Softmax (Logistic) Regression

Linear regression: Predict a continuous value.
Classification: Predict one of k classes.

- classes might not be ordered

  ~~"cat" > "dog"~~

- might want to predict the probability for each possible class.

  <u>ex</u> 93% cat
  7% dog

ex Predict sell above, below or not sell from size, age, sq ft, prize

"one-hot" vectors:

$[1, 0, 0]$	$[0, 1, 0]$	$[0, 0, 1]$
below	above	not sell
1	2	3

Model predict $k = 3$ scores.

Larger score: "I think it's this class"

$$\underbrace{O_1}_{\text{score, class 1}} = \underbrace{W_{1,1}}_{\text{output}} \underbrace{x_1}_{\text{input, size}} + W_{1,2} \underbrace{x_2}_{\text{age}} + \ldots + b_1$$

$$o = Wx + b \quad W \in \mathbb{R}^{3 \times 4},\ b \in \mathbb{R}^3,\ x \in \mathbb{R}^4$$

$$o \in \mathbb{R}^3$$

have unnormalized scores (any real number)
want probabilities:

1. Sum to 1
2. Lie in $[0, 1]$

$\hat{y} = \text{softmax}(o) \quad o \in \mathbb{R}^k \quad \hat{y} \in \mathbb{R}^k$

$$\text{softmax}(o)_j = \frac{\exp(o_j)}{\sum_k \exp(o_k)}$$

1. $\sum_j \text{sofmax}(o)_j = \dfrac{\sum_j \exp(o_j)}{\sum_k \exp(o_k)} = 1$
2. $\exp(\ldots) \geq 0$

argmax gives us the highest-probability class

$$\mathrm{argmax}(o) = \mathrm{argmax}(\mathrm{softmax}(o))$$

softmax regression

$$p(\underset{\text{class label}}{y} \mid \underset{\text{input}}{x}) = \mathrm{softmax}(\underset{\text{params}}{W} x + b)$$

"linear" because the $\overset{o}{\text{scores}}$ are a linear function of the input

Want to find parameters that maximize $p(y | x)$ over training dataset.

↳ ground-truth label

$\prod_i p(y^{(i)} | x^{(i)})$ $(x^{(i)}, y^{(i)}$ i^{th} training example)

$\sum_i \log p(y^{(i)} | x^{(i)})$ (because log monotonic)

$-\sum_i \log p(y^{(i)} | x^{(i)})$ (because we minimize)

$-\sum_i \sum_j y_j^{(i)} \log \overbrace{\text{softmax}(Wx^{(i)} + b)}^{\text{model}}_j$

"cross-entropy"

measures how "wrong" a probability distribution is

$$-\sum_i \sum_j y_j^{(i)} \log \hat{y}_j^{(i)}$$

$$\hat{y}_j^{(i)} = \mathrm{softmax}(\overbrace{W x^{(i)} + b}^{o})_j$$

for one particular example:

$$L(y, \hat{y}) = -\sum_j y_j \log \frac{\exp(o_j)}{\sum_k \exp(o_k)}$$

$$= \sum_j y_j \log \sum_k \exp(o_k) - \sum_j y_j o_j$$

$$= \log \sum_k \exp(o_k) - \sum_j y_j o_j$$

$$\frac{d\,L(y,\hat{y})}{d\,o_j} \rightarrow \text{only new term}$$

$$d_{o_j}[L(y,\hat{y})] = d_{o_j}\left[\log \sum_k \exp(o_k) - \sum_j y_j o_j\right]$$

$$= \frac{d_{o_j} \sum_k \exp(o_k)}{\sum_k \exp(o_k)} - y_j$$

$$= \frac{\exp(o_j)}{\sum_k \exp(o_k)} - y_j$$

$$= \operatorname{softmax}(o)_j - y_j$$

Gradient Descent

We want to minimize $f(x)$

$$f(x+e) = \sum_{n=0}^{\infty} \frac{e^n f^{(n)}(x)}{n!}$$

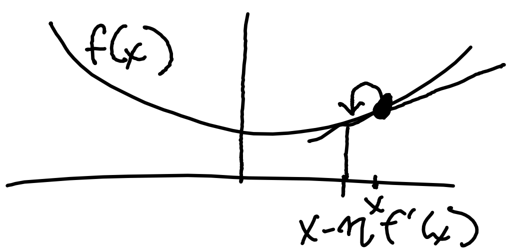
$$f(x+\underbrace{\varepsilon}_{\text{small}}) = f(x) + \varepsilon f'(x) + O(\varepsilon^2)$$

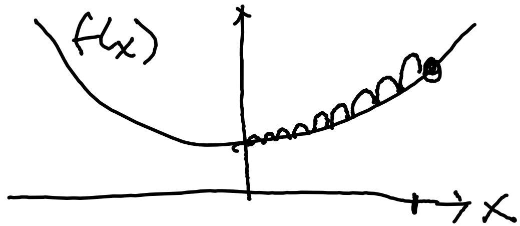
set $\varepsilon = -\eta f'(x)$ $\quad \eta$ small, positive

$$f(x-\eta f'(x)) = f(x) - \underbrace{\eta(f'(x))^2}_{\text{non-negative}} + O(\eta^2 (f'(x))^2)$$

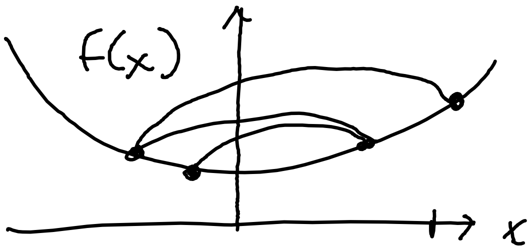
$$f(x-\eta f'(x)) \lesssim f(x)$$

Therefore: $x \leftarrow x - \eta f'(x)$

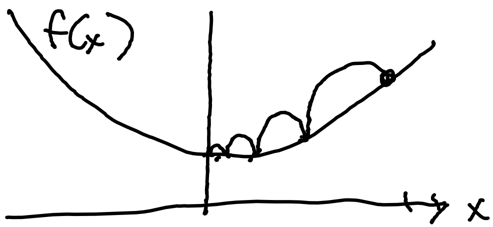




small η



big η



"just right" η

In the multivariate

$$\nabla_x f(x) = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_n}, \dots\right]^T$$

vector

$$f(x+\varepsilon) = f(x) + \varepsilon^T \nabla f(x) + O(\|\varepsilon\|^2)$$

$$x \leftarrow x - \eta \nabla f(x)$$

scalar

In ML, we have $L(y^{(i)}, x^{(i)}, \theta) = L_i(\theta)$

we want to minimize L by changing θ

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n} L_i(\theta)$$

$$\nabla_\theta L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta L_i(\theta)$$

"Batch" gradient descent

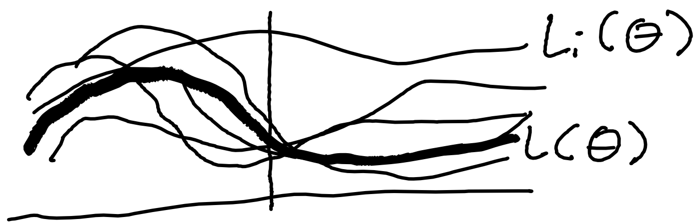
Stochastic gradient descent:

$$\Theta \leftarrow \Theta - \eta \nabla_\Theta L_i(\Theta)$$

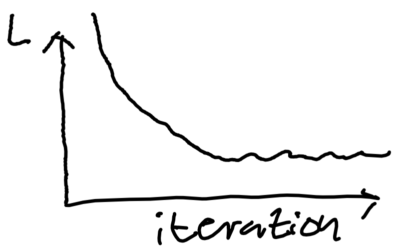
$$i \sim \text{uniform categorical}(n)$$

$$\mathbb{E}_i[\nabla_\Theta L_i(\Theta)] = \frac{1}{n}\sum_{i=1}^{n} \nabla_\Theta L_i(\Theta)$$

We are optimizing a different function at each iteration.



Progress can stall as we approach minimum of $L(\Theta)$



One option: Use a "learning rate schedule"

→ different η at each iteration t

$\eta(t) = \eta$ (constant)

$\eta(t) = \eta_i$ if $t_i \leq t \leq t_{i+1}$ (piecewise)

$\eta(t) = \eta_0 e^{-\lambda t}$ (exponential)

Another option: Use a minibatch.

Sample B examples randomly and use their average gradient.

$$\theta \leftarrow \theta - \eta \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta L_i(\theta)$$

Note: SGD can be more efficient BGD if $\nabla_\theta L_i$ is generally aligned with $\nabla_\theta L(\theta)$

Note: SGD can be noisy.

Note: SGD is less parallelizable.

Why might it be a bad idea to use the same η for every parameter?

ex/ $\hat{y} = w^T x$

$$L(y, \hat{y}) = \frac{1}{2}\|y - \hat{y}\|^2$$

$$\frac{dL}{d\hat{y}} = \hat{y} - y$$

$$\frac{dL}{dw} = (\hat{y} - y)x$$

Imagine that $|x_i| \gg |x_j|$ e.g.

$x_i \sim N(0, \sigma_i)$ $x_j \sim N(0, \sigma_j)$ with $\sigma_i \gg \sigma_j$

Then we will often have $\left|\frac{\partial L}{\partial w_i}\right| \gg \left|\frac{\partial L}{\partial w_j}\right|$