Multi-class linear prediction

COMS 4771 Fall 2025

Multi-class prediction

Multi-class prediction usually means classification problems with $|\mathcal{Y}| \geq 3$ (. . . but technically also includes $|\mathcal{Y}| = 2$)

- Assume we use zero-one loss, so corresponding risk is error rate
- Best prediction of Y given $X = x$ is

$$f^\star(x) = \arg\max_{k \in \mathcal{Y}} \Pr(Y = k \mid X = x)$$

	$\lvert\mathcal{Y}\rvert = 2$	$\lvert\mathcal{Y}\rvert \geq 3$
nearest neighbors	✓	✓
decision trees	✓	✓
generative models	✓	✓
logistic regression	✓	?
Perceptron	✓	?

Linear classifiers are inherently binary output functions;
need some work to extend to $|\mathcal{Y}| \geq 3$

Multi-class logistic regression

Generalize logistic regression model to K classes, $[K] = \{1, 2, \ldots, K\}$

$$\Pr(Y = k \mid X = x) = \frac{\exp(x^\mathsf{T} w^{(k)})}{\sum_{c \in [K]} \exp(x^\mathsf{T} w^{(c)})}$$

- K weight vectors $w^{(1)}, \ldots, w^{(K)} \in \mathbb{R}^d$ are parameters of the model

  $$w^{(k)} = (w_1^{(k)}, \ldots, w_d^{(k)})$$

  Total of Kd parameters

In this model with parameters $w^{(1)}, \ldots, w^{(K)}$, classifier with smallest error rate is

$$f^\star(x) = \arg\max_{k \in [K]} \Pr(Y = k \mid X = x) = \arg\max_{k \in [K]} x^\mathsf{T} w^{(k)}$$

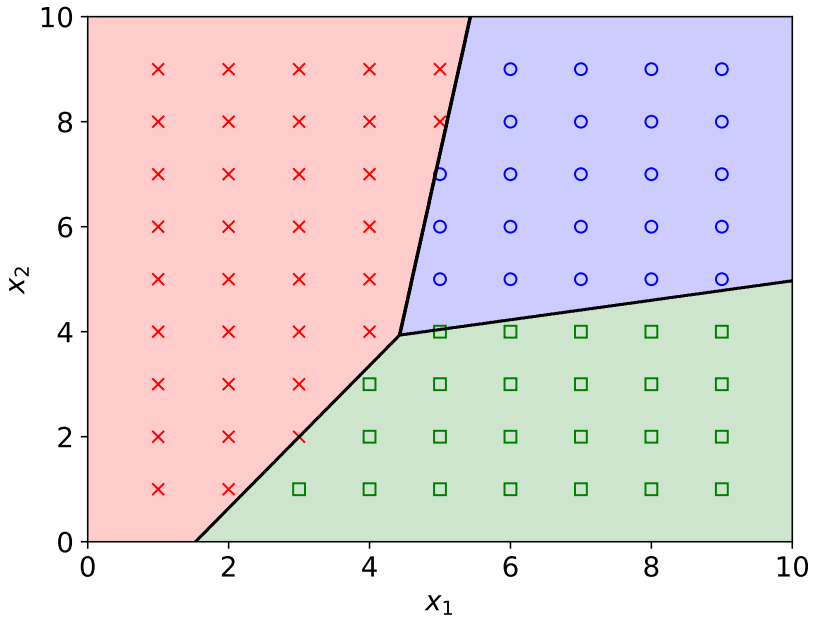
Maximum likelihood estimation: given training data $\mathcal{S}$ from $\mathbb{R}^d \times [K]$, log-likelihood of $w^{(1)}, \ldots, w^{(K)}$ is

$$\ln L(w^{(1)}, \ldots, w^{(K)}) = \sum_{(x,y)\in\mathcal{S}} \ln\left(\frac{\exp(x^\mathsf{T} w^{(y)})}{\sum_{c\in[K]} \exp(x^\mathsf{T} w^{(c)})}\right)$$

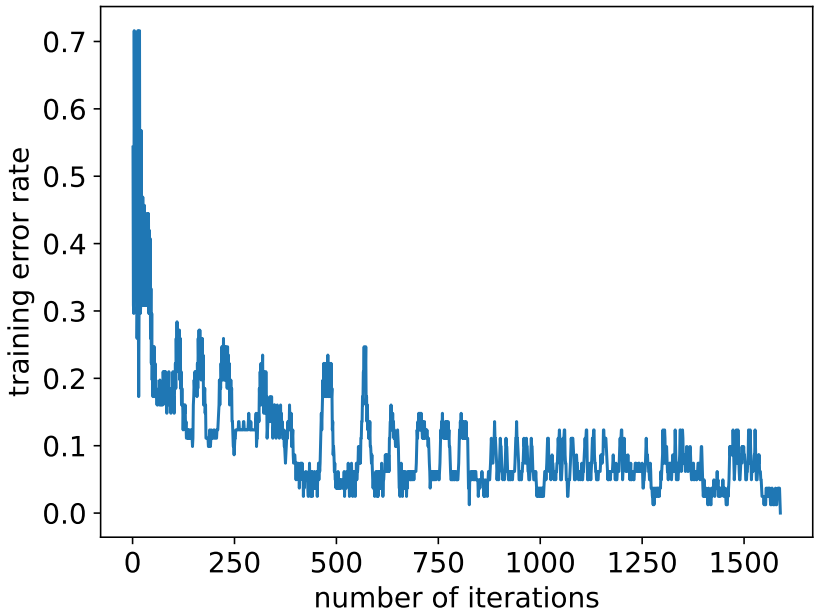
$$= \sum_{(x,y)\in\mathcal{S}} \left(x^\mathsf{T} w^{(y)} - \ln\left(\sum_{c\in[K]} \exp(x^\mathsf{T} w^{(c)})\right)\right)$$

- Negative log-likelihood turns out to be a convex objective function

Synthetic example: "linearly separable" dataset



Synthetic example: "linearly separable" dataset



Can also interpret negative log-likelihood as sum of log losses, like in binary case, for prediction function

$$\hat{p}_W(x) = \mathrm{softmax}(Wx)$$

where $\mathrm{softmax}\colon \mathbb{R}^K \to \mathbb{R}^K$ (soft(arg)max function) is defined by

$$\mathrm{softmax}(u)_k = \frac{\exp(u_k)}{\sum_{c\in[K]} \exp(u_c)}$$

- $W \in \mathbb{R}^{K\times d}$ is matrix of parameters, one row per class
- Negative log-likelihood is

$$J(W) = \sum_{(x,y)\in\mathcal{S}} -\ln(\hat{p}_W(x)_y)$$

Setup

```
import torch

W = torch.zeros(d, K)
W.requires_grad_(True)

def f(x):
  return torch.softmax(x.matmul(W),
  ↪  dim=1)

loss = torch.nn.NLLLoss(reduction='sum')
```

Gradient descent code

```
for t in range(num_iter):
  J = loss(f(x), y)
  J.backward()
  with torch.no_grad():
    W -= eta * W.grad
    W.grad.zero_()
```


Example: iris dataset

- Multi-class logistic regression ($\eta = 0.01$, $T = 2^{17}$ iterations)
- Test error rate: 3.33%