UiO : University of Oslo

institutt for informatikk

IN4050 - Introduction to Artificial Intelligence and Machine Learning

Multi-class Classification

Ali Ramezani-Kebrya

Multi-Class Classification One vs. Rest

IN4050 Introduction to Artificial Intelligence and Machine Learning

Multi-class classification

Classification

- Assign a label (class) from a finite set of labels to an observation

?

- So far, many algorithms and examples have been binary: *yes-no, 1-0*
- But many classification tasks are multi-class:
  - To each observation $\boldsymbol{x}$ choose one label from a finite set **C**
- What is different?

Multi-class classification

- A finite set, **C**, of *n* different labels ($n > 2$)
- To each observation $\boldsymbol{x}$ choose one label from the set **C**

We will consider two approaches:

- *One vs. rest classifier*
  - (also called one vs. all)
- *Multinomial logistic regression*, or *softmax regression*

1-of-N or "one hot encoding"

- The labels might be categorical:
  - 'apple', tomato', 'dog', 'horse'
- The algorithms demand numerical attributes.
- First attempt
  - 'apple' = 1
  - 'tomato' = 2
  - 'dog' = 3
  - etc.
- Why isn't this a good idea?

treats classification like regression

- Better:
  - 'apple' = (1, 0, 0, 0, 0, 0)
  - 'tomato' = (0, 1, 0, 0, 0, 0)
  - 'dog' = (0, 0, 1, 0, 0, 0)
  - etc.
- Both the target and the predicted value are vectors.

From multi-label to multi-class

Multi-label classifier

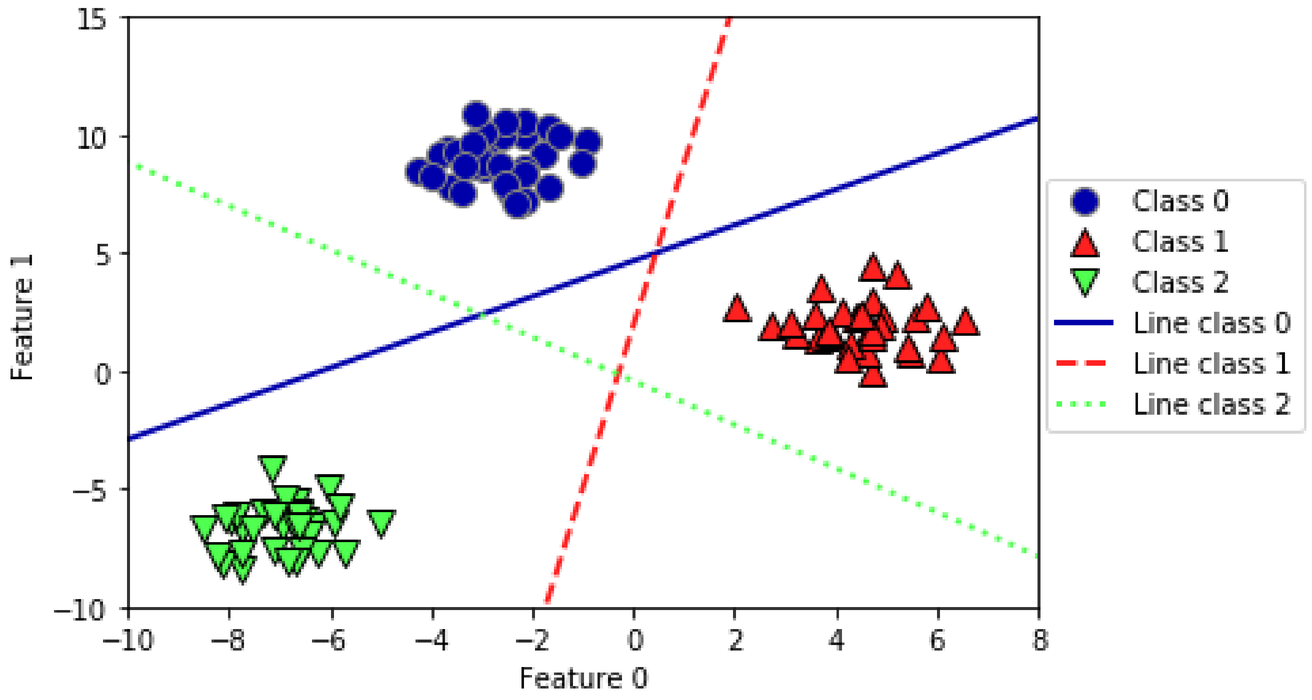
- Make *n* different classifiers, one for each class
- For classifier *j*:
  - consider class *j* the positive class
  - all other items in the negative class
  - train a classifier f_j
- Application
  - Assign a label c_j to an item if and only if it is classified as positive by f_j.

Multi-class classifier

- "To each observation $\boldsymbol{x}$ choose one label from the set **C**"
- How can a multi-label classifier be turned into a multi-class classifier?

One vs. rest (also called one vs. all)

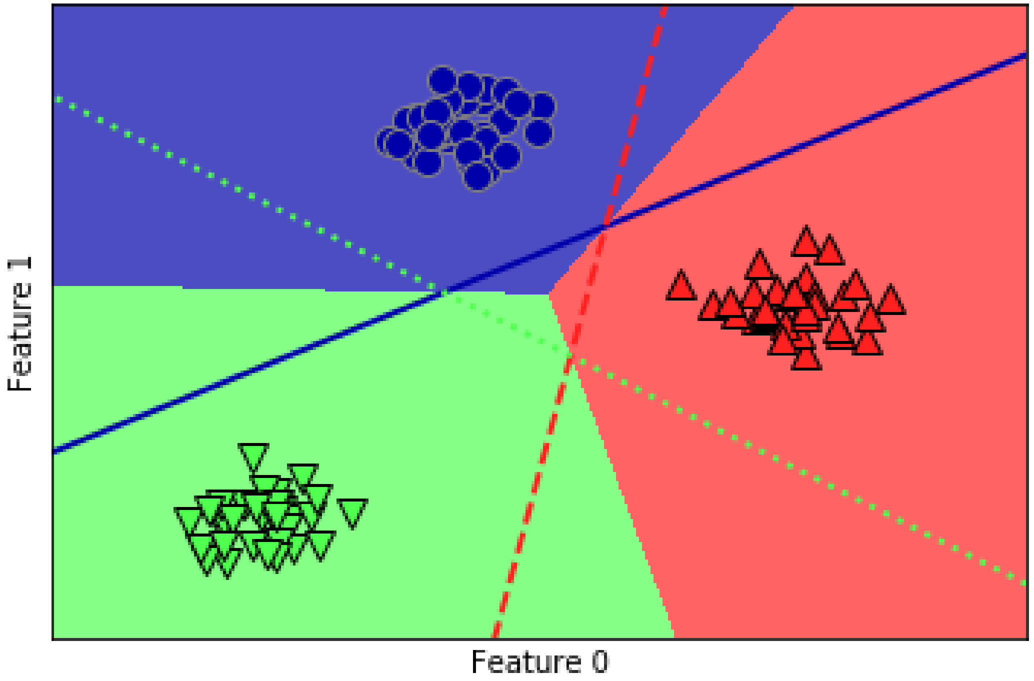
- Start like the multi-label classifier: make one classifier for each class.
- It is easy to decide for items which fall into exactly one class
- But what if they fall into
  - More than one class?
  - No classes?



https://github.com/amueller/introduction_to_ml_with_python

One vs. rest

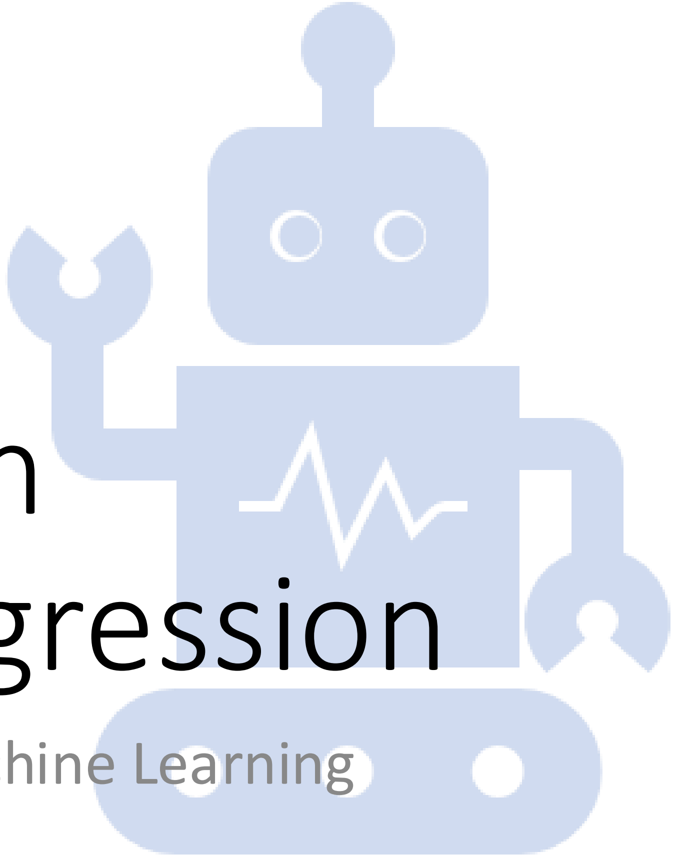
- If each classifier predicts a score, compare the scores for the classes
- Choose the class with the highest score.
- E.g., log. reg.:
  - Probability of being red: 0.8
  - Probability of being blue: 0.7
  - Choose red



https://github.com/amueller/introduction_to_ml_with_python

Multi-Class Classification
Multinomial Logistic Regression

IN4050 Introduction to Artificial Intelligence and Machine Learning



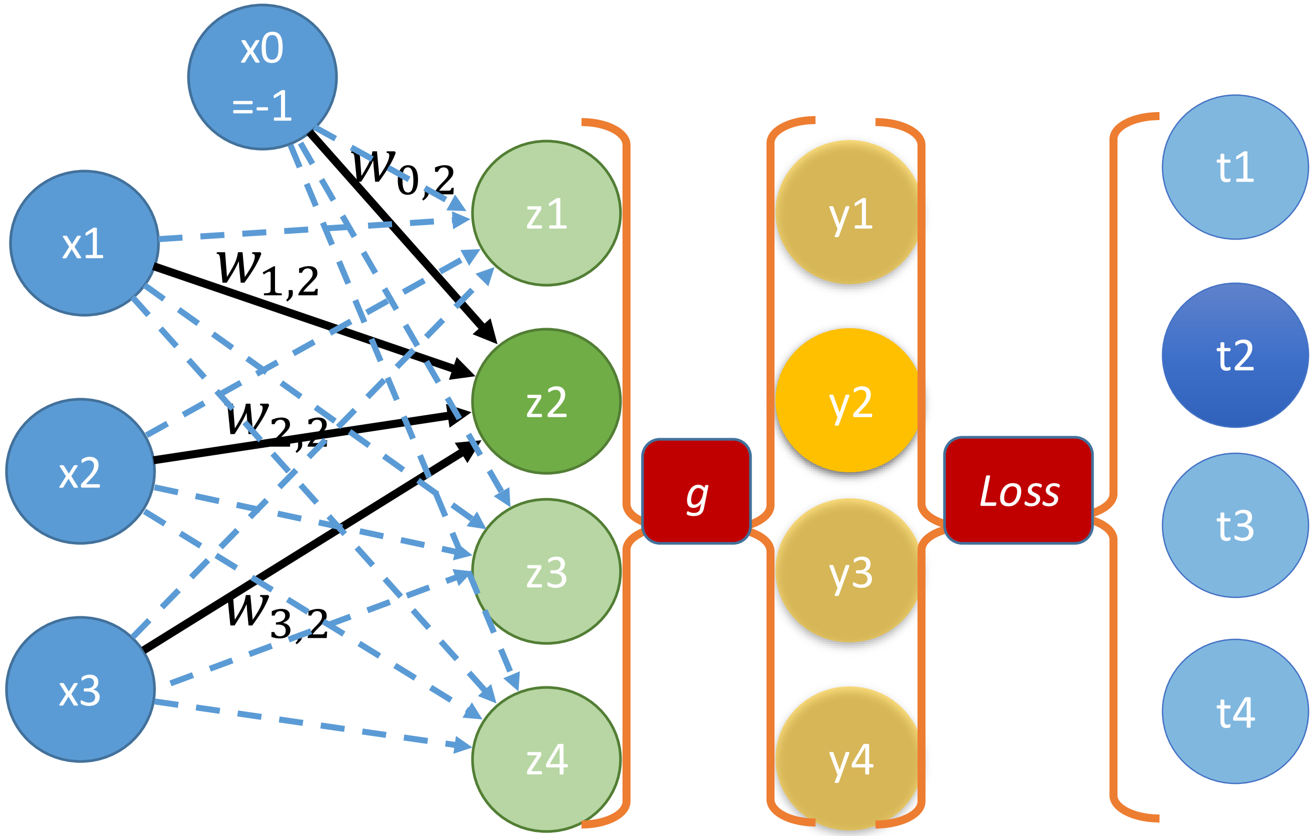
Towards Multinomial Logistic Regression

- On the way, we will adopt a broader perspective
- Comparing perceptron, linear regression, logistic regression
- Compare to Marsland
- Shortly describe a multi-class perceptron

A general view

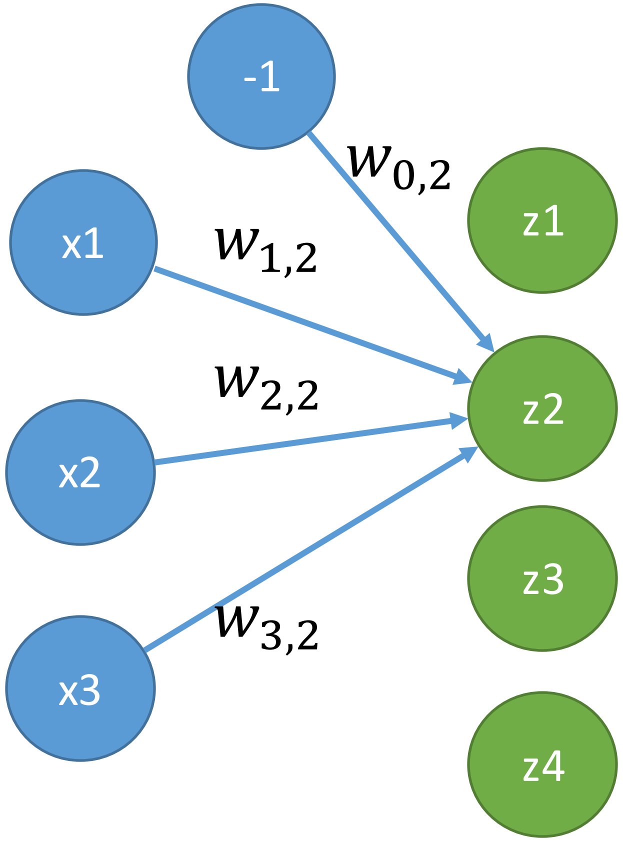
- Common
  - $z_j = \sum_{i=0}^{n} w_{i,j} x_i$
  - $w_{i,j}$ is the weight into node j from node i
    - Some index in opposite order

Binary classifiers		
Classifier	***g***	**Loss**
Perceptron	Step	0-1 loss
Lin. Regr.	Identity	MSE
Log.Regr.	Logistic	Cross-entropy



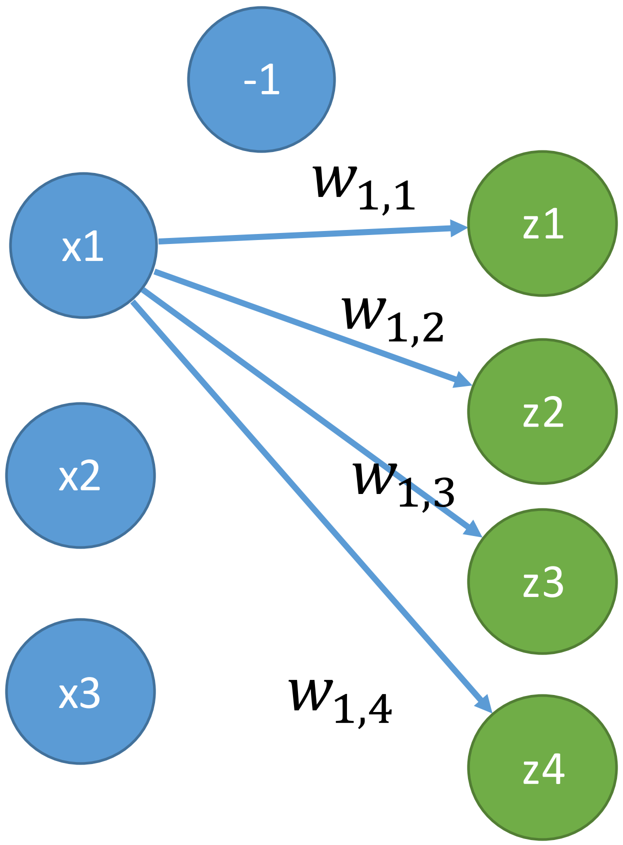
Connections going into a node

$$\begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_m \end{bmatrix} \begin{bmatrix} w_{0,1} & w_{0,2} & \cdots & w_{0,n} \\ w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} = \begin{bmatrix} z_1 & z_2 & \cdots & z_n \end{bmatrix}$$



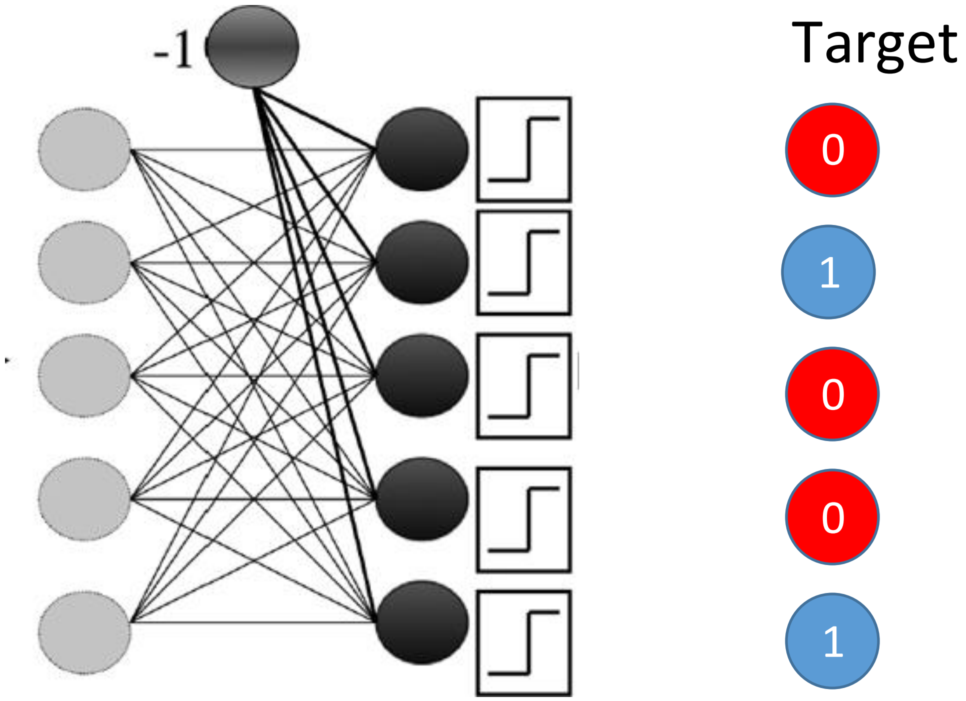
Connections going out of a node

$$\begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_m \end{bmatrix} \begin{bmatrix} w_{0,1} & w_{0,2} & \cdots & w_{0,n} \\ \boxed{w_{1,1} \quad w_{1,2} \quad \cdots \quad w_{1,n}} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} = \begin{bmatrix} z_1 & z_2 & \cdots & z_n \end{bmatrix}$$



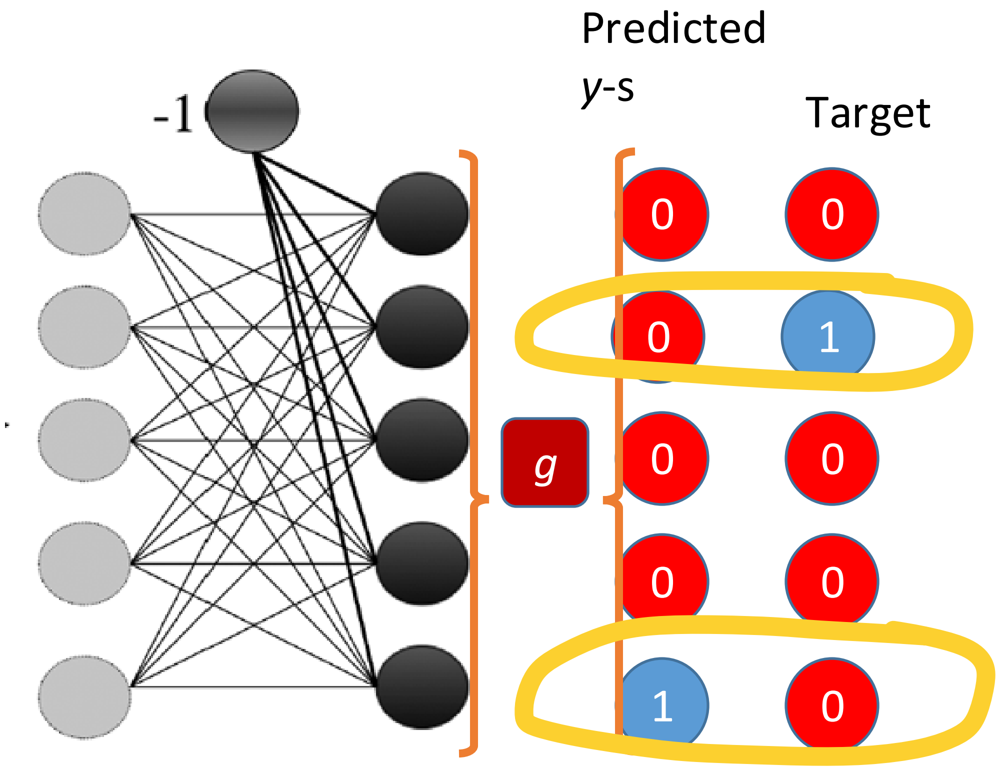
Multi-label perceptron

- Marsland's description of perceptron:
  - Possible targets: (0,1,0,0,1)
  - $y_j = g(z_j) = \begin{matrix} 1 \; if \; z_j > 0 \\ 0 \; if \; z_j \leq 0 \end{matrix}$
  - Loss is 0-1 loss for each *j*
- Describes a multi-label classifier
  - Each y_j depends only on the $w_{i,j}$'s
  - This could have been described as *n* independent binary perceptrons



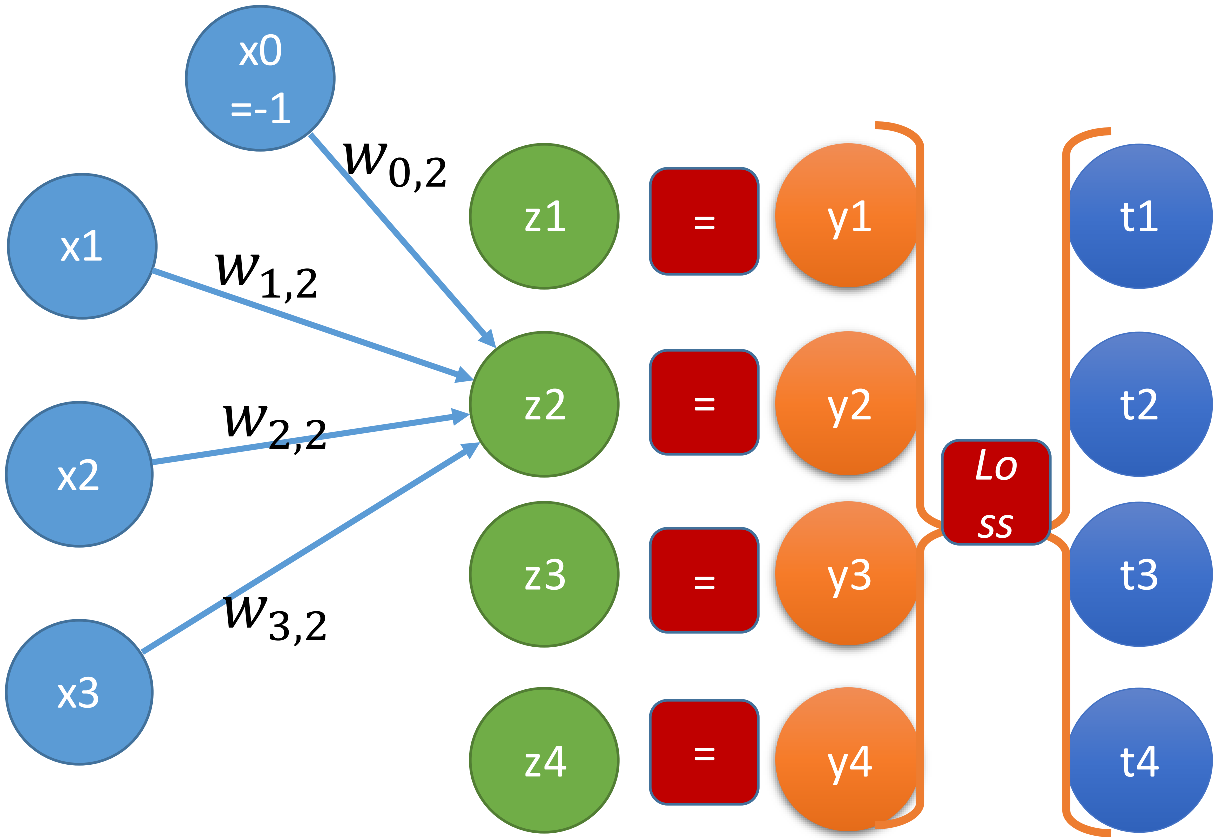
Multi-class perceptron

- The target contains one 1, the rest are 0s
- $g(z_1, z_2, \dots, z_n) =$
- $argmax\,(z_1, z_2, \dots, z_n)$
  - The index with the max value
- The update rule (0-1 loss)
  - $w_{i,j} = w_{i,j} - \eta(y_j - t_j)x_i$
  - will correct for $j = 2$ and $j = 5$
  - leaves the other weights unaltered



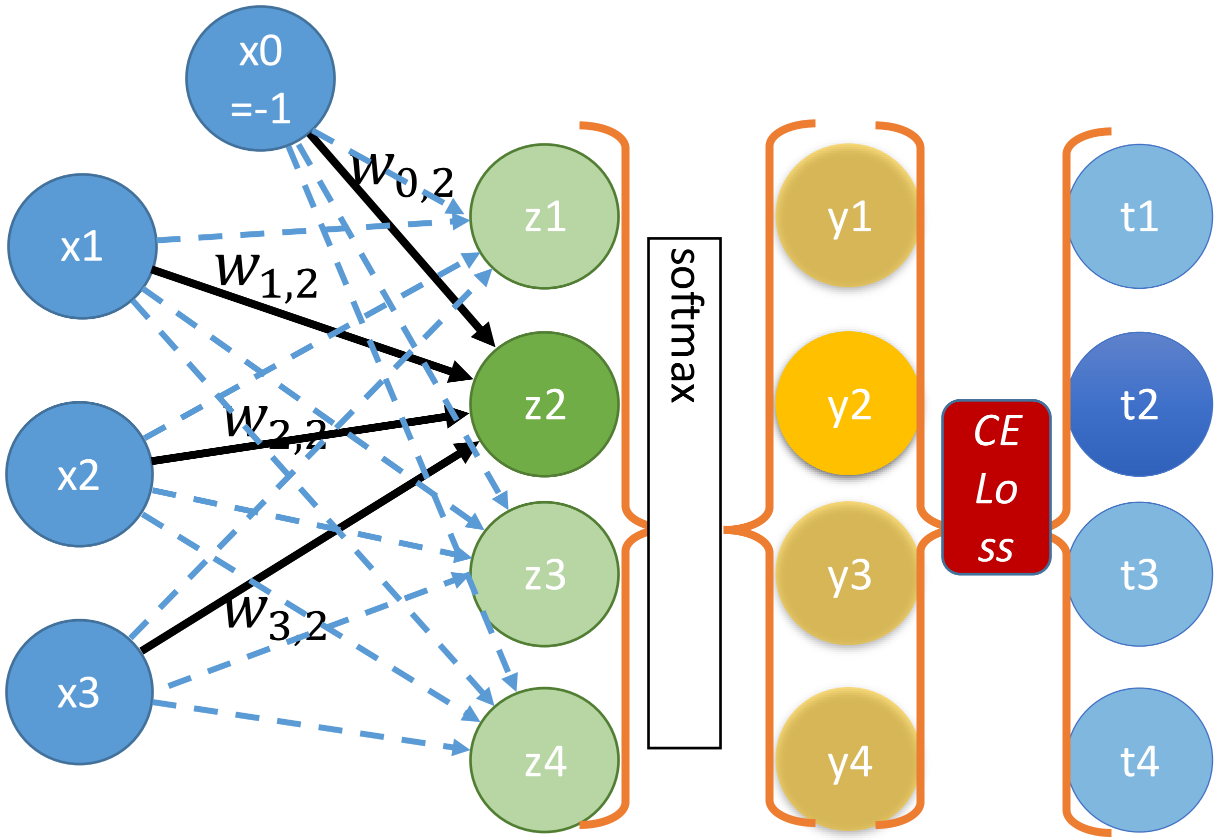
Multi-output linear regression

- $y_j = g(z_j) = z_j$
- MSE-loss: $\sum_{k=1}^{N}(\sum_{j=1}^{n}(y_{k,j} - t_{k,j})^2)$
  - n output nodes
  - N input items
- y_j independent of $w_{i,k}$ $k \neq j$,
  - hence corresponds to n independent models
  - (Gets more interesting for multi-layer networks)



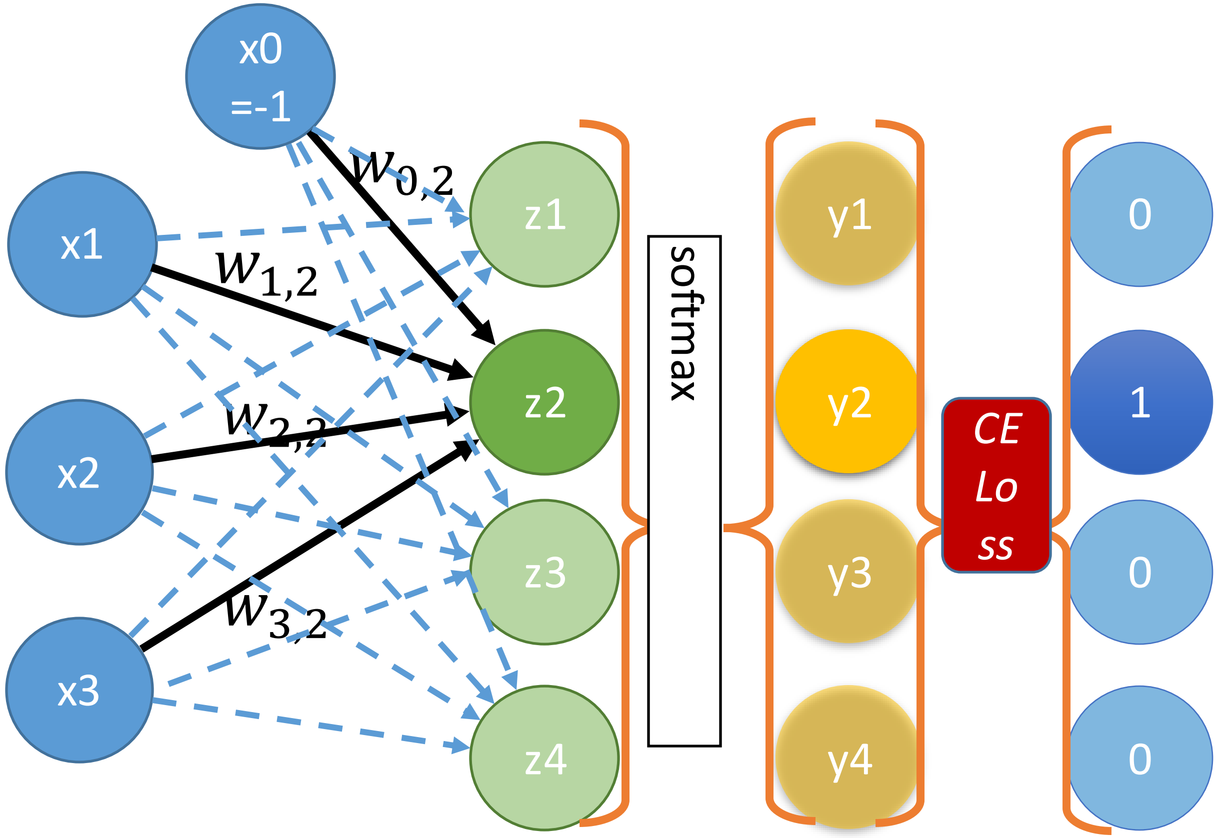
Multinomial Logistic Regression

- $z_j = \sum_{i=0}^{n} w_{i,j} x_i$
- Apply the softmax-function, S, where
  - $y_j = (S(z_1, \dots, z_n))_j = \frac{e^{z_j}}{\sum_{k=1}^{n} e^{z_k}}$
- Observe:
  - y_j depends on all the z_k
  - If $z_h > z_k$ then $y_h > y_k$
  - $0 < y_j < 1$
  - $\sum_{j=1}^{n} y_j = 1$
  - A probability distribution
  - $\hat{P}\left(C_j \middle| \vec{x}\right) = \frac{e^{\vec{w}_j \cdot \vec{x}}}{\sum_{k=1}^{n} e^{\vec{w}_k \cdot \vec{x}}}$



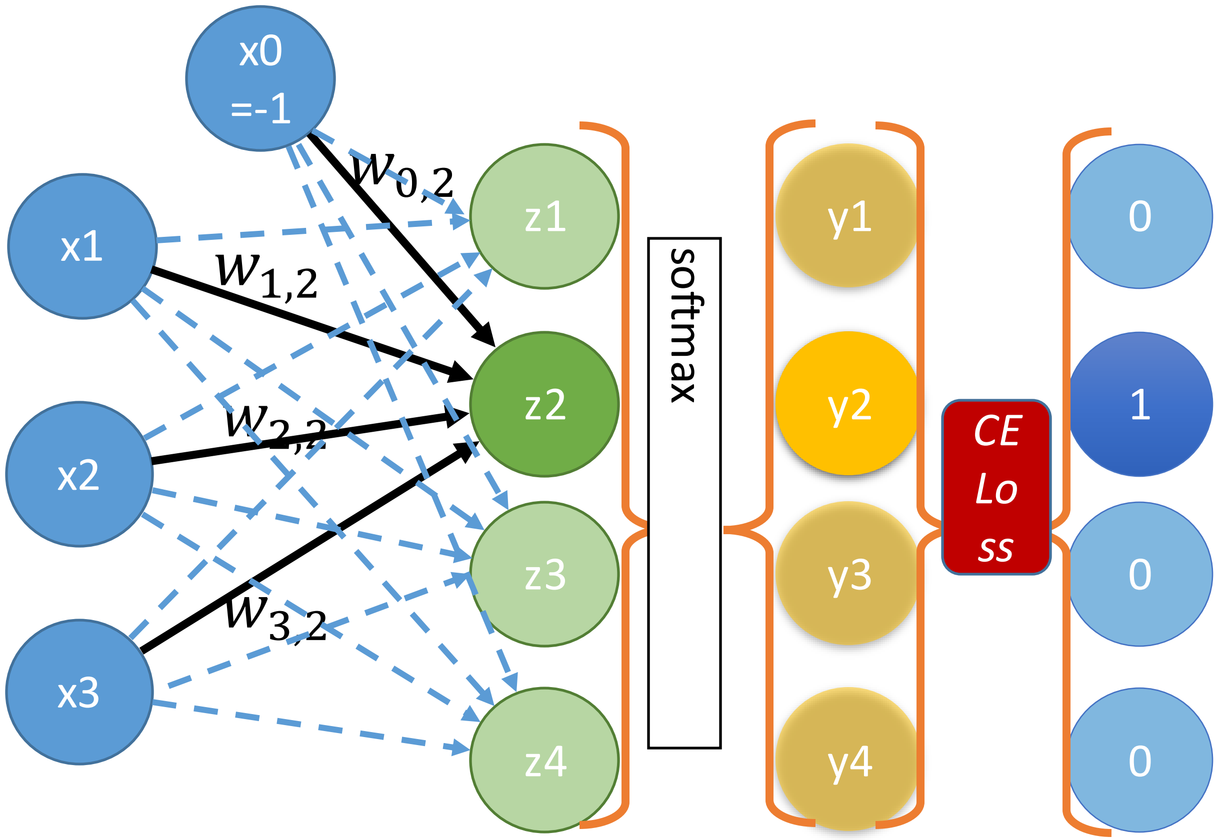
Training Multinomial Logistic Regression 1

- The target has the form $(0,0, \dots, 0,1,0, \dots, 0)$, say
  - $t_s = 1$ and $t_j = 0$ for $j \neq s$
- We compare
  - $\mathbf{y} = (y_1, y_2, \dots y_n)$
- to the target labels
  - $\mathbf{t} = (t_1, t_2, \dots t_n)$
- using cross-entropy loss
  - $L_{CE}(\boldsymbol{y}, \boldsymbol{t}) = -\sum_{j=1}^{n} t_j \log y_j = -\log y_s$



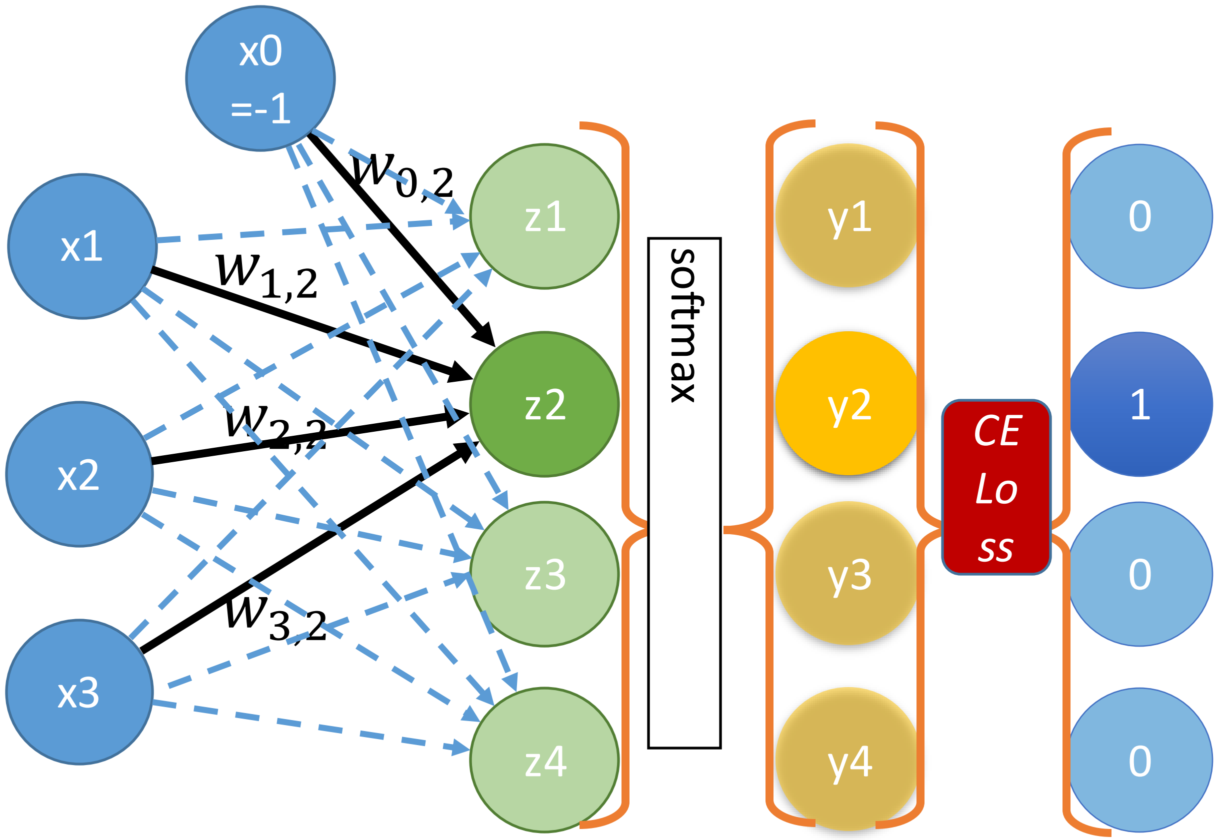
Training Multinomial Logistic Regression 2

- $y_j = \frac{e^{z_j}}{\sum_{k=1}^{n} e^{z_k}}$
- $L_{CE}(\boldsymbol{y}, \boldsymbol{t}) = -\sum_{j=1}^{n} t_j \log y_j = -\log y_s$
- Goal: to find $\frac{\partial}{\partial w_{i,j}} L_{CE}(\boldsymbol{x}, t, \boldsymbol{w})$ for all $w_{i,j}$
- Use the chain-rule for derivatives
- A little more complicated than for LogReg
- The result is simple, though
- $\frac{\partial}{\partial w_{i,j}} L_{CE}(\boldsymbol{x}, t, \boldsymbol{w}) = (y_j - t_j) x_i$



Training Multinomial Logistic Regression 3

- $w_{i,j} = w_{i,j} + \eta(t_j - y_j)x_i$
- if $t_s = 1$:
  - $w_{i,s} = w_{i,s} + \eta(1 - y_s)x_i$
  - $w_{i,j} = w_{i,j} - \eta(y_j)x_i$
    - for $j \neq s$
- Here
  - $z_j = \sum_{i=0}^{m} w_{i,j} x_i$
  - $y_j = \frac{e^{z_j}}{\sum_{k=1}^{n} e^{z_k}}$
- Observe: All weights are updated for each observation



Applying Multinomial Logistic Regression

- We can use this as a probabilistic classifier
  - $\hat{P}\left(C_j \middle| \vec{x}\right) = \frac{e^{\vec{w_j}\cdot\vec{x}}}{\sum_{k=1}^{n} e^{\vec{w_k}\cdot\vec{x}}}$
- To make hard decisions use

$$argmax_{j=1,\ldots,n} \frac{e^{\vec{w_j}\cdot\vec{x}}}{\sum_{k=1}^{n} e^{\vec{w_k}\cdot\vec{x}}}$$