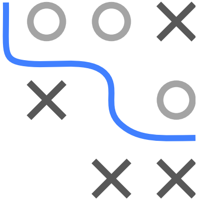
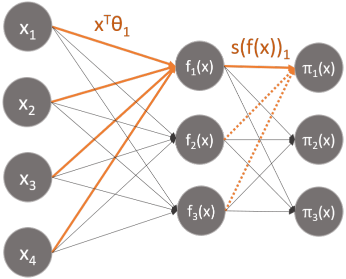


Introduction to Machine Learning

Multiclass Classification
Softmax Regression

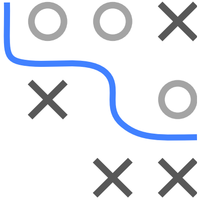


Learning goals

- 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.



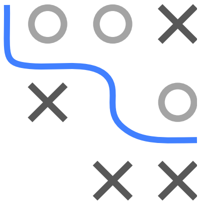
$$\mathcal{H} = \left\{\pi : \mathcal{X} \to \mathbb{R} \mid \pi(\mathbf{x}) = s(\boldsymbol{\theta}^\top \mathbf{x})\right\} ,$$

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

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

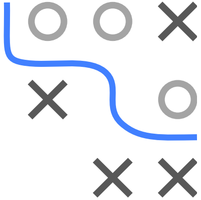
each indicating the confidence in class k.

- 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.

... TO SOFTMAX REGRESSION / 2



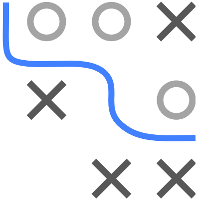
- The softmax function is a generalization of the logistic function. For $g = 2$, the logistic function and the softmax function are equivalent.
- Instead of the **Bernoulli** loss, we use the multiclass **logarithmic loss**

$$L(y, \pi(\mathbf{x})) = -\sum_{k=1}^{g} \mathbb{1}_{\{y=k\}} \log\left(\pi_k(\mathbf{x})\right).$$

- Note that the softmax function is a "smooth" approximation of the arg max operation, so $s((1, 1000, 2)^T) \approx (0, 1, 0)^T$ (picks out 2nd element!).
- Furthermore, it is invariant to constant offsets in the input:

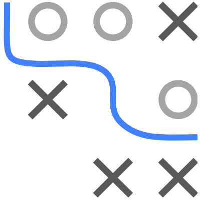
$$s(f(\mathbf{x})+\mathbf{c}) = \frac{\exp(\boldsymbol{\theta}_k^\top \mathbf{x} + c)}{\sum_{j=1}^{g} \exp(\boldsymbol{\theta}_j^\top \mathbf{x} + c)} = \frac{\exp(\boldsymbol{\theta}_k^\top \mathbf{x}) \cdot \exp(c)}{\sum_{j=1}^{g} \exp(\boldsymbol{\theta}_j^\top \mathbf{x}) \cdot \exp(c)} = s(f(\mathbf{x}))$$

LOGISTIC VS. SOFTMAX REGRESSION

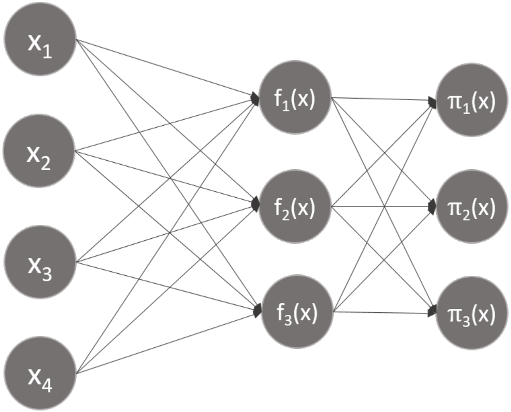


	Logistic Regression	Softmax Regression
$\mathcal{Y}$	$\{0, 1\}$	$\{1, 2, ..., g\}$
Discriminant fun.	$f(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$	$f_k(\mathbf{x}) = \boldsymbol{\theta}_k^\top \mathbf{x}, k = 1, 2, ..., g$
Probabilities	$\pi(\mathbf{x}) = \frac{1}{1+\exp(-\boldsymbol{\theta}^\top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

LOGISTIC VS. SOFTMAX REGRESSION

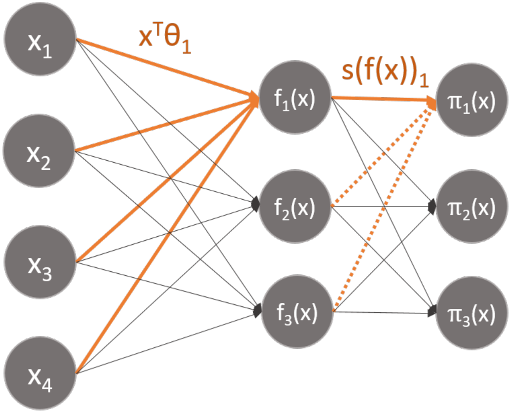


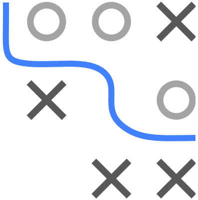
We can schematically depict softmax regression as follows:



LOGISTIC VS. SOFTMAX REGRESSION

We can schematically depict softmax regression as follows:



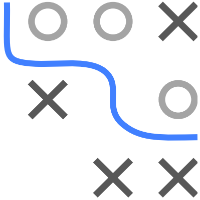


LOGISTIC VS. SOFTMAX REGRESSION

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 $\boldsymbol{\theta}_k$, 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 $\mathcal{R}_{\text{emp}}(\boldsymbol{\theta})$ above is not unique! Hence, a numerical trick is to set $\boldsymbol{\theta}_g = 0$ and only optimize the other $\boldsymbol{\theta}_k$. This does not restrict our hypothesis space, but the constrained problem is now 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**.

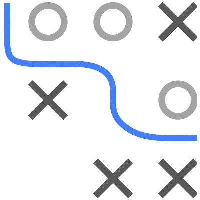
- The softmax function $s(\mathbf{z})_k = \frac{\exp(\mathbf{z}_k)}{\sum_{j=1}^{g} \exp(\mathbf{z}_j)}$ is
  - a rank-preserving function, i.e. the ranks among the elements of the vector $\mathbf{z}$ are the same as among the elements of $s(\mathbf{z})$. This is because softmax transforms all scores 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.

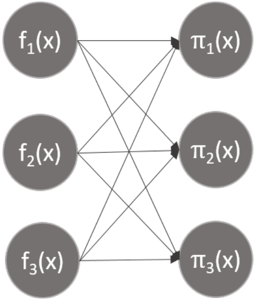
GENERALIZING SOFTMAX REGRESSION



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

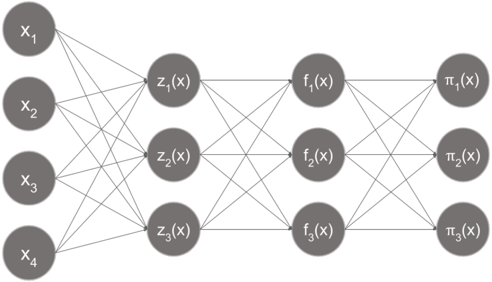
$$f_k(\mathbf{x}) \in \mathbb{R}, k = 1, 2, ..., g$$

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.



GENERALIZING SOFTMAX REGRESSION / 2

For example for a **neural network** (note that softmax regression is also a neural network with no hidden layers):



Remark: For more details about neural networks please refer to the lecture **Deep Learning**.