Classification

Lecture 04 — CS 577 Deep Learning

Instructor: Yutong Wang

Computer Science
Illinois Institute of Technology

September 11, 2024

Administrative matter

- For the course project, you can form your own groups (2-4 people) or have the groups be assigned to you randomly.

Binary classification

Let $i = 1, \ldots, N$ (the sample index)

- Training samples $\mathbf{x}^{(i)} \in \mathcal{X} \subseteq \mathbb{R}^d$
- Labels $y^{(i)} \in \mathcal{Y} = \{\pm 1\}$
- $f(\cdot; \boldsymbol{\theta}) : \mathcal{X} \to \mathcal{Y}$ $\quad = \mathbb{R}$

Previously

$$\begin{array}{ccccc} \text{Sample} & \mapsto & \text{Model output} & \mapsto & \text{Model prediction} \\ \mathbf{x} & \mapsto & \mathbf{z} = f(\mathbf{x}; \boldsymbol{\theta}) & \mapsto & \hat{y} = \mathrm{Pred}(\mathbf{z}) \approx y \text{ ground truth} \end{array}$$

Task	Model Output	Pred	Loss function†
Regression	Number	identity	squared errror
Binary classification	Number	sign	perceptron binary cross entropy
Multiclass classification	Vector	argmax	cross entropy

† there are other choices of loss functions that are valid. These are just examples.

Perceptron: an example

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

$x^{(1)}, \dots, x^{(N)}, x^{(N+1)\%N)}$

$x^{(N+2\%N)}$

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots,$ time T,

could be larger than N

wrap around

1. Initialize $\mathbf{w} = \mathbf{0}$.
2. For $t = 1, 2, \dots, T$
   - 2.1 If $y^{(t)} \mathbf{w}^\top \mathbf{x}^{(t)} > 0$, then $\mathbf{w} \leftarrow \mathbf{w}$, ← no mistake
   - 2.2 Else, then $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)}$. ← mistake

Output: $\mathbf{w}$

Stochastic gradient descent (SGD)

Let $\eta_t > 0$ be learning rates, $t = 1, 2, \ldots$
Let $m \geq 1$ be an integer (mini-batch)

- Initialize $\boldsymbol{\theta}$
- While not converged (t = iteration counter):
  - Select m samples $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$ and matching labels $\{y^{(1)}, \ldots, y^{(m)}\}$
  - Compute gradient $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} \frac{1}{m} \sum_{i=1}^{m} L(f(\mathbf{x}^{(i)}, \boldsymbol{\theta}), y^{(i)})$
  - Compute update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_t \mathbf{g}$

not the Full ERM
but a sampled version of
the ERM

Hope: the sampling is not too off.

Stochastic gradient descent (SGD)

Let $\eta_t > 0$ be learning rates, $t = 1, 2, \ldots$
Let $m \geq 1$ be an integer   Size 1   minibatch

- Initialize $\boldsymbol{\theta}$
- While not converged (t = iteration counter):
  - Select 1 sample $\{\mathbf{x}^{(t)}\}$ and its label $\{y^{(t)}\}$
  - Compute gradient $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} L(f(\mathbf{x}^{(t)}, \boldsymbol{\theta}), y^{(t)})$
  - Compute update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_t \mathbf{g}$

cheap

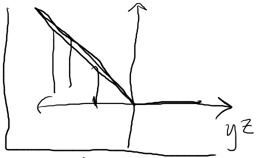
but

not as performant

Perceptron loss

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} L(f(x^{(i)};\theta), y)$$

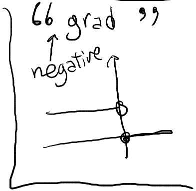
u



Consider

$$L(z, y) := \max\{0, -yz\}$$

and $\boldsymbol{\theta} = \mathbf{w}$ as in the perceptron algorithm. solve with $m=1$ SGD.



$$\nabla_{\boldsymbol{\theta}} L(f(\mathbf{x}^{(t)}, \boldsymbol{\theta}), y^{(t)}) = \nabla_{\mathbf{w}} \max\{0, -y^{(t)}\mathbf{w}^\top \mathbf{x}^{(t)}\}$$

chain rule

case 1: $(-1) \cdot y^{(t)} x^{(t)}$ if $y^{(t)} w^\top x^{(t)} < 0$

wrong sign predict

der of $\max\{0, -\square\}$

case 2: $0 \cdot \square$ if $y^{(t)} w^\top x^{(t)} > 0$

Perceptron loss

$\max\{0, -u\}$, where $u = y^{(t)} \underbrace{w^\top x^{(t)}}_{z^{(t)}}$

Consider

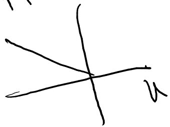
$$L(z, y) := \max\{0, -\overbrace{yz}^{u}\}$$

$$\nabla_w \max\{0, -u\} = \left(\frac{\partial}{\partial u} \max\{0, -u\}\right) \nabla_w \left(y^{(t)} w^\top x^{(t)}\right)$$

and $\boldsymbol{\theta} = \mathbf{w}$ as in the perceptron algorithm.

$$\nabla_{\boldsymbol{\theta}} L(f(\mathbf{x}^{(t)}, \boldsymbol{\theta}), y^{(t)}) = \nabla_{\mathbf{w}} \max\{0, -y^{(t)} \mathbf{w}^\top \mathbf{x}^{(t)}\}$$

$\max\{0, -u\}$



Perceptron as stochastic gradient descent (SGD)

Let $\eta_t = 1$ be learning rates, $t = 1, 2, \ldots$

- Initialize $\boldsymbol{\theta}$
- While not converged (t = iteration counter):
  - Select 1 sample $\{\mathbf{x}^{(t)}\}$ and its label $\{y^{(t)}\}$
  - Compute gradient $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} L(f(\mathbf{x}^{(t)}, \boldsymbol{\theta}), y^{(t)})$
  - Compute update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \mathbf{g}$

update rule
in percep
alg

$\equiv$

SGD with
$m = 1$

Breakout session 1

1. Implement the "full-batch" version of perceptron, i.e., $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ where

   $m = N$

   $$J(\boldsymbol{\theta}) := \frac{1}{N} \sum_{i=1}^{N} J_i(\boldsymbol{\theta})$$

   and

   $$J_i(\boldsymbol{\theta}) := L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), y^{(i)})$$

   Does it work?
2. What if you use a smaller stepsize like $\eta_t = 0.1$?
3. Is this a good loss function? What would be a better choice? $\boxed{\checkmark}$

Probability

Keep in mind – quest to discover cross entropy

Note: For reference, see §5.5.1 of [GBC16]

- How does the label depend on the label?
- *Conditional probability*: $p_{\text{data}}(y \mid \mathbf{x})$ probability of y given $\mathbf{x}$
- "Recipe" for generating synthetic data

label mechanical turk

$$\underbrace{p_{\text{data}}(\mathbf{x})}_{\text{sample raw data}} \cdot \underbrace{p_{\text{data}}(y \mid \mathbf{x})}_{\text{sample the label}} = p_{\text{data}}(\mathbf{x}, y)$$

Suppose pictures of animal internet archive

Gaussian/normal distribution

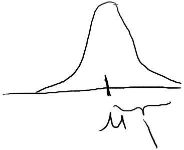
Note: §3.9.3 of [GBC16]

- Gaussian random variable with mean μ and variance σ^2

$$\epsilon \sim \mathcal{N}(\mu, \sigma^2) \qquad (\epsilon \text{ for "error"})$$

- The probability density function (PDF)

$$\mathcal{N}(\epsilon; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(\epsilon-\mu)^2}{\sigma^2}\right)$$



σ^2

"secretly"

$\mu = f(x;\theta)$

What exactly is $p_{\text{model}}(y \mid \mathbf{x}; \boldsymbol{\theta})$?

- Model params in linear regression: $\boldsymbol{\theta} = \mathbf{w}$ (without the bias term for now!)
- The model: $f(\mathbf{x}; \boldsymbol{\theta}) := \mathbf{w}^\top \mathbf{x}$
- Gaussian/normally distributed noise: there exists $\sigma^2 > 0$ such that

$$p_{\text{data}}(y \mid \mathbf{x}) = \mathcal{N}(y; \underbrace{f(\mathbf{x}; \boldsymbol{\theta})}_{\mu}, \sigma^2)$$

Note: Model the mean using a model

Maximum likelihood

θ explains the data well

Likelihood: Want high prob

PDF

max $$\prod_{i=1}^{N} p(y^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(y^{(i)} - f(\mathbf{x}^{(i)}; \boldsymbol{\theta}))^2}{\sigma^2}\right)$$

Previously, apply $-\log(\cdot)$ and get (up to scaling)

min $$\frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - f(\mathbf{x}^{(i)}; \boldsymbol{\theta}))^2$$

Mean square error

Note: That was for regression, what about classification?

Max Likelihood framework to motivate cross entropy

Bernoulli distribution

-1 $+1$ $p =$ prob of $+1$ class
$1-p =$ " -1 class

Note: §3.9.1 of [GBC16]

- Bernoulli random variable $\epsilon \in \{0, 1\}$ with parameter $p \in [0, 1]$

$$\epsilon \sim \text{Bern}(p)$$

secretly want to model this

- The probability density function (PDF)

$$\text{Bern}(\epsilon; p) = p^{\epsilon}(1-p)^{1-\epsilon}$$

$=$ prob of $\epsilon = 1$

What exactly is $p_{\text{model}}(y \mid \mathbf{x}; \boldsymbol{\theta})$?

- Model params in *logistic* regression: $\boldsymbol{\theta} = \mathbf{w}$ (without bias)

What exactly is $p_{\text{model}}(y \mid \mathbf{x}; \boldsymbol{\theta})$?

$p \in [0,1]$?

$w^T x \in \mathbb{R}$

- Model params in *logistic* regression: $\boldsymbol{\theta} = \mathbf{w}$ (without bias)
- The model: $f(\mathbf{x}; \boldsymbol{\theta}) := \mathbf{w}^\top \mathbf{x}$ Can this be a model for p?

Note: We want to model p in the Bernoulli parameter.

What exactly is $p_{\text{model}}(y \mid \mathbf{x}; \boldsymbol{\theta})$?

- Model params in *logistic* regression: $\boldsymbol{\theta} = \mathbf{w}$ (without bias)
- The model: $f(\mathbf{x}; \boldsymbol{\theta}) := \mathbf{w}^\top \mathbf{x}$

Note: We want to model p in the Bernoulli parameter.

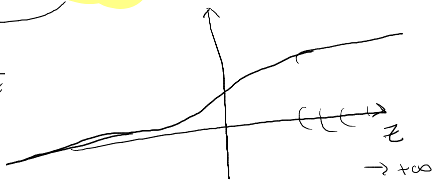
- There exists $p \in [0, 1]$ such that

aka logistic function

$$p_{\text{data}}(y \mid \mathbf{x}) = \text{Bern}\left((y+1)/2\,;\, \underline{\text{sigmoid}(f(\mathbf{x}; \boldsymbol{\theta}))}\right)$$

where

$$\text{sigmoid}(z) = \frac{1}{1 + e^{-z}}$$



idea

$f(x; \theta)$ models p

but through a transformation

Breakout session: Maximum likelihood

$\mathrm{Bern}(\epsilon; p) = p^{\epsilon}(1-p)^{\epsilon}$

Likelihood:

$$\prod_{i=1}^{N} p(y^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}) = \prod_{i=1}^{N}$$

Previously, apply $-\log(\cdot)$ and get (up to scaling)

ans

$$\frac{1}{N}\sum_{i=1}^{N} \log(1 + \exp(-y^{(i)}\mathbf{w}^{\top}\mathbf{x}^{(i)}))$$

Exercise: Fill in the new likelihood.

Breakout session: Maximum likelihood

$f(x;\theta)$

Likelihood:

Case $y=+1$, $y=-1$

$$\prod_{i=1}^{N} p(y^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}) = \prod_{i=1}^{N} \text{Bern}(1;\ \text{Sigmoid}(z))$$

$\text{sigmoid}(z) = \frac{1}{1+e^{-z}}$

$-\log(\cdot)$

$-\log((1+e^{-z})^{-1})$

$=\log(1+e^{-z})$

Previously, apply $-\log(\cdot)$ and get (up to scaling)

$$\frac{1}{N}\sum_{i=1}^{N} \log(1+\exp(-y^{(i)}\mathbf{w}^{\top}\mathbf{x}^{(i)}))$$

Exercise: Fill in the new likelihood.

Breakout session: Maximum likelihood

Likelihood:

$$\prod_{i=1}^{N} p(y^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}) = \prod_{i=1}^{N}$$

Case $y = -1$

$\text{Bern}(0; \text{sigmoid}(z))$

$= (1 - \text{sigmoid}(z))$

$= 1 - \frac{1}{1+e^{-z}}$

$= \frac{e^{-z}}{1+e^{-z}} = \frac{1}{1+e^{z}}$

Previously, apply $-\log(\cdot)$ and get (up to scaling)

$$\frac{1}{N} \sum_{i=1}^{N} \log(1 + \exp(-y^{(i)} \mathbf{w}^\top \mathbf{x}^{(i)}))$$

$-\log(\cdot)$

$L(z, y)$

Exercise: Fill in the new likelihood.

$= \log(1 + e^{-yz}) = \begin{cases} \log(1+e^{-z}) & \text{if } y = 1 \\ \log(1+e^{z}) & y = -1 \end{cases}$

$\log(1 + e^{z})$

Breakout session

1. Implement the "full-batch" version of perceptron, i.e., $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ where

$$J(\boldsymbol{\theta}) := \frac{1}{N} \sum_{i=1}^{N} J_i(\boldsymbol{\theta})$$

Perceptron

and

$$J_i(\boldsymbol{\theta}) := L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), y^{(i)})$$

Does it work? Nope.

Breakout session

1. Implement the "full-batch" version of perceptron, i.e., $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ where

   $$J(\boldsymbol{\theta}) := \frac{1}{N} \sum_{i=1}^{N} J_i(\boldsymbol{\theta})$$

   and

   $$J_i(\boldsymbol{\theta}) := L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), y^{(i)})$$

   Does it work? Nope.
2. What if you use a smaller stepsize like $\eta_t = 0.1$? Nope.

Breakout session

$$\frac{\partial}{\partial w} J_i(w) = \frac{\partial}{\partial z} L(\underbrace{z}_{w^T x^{(i)}}, \underbrace{y}_{y^{(i)}}) \cdot \frac{\partial z}{\partial w}$$

1. Implement the "full-batch" version of perceptron, i.e., $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ where

   $$J(\boldsymbol{\theta}) := \frac{1}{N} \sum_{i=1}^{N} J_i(\boldsymbol{\theta})$$

   and

   $$J_i(\boldsymbol{\theta}) := L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), y^{(i)})$$

   $$L(z,y) = \log(1+e^{-yz})$$

   Does it work? Nope.
2. What if you use a smaller stepsize like $\eta_t = 0.1$? Nope.
3. Is this a good loss function? What would be a better choice? The logistic loss.

Breakout session

$\frac{\partial}{\partial z} L(z,y) = \frac{\partial}{\partial z}\left(\log(1+e^{-yz})\right) = \frac{1}{1+e^{-yz}} \cdot e^{-yz} \cdot (-y)$

$\|$

$-y \cdot \frac{e^{-yz}}{1+e^{-yz}}$

1. Implement the "full-batch" version of perceptron, i.e., $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ where

$$J(\boldsymbol{\theta}) := \frac{1}{N} \sum_{i=1}^{N} J_i(\boldsymbol{\theta})$$

   and

$$J_i(\boldsymbol{\theta}) := L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), y^{(i)})$$

$\frac{\partial\, w^T x^{(i)}}{\partial w} = x^{(i)}$

   Does it work? Nope.

2. What if you use a smaller stepsize like $\eta_t = 0.1$? Nope.
3. Is this a good loss function? What would be a better choice? The logistic loss.

Activation function

 bad loss  good act

Rectified linear unit or “relu”

$$\mathrm{relu}(z) := \max\{0, z\}$$

Note: Plot “relu” and its derivative

Linearity

2-layer neural network with "non-linear" activation $g : \mathbb{R} \to \mathbb{R}$.

$$f\left(\mathbf{x}; \mathbf{w}^{(2)}, b^{(2)}, \mathbf{W}^{(1)}, \mathbf{b}^{(1)}\right) := \mathbf{w}^{(2)\top} g(\underbrace{\mathbf{W}^{(1)\top}\mathbf{x} + \mathbf{b}^{(1)}}_{\in \mathbb{R}^m}) + b^{(2)} \in \mathbb{R}$$

T

$m = 10$ relu

$W^{(1)} \in \mathbb{R}^{d \times m}$ $w^{(2)} \in \mathbb{R}^m$

1-layer neural network (Regression)

$L(z^{(i)}, y^{(i)})$ $\qquad$ $L(z,y) = (y-z)^2$ MSE

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - z^{(i)})^2 \quad \text{where} \quad z_i = \mathbf{w}^{(2)\top} g(\mathbf{h}^{(i)}) + b^{(2)} \quad \text{and} \quad \mathbf{h}^{(i)} = \mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)}$$

$$\frac{\partial L}{\partial z}(z,y) = -2(y-z)$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(2)}} = -2(y^{(i)} - z^{(i)}) g(\mathbf{h}^{(i)})$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial b^{(2)}} = -2(y^{(i)} - z^{(i)})$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(1)}} = -2(y^{(i)} - z^{(i)})(\mathbf{w}^{(2)} \odot g'(\mathbf{h}^{(i)})) x^{(i)}$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{b}^{(1)}} = -2(y^{(i)} - z^{(i)})(\mathbf{w}^{(2)} \odot g'(\mathbf{h}^{(i)}))$$

1-layer neural network (1-dim data)

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - z^{(i)})^2 \quad \text{where} \quad z_i = \mathbf{w}^{(2)\top} g(\mathbf{h}^{(i)}) + b^{(2)} \quad \text{and} \quad \mathbf{h}^{(i)} = \mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)}$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(2)}} = -2(y^{(i)} - z^{(i)}) g(\mathbf{h}^{(i)})$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial b^{(2)}} = -2(y^{(i)} - z^{(i)})$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(1)}} = -2(y^{(i)} - z^{(i)})(\mathbf{w}^{(2)} \odot g'(\mathbf{h}^{(i)})) x^{(i)}$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{b}^{(1)}} = -2(y^{(i)} - z^{(i)})(\mathbf{w}^{(2)} \odot g'(\mathbf{h}^{(i)}))$$

1-layer neural network (higher dim data)

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - z^{(i)})^2 \quad \text{where} \quad z_i = \mathbf{w}^{(2)\top} g(\mathbf{h}^{(i)}) + b^{(2)} \quad \text{and} \quad \mathbf{h}^{(i)} = \mathbf{W}^{(1)}\mathbf{x}^{(i)} + \mathbf{b}^{(1)}$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(2)}} = -2(y^{(i)} - z^{(i)}) g(\mathbf{h}^{(i)})$$

$W^{(1)} \in \mathbb{R}^{m \times d}$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial b^{(2)}} = -2(y^{(i)} - z^{(i)})$$

d

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{W}^{(1)}} = -2(y^{(i)} - z^{(i)})(\mathbf{w}^{(2)} \odot g'(\mathbf{h}^{(i)}))\mathbf{x}^{(i)\top}$$

m

(m, d)

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{b}^{(1)}} = -2(y^{(i)} - z^{(i)})(\mathbf{w}^{(2)} \odot g'(\mathbf{h}^{(i)}))$$

1-layer neural network (higher dim data)

$$J_i(\boldsymbol{\theta}) := \log(1+e^{-y^{(i)}z^{(i)}}) \quad \text{where} \quad z_i = \mathbf{w}^{(2)\top}g(\mathbf{h}^{(i)})+b^{(2)} \quad \text{and} \quad \mathbf{h}^{(i)} = \mathbf{W}^{(1)}\mathbf{x}^{(i)}+\mathbf{b}^{(1)}$$

$$\frac{\partial L(z,y)}{\partial z} = \frac{-y e^{-yz}}{1+e^{-yz}}$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(2)}} = \qquad g(\mathbf{h}^{(i)})$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial b^{(2)}} =$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{W}^{(1)}} = \qquad (\mathbf{w}^{(2)} \odot g'(\mathbf{h}^{(i)}))\mathbf{x}^{(i)\top}$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{b}^{(1)}} = \qquad (\mathbf{w}^{(2)} \odot g'(\mathbf{h}^{(i)}))$$

Breakout session

- The "two-moons" toy dataset

Multiclass classification

Let $i = 1, \ldots, N$ (the sample index)

- Training samples $\mathbf{x}^{(i)} \in \mathcal{X} \subseteq \mathbb{R}^d$
- Labels $y^{(i)} \in \mathcal{Y} = \{1, 2, \ldots, K\}$
- $f(\cdot; \boldsymbol{\theta}) : \mathcal{X} \to \mathcal{Y}$

$\{\pm 1\}$

becomes

Multiclass linear classifier (second attempt)

$\mathbf{W} = \left[\mathbf{w}_1 \quad \cdots \quad \mathbf{w}_K\right] \in \mathbb{R}^{d\times K}$ with classifier given by

sign$(w^T x)$

$$f(\mathbf{x};\mathbf{W}) := \operatorname{argmax}_{\hat{y}=1,\dots,K} \quad \mathbf{w}_{\hat{y}}^\top \mathbf{x} \quad \in \{1,\dots,K\} \tag{1}$$

1 col per class

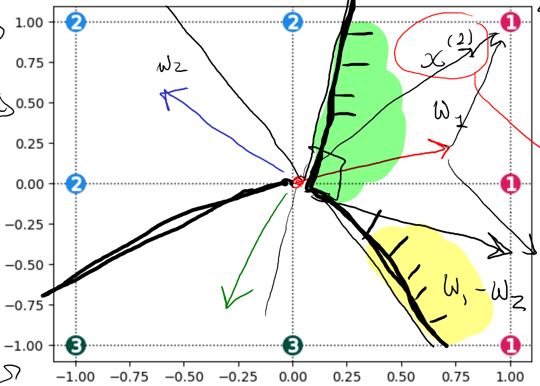
class 2

generic

$w_1^T x - w_2^T x = (w_1 - w_2)^T x > 0$

&

$w_1^T x - w_3^T x > 0$



class 3

class 1

w_1 ?

$w_1^T x^{(2)}$

$>$ both

$w_2^T x^{(2)}$

& $w_3^T x^{(2)}$

Multiclass perceptron

multiclass perceptron suffers from instability

Perceptron update

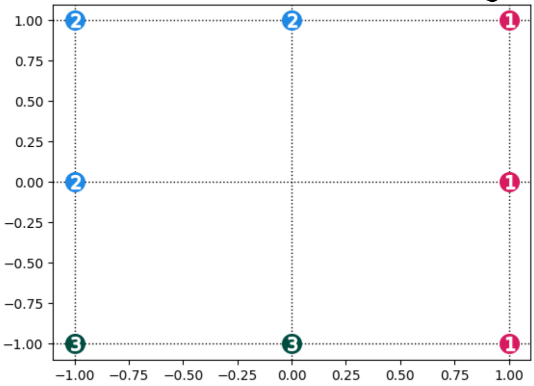
Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots$, time T

1. Initialize $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 If $\underbrace{\operatorname{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}}_{\text{pred}} \setminus \{\underbrace{y^{(t)}}_{\text{truth}}\}$ is empty, then pass
   - 2.2 Else $\hat{y}^{(t)} \leftarrow \operatorname{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}$ then

$\hat{y}^{(t)} \neq y^{(t)}$

$$\mathbf{w}_{\hat{y}^{(t)}} \leftarrow \mathbf{w}_{\hat{y}^{(t)}} - \mathbf{x}^{(t)}$$
$$\mathbf{w}_{y^{(t)}} \leftarrow \mathbf{w}_{y^{(t)}} + \mathbf{x}^{(t)}$$

Output: $\mathbf{W}$



Multinoulli distribution

Bernoulli $\epsilon \in \{0,1\}$ $p \in [0,1]$

Note: §3.9.1 of [GBC16]

- Multinoulli random variable $\epsilon \in \{1, 2, \ldots, K\}$ with parameter $p_1, \ldots, p_K \in [0,1]$ such that $p_1 + \cdots + p_K = 1$

$$\epsilon \sim \mathrm{Multi}(p_1, \ldots, p_K) = \mathrm{Multi}(\mathbf{p})$$

p_1	p_2	p_3
cat	bird	dog
$= 0.9$	$= 0.05$	$= 0.05$

- The probability mass function (PMF)

$$\mathrm{Multi}(\epsilon; \mathbf{p}) = p_\epsilon$$

$=$ picks one entry of p.

What exactly is $p_{\text{model}}(y \mid \mathbf{x}; \boldsymbol{\theta})$?

$$\begin{array}{cc} & k \\ d & \left[\qquad\right] \end{array}$$

- Model params in *multinomial logistic* regression: $\boldsymbol{\theta} = \mathbf{W}$ (without bias)

What exactly is $p_{\text{model}}(y \mid \mathbf{x}; \boldsymbol{\theta})$?

- Model params in *multinomial logistic* regression: $\boldsymbol{\theta} = \mathbf{W}$ (without bias)
- The model: $f(\mathbf{x}; \boldsymbol{\theta}) := \mathbf{W}^\top \mathbf{x} \in \mathbb{R}^K$

Note: We want to model $\mathbf{p}$ in the Multinoulli parameter.

$W^T x$ could be negative

doesn't necessarily sum to 1

Sigmoid? For Multiclass

What exactly is $p_{\text{model}}(y \mid \mathbf{x}; \boldsymbol{\theta})$?

- Model params in *multinomial logistic* regression: $\boldsymbol{\theta} = \mathbf{W}$ (without bias)
- The model: $f(\mathbf{x}; \boldsymbol{\theta}) := \mathbf{W}^\top \mathbf{x}$

Note: We want to model $\mathbf{p}$ in the Multinoulli parameter.

- There exists $p \in [0, 1]$ such that

$$p_{\text{data}}(y \mid \mathbf{x}) = \text{Multi}(y\,;\, \underline{\mathsf{softmax}}(\mathbf{W}^\top \mathbf{x}))$$

where...

$\text{softmax} : \mathbb{R}^K \rightsquigarrow \text{Prob}(K)$

$\Downarrow$

"logits" $= W^T x \rightsquigarrow$ K p

$\sum_{\ell=1} p_\ell = 1$

Viewing the "cross-entropy" as a NLL

Note: [GBC16, §6.2.2.3] "Softmax Units for Multinoulli Output Distributions"

$$\text{softmax}(\mathbf{z}) = \underbrace{\frac{1}{\sum_{j=1}^{K} \exp(z_j)}}_{\text{sum to one}} \underbrace{\begin{bmatrix} \exp(z_1) \\ \vdots \\ \exp(z_K) \end{bmatrix}}_{\text{makes positive}} = \underbrace{\begin{bmatrix} \text{softmax}(\mathbf{z})_1 \\ \vdots \\ \text{softmax}(\mathbf{z})_K \end{bmatrix}}_{\text{notation}} = \underbrace{\begin{bmatrix} p_1 \\ \vdots \\ p_K \end{bmatrix}}_{\text{understand that they implicitly depend on } z.} = \mathbf{p}$$

$$z = \begin{bmatrix} z_1 \\ \vdots \\ z_K \end{bmatrix}$$

binary

z was scalar

Derivative

$\prod_{i=1}^{N}$ softmax$(z^{(i)})_{y^{(i)}}$

log(·)

without the Σ.

$$\begin{bmatrix} z_1^{(i)} \\ \vdots \\ z_K^{(i)} \end{bmatrix}$$

Cross entropy $L(\mathbf{z}, y) = -\log(\mathrm{softmax}(\mathbf{z})_y)$

$L(z,y)$

no longer a scalar

$$\frac{\partial L}{\partial \mathbf{z}}(\mathbf{z}, y) = \frac{\partial}{\partial \mathbf{z}}(-\log(\mathrm{softmax}(\mathbf{z})_y))$$

$$\frac{\partial}{\partial z_j}(-\log(\mathrm{softmax}(\mathbf{z})_y)) = -\frac{1}{\underbrace{\mathrm{softmax}(\mathbf{z})_y}_{p_y}} \boxed{\frac{\partial}{\partial z_j}(\mathrm{softmax}(\mathbf{z})_y)}$$

$y \neq j$: $-p_y p_j$

$y = j$: $p_y(1-p_y)$

$$\frac{\partial}{\partial z_j}(\mathrm{softmax}(\mathbf{z})_y) = \frac{\partial}{\partial z_j}\left(\frac{e^{z_y}}{\sum_{\ell=1}^{K} e^{z_\ell}}\right)$$

2 cases $\begin{cases} y = j \\ y \neq j \end{cases}$

$\frac{\partial L}{\partial z_j}$ ← Need for each $j = 1, \cdots, K$

Derivative of the softmax

$$\frac{\partial}{\partial z_j}\left(\frac{e^{z_y}}{\sum_{\ell=1}^K e^{z_\ell}}\right)$$

Case 1: $j \neq y$

constant

$$\frac{\partial}{\partial z_j}\left(\frac{e^{z_y}}{\sum_{\ell=1}^K e^{z_\ell}}\right) = -\frac{e^{z_y}}{(\sum_{\ell=1}^K e^{z_\ell})^2} e^{z_j} = -\overbrace{\frac{e^{z_y}}{\sum_{\ell=1}^K e^{z_\ell}}}^{=: p_y} \cdot \overbrace{\frac{e^{z_j}}{\sum_{\ell=1}^K e^{z_\ell}}}^{=: p_j} = -p_y \cdot p_j$$

e^{z_y} appears here once

Derivative of the softmax

$$\frac{\partial}{\partial z_y}\left(\frac{e^{z_y}}{\sum_{\ell=1}^K e^{z_\ell}}\right)$$

e^0

Case 2: $j = y$

bring to denom

$$\frac{\partial}{\partial z_y}\left(\frac{e^{z_y}}{\sum_{\ell=1}^K e^{z_\ell}}\right) = \frac{\partial}{\partial z_y}\left(\frac{1}{\sum_{\ell=1}^K e^{z_\ell - z_y}}\right) = -\frac{1}{(\sum_{\ell=1}^K e^{z_\ell - z_y})^2}\frac{\partial}{\partial z_y}\sum_{\ell=1}^K e^{z_\ell - z_y}$$

$$\frac{\partial}{\partial z_y}\sum_{\ell=1}^K e^{z_\ell - z_y} = -\sum_{\ell=1:\ell\neq y}^K e^{z_\ell - z_y} = -\boxed{e^{-z_y}}\sum_{\ell=1:\ell\neq y}^K e^{z_\ell}$$

Constant

ℓ now index

Derivative of the softmax

$$\frac{\partial}{\partial z_y}\left(\frac{e^{z_y}}{\sum_{\ell=1}^{K} e^{z_\ell}}\right)$$

Case 2: $j = y$

$$\frac{\partial}{\partial z_y}\left(\frac{e^{z_y}}{\sum_{\ell=1}^{K} e^{z_\ell}}\right) = \frac{1}{(\sum_{\ell=1}^{K} e^{z_\ell - z_y})^2} e^{-z_y} \sum_{\ell=1:\ell\neq y}^{K} e^{z_\ell}$$

$$= \frac{(e^{z_y})^{\cancel{2}}}{(\sum_{\ell=1}^{K} e^{z_\ell})^2} \cancel{e^{-z_y}} \sum_{\ell=1:\ell\neq y}^{K} e^{z_\ell}$$

$$= \frac{e^{z_y}}{\sum_{\ell=1}^{K} e^{z_\ell}} \cdot \boxed{\frac{\sum_{\ell=1:\ell\neq y}^{K} e^{z_\ell}}{\sum_{\ell=1}^{K} e^{z_\ell}}} = \overbrace{\frac{e^{z_y}}{\sum_{\ell=1}^{K} e^{z_\ell}}}^{p_y} \cdot \overbrace{\boxed{\left(1 - \frac{e^{z_y}}{\sum_{\ell=1}^{K} e^{z_\ell}}\right)}}^{1 - p_y} = p_y(1 - p_y)$$

Derivative

Cross entropy $L(\mathbf{z}, y) = -\log(\mathsf{softmax}(\mathbf{z})_y)$

$$\begin{cases} -\frac{1}{p_y} p_y p_j \\ -\frac{1}{p_y} p_y (1-p_y) \end{cases} = \begin{cases} p_j & j \neq y \\ (1-p_y) & j = y \end{cases}$$

$$\frac{\partial L}{\partial \mathbf{z}}(\mathbf{z}, y) = \mathbf{p} - \underbrace{\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{\text{one-hot}(y^{(i)})} \begin{matrix} \\ \\ \\ \leftarrow \quad y\text{-th position} \\ \\ \\ \\ \end{matrix}$$

p − one hot

1-layer neural network (multicategory data)

$W^{(2)} \in \mathbb{R}^{m \times K}$

$$J_i(\boldsymbol{\theta}) := L(\mathbf{z}^{(i)}, y^{(i)}) \quad \text{where} \quad z_i = \mathbf{W}^{(2)\top} g(\mathbf{h}^{(i)}) + \mathbf{b}^{(2)} \quad \text{and} \quad \mathbf{h}^{(i)} = \mathbf{W}^{(1)\top} \mathbf{x}^{(i)} + \mathbf{b}^{(1)}$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{W}^{(2)}} = \boxed{g(\mathbf{h}^{(i)})} \frac{\partial L(\mathbf{z}^{(i)}, y)}{\partial \mathbf{z}}^{\top}$$

K-dim

K

m

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{b}^{(2)}} = \frac{\partial L(\mathbf{z}^{(i)}, y)}{\partial \mathbf{z}}$$

matrix vector mul

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{W}^{(1)}} = ((\mathbf{W}^{(2)} \frac{\partial L(\mathbf{z}^{(i)}, y)}{\partial \mathbf{z}}) \odot g'(\mathbf{h}^{(i)}))\mathbf{x}^{(i)\top}$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{b}^{(1)}} = (\mathbf{W}^{(2)} \frac{\partial L(\mathbf{z}^{(i)}, y)}{\partial \mathbf{z}}) \odot g'(\mathbf{h}^{(i)})$$

1-layer neural network (higher dim data)

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - z^{(i)})^2 \quad \text{where} \quad z_i = \mathbf{w}^{(2)\top} g(\mathbf{h}^{(i)}) + b^{(2)} \quad \text{and} \quad \mathbf{h}^{(i)} = \mathbf{W}^{(1)}\mathbf{x}^{(i)} + \mathbf{b}^{(1)}$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(2)}} = -2(y^{(i)} - z^{(i)}) g(\mathbf{h}^{(i)})$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial b^{(2)}} = -2(y^{(i)} - z^{(i)})$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{W}^{(1)}} = -2(y^{(i)} - z^{(i)})(\mathbf{w}^{(2)} \odot g'(\mathbf{h}^{(i)}))\mathbf{x}^{(i)\top}$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{b}^{(1)}} = -2(y^{(i)} - z^{(i)})(\mathbf{w}^{(2)} \odot g'(\mathbf{h}^{(i)}))$$

References I

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT press, 2016.