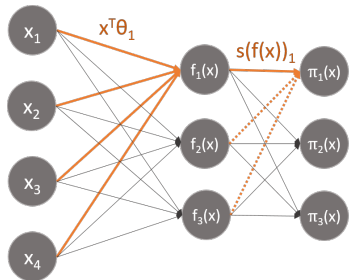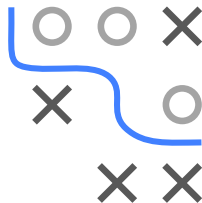# Introduction to Machine Learning

Multiclass Classification
Softmax Regression

Learning goals

- Know softmax regression
- Understand that softmax regression is a generalization of logistic regression
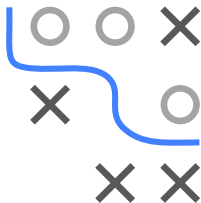
FROM LOGISTIC REGRESSION ...

Remember **logistic regression** ($\mathcal{Y} = \{0, 1\}$): We combined the
hypothesis space of linear functions, transformed by the logistic
function $s(z) = \frac{1}{1+\exp(-z)}$, i.e.

$$\mathcal{H} = \left\{ \pi : \mathcal{X} \to \mathbb{R} \mid \pi(\mathbf{x}) = s(\boldsymbol{\theta}^\top \mathbf{x}) \right\},$$

with the Bernoulli (logarithmic) loss:

$$L(y, \pi(\mathbf{x})) = -y \log\left(\pi(\mathbf{x})\right) - (1 - y) \log\left(1 - \pi(\mathbf{x})\right).$$

Remark: We suppress the intercept term for better readability. The intercept term can
be easily included via $\boldsymbol{\theta}^\top \tilde{\mathbf{x}}$, $\boldsymbol{\theta} \in \mathbb{R}^{p+1}$, $\tilde{\mathbf{x}} = (1, \mathbf{x})$.

... TO SOFTMAX REGRESSION

There is a straightforward generalization to the multiclass case:

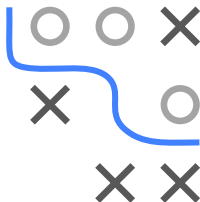- Instead of a single linear discriminant function we have g linear discriminant functions

$$f_k(\mathbf{x}) = \boldsymbol{\theta}_k^\top \mathbf{x}, \quad k = 1, 2, ..., g,$$

  each indicating the confidence in class k.

- The g score functions are transformed into g probability functions by the **softmax** function $s : \mathbb{R}^g \to [0, 1]^g$

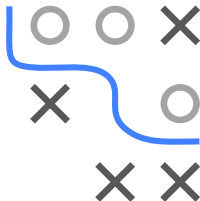$$\pi_k(\mathbf{x}) = s(f(\mathbf{x}))_k = \frac{\exp(\boldsymbol{\theta}_k^\top \mathbf{x})}{\sum_{j=1}^{g} \exp(\boldsymbol{\theta}_j^\top \mathbf{x})},$$

  instead of the **logistic** function for $g = 2$. The probabilities are well-defined: $\sum \pi_k(\mathbf{x}) = 1$ and $\pi_k(\mathbf{x}) \in [0, 1]$ for all k.
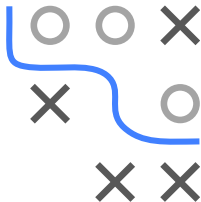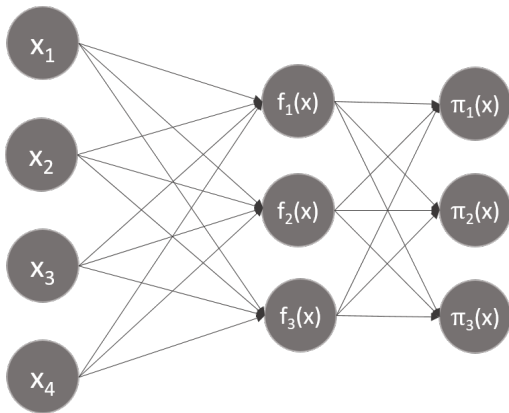
LOGISTIC VS. SOFTMAX REGRESSION

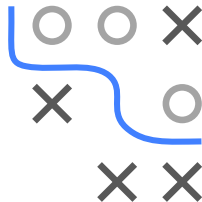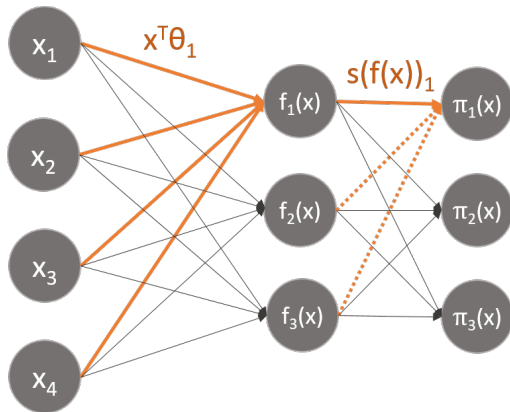|	Logistic Regression	Softmax Regression
$\mathcal{Y}$	$\{0, 1\}$	$\{1, 2, ..., g\}$
Discriminant fun.	$f(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$	$f_k(\mathbf{x}) = \boldsymbol{\theta}_k^\top \mathbf{x}, k = 1, 2, ..., g$
Probabilities	$\pi(\mathbf{x}) = \frac{1}{1+\exp\left(-\boldsymbol{\theta}^\top \mathbf{x}\right)}$	$\pi_k(\mathbf{x}) = \frac{\exp(\boldsymbol{\theta}_k^\top \mathbf{x})}{\sum_{j=1}^{g} \exp(\boldsymbol{\theta}_j^\top \mathbf{x})}$
$L(y, \pi(\mathbf{x}))$	Bernoulli / logarithmic loss $-y \log\left(\pi(\mathbf{x})\right) - (1 - y) \log\left(1 - \pi(\mathbf{x})\right)$	Multiclass logarithmic loss $-\sum_{k=1}^{g} [y = k] \log\left(\pi_k(\mathbf{x})\right)$

LOGISTIC VS. SOFTMAX REGRESSION

We can schematically depict softmax regression as follows:
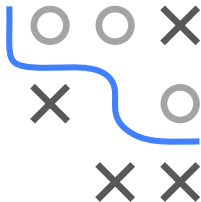
LOGISTIC VS. SOFTMAX REGRESSION

We can schematically depict softmax regression as follows:

LOGISTIC VS. SOFTMAX REGRESSION

Further comments:

- We can now, for instance, calculate gradients and optimize this with standard numerical optimization software.

- Softmax regression has an unusual property in that it has a "redundant" set of parameters. If we subtract a fixed vector from all θ_k, the predictions do not change at all. Hence, our model is "over-parameterized". For any hypothesis we might fit, there are multiple parameter vectors that give rise to exactly the same hypothesis function. This also implies that the minimizer of $\mathcal{R}_{\text{emp}}(\theta)$ above is not unique! Hence, a numerical trick is to set $\theta_g = 0$ and only optimize the other θ_k. This does not restrict our hypothesis space, but the constrained problem is now strictly convex, i.e., there exists exactly one parameter vector for every hypothesis.

- A similar approach is used in many ML models: multiclass LDA, naive Bayes, neural networks and boosting.

SOFTMAX: LINEAR DISCRIMINANT FUNCTIONS

Softmax regression gives us a **linear classifier**.

- The softmax function $s(\boldsymbol{z})_k = \frac{\exp(\boldsymbol{z}_k)}{\sum_{j=1}^{g} \exp(\boldsymbol{z}_j)}$ is

  - a rank-preserving function, i.e. the ranks among the elements of the vector $\boldsymbol{z}$ are the same as among the elements of $s(\boldsymbol{z})$. This is because softmax transforms all scores by taking the $\exp(\cdot)$ (rank-preserving) and divides each element by **the same** normalizing constant.

  Thus, the softmax function has a unique inverse function $s^{-1} : \mathbb{R}^g \to \mathbb{R}^g$ that is also monotonic and rank-preserving.

  Applying s_k^{-1} to $\pi_k(\mathbf{x}) = \frac{\exp(\boldsymbol{\theta}_k^\top \mathbf{x})}{\sum_{j=1}^{g} \exp(\boldsymbol{\theta}_j^\top \mathbf{x})}$ gives us $f_k(\mathbf{x}) = \boldsymbol{\theta}_k^\top \mathbf{x}$.

  Thus, softmax regression is a linear classifier.
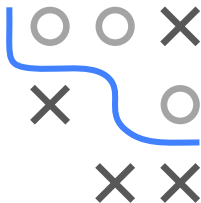
GENERALIZING SOFTMAX REGRESSION

Instead of simple linear discriminant functions we could use **any** model that outputs g scores

$$f_k(\mathbf{x}) \in \mathbb{R}, k = 1, 2, ..., g$$

We can choose a multiclass loss and optimize the score functions $f_k, k \in \{1, ..., g\}$ by multivariate minimization. The scores can be transformed to probabilities by the **softmax** function.