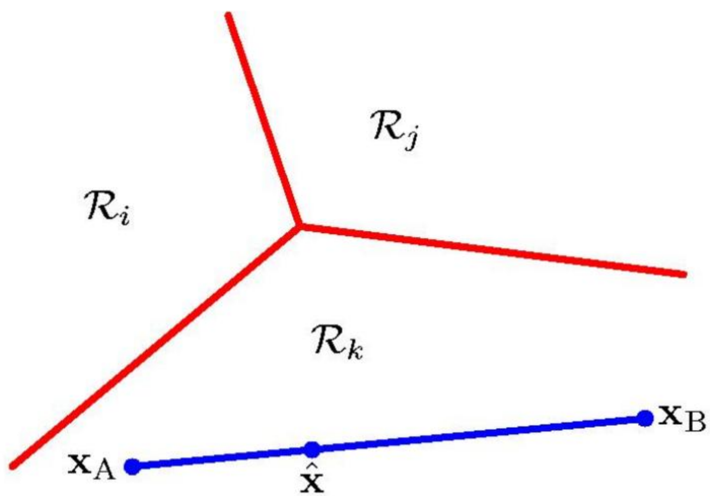


> Which object is convex?

Decision Region

- The decision regions of such a discriminant are always **singly connected and convex**
- Consider 2 points x_A and x_B that lie inside decision region $\mathcal{R}_k$
- Any convex combination $\hat{x}$ of those points also will be in $\mathcal{R}_k$

$$\hat{x} = \lambda x_A + (1 - \lambda) x_B$$



Proof

- A convex combination point, i.e., $\lambda \in [0, 1]$

$$\hat{x} = \lambda x_A + (1 - \lambda) x_B$$

- From the linearity of the classifier $y(x)$

$$y_k(\hat{x}) = \lambda y_k(x_A) + (1 - \lambda) y_k(x_B)$$

- Since x_A and x_B are in $\mathcal{R}_k$ it follows that

$$y_k(x_A) > y_j(x_A), \quad y_k(x_B) > y_j(x_B), \quad \forall j \neq k$$

- Since λ and $1 - \lambda$ are positive, then $\hat{x}$ is inside $\mathcal{R}_k$

- Thus $\mathcal{R}_k$ is singly connected and convex

1-of-K Encoding

> 1-of-K encoding:
>
> For multi-class problems (with K classes), instead of using $t = k$ (target has label k) we often use a **1-of-K encoding**, i.e., a vector of K target values containing a single 1 for the correct class and zeros elsewhere

- Example:
  - For a 4-class problem, we would write a target with class label 2 as:

$$t = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Linear Regression for MC Classification

From before we have:

$$y_k(x) = w_k^T x + b_k, \quad k = 1, \ldots, K$$

which can be rewritten as:

$$y(x) = \bar{W}^T \bar{x}$$

where the k-th column of $\bar{W}$ is $[b_k, w_k^T]$, and $\bar{x} = [1, x]^T$

- **Training:** How can I find the weights $\bar{W}$ with the standard sum-of-squares regression loss?

Least square solution

- Sum-of-least-squares loss:

$$\begin{aligned} l(\bar{W}) &= \sum_{n=1}^{N} ||\bar{W}^T \bar{x}^{(n)} - t^{(n)}||^2 \\ &= ||X\,\bar{W} - T||_F^2 \end{aligned}$$

where the n-th row of X is $\bar{x}^{(n)}$, and the n-th row of T is $t^{(n)}$.

- Setting derivative w.r.t. $\bar{W}$ to 0, we get:

$$\tilde{W} = (X^T X)^{-1} X^T T$$

Logistic Regression for MC Classification

- Associate a set of weights with each class, then use a normalized exponential output

$$p(C_k|x) = y_k(x) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

where **the activations** are given **on**

$$z_k = w_k^T x + b_k$$

> The functione $\frac{\exp(z_k)}{\sum_j \exp(z_j)}$ is called a **softmax function**

Maximum Likelyhood Esitmation

The likelihood

$$L(T|x^{(1)}, \ldots, x^{(N)}, \bar{W}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \left[p(C_k|x^{(n)})\right]^{t_k^{(n)}} = \prod_{n=1}^{N} \prod_{k=1}^{K} \left[y_k(x^{(n)})\right]^{t_k^{(n)}}$$

with

$$p(C_k|x) = y_k(x) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

where k-th row of T is 1-of-K encoding of example k and

$$z_k = w_k^T x^{(k)} + b_k$$

Multi-class Logistic Regression

- Derive the loss by computing the nagative log-likelihood:

$$E(\bar{W}) = -\log L(T|\bar{W}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_k^{(n)} \log\left[y_k(x^{(n)})\right]$$

> This is known as the **cross-entropy error** for multi-class classification

- How do we obtain the weights?

Training Multi-class Logistic Regression

- Conisder one example.

$$E(\bar{W}) = -\sum_{k=1}^{K} t_k \log\left[y_k(x)\right]$$

- Then the derivate of E w.r.t. y_k is obtained

$$\frac{\partial E}{\partial y_k} = -\frac{t_k}{y_k}$$

Training Multi-class Logistic Regression

- Computer the gradient the softmax function:

$$y_k(z) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

where the derivatives are

$$(1)\ \frac{\partial y_k}{\partial z_k} = \frac{\exp(z_k)}{\sum_j \exp(z_j)} - \frac{\exp(z_k)\exp(z_k)}{\left[\sum_j \exp(z_j)\right]^2} = y_k - y_k^2$$

$$(2)\ \frac{\partial y_k}{\partial z_m} = -\frac{\exp(z_k)\exp(z_m)}{\left[\sum_j \exp(z_j)\right]^2} = -y_k \cdot y_m, \quad m \neq k$$

- The overall form is

$$\frac{\partial y_k}{\partial z_m} = \delta(k, m) y_k - y_k y_m$$

Training Multi-class Logistic Regression

- Now move backward one more step:

$$
\begin{aligned}
\frac{\partial E}{\partial z_m} &= \sum_{k=1}^{K} \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial z_m} \\
&= -\sum_{k=1}^{K} \frac{t_k}{y_k} [\delta(k,m) y_k - y_k y_m] \\
&= -\sum_{k=1}^{K} t_k \delta(k,m) - \left[\sum_{k=1}^{K} t_k\right] y_m \\
&= y_m^{(n)} - t_m^{(n)}
\end{aligned}
$$

Training Multi-class Logistic Regression

Thus, the gradient of E with respect to the parameter is given by

$$\frac{\partial E}{\partial w_{m,i}} = \frac{\partial E}{\partial z_m}\frac{\partial z_m}{\partial w_{m,i}} = [y_m - t_m] \cdot x_i$$

$$\frac{\partial E}{\partial b_m} = \frac{\partial E}{\partial z_m}\frac{\partial z_m}{\partial b_m} = [y_m - t_m]$$

> The derivative is the error times the input

The gradient for a batch

- In a matrix form, the gradient for a batch becomes

$$\nabla E(\bar{W}) = -X^T(T - Y)$$

> $$X = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix}, \quad T = \begin{bmatrix} t_1^{(1)} & \cdots & t_K^{(1)} \\ \vdots & \cdots & \vdots \\ t_1^{(N)} & \cdots & t_K^{(N)} \end{bmatrix}, \text{ and } Y = \mathrm{softmax}(X\bar{W})$$

> The softmax takes the ith row of $X\bar{W}$ and outputs the ith row of Y.

- This is all there is to learning in logistic regression for multi-class classification.

Softmax for 2 Classes

- Let's write the probability of one of the classes

$$p(C_1|x) = y_1(x) = \frac{\exp(z_1)}{\sum_j \exp(z_j)} = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)}$$

- I can equivalently write this as

$$p(C_1|x) = y_1(x) = \frac{\exp(z_1)}{\sum_j \exp(z_j)} = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)} = \frac{1}{1 + \exp(-(z_1 - z_2))}$$

> So the logistic is just a special case that avoids using redundant parameters

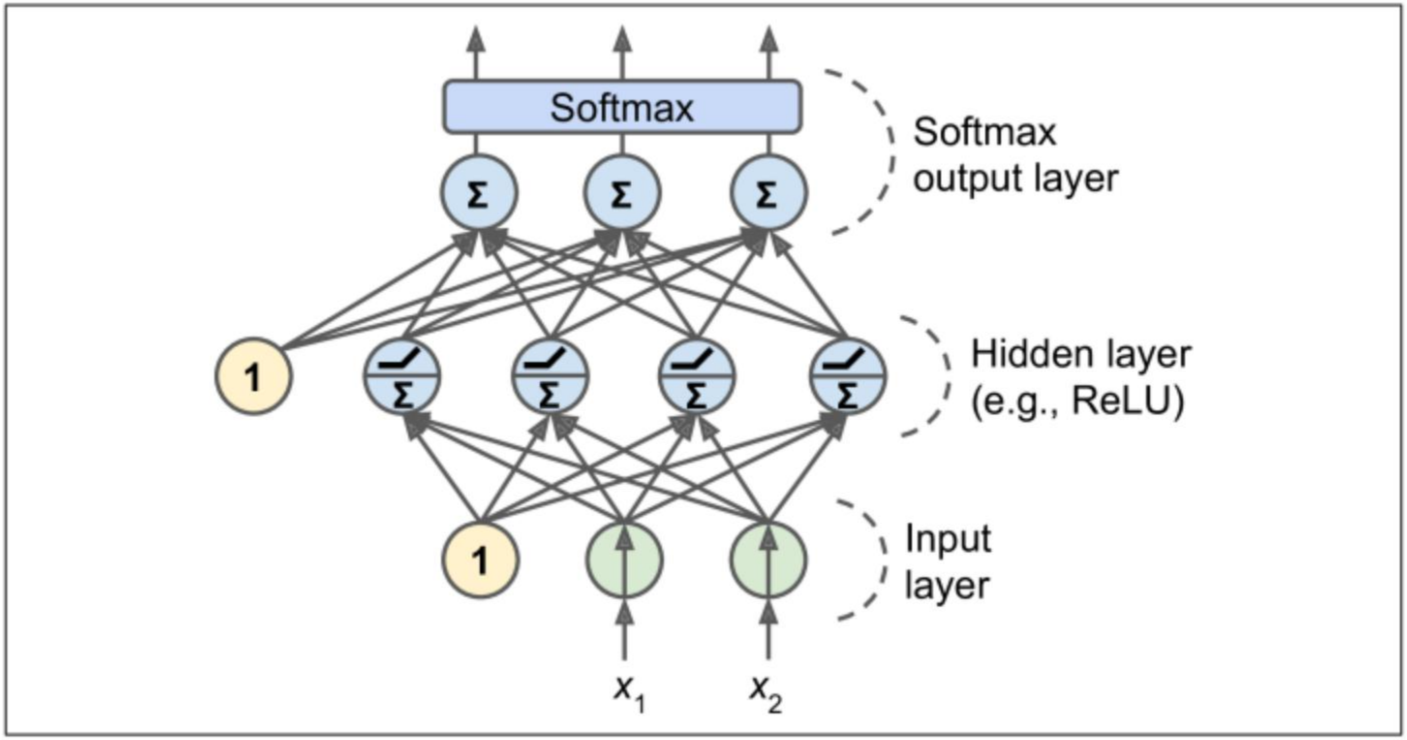
Softmax for 2 Classes

- Rather than having two separate set of weights for the two classes, combine into one

$$z' = z_1 - z_2 = (w_1^T x + b_1) - (w_2^T x + b_2) = w^T x + b$$

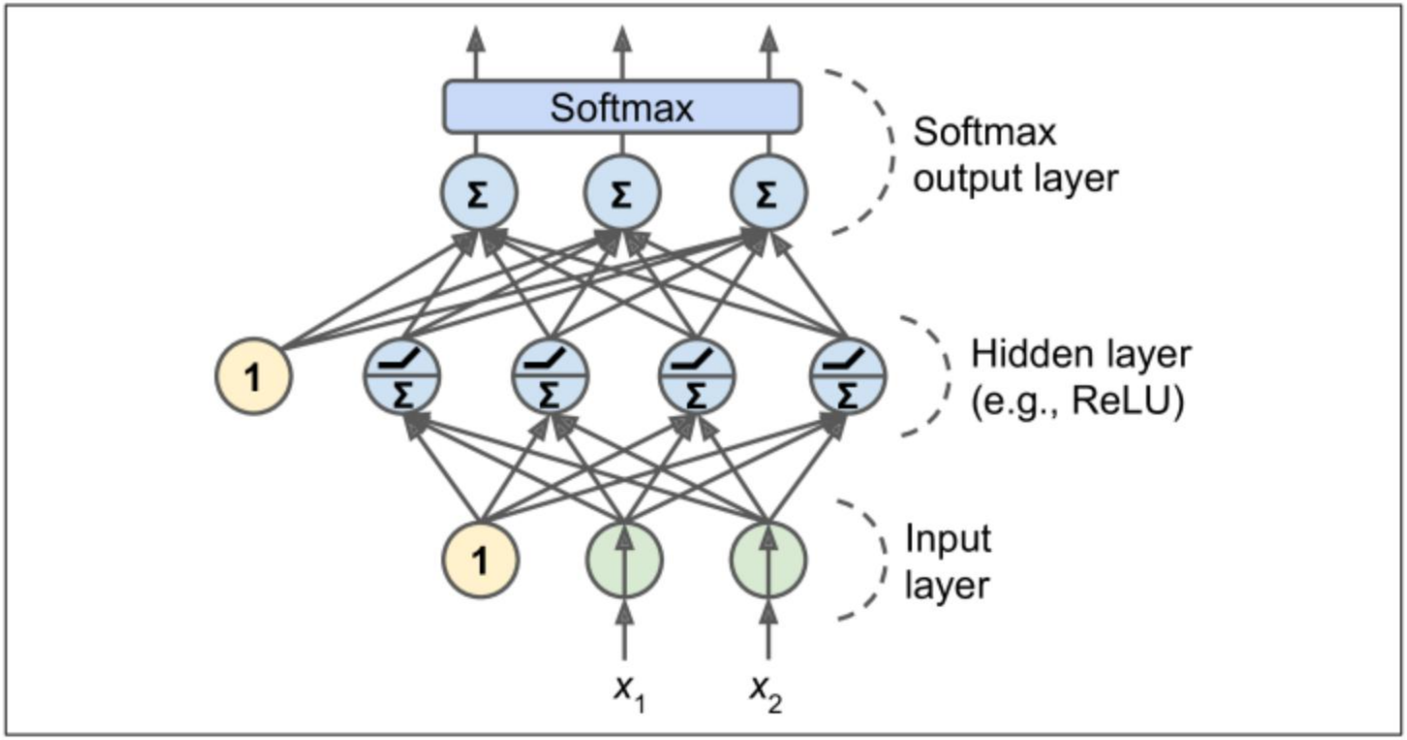
> The over-parameterization of the softmax is because the probabilities must add to 1.

MLP for MC Classification



- Foward propagation
- Backward propagation

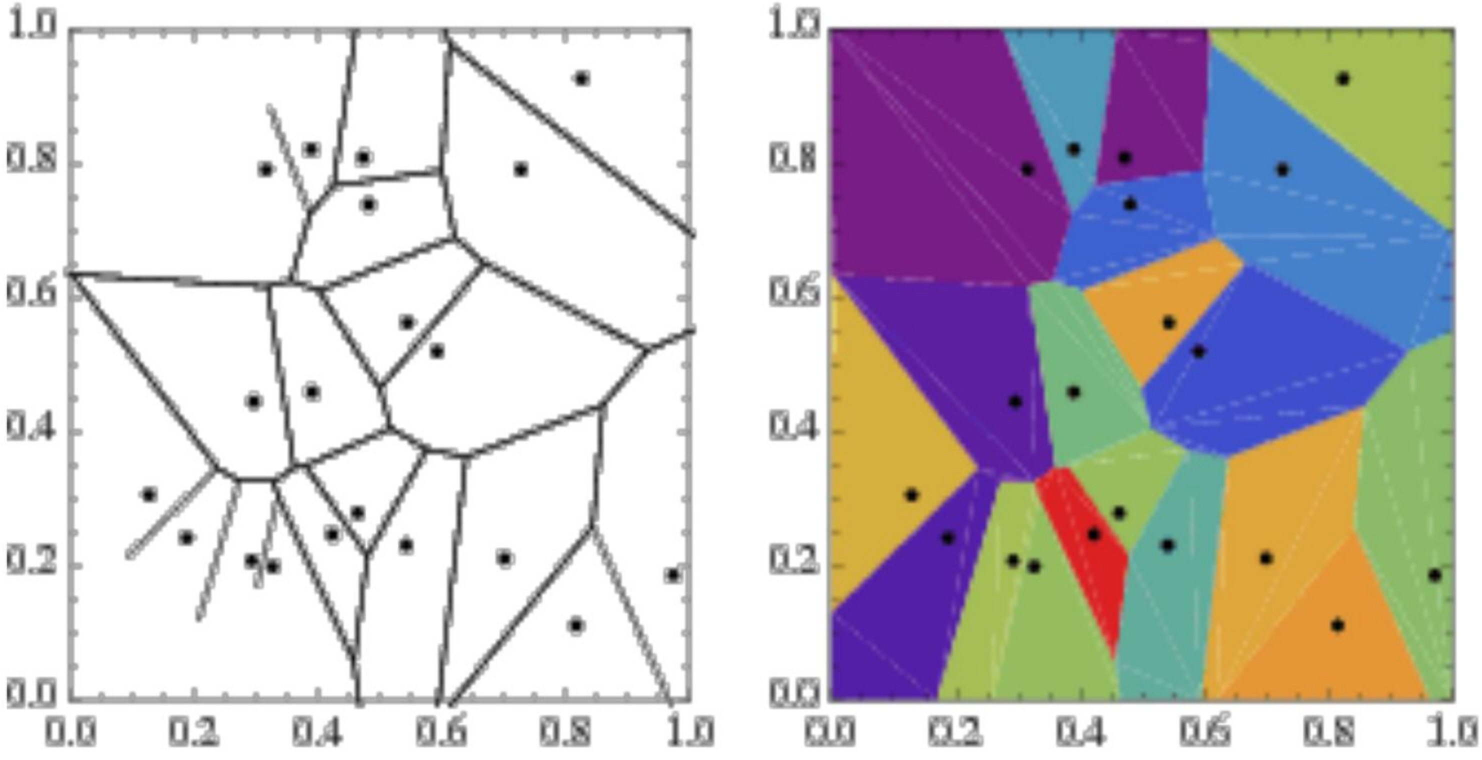
MLP for MC Classification



- The only difference is at the output layer with softmax
- The K outputs correspond to the probability of belonging to these classes
- The K outputs are **fully connected** with the linear output at the output layer via softmax

Multi-class k-NN

- Can directly handle multi class problems



Multi-class Decision Trees

- Can directly handle multi class problems