Logistic Regression

A classification problem asks for a prediction $\mathcal{X} \to \Delta(\mathcal{C})$

prob. simplex on $\mathcal{C}$.

$\mathcal{X}$ = input space $\subseteq \mathbb{R}^d$

$\mathcal{C}$ = classes / labels

$x \in \mathcal{X}$ $\quad x = (x_1, \ldots, x_d)$

pos's ↖ ↗ dims = features $\quad x_i$ = value @ i^{th} feature.

Def' A probabilistic classifier for $\mathcal{X}, \mathcal{C}$ is a fn

$$P : \Theta \times \mathcal{X} \to \Delta(\mathcal{C}) \qquad (\theta, \vec{x}) \mapsto p_\theta(\cdot \mid \vec{x})$$

differentiable in θ for any fixed $\vec{x}$.

Eg. spam filter: $\mathcal{C} = \{\text{spam}, \text{not spam}\}$, $\mathcal{X}$ = emails

Multinomial logistic regression

$\mathcal{X} = \mathbb{R}^d$ $\quad$ $\mathcal{C}$ = classes / categories

$\Delta(\mathcal{C})$ = prob. distr.'s on $\mathcal{C}$. $\subseteq \mathbb{R}^{|\mathcal{C}|}$

relint $\Delta(\mathcal{C})$ = relative interior (in space it spans)

$$\text{softmax}: \mathbb{R}^{|\mathcal{C}|} \longrightarrow \text{relint}\,\Delta(\mathcal{C})$$

$$\vec{z} \longmapsto \frac{e^{\vec{z}}}{\sum_{c=1}^{|\mathcal{C}|} e^{z_c}} \quad \begin{matrix}(\text{vector}) \\ (\#)\end{matrix}$$

vector $\longrightarrow$ prob. distr. w/ weight on largest entries

Def^n A softmax regression model

$$P: \Theta \times \mathcal{X} \longrightarrow \text{relint}\,\Delta(\mathcal{C}) \subseteq \mathbb{R}^{|\mathcal{C}|}$$

$$(\theta, \vec{x}) \longmapsto \vec{p} = \text{softmax}\underbrace{(W^T \vec{x} + \beta)}_{!!}$$

$z_\theta : \mathcal{X} \to \mathbb{R}^{|\mathcal{C}|}$ "logit"

weights ↓ biases ↓

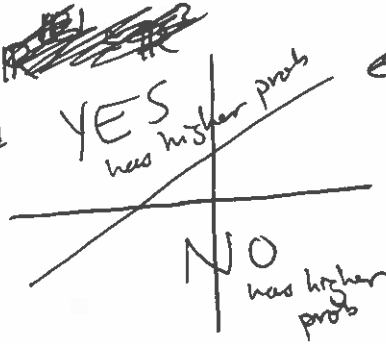
i.e. $\theta = (W, \vec{\beta})$ parameters

$\vec{\beta} \in \mathbb{R}^{|\mathcal{C}|}$

W is $d \times |\mathcal{C}|$

Eg. If $\mathcal{C} = \{\text{yes}, \text{no}\}$.

$\vec{x} \in \mathbb{R}^d$



← decision boundary (where prob's equal)

$\text{softmax}(W^T\vec{x} + \vec{\beta}) \in \mathbb{R}^2$

has 1st coord $\geq$ 2nd

i.e. $W^T\vec{x} + \vec{\beta}$ has slope 1

linear eq'n

Note: $\text{softmax}(1,1,\ldots,1) = (\underbrace{\tfrac{1}{n},\ldots,\tfrac{1}{n}}_{n})$

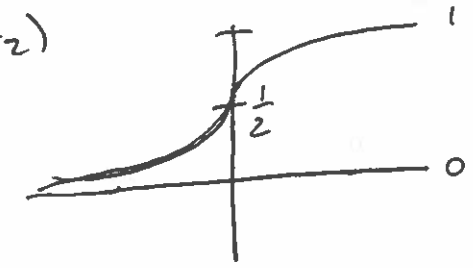
$\text{softmax}(\vec{z} + (1,1,\ldots,1)) = \text{softmax}(\vec{z})$

$= e^{(z_1,z_2)}\left(\frac{1}{e^{z_1}+e^{z_2}}\right)$

n=2 case

$\text{Softmax}(z_1, z_2) = (\underbrace{1-\sigma(z_2-z_1)}_{=\sigma(z_1-z_2)}, \sigma(z_2-z_1))$

with $\sigma(z) = \frac{e^z}{1+e^z}$



$\sigma: \mathbb{R} \to (0,1)$

inverse $\text{logit}(p) = \log\frac{p}{1-p}$ "log odds"

$((0,1) \to \mathbb{R})$

Loss $\ell(\underbrace{w, \beta}_{\theta}; \underbrace{\vec{x}, y}_{\in \mathcal{X} \times \mathcal{C}}) = -\log p_y = -z_y + \log \sum_{j=1}^{|\mathcal{C}|} e^{z_j}$

training data

y^{th} entry of $\vec{p}$

w/ $z = z_\theta(\vec{x})$

For many samples $(\vec{x}_i, y_i)$ ($\vec{x}_i$: observation, y_i: gold label)

$$\mathcal{L}(w, \beta) = \frac{1}{N} \sum_{i=1}^{N} \ell(w, \beta; \vec{x}_i, y_i)$$

We wish to minimize this.

Gradient Descent

$F: \mathbb{R}^k \to \mathbb{R}$ differentiable $\quad \theta \mapsto F(\theta)$ $\quad$ eg. $(w, \beta) \mapsto \mathcal{L}(w, \beta)$

gradient $\nabla F(\theta) := \begin{pmatrix} \frac{\partial F}{\partial \theta_1}(\theta) \\ \vdots \\ \frac{\partial F}{\partial \theta_2}(\theta) \end{pmatrix}$

direction of steepest descent @ θ:

$$-\frac{\nabla F(\theta)}{\|\nabla F(\theta)\|}.$$

Gradient Descent Algorithm:

$$\theta_{new} = \theta_{old} - \eta \nabla F(\theta)$$

η: learning rate

Gradient of loss

one hot vector!

$$\nabla_z \ell = \vec{p} - \vec{e}_y$$

$$\nabla_W \ell = \vec{x} (\vec{p} - \vec{e}_y)^T$$

$$\nabla_b \ell = \vec{p} - \vec{e}_y$$

Proposition

$\mathcal{L}$ is convex as a f^n in (W, b)

Artificial Neurons ("units")

Def'n An  is a f^n $\mathbb{R}^d \to \mathbb{R}$ of shape $\vec{x} \mapsto \phi(\vec{w}^T\vec{x} + b)$

Choice of activation = type of unit

eg. $H(t) = \begin{cases} 0 & t<0 \\ 1 & t \geq 0 \end{cases}$ gives a perceptron

Def'n A feedforward network or multilayer perceptron (MLP)

w/ layer widths $n_0, n_1, \ldots, n_L$ is a f^n $f_\theta : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$

(L = depth)

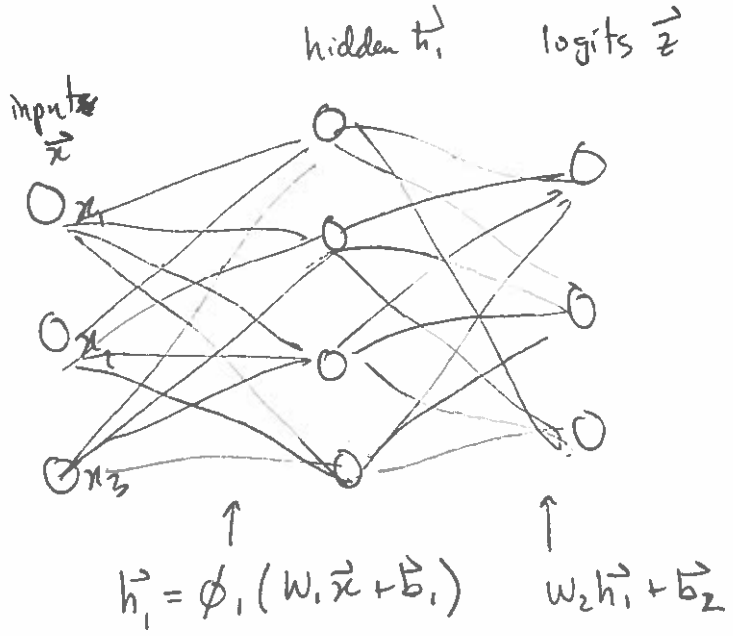
defined recursively by:

$h_0 = \vec{x} \in \mathbb{R}^{n_0}$

$a_\ell = W_\ell h_{\ell-1} + b_\ell \qquad h_\ell = \phi_\ell(a_\ell) \qquad 1 \leq \ell < L$ (ϕ_ℓ applied coordinatewise)

"logits" $\longrightarrow$ $a_L = W_L h_{L-1} + b_L \qquad f_\theta(x) = a_L$

with $W_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$, $a_\ell, b_\ell \in \mathbb{R}^{n_\ell}$, $h_\ell \in \mathbb{R}^{n_\ell}$



other common activations ϕ:

$\tanh t$

$\mathrm{ReLU} = \max\{0, t\}$

others

DEMO: Handwriting rec.

DEMO: Playground.

Def" **Feedforward neural language model** consists of:

① encoder $e_{\theta_1} : \mathcal{V}^* \longrightarrow \mathbb{R}^{n_0}$
context $\longmapsto$ feature vec

② feed f. net. $f_{\theta_2} : \mathbb{R}^{n_0} \to \mathbb{R}^V$

giving $P(\theta, c) = p_\theta(\cdot \mid c) = \text{softmax}(f_{\theta_2}(e_{\theta_1}(c)))$

Example "Bengio-style" "neural n-gram"

encoder: $C \in \mathbb{R}^{V \times d}$ embedding matrix gives $\left\{ \begin{array}{l} \mathcal{V} \longrightarrow \mathbb{R}^d \\ c \longmapsto r_c \ (c^{th} \text{ row}) \end{array} \right\}$

So tokens $a_1 \cdots a_{n-1} \longmapsto \vec{x} = r_{a_1} \cdots r_{a_{n-1}}$ concatenated into vector
$\in \mathbb{R}^{(n-1)d}$

ff. net: one hidden layer:

$$h_{ctx} = \tanh(W_1 \vec{x} + b_1) \qquad z = W_2 h_{ctx} + b_2$$

$$\theta = C, W_1, b_1, W_2, b_2$$

$V \times d,\ h \times (n-1)d,\ h,\ V \times h,\ V$