COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

From Logistic Regression to Neural Networks

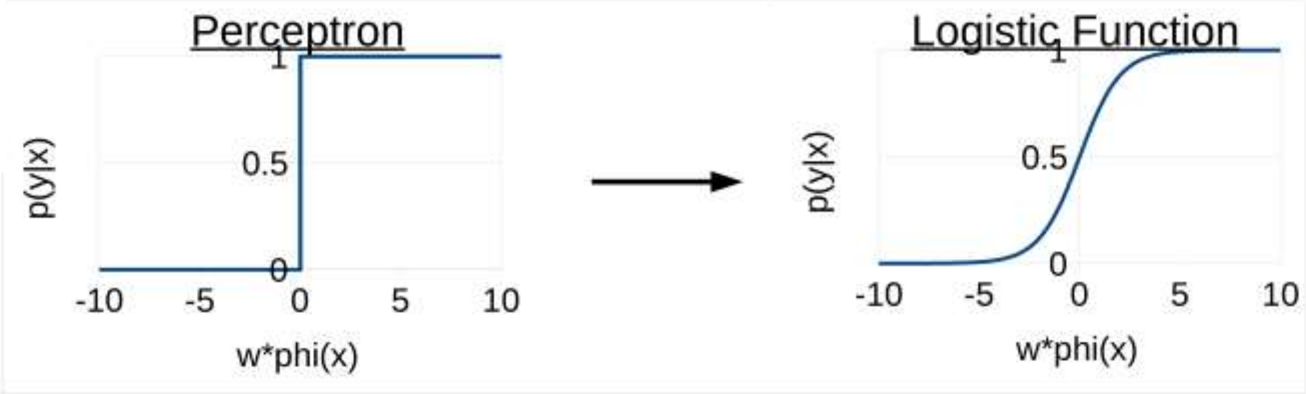
CMSC 470

Marine Carpuat

Slides credit: Jacob Eisenstein

The logistic function

$$P(y=1|x)=\frac{e^{w\cdot\varphi(x)}}{1+e^{w\cdot\varphi(x)}}$$



- x: the input
- **φ(x)**: vector of feature functions $\{\varphi_1(x), \varphi_2(x), \ldots, \varphi_I(x)\}$
- **w**: the weight vector $\{w_1, w_2, \ldots, w_I\}$
- y: the prediction, +1 if "yes", -1 if "no"

- "Softer" function than in perceptron
- Can account for uncertainty
- Differentiable

Logistic regression: how to train?

- Train based on **conditional likelihood**
- Find parameters w that maximize conditional likelihood of all answers y_i given examples x_i

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \prod_i P(y_i | \mathbf{x}_i ; \mathbf{w})$$

Stochastic gradient ascent (or descent)

- Online training algorithm

```
create map w
for I iterations
    for each labeled pair x, y in the data
        w += α * dP(y|x)/dw
```

- Update weights for every training example
- Move in direction given by gradient
- Size of update step scaled by learning rate

Gradient of the logistic function

$$\frac{d}{dw}P(y=1|x) = \frac{d}{dw}\frac{e^{w\cdot\varphi(x)}}{1+e^{w\cdot\varphi(x)}}$$
$$= \varphi(x)\frac{e^{w\cdot\varphi(x)}}{(1+e^{w\cdot\varphi(x)})^2}$$

$$\frac{d}{dw}P(y=-1|x) = \frac{d}{dw}(1-\frac{e^{w\cdot\varphi(x)}}{1+e^{w\cdot\varphi(x)}})$$
$$= -\varphi(x)\frac{e^{w\cdot\varphi(x)}}{(1+e^{w\cdot\varphi(x)})^2}$$

How to set the learning rate?

- Various strategies
  - decay over time

$$\alpha = \frac{1}{C + t}$$

Parameter

Number of samples

  - Use held-out test set, increase learning rate when likelihood increases

Logistic Regression for **Multiclass** Classification

Logistic Regression: Prediction

- Find y that maximizes

$$p(y \mid \boldsymbol{x}; \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}, y))}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}, y'))}.$$

Logistic Regression: Training

- Find parameters that
  - maximize the conditional likelihood
  - of a training dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$

$$\log \mathrm{p}(\boldsymbol{y}^{(1:N)} \mid \boldsymbol{x}^{(1:N)}; \boldsymbol{\theta}) = \sum_{i=1}^{N} \log \mathrm{p}(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

$$= \sum_{i=1}^{N} \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y^{(i)}) - \log \sum_{y' \in \mathcal{Y}} \exp\left(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y')\right).$$

Logistic Regression: Gradient

$$\ell_{\text{LogReg}} = -\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y^{(i)}) + \log \sum_{y' \in \mathcal{Y}} \exp\left(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y')\right) \quad [2.60]$$

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}} = -\boldsymbol{f}(\boldsymbol{x}^{(i)}, y^{(i)}) + \frac{1}{\sum_{y'' \in \mathcal{Y}} \exp\left(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y'')\right)} \times \sum_{y' \in \mathcal{Y}} \exp\left(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y')\right) \times \boldsymbol{f}(\boldsymbol{x}^{(i)}, y') \quad [2.61]$$

$$= -\boldsymbol{f}(\boldsymbol{x}^{(i)}, y^{(i)}) + \sum_{y' \in \mathcal{Y}} \frac{\exp\left(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y')\right)}{\sum_{y'' \in \mathcal{Y}} \exp\left(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y'')\right)} \times \boldsymbol{f}(\boldsymbol{x}^{(i)}, y') \quad [2.62]$$

$$= -\boldsymbol{f}(\boldsymbol{x}^{(i)}, y^{(i)}) + \sum_{y' \in \mathcal{Y}} \mathrm{p}(y' \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \times \boldsymbol{f}(\boldsymbol{x}^{(i)}, y') \quad [2.63]$$

$$= -\boldsymbol{f}(\boldsymbol{x}^{(i)}, y^{(i)}) + E_{Y|X}[\boldsymbol{f}(\boldsymbol{x}^{(i)}, y)]. \quad [2.64]$$

Observed feature counts

Expected feature counts under the current model

Learning as optimization: Loss Functions

- Loss function scores how bad a model predictions are on a training set (or on a single example)
- Each of the linear models we've seen so far optimize a different loss function

- Logistic regression minimizes the logistic loss

$$\ell_{\text{LOGREG}}(\boldsymbol{\theta}; \boldsymbol{x}^{(i)}, y^{(i)}) = -\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y^{(i)}) + \log \sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y'))$$

Learning as optimization: Loss Functions

- Naïve Bayes loss

$$\ell_{\text{NB}}(\boldsymbol{\theta}; \boldsymbol{x}^{(i)}, y^{(i)}) = -\log \text{p}(\boldsymbol{x}^{(i)}, y^{(i)}; \boldsymbol{\theta})$$

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}} \sum_{i=1}^{N} \ell_{\text{NB}}(\boldsymbol{\theta}; \boldsymbol{x}^{(i)}, y^{(i)})$$

$$= \underset{\boldsymbol{\theta}}{\text{argmax}} \sum_{i=1}^{N} \log \text{p}(\boldsymbol{x}^{(i)}, y^{(i)}; \boldsymbol{\theta}).$$

- Zero-one loss

$$\ell_{\text{perceptron}}(\boldsymbol{\theta}; \boldsymbol{x}_i, y_i) = \begin{cases} 0, & y_i = \arg\max_y \boldsymbol{\theta}^\top \boldsymbol{f}(x_i, y) \\ 1, & \text{otherwise} \end{cases}$$

Learning as optimization: Loss Functions

- Naïve bayes loss
  - can suffer infinite loss on a single example
  - But the optimization problem has a closed form solution

- Zero-one loss
  - most closely related to error rate
  - but non-convex and not continuous

- Logistic loss
  - Never zero: the objective can always be improved by assigning higher confidence to the correct label
  - Convex and continuous

Regularization

Some models are better then others…

- Consider these 2 examples

-1	he saw a bird in the park
+1	he saw a robbery in the park

- Which of the 2 models below is better?

Classifier 1	Classifier 2
he +3	bird -1
saw -5	robbery +1
a +0.5	
bird -1	
robbery +1	
in +5	
the -3	
park -2	

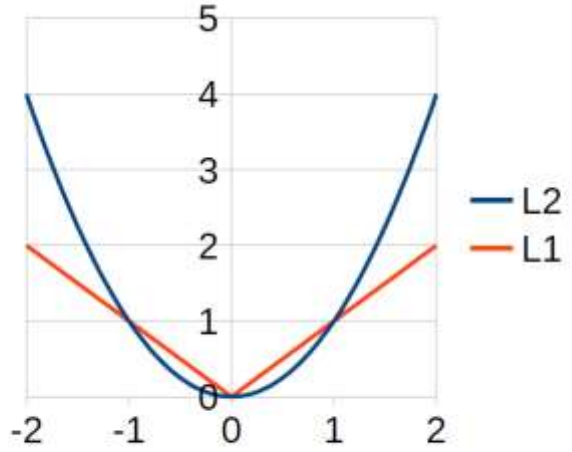
Classifier 2 will probably generalize better!
It does not include irrelevant information
=> Smaller model is better

Regularization

- Encodes a preference towards simpler models to avoid overfitting
- By augmenting the loss with a penalty on adding extra weights

- L2 regularization: $\|w\|_2$
  - big penalty on large weights
  - small penalty on small weights

- L1 regularization: $\|w\|_1$
  - Uniform increase when large or small
  - Will cause many weights to become zero



What you should know about linear models

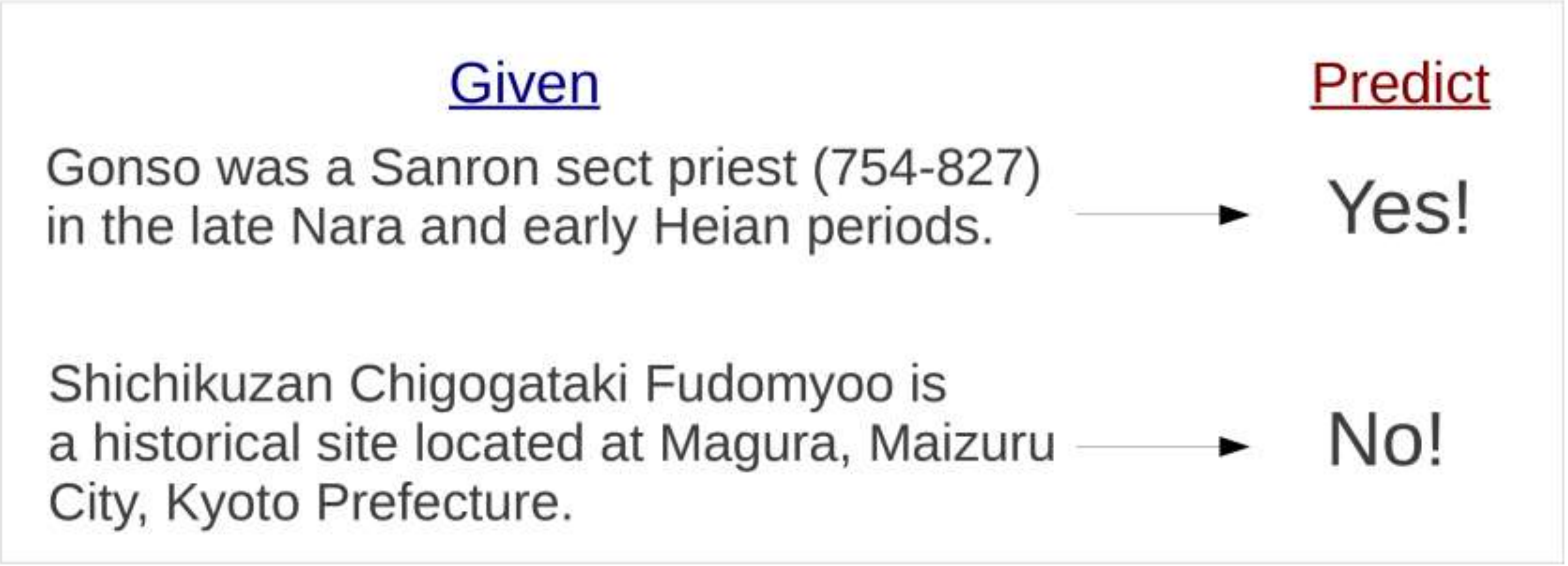
- Standard supervised learning set-up for text classification
  - Difference between train vs. test data
  - How to evaluate
- 3 examples of linear classifiers
  - Naïve Bayes, Perceptron, Logistic Regression
    - How to make predictions, how to train, strengths and weaknesses
  - Learning as optimization: loss functions and their properties
  - Difference between generative vs. discriminative classifiers
- General machine learning concepts
  - Smoothing, regularization, overfitting, underfitting

Neural Networks

"Machines" that learn combinations of features

Let’s go back to our Binary Classification Problem

Given an introductory sentence in Wikipedia
predict whether the article is about a person



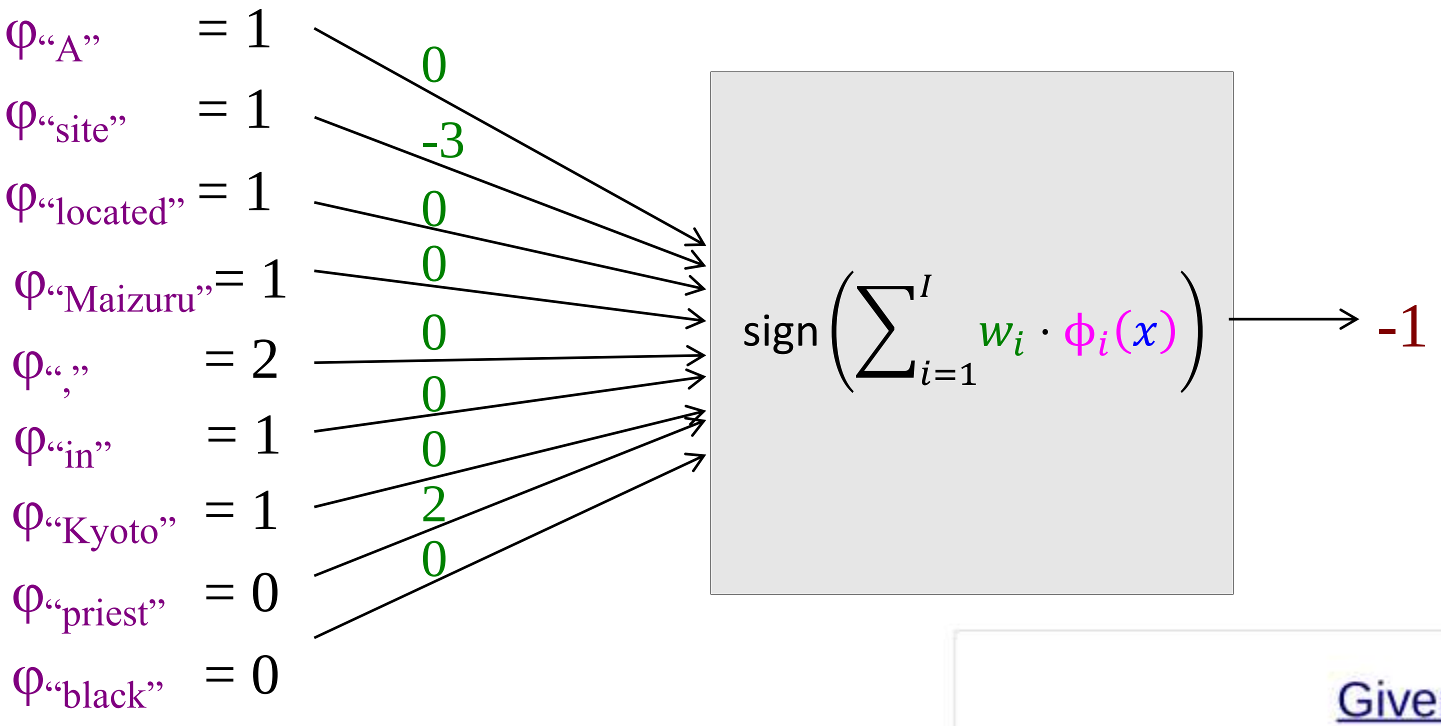
Example & figures by Graham Neubig

Binary Classification with the Perceptron

$$y = \text{sign}(\mathbf{w} \cdot \boldsymbol{\varphi}(x))$$
$$= \text{sign}\left(\sum_{i=1}^{I} w_i \cdot \varphi_i(x)\right)$$

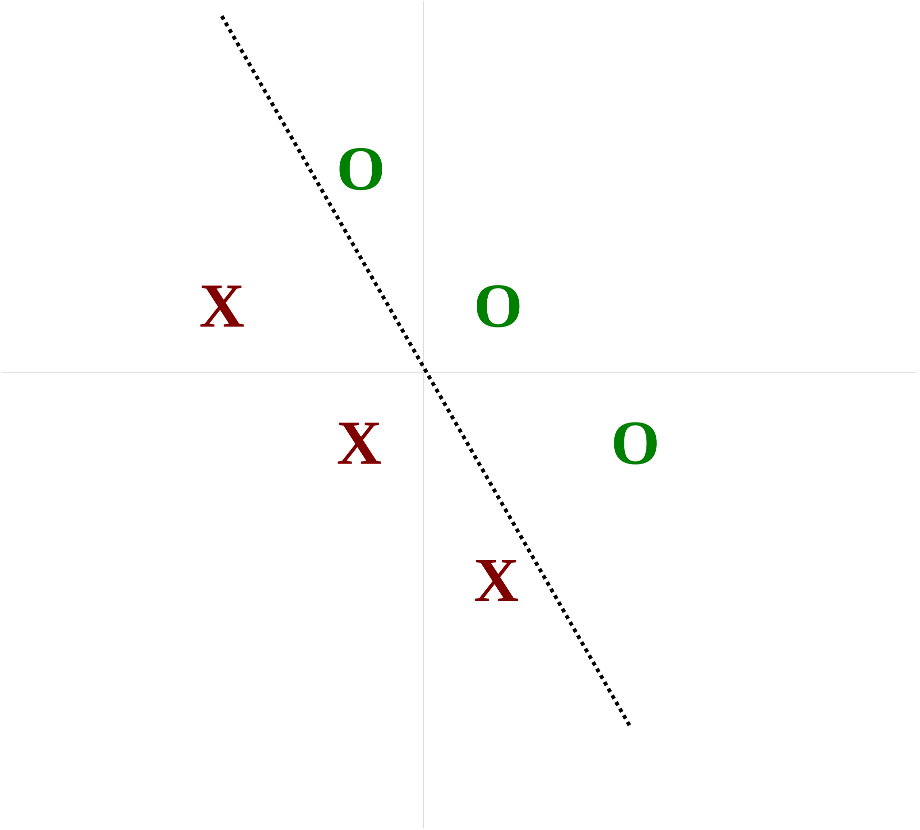
- x: the input
- **φ(x)**: vector of feature functions $\{\varphi_1(x), \varphi_2(x), \ldots, \varphi_I(x)\}$
- **w**: the weight vector $\{w_1, w_2, \ldots, w_I\}$
- y: the prediction, +1 if "yes", -1 if "no"
  - (sign(v) is +1 if v >= 0, -1 otherwise)

Making Predictions with the Perceptron



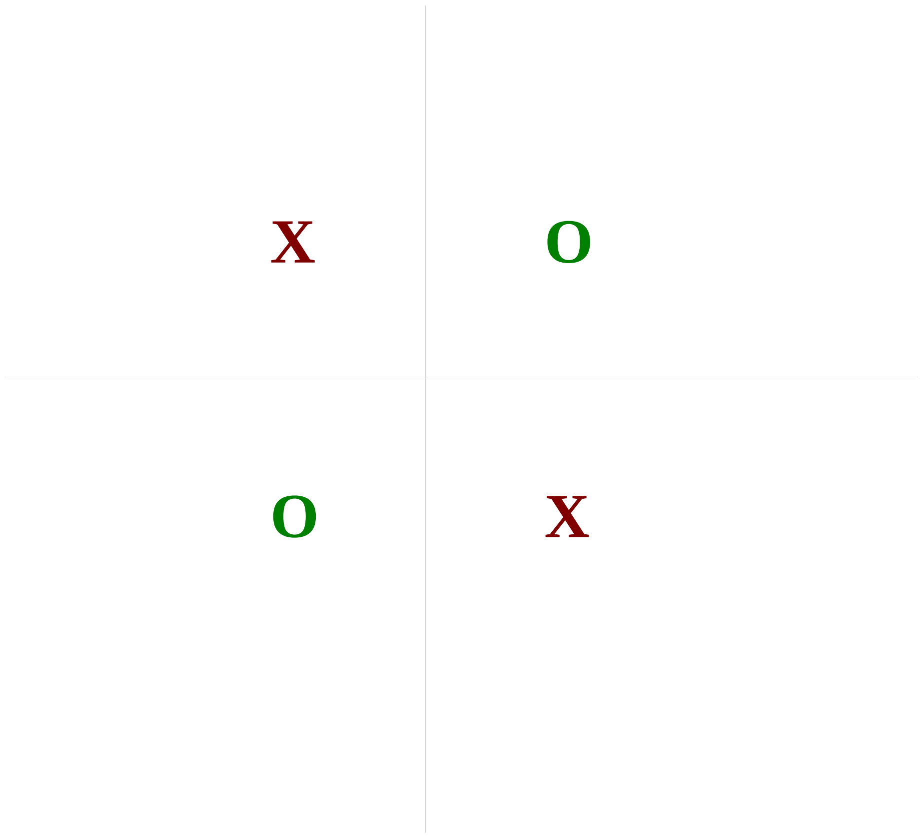
Given	Predict
Gonso was a Sanron sect priest (754-827) in the late Nara and early Heian periods. →	
Shichikuzan Chigogataki Fudomyoo is a historical site located at Magura, Maizuru City, Kyoto Prefecture. →	

The Perceptron: Geometric interpretation

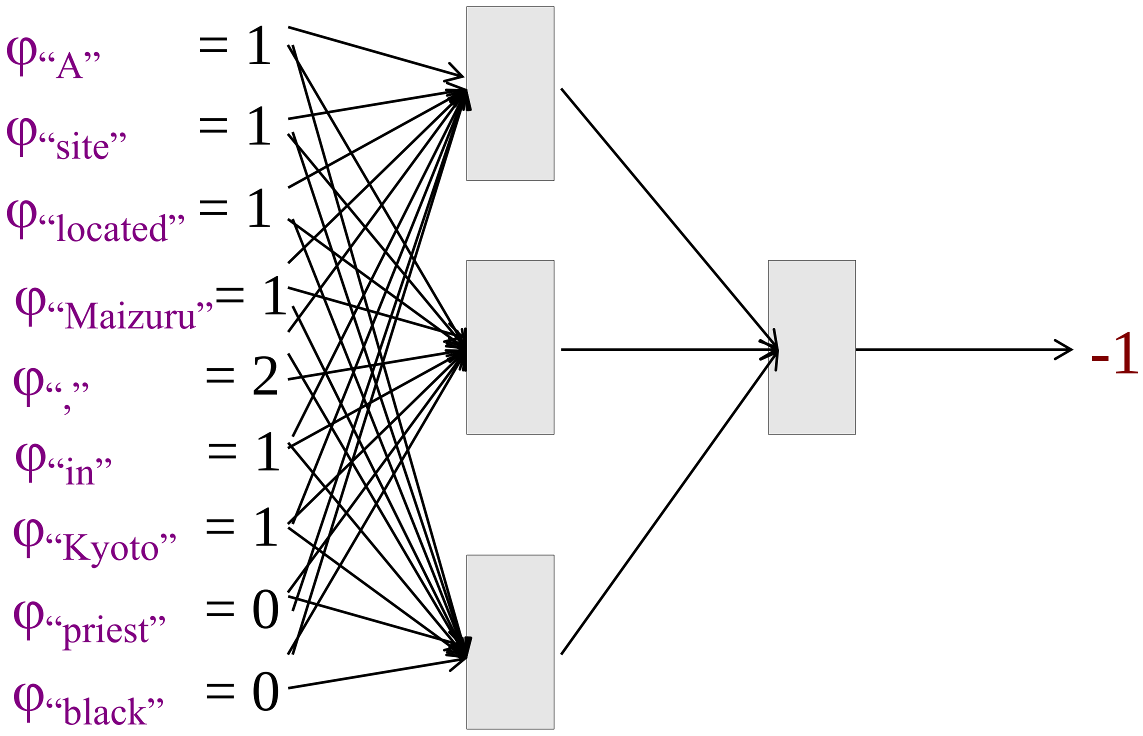


Limitation of perceptron

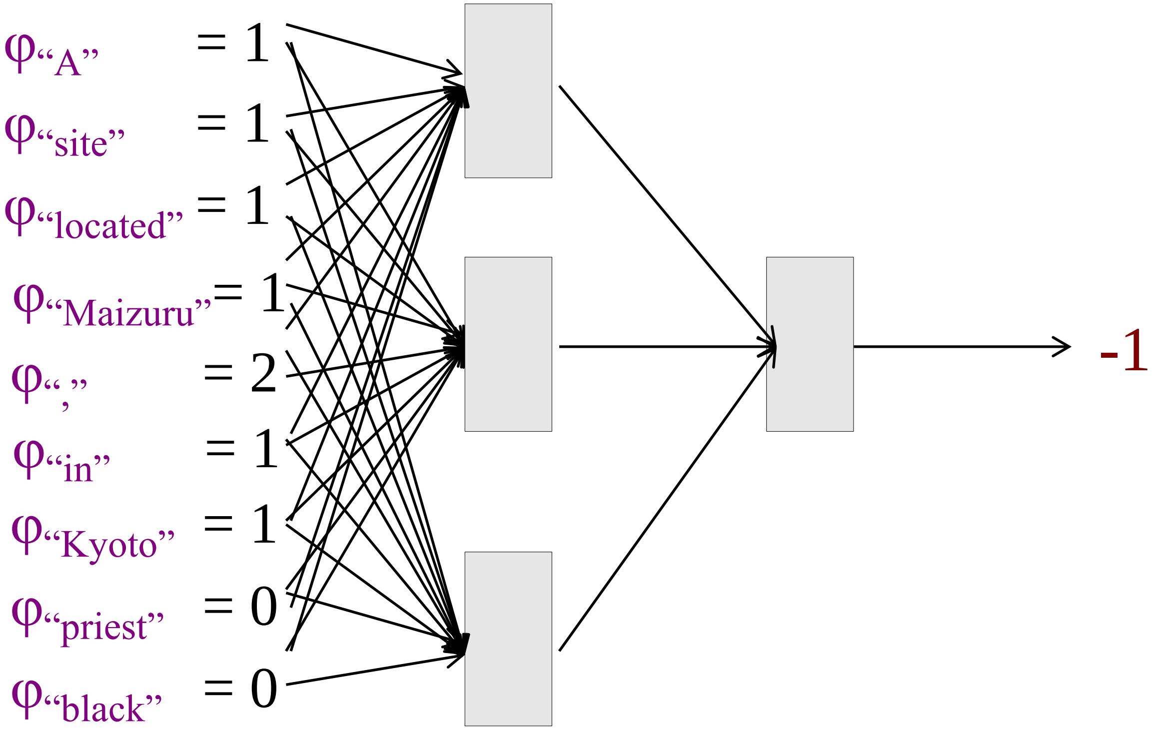
- can only find **linear separations** between positive and negative examples



Binary Classification with a Multi-layer Perceptron



Multi-layer Perceptrons are a kind of “Neural Network” (NN)

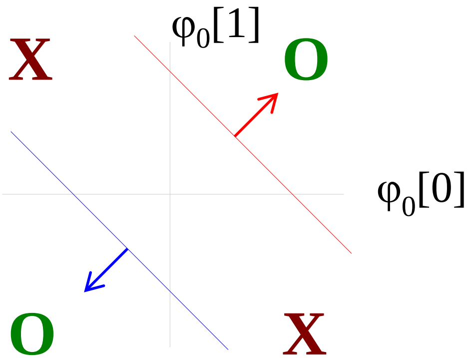


- Input (aka features)
- Output
- Nodes
- Layers
- Hidden layers
- Activation function (non-linear)

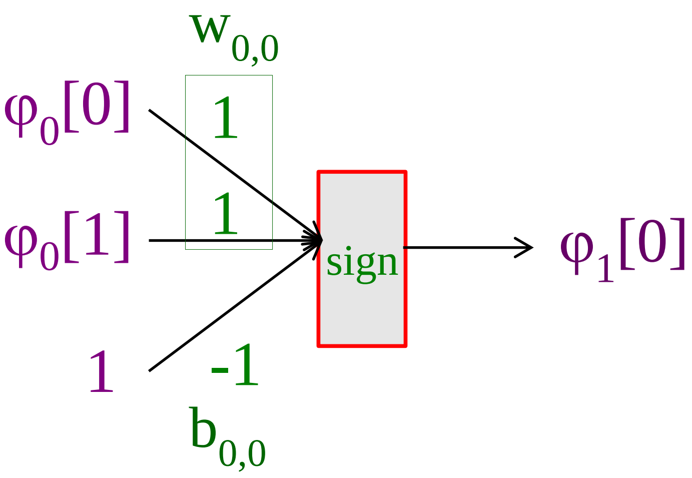
Example: binary classification with a NN

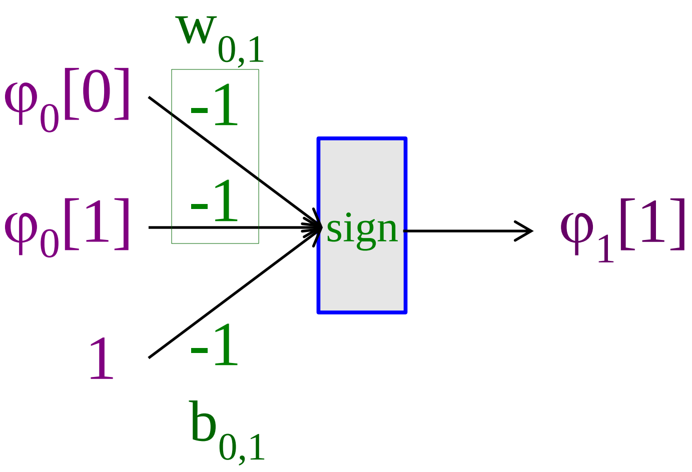
- Create two classifiers

$\varphi_0(x_1) = \{-1, 1\}$ $\varphi_0(x_2) = \{1, 1\}$



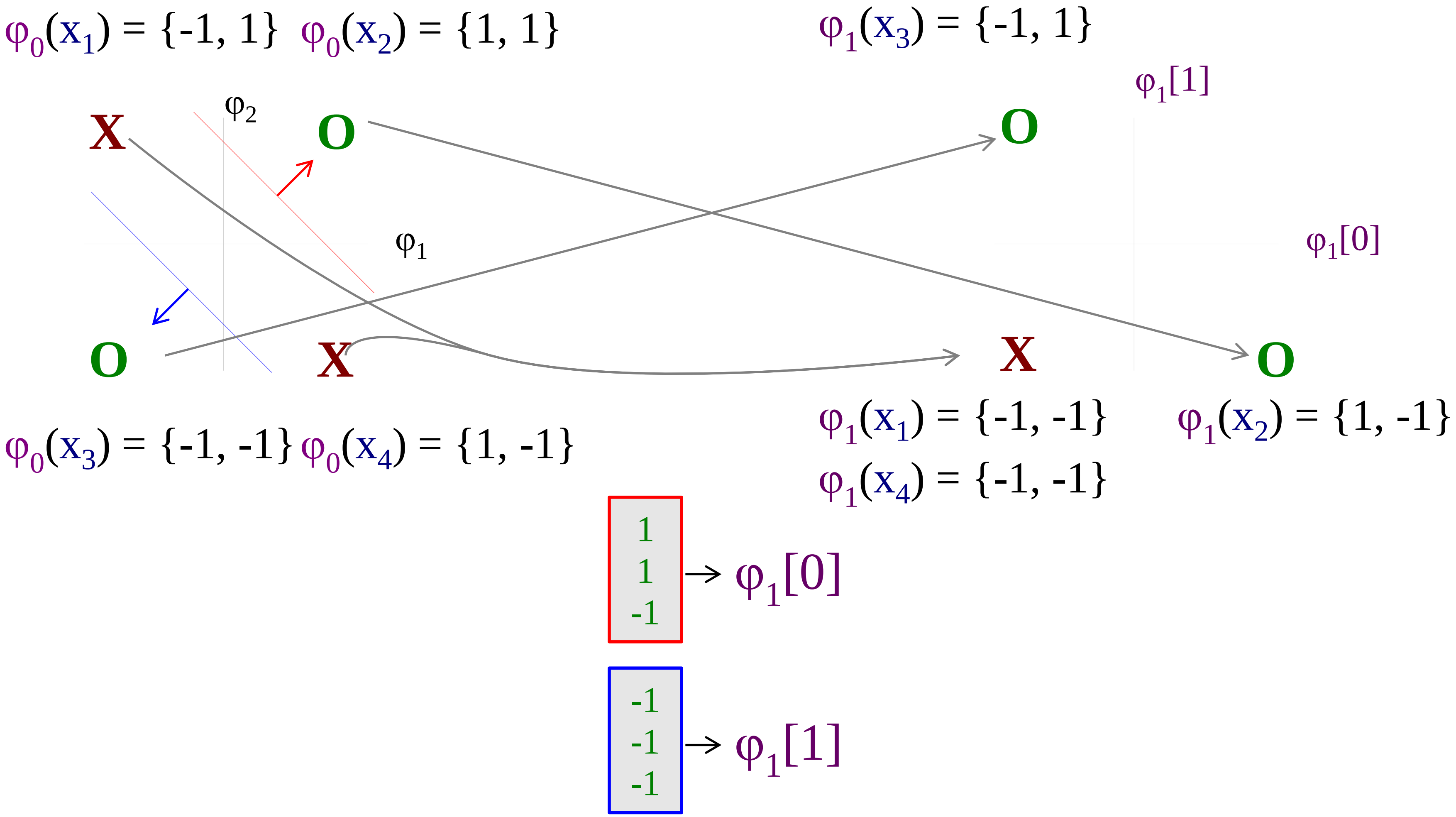
$\varphi_0(x_3) = \{-1, -1\}$ $\varphi_0(x_4) = \{1, -1\}$



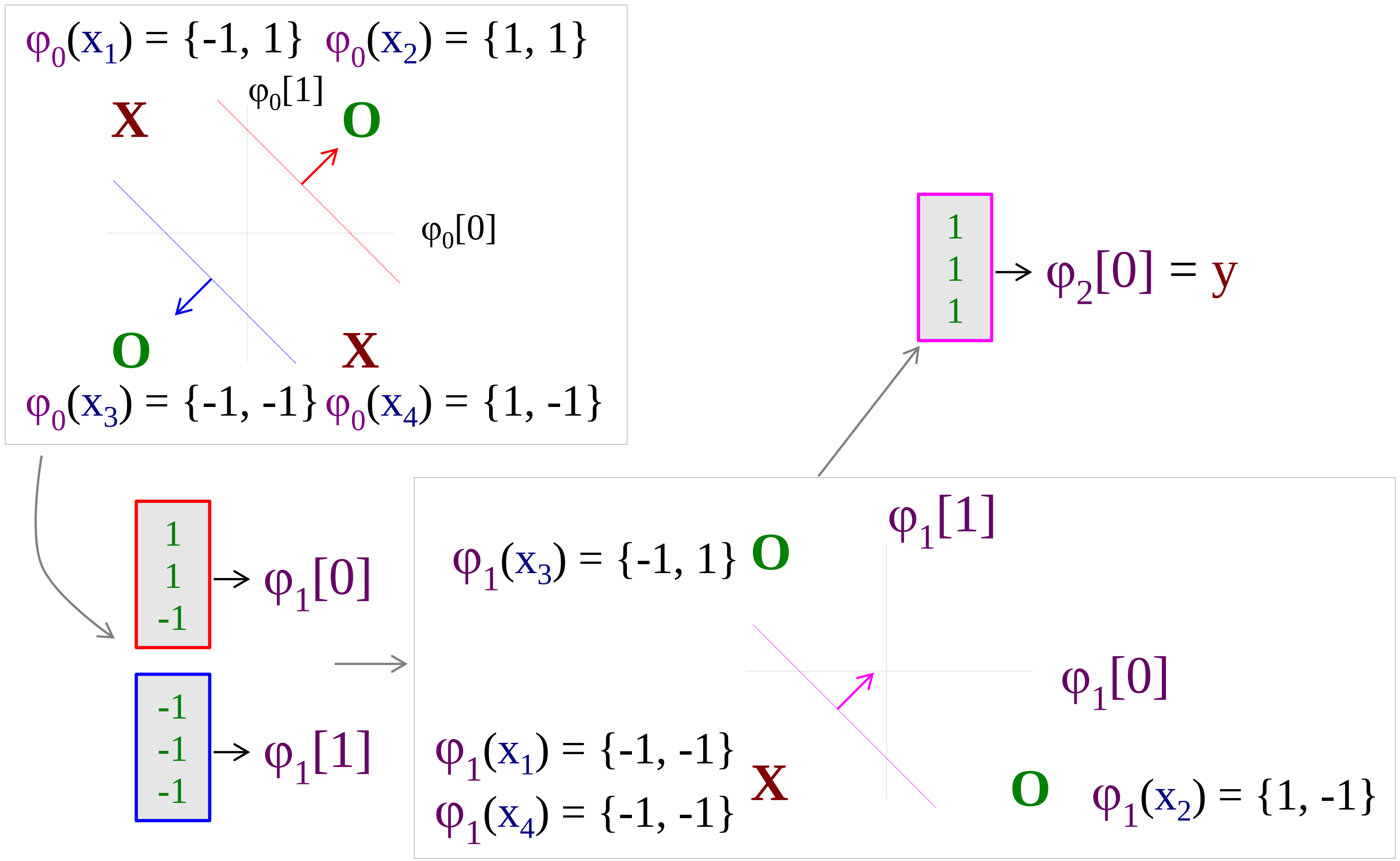


Example: binary classification with a NN

- These classifiers map to a new space



Example: binary classification with a NN



Example: the Final Net