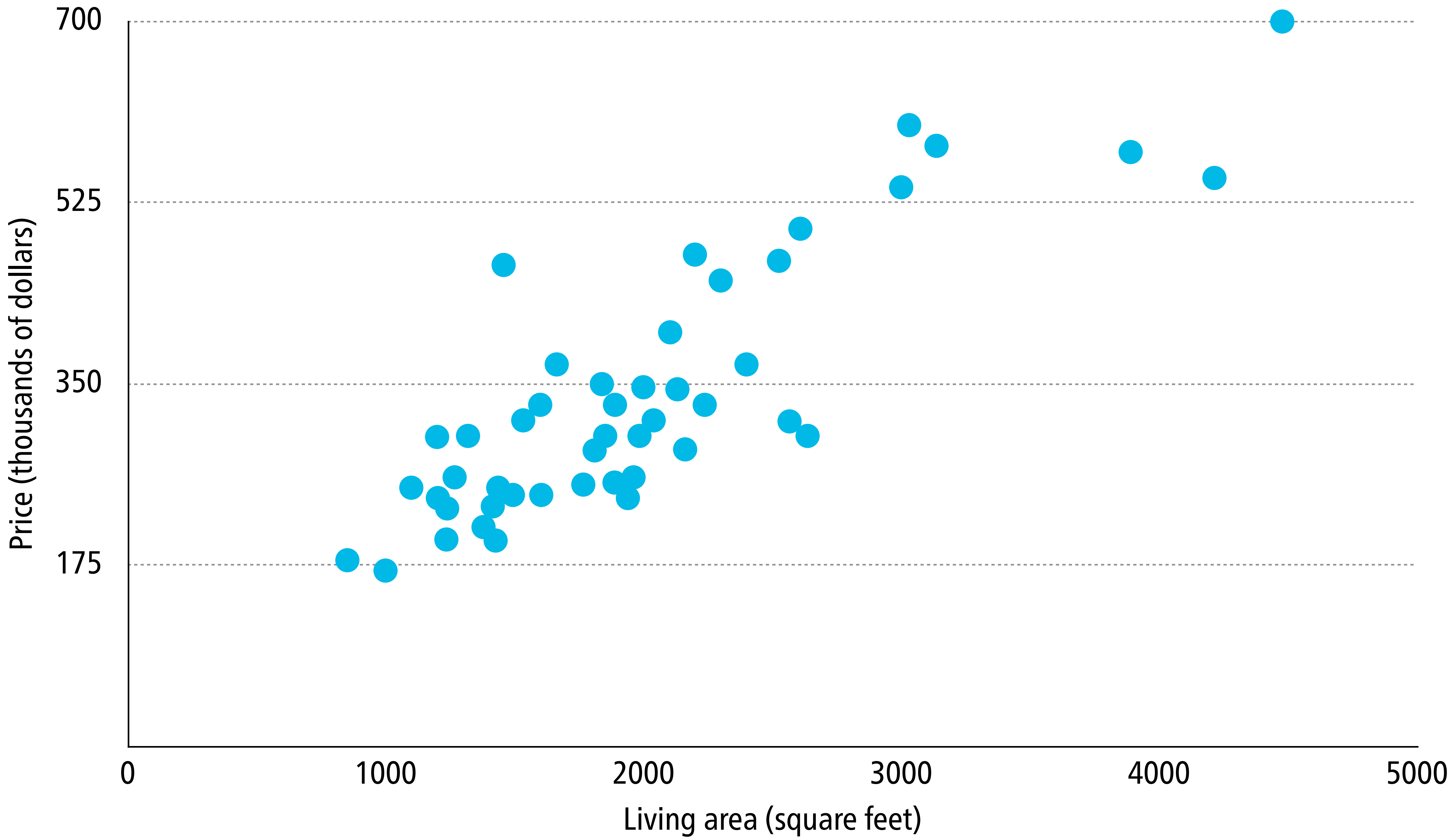
Natural Language Processing

Softmax regression

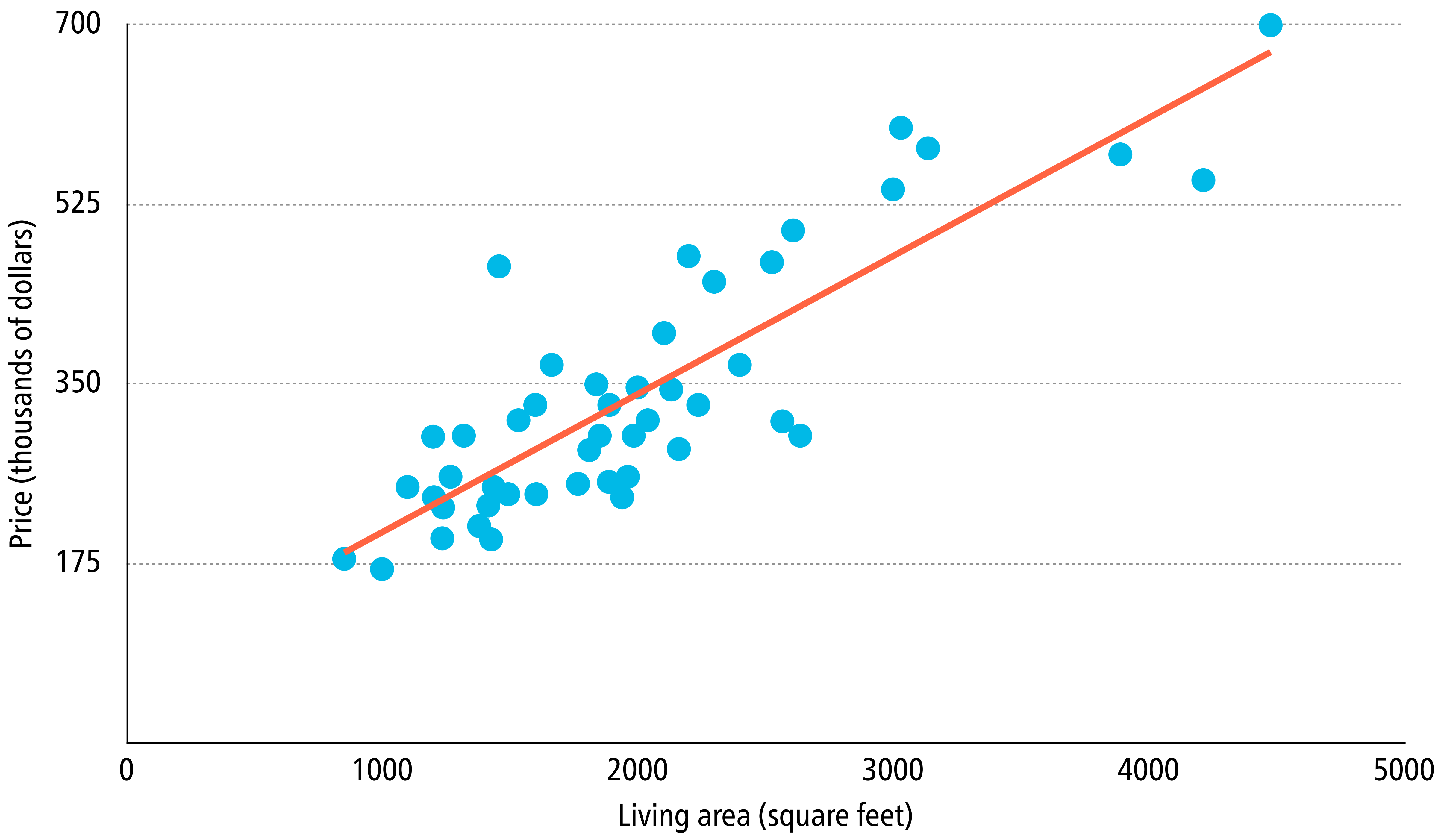
Marco Kuhlmann
Department of Computer and Information Science

LINKÖPING UNIVERSITY

Linear regression with one variable



Linear regression with one variable



The simple linear model

input vector (1-by-F)

Convention: Input vectors are row vectors

output value

$$\hat{y} = \boldsymbol{x}\boldsymbol{w} + b$$

bias

weight vector (F-by-1)

F = number of features (independent variables)

The general linear model

input vector (1-by-m)

output vector (1-by-n)

$$\hat{\boldsymbol{y}} = \boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}$$

bias vector (1-by-n)

weight matrix (m-by-n)

m = number of input features, n = number of output features

Linear classification

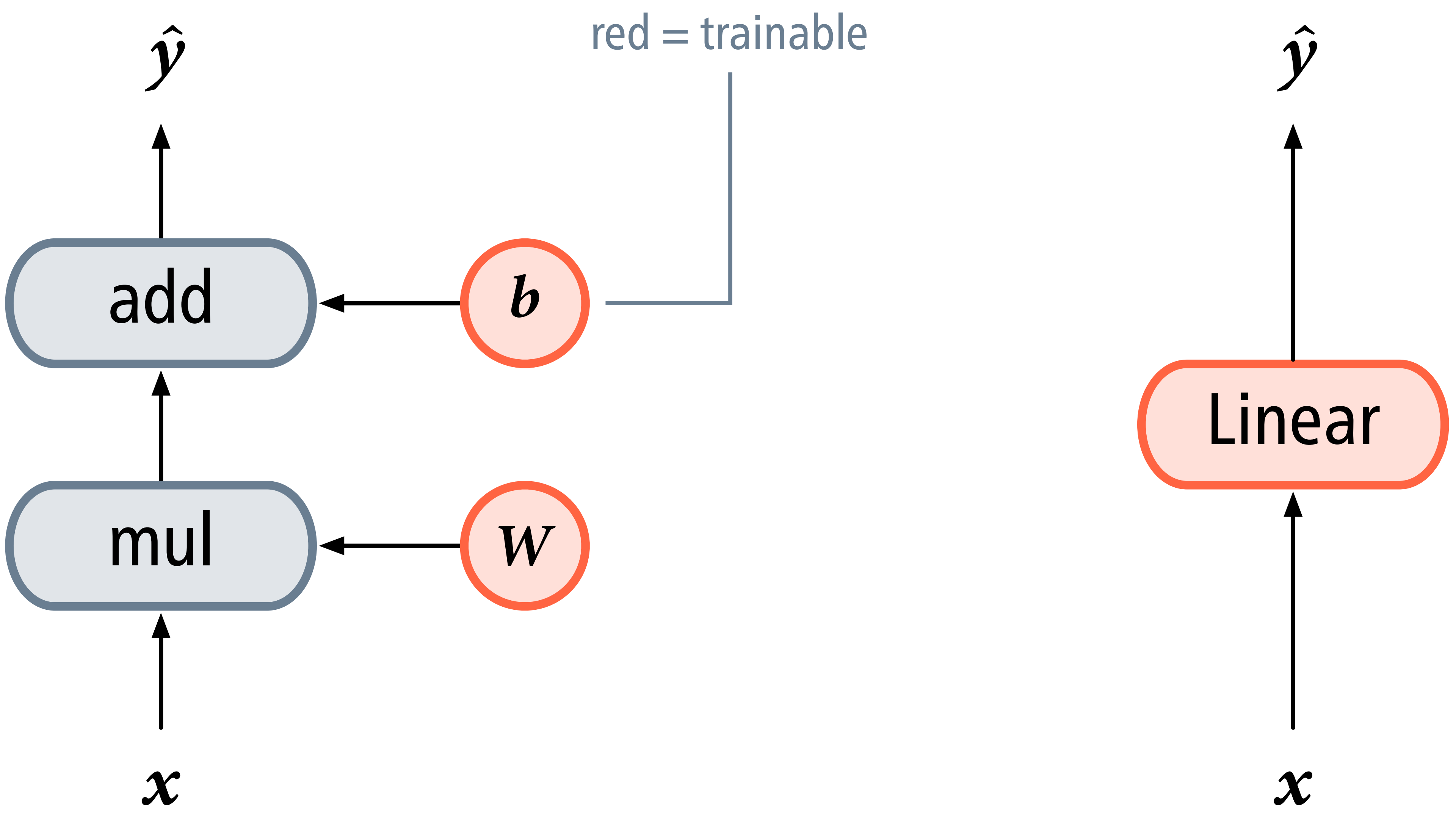
- We think of $\boldsymbol{z} = \boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}$ as an n-dimensional vector of scores that quantify the compatibility of the input $\boldsymbol{x}$ with each class k.

  higher score = higher compatibility

- In **linear classification**, we predict the input $\boldsymbol{x}$ to belong to the highest-scoring class k.

- With linear models, we can only solve a rather restricted class of classification problems (linearly separable).

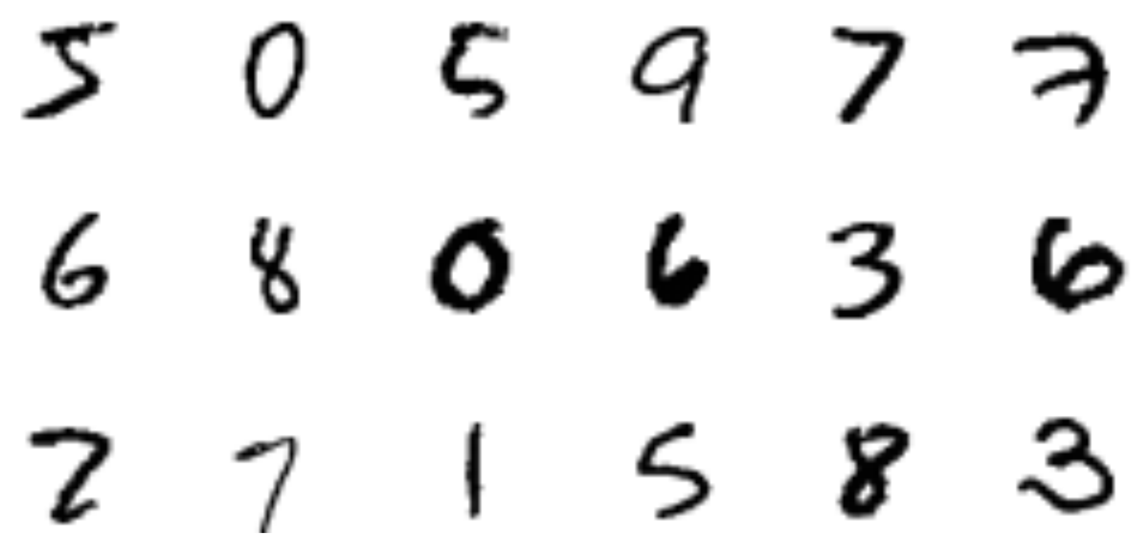
Graphical notation



computation graph

shorthand notation

Handwritten digit recognition



Input: an image of a digit, represented as a 768-dimensional vector of greyscale values.

Output: the digit depicted in the image

Softmax regression

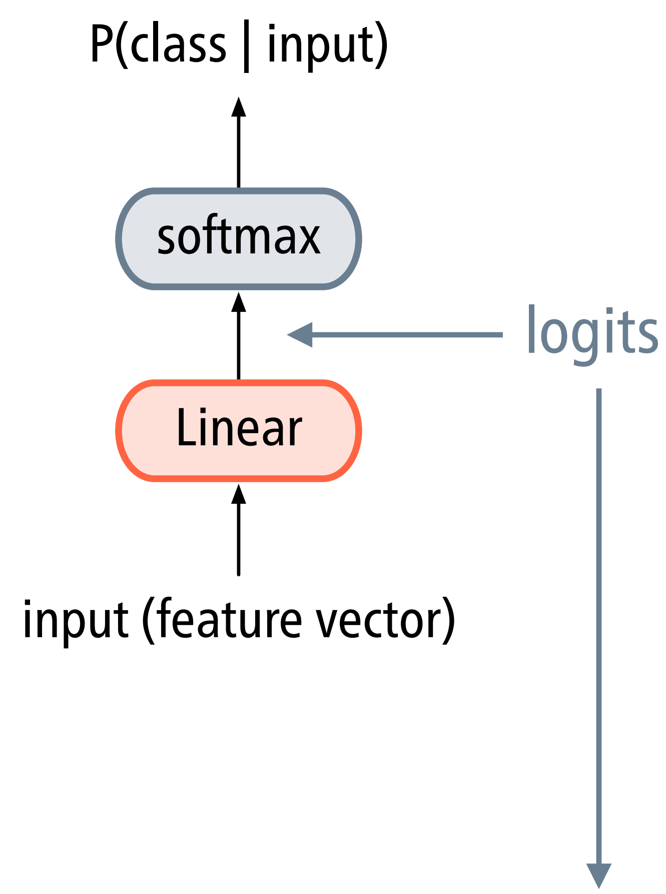
- We convert the scores into a probability distribution $P(k \mid \boldsymbol{x})$ over the classes by sending them through the **softmax function**:

$$\text{softmax}(\boldsymbol{z})[k] = \frac{\exp(\boldsymbol{z}[k])}{\sum_i \exp(\boldsymbol{z}[i])}$$

- Similar to the case of linear classification, we now predict the input $\boldsymbol{x}$ to belong to the highest-probability class k.

- In this context, the unnormalised (raw) scores are called **logits**.

Softmax regression as a neural network



$$P(k \mid \boldsymbol{x}) = \text{softmax}(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{b})$$

Training a softmax regression model

- We present the model with training samples of the form ($\boldsymbol{x}$, y) where $\boldsymbol{x}$ is a feature vector and y is the gold-standard class.
- The output of the model is a vector of conditional probabilities $P(k \mid \boldsymbol{x})$ where k ranges over the possible classes.
- We want to train the model so as to maximise the likelihood of the training data under this probability distribution.

Cross-entropy loss

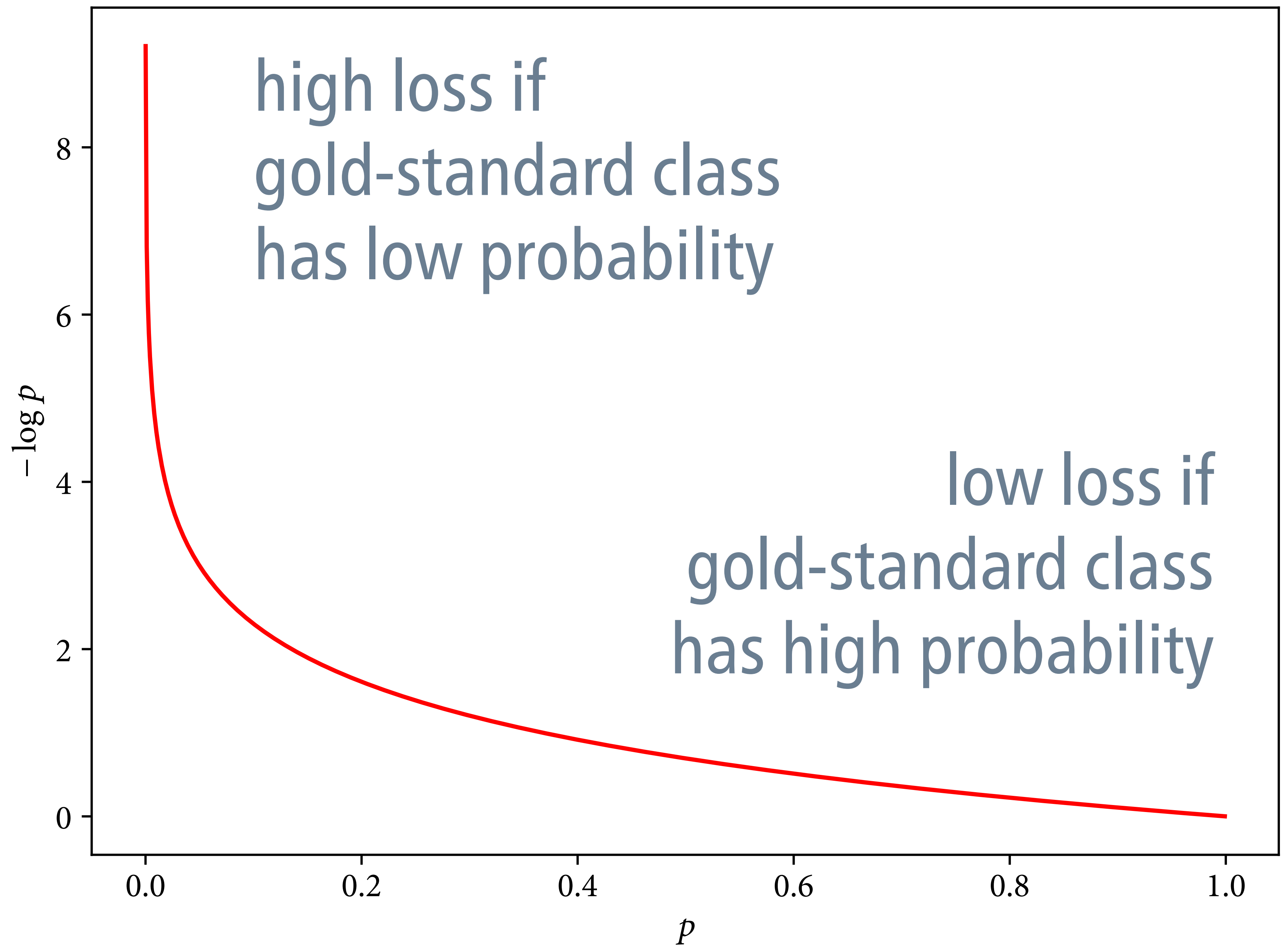
- Instead of maximising the likelihood of the training data, we minimise the model's **cross-entropy loss**.

- The cross-entropy loss for a specific sample $(\boldsymbol{x}, y)$ is the negative log probability of the gold-standard class y, in our case:

$$L(\boldsymbol{\theta}) = -\log \operatorname{softmax}(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{b})[y]$$

all trainable parameters

Cross-entropy loss



Gradient descent

'Follow the gradient into the valley of error.'

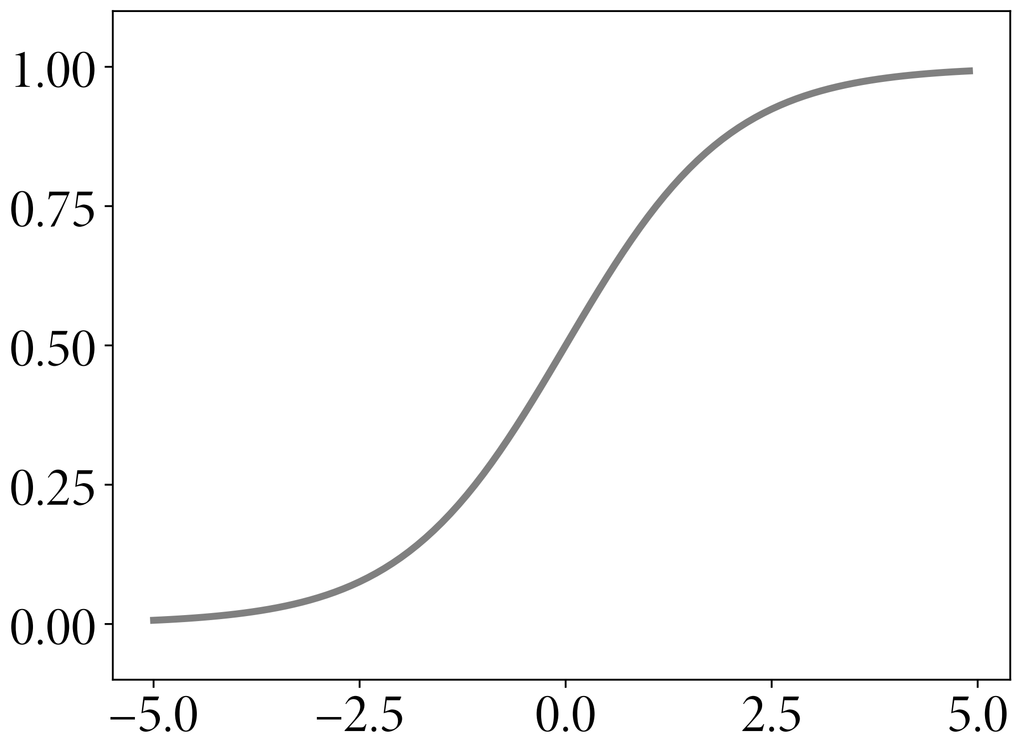
- **Step 0:** Start with random values for the parameters $\boldsymbol{\theta}$.
- **Step 1:** Compute the gradient of the loss function for the current parameter settings, $\nabla L(\boldsymbol{\theta})$.
- **Step 2:** Update the parameters $\boldsymbol{\theta}$ as follows: $\boldsymbol{\theta} \coloneqq \boldsymbol{\theta} - \alpha \, \nabla L(\boldsymbol{\theta})$

  The parameter α is the learning rate.
- Repeat step 1–2 until the loss is sufficiently low.

A note on terminology

- The softmax function can be viewed as a generalisation of the standard logistic function to more than two classes.
- What we call “softmax regression” is sometimes also called **multinomial logistic regression**.

or simply “logistic regression”



$$y = \frac{1}{1 + \exp(-z)}$$