Logistic Regression & Multinomial Logistic Regression

Suleyman Demirel University

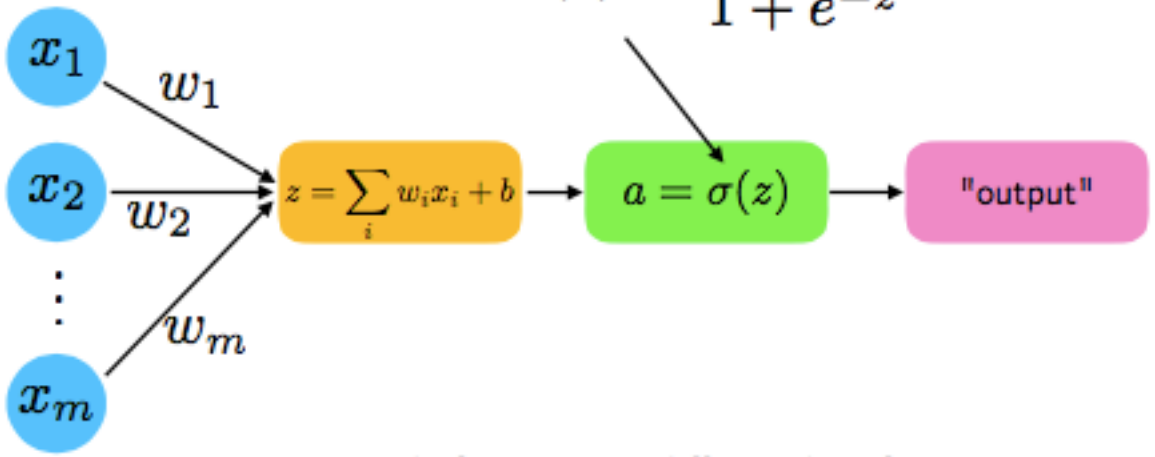
CSS634: Deep Learning

PhD Abay Nussipbekov

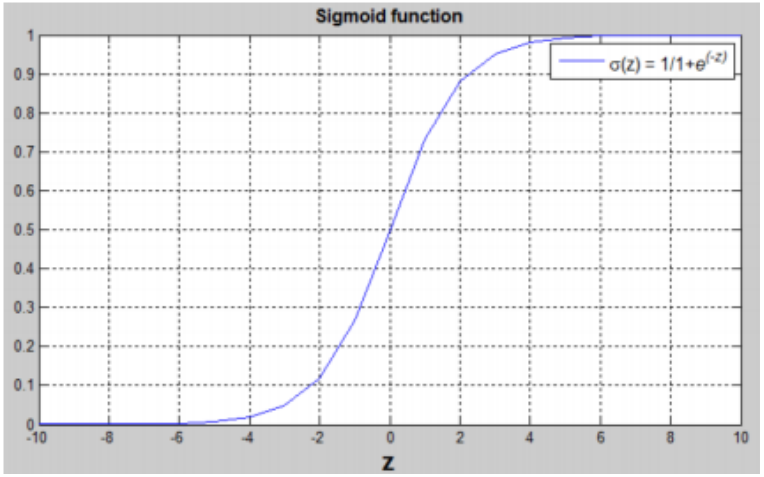
Logistic Regression

For binary classes $y \in \{0, 1\}$

"logistic sigmoid" $\sigma(z) = \dfrac{1}{1 + e^{-z}}$



Sigmoid Function



Some observations from the graph:

If z is a large positive number, then $\sigma(z) = 1$

If z is small or large negative number, then $\sigma(z) = 0$

If $z = 0$, then $\sigma(z) = 0$

Logistic Regression

$$h(x) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$

$$P(y|\mathbf{x}) = \begin{cases} h(\mathbf{x}) \;\; if \; y = 1 \\ 1 - h(\mathbf{x}) \;\; if \; y = 0 \end{cases}$$

want $P(y = 0|\mathbf{x}) \approx 1 \; if \; y = 0$

$$P(y = 1|\mathbf{x}) \approx 1 \; if \; y = 1$$

Logistic Regression

$$h(x) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$

can be rewrited as

$$P(y|\mathbf{x}) = \begin{cases} h(\mathbf{x}) \;\; if \; y = 1 \\ 1 - h(\mathbf{x}) \;\; if \; y = 0 \end{cases}$$

want $P(y = 0|\mathbf{x}) \approx 1 \; if \; y = 0$

$$P(y = 1|\mathbf{x}) \approx 1 \; if \; y = 1$$

$$P(y|\mathbf{x}) = a^{y}(1 - a)^{(1-y)}$$

Logistic Regression

For multiple training examples, we want to maximize:

$$P\left(y^{(i)}, \ldots, y^{(n)} \middle| \mathbf{x}^{(i)}, \ldots, \mathbf{x}^{(n)}; \mathbf{w}\right) = \prod_{i=1}^{n} P(y^{(i)} | \mathbf{x}^{(i)})$$

What is the name of this?

Logistic Regression

For multiple training examples, we want to maximize:

$$P\left(y^{(i)}, \ldots, y^{(n)} \middle| \mathbf{x}^{(i)}, \ldots, \mathbf{x}^{(n)}; \mathbf{w}\right) = \prod_{i=1}^{n} P(y^{(i)} | \mathbf{x}^{(i)})$$

Maximum likelihood estimation

Likelihood “Loss”

$$L(\mathbf{w}) = P(y|\mathbf{x};\mathbf{w})$$

$$= \prod_{i=1}^{n} P(y^{(i)}|x^{(i)};\mathbf{w})$$

$$= \prod_{i=1}^{n} (\sigma(z^{(i)}))^{y^{(i)}} (1-\sigma(z^{(i)}))^{1-y^{(i)}}$$

Log-Likelihood "Loss"

$$L(\mathbf{w}) = P(y|\mathbf{x};\mathbf{w})$$
$$= \prod_{i=1}^{n} P(y^{(i)}|x^{(i)};\mathbf{w})$$
$$= \prod_{i=1}^{n} (\sigma(z^{(i)}))^{y^{(i)}} (1-\sigma(z^{(i)}))^{1-y^{(i)}}$$

In practice, it is easier to maximize the natural log of this equation, which is called log-likelihood:

$$l(w) = \log L(\mathbf{w})$$
$$= \sum_{i=1}^{n} [y^{(i)} \log\left(\sigma\left(z^{(i)}\right)\right) + \left(1-y^{(i)}\right) \log(1-\sigma(z^{(i)}))]$$

Negative Log-Likelihood Loss

In practice, it is even more convenient to minimize negative log-likelihood instead of maximizing log-likelihood:

Sometimes J

$$\mathcal{L}(\mathbf{w}) = -l(\mathbf{w})$$
$$= -\sum_{i=1}^{n}[y^{(i)} \log\left(\sigma\left(z^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log(1 - \sigma(z^{(i)}))]$$

(in code, we also usually add a $1/n$ scaling factor for further convenience, where n is the number of training examples or number of examples in a minibatch)

Loss function sometimes can be written as cost function or objective function
Sometimes researches use term "loss" to indicate it for one example and "cost" for multiple

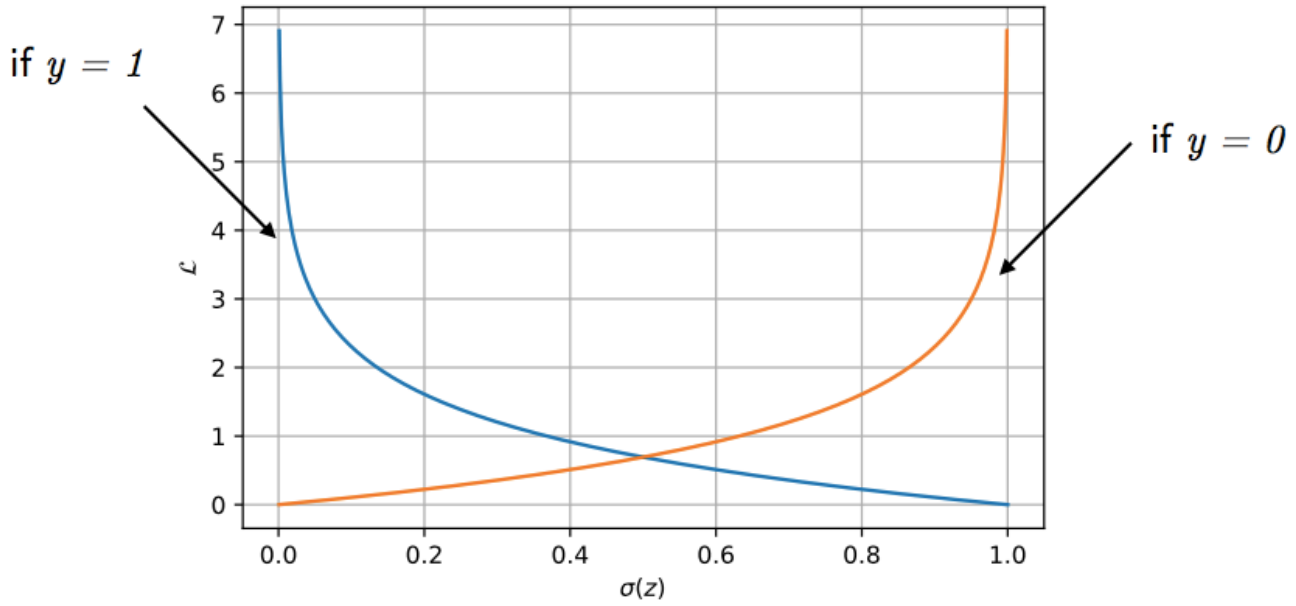
Logistic Regression Loss

- In ADALINE/linear regression we minimized the MSE (Mean Squared Error/L2) loss:

$$\text{MSE} = \frac{1}{n}\sum_{i}(a^{(i)} - y^{(i)})^2$$

- In Logistic Regression we maximize the likelihood
- Maximizing likelihood is the same as maximizing the log-likelihood, but the latter is numerically more stable
- Maximizing the log-likelihood is the same as minimizing the negative log-likelihood, which is convenient, so we don't have to change our code and can still use gradient descent (instead of gradient ascent)

Loss for a Single Training Example



$$\mathcal{L}(\mathbf{w}) = -y^{(i)} \log\left(\sigma\left(z^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(z^{(i)}\right)\right)$$

About the Term “Logits”

- "Logits" is a very commonly used Deep Learning jargon; probably inspired by the logits for logistic regression
- In deep learning, the logits are the net inputs of the the last neuron layer
- In logistic regression, the logits are naturally $w^T \mathrm{x}$ ...

About the Term "Binary Cross Entropy"

- Negative log-likelihood and binary cross entropy are equivalent
- Cross entropy comes from the "information theory" (computer science) perspective

$$H_{\mathbf{a}}(\mathbf{y}) = -\sum_{i} \left(y^{[i]} \log(a^{[i]}) + (1 - y^{[i]}) \log(1 - a^{[i]}) \right)$$

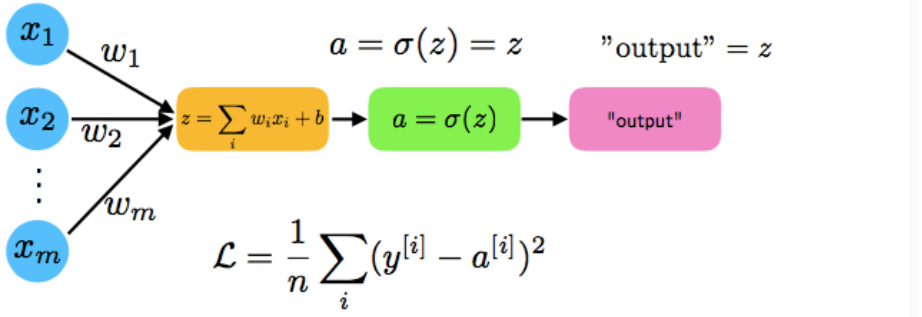
Binary Cross Entropy

$$H_{\mathbf{a}}(\mathbf{y}) = \sum_{i=1}^{n} \sum_{k=1}^{K} -y_k^{[i]} \log \left(a_k^{[i]} \right)$$

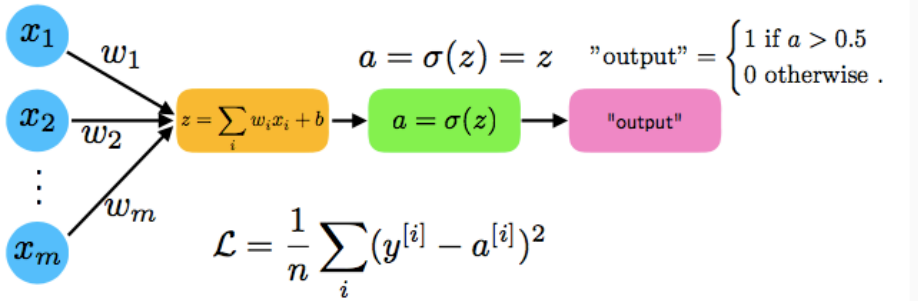
(Multi-category) Cross Entropy
for K different class labels

Perceptron vs. Adaline vs. Logistic Regression

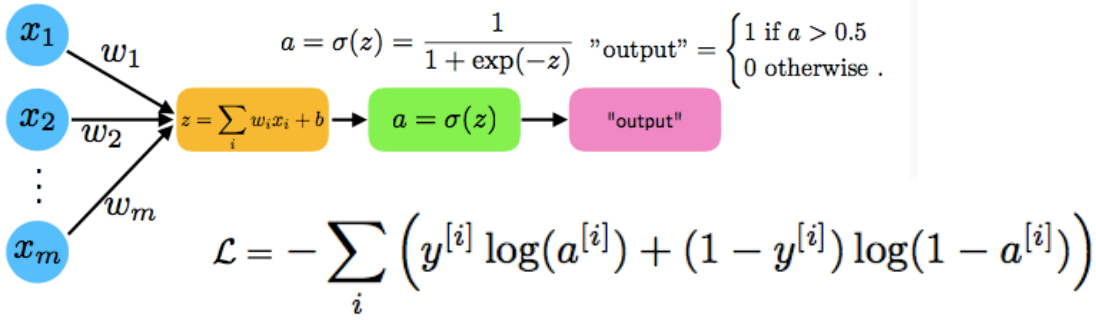
Linear regression



Adaline



Logistic Regression

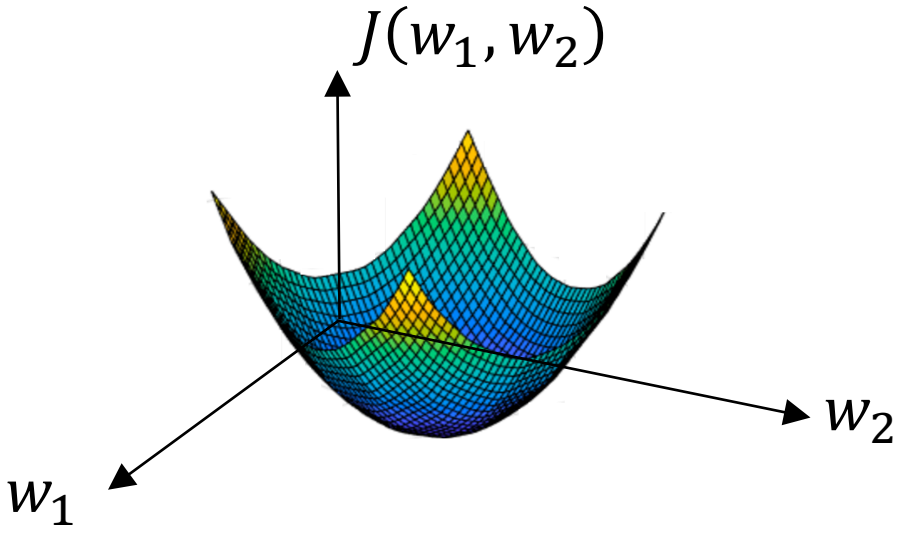


Optimization

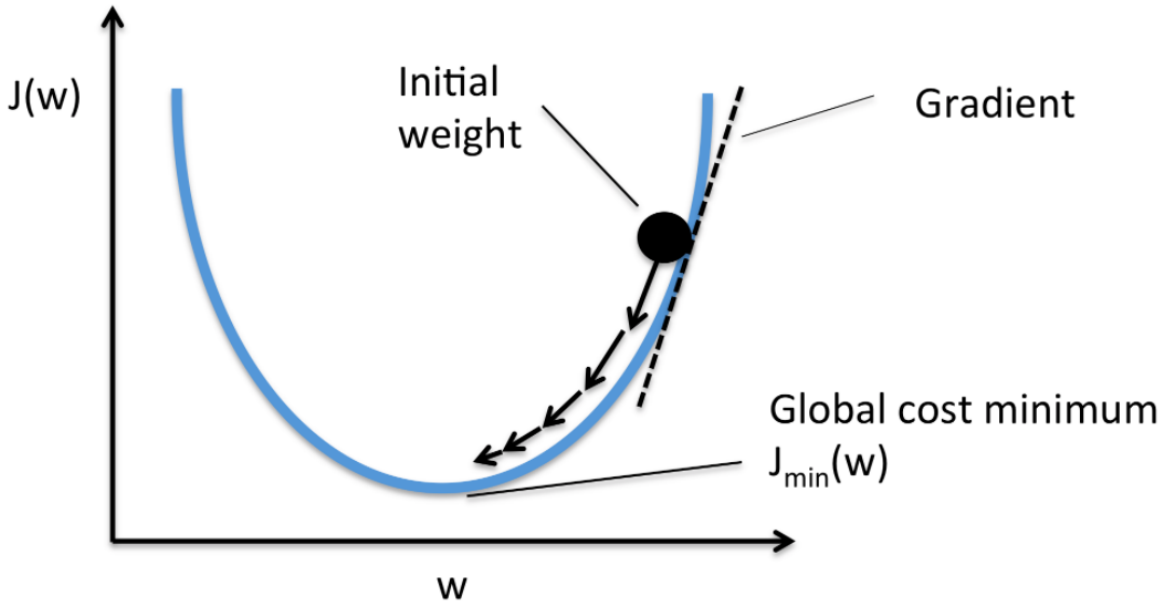
Optimization: Follow the Slope

Gradient Descent

- Cost function
- $J(w_1, w_2) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right) = -\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \log\left(\hat{y}^{(i)}\right) + (1-y)\log(1-\hat{y}^{(i)})$



Gradient Descent



repeat {

$$w := w - \alpha \frac{dJ(w)}{dw}$$

}

Gradient Descent

In 1-dimension, the derivative of a function:

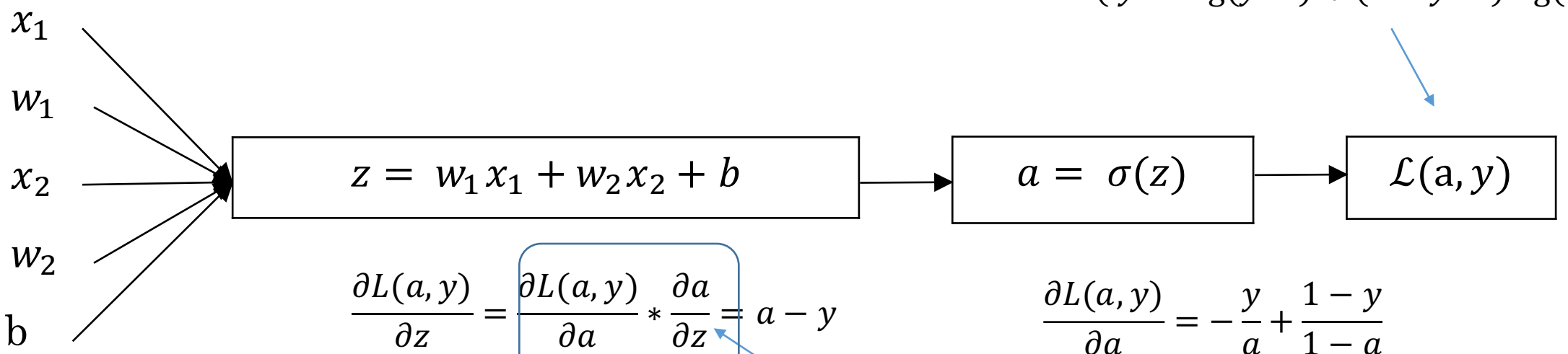
$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

In multiple dimensions, the gradient is the vector of (partial derivatives) along each dimension

The slope in any direction is the dot product of the direction with the gradient

The direction of steepest descent is the negative gradient

Computational graph



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

$$\frac{\partial L(a,y)}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$$

$a(1-a)$

Chain rule!

How much we should change w1, w2 and b?

$$\frac{\partial L(a,y)}{\partial w_1} = \frac{\partial L(a,y)}{\partial z} * \frac{\partial z}{\partial w_1} = \frac{\partial L(a,y)}{\partial z} x_1$$

$$\frac{\partial L(a,y)}{\partial w_2} = \frac{\partial L(a,y)}{\partial z} * \frac{\partial z}{\partial w_2} = \frac{\partial L(a,y)}{\partial z} x_2$$

$$\frac{\partial L(a,y)}{\partial b} = \frac{\partial L(a,y)}{\partial z} * \frac{\partial z}{\partial b} = \frac{\partial L(a,y)}{\partial z}$$

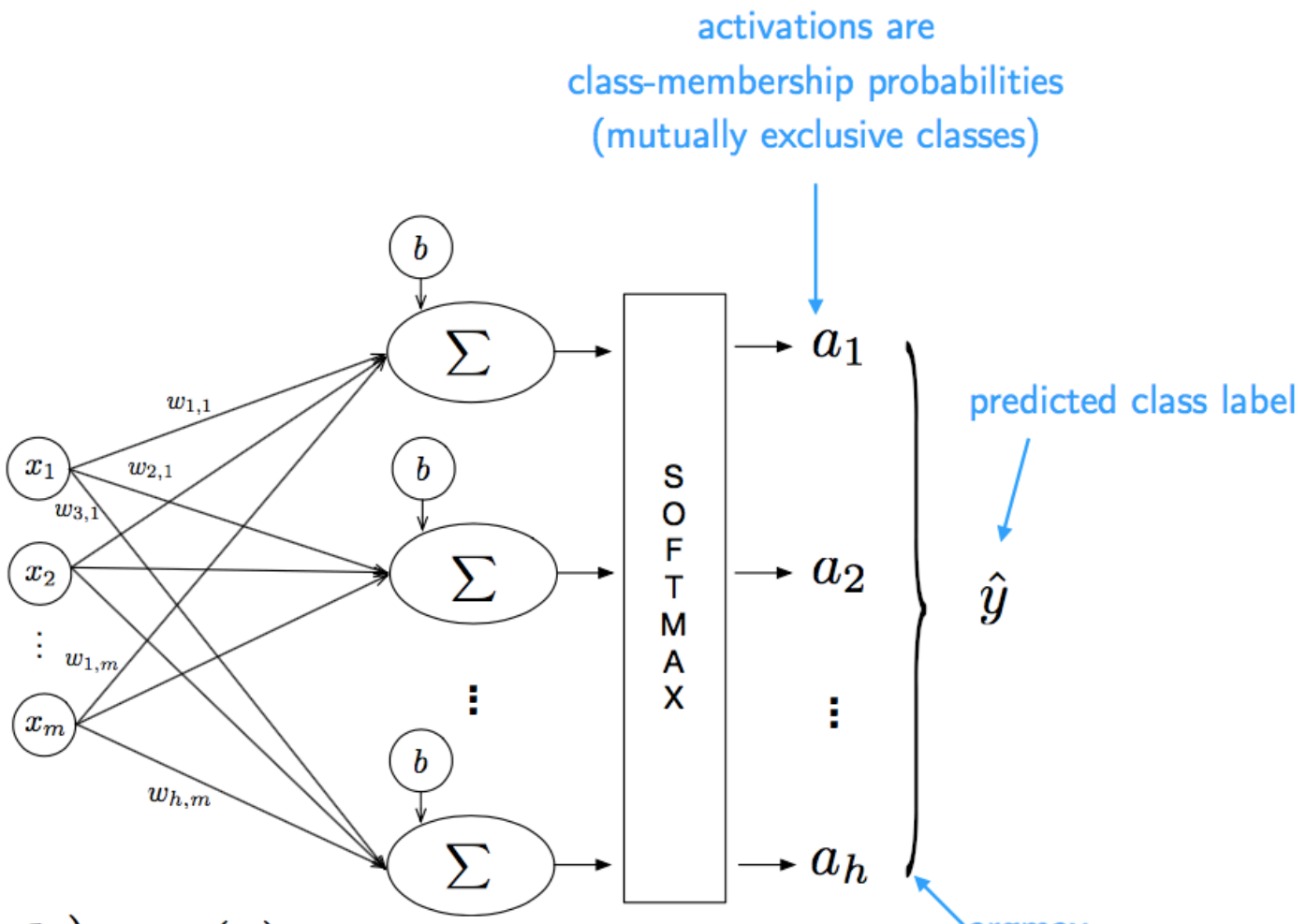
Make updates:

$$w_1 := w_1 - \alpha \frac{\partial L(a,y)}{\partial w_1}$$

$$w_2 := w_2 - \alpha \frac{\partial L(a,y)}{\partial w_2}$$

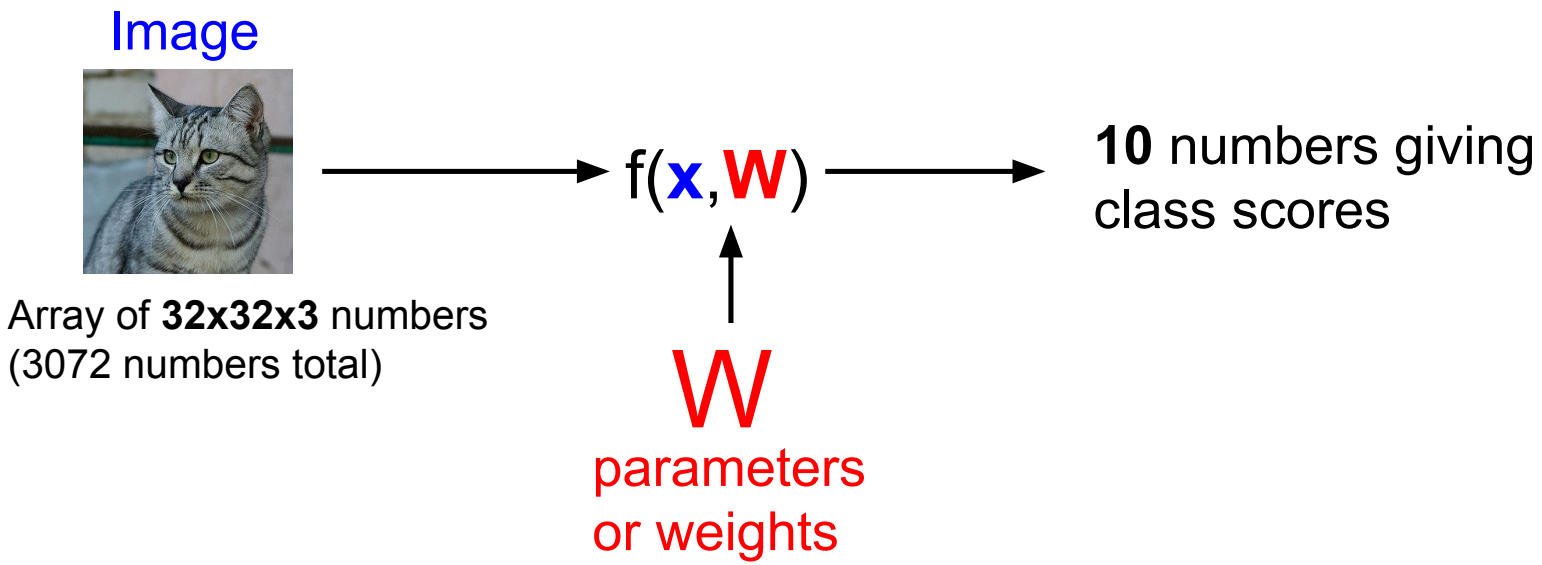
$$b := b - \alpha \frac{\partial L(a,y)}{\partial b}$$

Multinomial Logistic Regression/ Softmax Regression

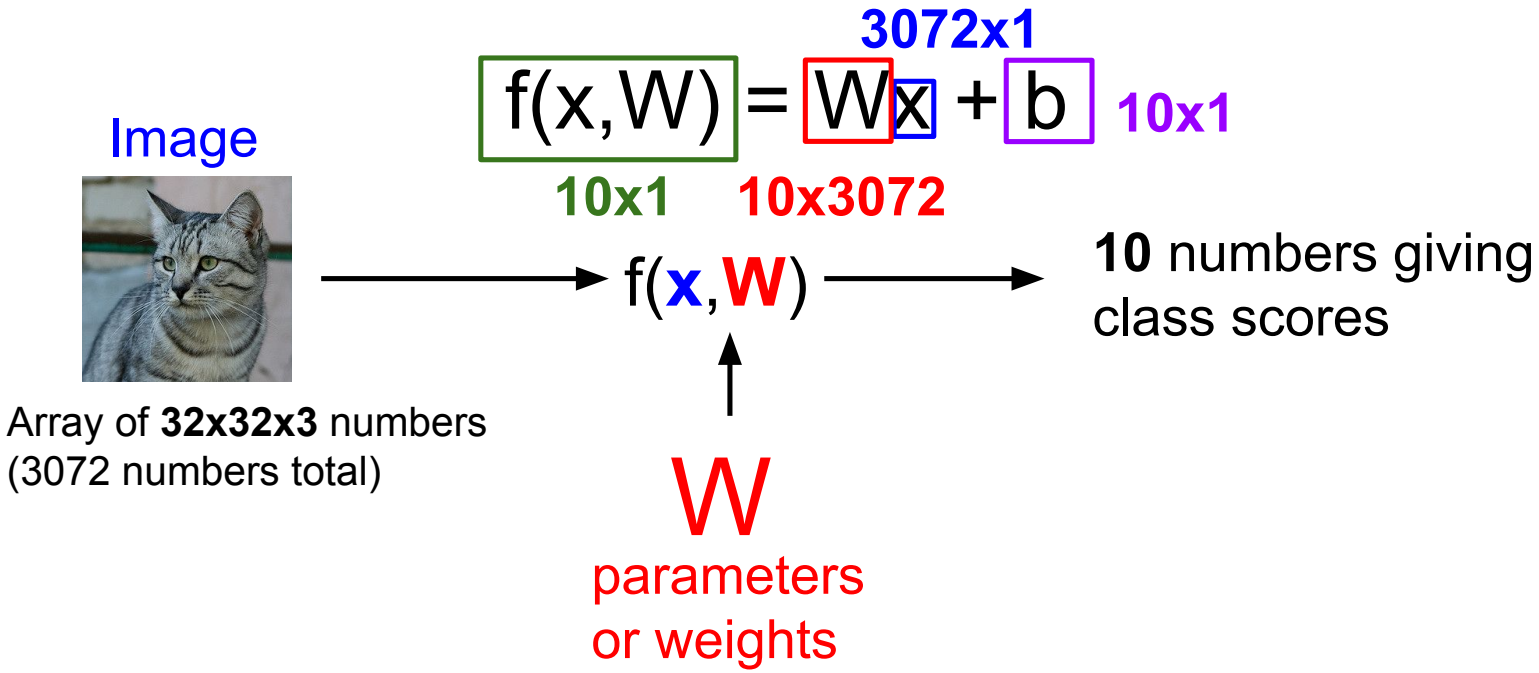


$$\sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) = \sigma(\mathbf{z}) = \mathbf{a}$$

Multinomial Logistic Regression/ Softmax Regression

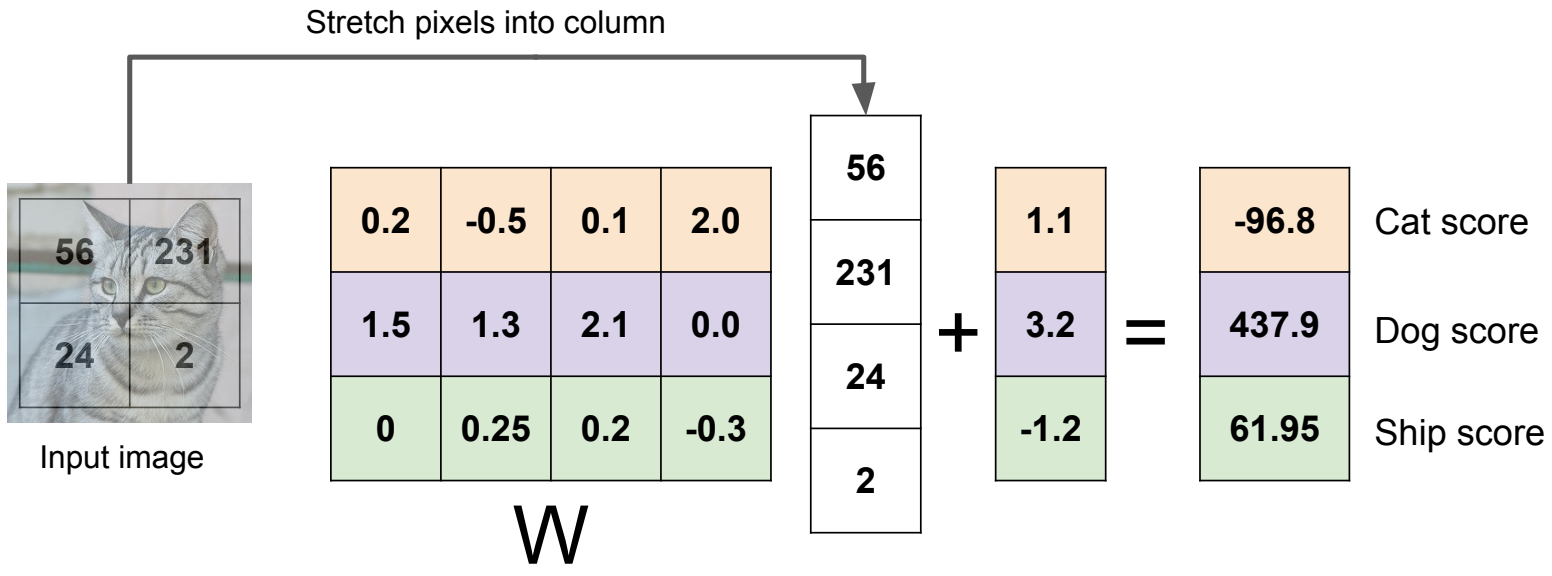


Multinomial Logistic Regression/ Softmax Regression



Multinomial Logistic Regression/ Softmax Regression

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)



Multinomial Logistic Regression/ Softmax Regression

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

cat	**3.2**
car	5.1
frog	-1.7

Multinomial Logistic Regression/ Softmax Regression

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

cat	**3.2**
car	5.1
frog	-1.7

Multinomial Logistic Regression/ Softmax Regression

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

cat	**3.2**	exp →	**24.5**
car	5.1		164.0
frog	-1.7		0.18

unnormalized probabilities

Multinomial Logistic Regression/ Softmax Regression

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

		exp		normalize	
cat	**3.2**	→	**24.5**	→	**0.13**
car	5.1	→	164.0	→	0.87
frog	-1.7	→	0.18	→	0.00

unnormalized probabilities

probabilities

Multinomial Logistic Regression/ Softmax Regression

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

	Unnormalized log-probabilities / logits		unnormalized probabilities		probabilities
cat	**3.2**	exp →	**24.5**	normalize →	**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

Multinomial Logistic Regression/ Softmax Regression

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i|X = x_i)$$

	Unnormalized log-probabilities / logits	exp	unnormalized probabilities	normalize	probabilities
cat	3.2		24.5		0.13
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

→ L_i = -log(0.13) = **2.04**

Multinomial Logistic Regression/ Softmax Regression

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i|X = x_i)$$

	Unnormalized log-probabilities / logits	exp	unnormalized probabilities	normalize	probabilities
cat	**3.2**		**24.5**		**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

→ L_i = -log(0.13) = **2.04**

Maximum Likelihood Estimation
Choose weights to maximize the likelihood of the observed data
(See CS 229 for details)

Multinomial Logistic Regression/ Softmax Regression

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i|X = x_i)$$

	Unnormalized log-probabilities / logits	exp	unnormalized probabilities	normalize	probabilities	compare	Correct probs
cat	**3.2**		**24.5**		**0.13**		**1.00**
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Multinomial Logistic Regression/ Softmax Regression

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i|X = x_i)$$

	Unnormalized log-probabilities / logits	exp	unnormalized probabilities	normalize	probabilities	compare	Correct probs
cat	**3.2**		**24.5**		**0.13**		**1.00**
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Kullback–Leibler divergence

$$D_{KL}(P||Q) = \sum_y P(y) \log \frac{P(y)}{Q(y)}$$

Multinomial Logistic Regression/ Softmax Regression

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i|X = x_i)$$

	Unnormalized log-probabilities / logits	exp	unnormalized probabilities	normalize	probabilities	compare	Correct probs
cat	**3.2**		**24.5**		**0.13**		**1.00**
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Cross Entropy

$$H(P, Q) = H(p) + D_{KL}(P||Q)$$

Cross Entropy

For single sample

S(Y)
0.7
0.2
0.1

$$D(S, L) = -\sum_i L_i \log(S_i)$$

L
1
0
0

Onehot Encoding Needed

class labels
0
1
3
2

class_0	class_1	class_2	class_3
1	0	0	0
0	1	0	0
0	0	0	1
0	0	1	0

Loss Function

Multiple samples

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{j=1}^{k} -y_j^{(i)} \log(a_j^{(i)})$$

(Multiple-category) Cross Entropy for k different class labels

This assumes one-hot encoded labels!

Loss Function

$$\mathcal{L}_{\text{binary}} = -\sum_{i=1}^{n} \left(y^{(i)} \log\left(a^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - a^{(i)}\right)\right)$$

This assumes one-hot encoded labels!

$$\mathcal{L}_{\text{multiclass}} = \sum_{i=1}^{n} \sum_{j=1}^{k} -y_j^{(i)} \log\left(a_j^{(i)}\right)$$

(Multiple-category) Cross Entropy for k different class labels

Cross Entropy Loss Function Example

$$\mathbf{Y}_{\text{onehot}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{A}_{\text{softmax outputs}} = \begin{bmatrix} 0.3792 & 0.3104 & 0.3104 \\ 0.3072 & 0.4147 & 0.2780 \\ 0.4263 & 0.2248 & 0.3490 \\ 0.2668 & 0.2978 & 0.4354 \end{bmatrix}$$

(4 training examples, 3 classes)

Cross Entropy Loss Function Example

1 training example

$$\mathbf{Y}_{\text{onehot}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{A}_{\text{softmax outputs}} = \begin{bmatrix} 0.3792 & 0.3104 & 0.3104 \\ 0.3072 & 0.4147 & 0.2780 \\ 0.4263 & 0.2248 & 0.3490 \\ 0.2668 & 0.2978 & 0.4354 \end{bmatrix}$$

(4 training examples, 3 classes)

$$\mathcal{L}_{\text{multiclass}} = \sum_{i=1}^{n} \sum_{j=1}^{k} -y_j^{[i]} \log\left(a_k^{[i]}\right)$$

$$\begin{aligned} \mathcal{L}^{[1]} &= [(-1) \cdot \log(0.3792)] \\ &\quad + [(-0) \cdot \log(0.3104)] \\ &\quad + [(-0) \cdot \log(0.3104)] \\ &\quad = 0.969692... \end{aligned}$$

Cross Entropy Loss Function Example

$$\mathbf{Y}_{\text{onehot}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{A}_{\text{softmax outputs}} = \begin{bmatrix} 0.3792 & 0.3104 & 0.3104 \\ 0.3072 & 0.4147 & 0.2780 \\ 0.4263 & 0.2248 & 0.3490 \\ 0.2668 & 0.2978 & 0.4354 \end{bmatrix}$$

$$\begin{aligned} \mathcal{L}^{[1]} &= [(-1) \cdot \log(0.3792)] \\ &+ [(-0) \cdot \log(0.3104)] \\ &+ [(-0) \cdot \log(0.3104)] \\ &= 0.969692... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[2]} &= [(-0) \cdot \log(0.3072)] \\ &+ [(-1) \cdot \log(0.4147)] \\ &+ [(-0) \cdot \log(0.2780)] \\ &= 0.880200... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[3]} &= [(-0) \cdot \log(0.4263)] \\ &+ [(-0) \cdot \log(0.2248)] \\ &+ [(-1) \cdot \log(0.3490)] \\ &= 1.05268... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[4]} &= [(-0) \cdot \log(0.2668)] \\ &+ [(-0) \cdot \log(0.2978)] \\ &+ [(-1) \cdot \log(0.4354)] \\ &= 0.831490... \end{aligned}$$

Cross Entropy Loss Function Example

$$\mathbf{Y}_{\text{onehot}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{A}_{\text{softmax outputs}} = \begin{bmatrix} 0.3792 & 0.3104 & 0.3104 \\ 0.3072 & 0.4147 & 0.2780 \\ 0.4263 & 0.2248 & 0.3490 \\ 0.2668 & 0.2978 & 0.4354 \end{bmatrix}$$

$$\begin{aligned} \mathcal{L}^{[1]} &= [(-1) \cdot \log(0.3792)] \\ &+ [(-0) \cdot \log(0.3104)] \\ &+ [(-0) \cdot \log(0.3104)] \\ &= 0.969692... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[2]} &= [(-0) \cdot \log(0.3072)] \\ &+ [(-1) \cdot \log(0.4147)] \\ &+ [(-0) \cdot \log(0.2780)] \\ &= 0.880200... \end{aligned}$$

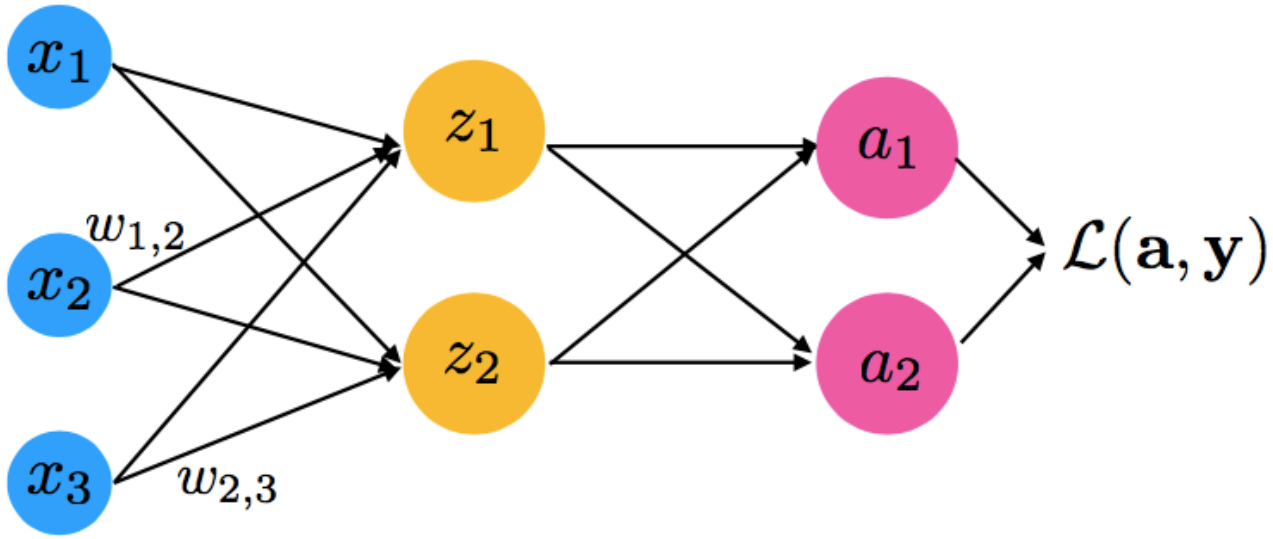
$$\begin{aligned} \mathcal{L}^{[3]} &= [(-0) \cdot \log(0.4263)] \\ &+ [(-0) \cdot \log(0.2248)] \\ &+ [(-1) \cdot \log(0.3490)] \\ &= 1.05268... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[4]} &= [(-0) \cdot \log(0.2668)] \\ &+ [(-0) \cdot \log(0.2978)] \\ &+ [(-1) \cdot \log(0.4354)] \\ &= 0.831490... \end{aligned}$$

$$\mathcal{L}_{\text{multiclass}} = \sum_{i=1}^{n} \sum_{j=1}^{k} -y_j^{[i]} \log\left(a_k^{[i]}\right)$$

$$\approx 0.9335$$

Softmax Regression Backward Propagation



Multivariable chain rule

$$\frac{\partial L}{\partial w_{1,2}} = \frac{\partial L}{\partial a_1}\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}} + \frac{\partial L}{\partial a_2}\frac{\partial a_2}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}}$$

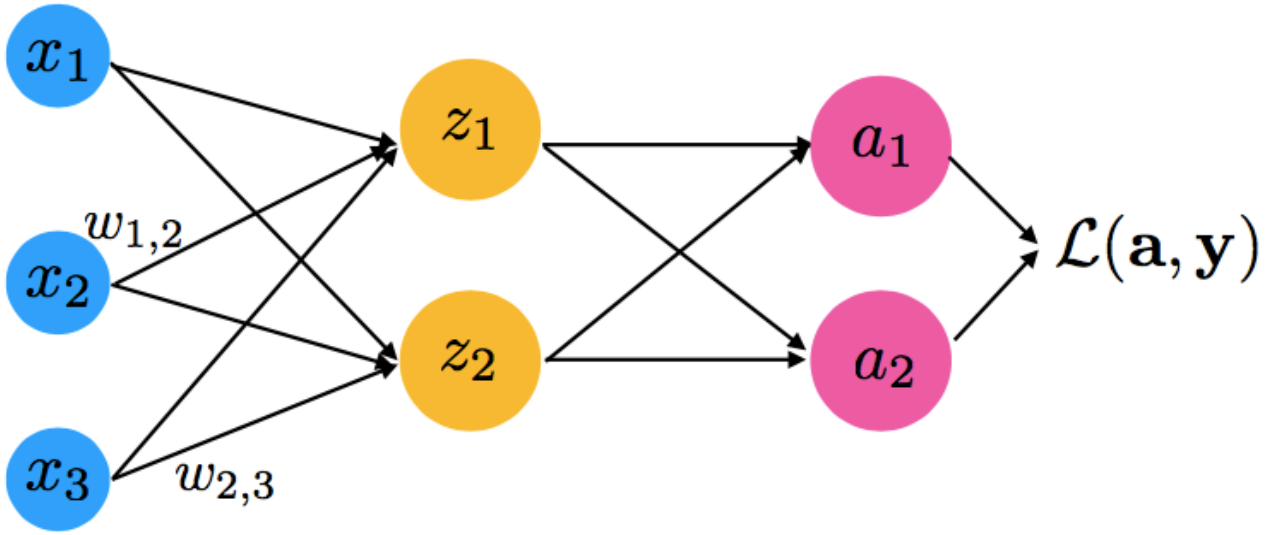
$$= \frac{-y_1}{a_1}[a_1(1-a_1)]x_2 + \frac{-y_2}{a_2}(-a_2 a_1)x_2$$

$$= (y_2 a_1 - y_1 + y_1 a_1)x_2$$

$$= (a_1(y_1 + y_2) - y_1)x_2$$

$$= -(y_1 - a_1)x_2$$

Softmax Regression Backward Propagation



$$\frac{\partial L}{\partial w_{1,2}} = \frac{\partial L}{\partial a_1}\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}} + \frac{\partial L}{\partial a_2}\frac{\partial a_2}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}}$$
$$= \frac{-y_1}{a_1}[a_1(1-a_1)]x_2 + \frac{-y_2}{a_2}(-a_2 a_1)x_2$$
$$= (y_2 a_1 - y_1 + y_1 a_1)x_2$$
$$= (a_1(y_1 + y_2) - y_1)x_2$$
$$= -(y_1 - a_1)x_2$$

Vectorized Form:

$$\nabla_{\mathbf{W}}\mathcal{L} = -(\mathbf{X}^\top(\mathbf{Y} - \mathbf{A}))^\top$$

where $\mathbf{W} \in \mathbb{R}^{k \times m}$

$\mathbf{X} \in \mathbb{R}^{n \times m}$

$\mathbf{A} \in \mathbb{R}^{n \times k}$

$\mathbf{Y} \in \mathbb{R}^{n \times k}$

Batch vs. Stochastic (Online) vs. Mini-Batch Gradient Descent

The minibatch and on-line modes are stochastic versions of gradient descent (batch mode)

Minibatch mode

(mix between on-line and batch)

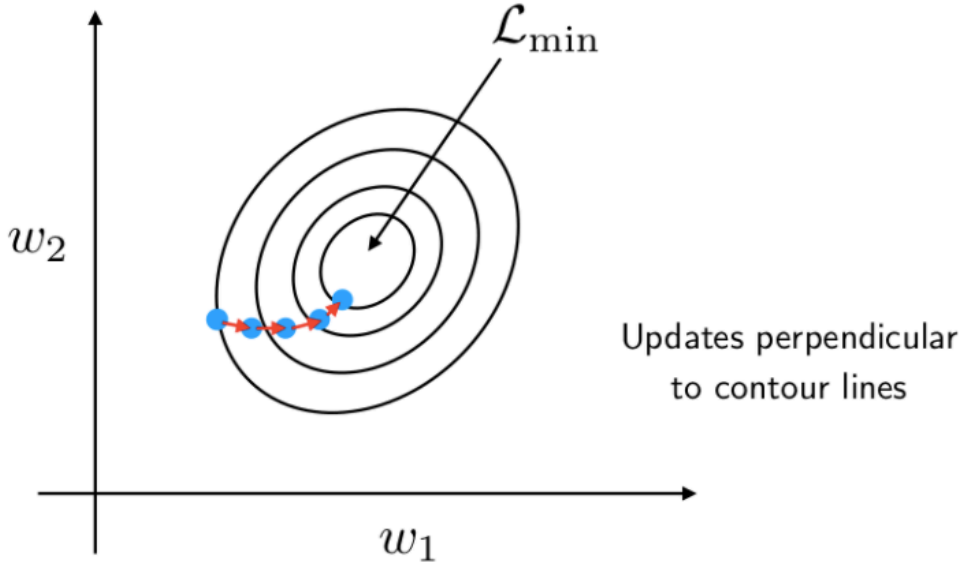
1. Initialize $\mathbf{w} := 0^{m-1}$, $\mathbf{b} := 0$
2. For every training epoch:
   - A. Initialize $\Delta\mathbf{w} := 0,\ \Delta b := 0$
   - B. For every $\{\langle \mathbf{x}^{[i]}, y^{[i]} \rangle, ..., \langle \mathbf{x}^{[i+k]}, y^{[i+k]} \rangle\} \subset \mathcal{D}$:
     - (a) Compute output (prediction)
     - (b) Calculate error
     - (c) Update $\Delta\mathbf{w}, \Delta b$
   - C. Update $\mathbf{w}, b$:

$$\mathbf{w} := \mathbf{w} + \Delta\mathbf{w}, b := +\Delta b$$

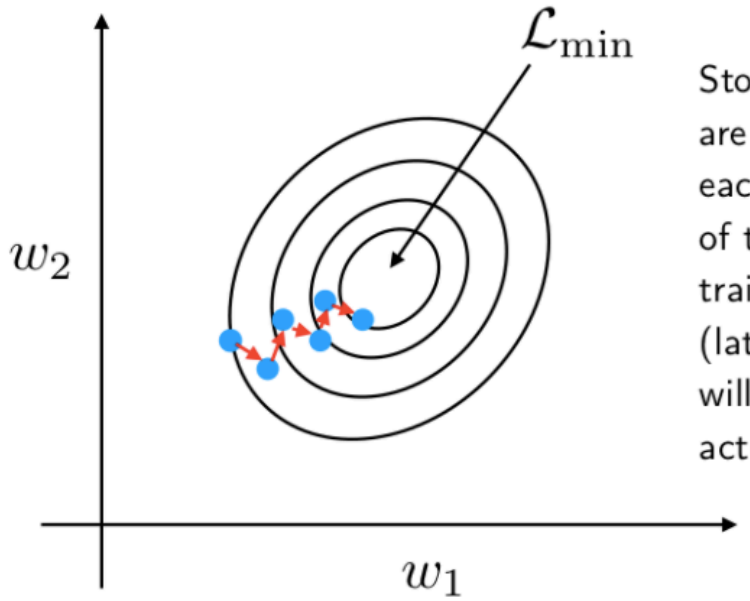
Most commonly used in DL, because

1. Choosing a subset (vs 1 example at a time) takes advantage of vectorization (faster iteration through epoch than on-line)
2. having fewer updates than "on-line" makes updates less noisy
3. makes more updates/ epoch than "batch" and is thus faster

Batch vs. Stochastic (Online) vs. Mini-Batch Gradient Descent



Updates perpendicular to contour lines



Stochastic updates are a bit noisier, because each batch is an approximation of the overall loss on the training set (later, in deep neural nets, we will see why noisier updates are actually helpful)

Resources Used

- CS231n Convolutional Neural Networks for Visual Recognition by Fei-Fei Li, Justin Johnson, Seran Yeung
- Deeplearningbook by Ian Godfellow
- CMSC 35246: Deep Learning by Shubhendu Trivedi and Risi Kondor
- STAT 479: Deep Learning by Sebastian Raschka