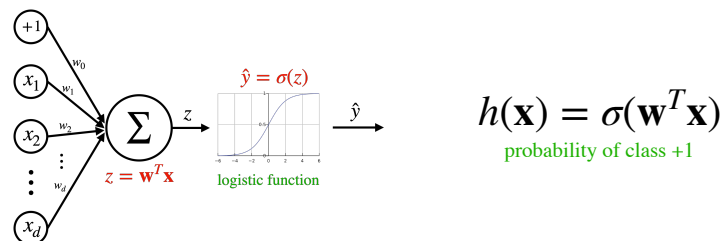


Multinomial logistic regression

Prof. Marco Alvarez, Computer Science
University of Rhode Island

Logistic regression



cross-entropy loss

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^n \log \left(1 + e^{-y^{(i)} \mathbf{w}^T \mathbf{x}^{(i)}} \right)$$

Gradient (partial derivatives)

$$L(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n \log \left(1 + e^{-y^{(i)} \mathbf{w}^T \mathbf{x}^{(i)}} \right)$$

$$\frac{\partial L(\mathbf{w})}{\partial w_j} = -\frac{1}{n} \sum_{i=1}^n \sigma \left(-y^{(i)} \mathbf{w}^T \mathbf{x}^{(i)} \right) y^{(i)} x_j^{(i)}$$

$$\nabla_{\mathbf{w}} L(\mathbf{w}) = \left[\frac{\partial L(\mathbf{w})}{\partial w_0}, \dots, \frac{\partial L(\mathbf{w})}{\partial w_d} \right]$$

Handling multiple classes

MNIST

- ▶ The MNIST database is a large database of handwritten digits

- contains 60,000 training images and 10,000 testing images
- convolutional neural networks, manages to get an error rate of 0.23%
- original paper reports an error rate of 0.8% with SVMs



<http://yann.lecun.com/exdb/mnist/>

Multinomial logistic regression

- ▶ Data

$$\mathbf{x} \in \mathbb{R}^d, \quad y \in \{1, 2, \dots, C\}$$

- ▶ Binary logistic regression

$$P(y = +1 \mid \mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}} = \frac{e^{\mathbf{w}^T \mathbf{x}}}{e^{\mathbf{w}^T \mathbf{x}} + 1}$$

- ▶ Multiclass logistic regression

$$P(y = c \mid \mathbf{x}; W) = \frac{e^{\mathbf{w}_c^T \mathbf{x}}}{\sum_{k=1}^C e^{\mathbf{w}_k^T \mathbf{x}}}$$

softmax function

$W_{C \times d+1}$ is a matrix where every row is a "class" weight vector

Softmax

$$\sigma(\mathbf{z}) = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$$

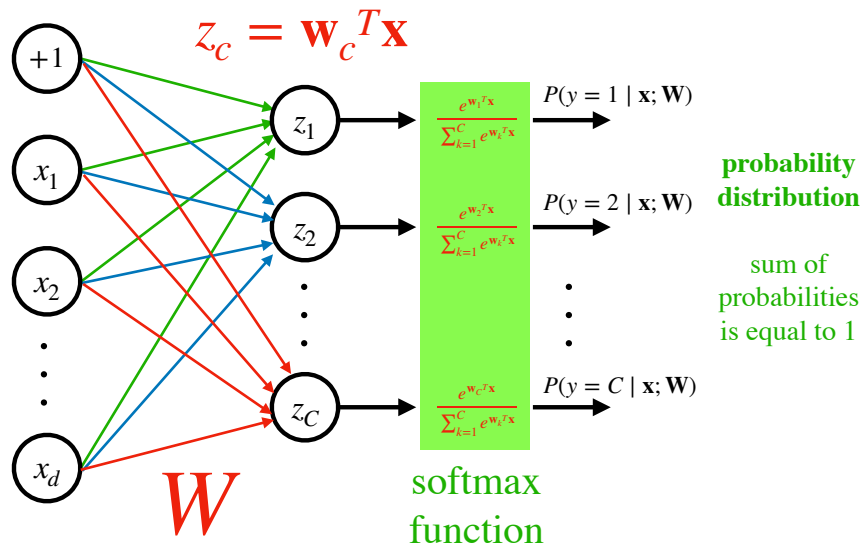
for $i = 1, \dots, K$ and $\mathbf{z} \in \mathbb{R}^K$

converts a vector of K real numbers into a **probability distribution** of K possible outcomes

Practice

- ▶ What is the value of $\text{softmax}(\mathbf{z})$, given that $\mathbf{z}^T = [-10, 10, 5, 4.3, 7]$?

Multinomial logistic regression



Multinomial logistic regression

- Use the **softmax function** for activation
- Predict the label with the highest probability score

$$\hat{y} = \arg \max_c P(y = c | \mathbf{x}; \mathbf{W})$$

- How to learn the weights?
 - need to define a **Loss Function** ... then apply **gradient descent**
 - loss function can be derived using **MLE** (similar to binary logistic regression)

Applying MLE

$$\mathbf{W}^* = \arg \max_{\mathbf{W}} \frac{1}{n} \prod_{i=1}^n P(y^{(i)} | \mathbf{x}^{(i)}; \mathbf{W})$$

$$= \arg \max_{\mathbf{W}} \frac{1}{n} \prod_{i=1}^n \prod_{c=1}^C P(y^{(i)} = c | \mathbf{x}^{(i)}; \mathbf{W})^{t_{i,c}}$$

$$= \arg \max_{\mathbf{W}} \frac{1}{n} \sum_{i=1}^n \sum_{c=1}^C t_{i,c} \log (P(y^{(i)} = c | \mathbf{x}^{(i)}; \mathbf{W}))$$

$$= \arg \min_{\mathbf{W}} - \frac{1}{n} \sum_{i=1}^n \sum_{c=1}^C t_{i,c} \log (P(y^{(i)} = c | \mathbf{x}^{(i)}; \mathbf{W}))$$

$$= \arg \min_{\mathbf{W}} - \frac{1}{n} \sum_{i=1}^n \sum_{c=1}^C t_{i,c} \log \left(\frac{e^{\mathbf{w}_c^T \mathbf{x}^{(i)}}}{\sum_{k=1}^C e^{\mathbf{w}_k^T \mathbf{x}^{(i)}}} \right)$$

individual loss
 $e(h_{\mathbf{w}}(\mathbf{x}), y) = -\log p_c$
 predicted probability of the correct class

Cross-entropy loss

Consider a matrix $t_{n \times C}$ where every row is a **one-hot encoding** of the target variable

Softmax and loss



cat
car
frog

3.2
5.1
-1.7

Unnormalized log-probabilities / logits

exp

24.5
164.0
0.18

unnormalized probabilities

normalize

0.13
0.87
0.00

probabilities

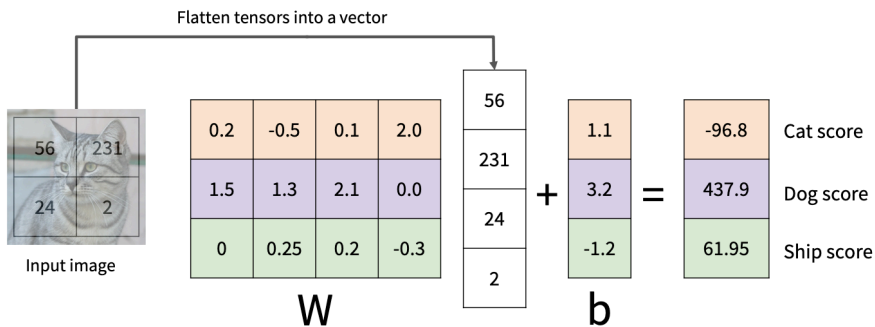
Probabilities must be ≥ 0

Probabilities must sum to 1

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

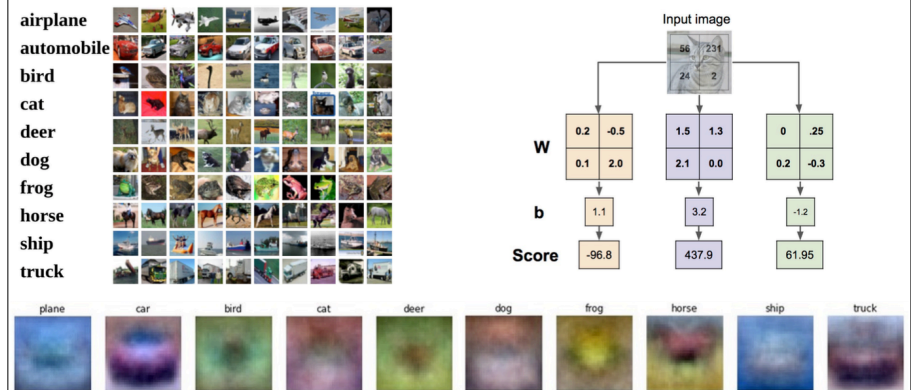
$\rightarrow L_i = -\log(0.13) = 2.04$

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



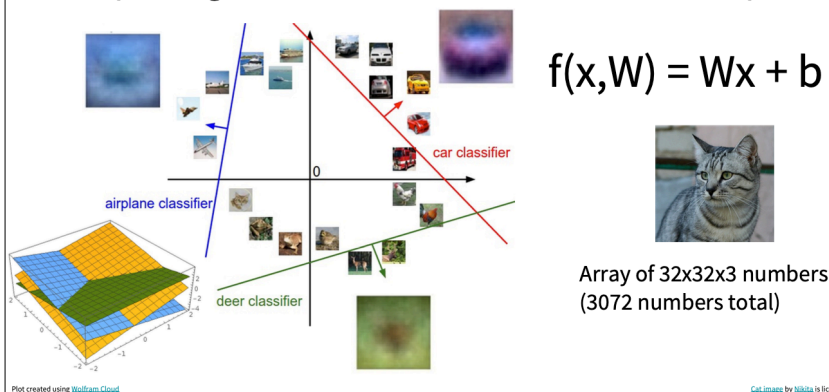
<https://cs231n.stanford.edu/schedule.html>

Interpreting a Linear Classifier: Visual Viewpoint



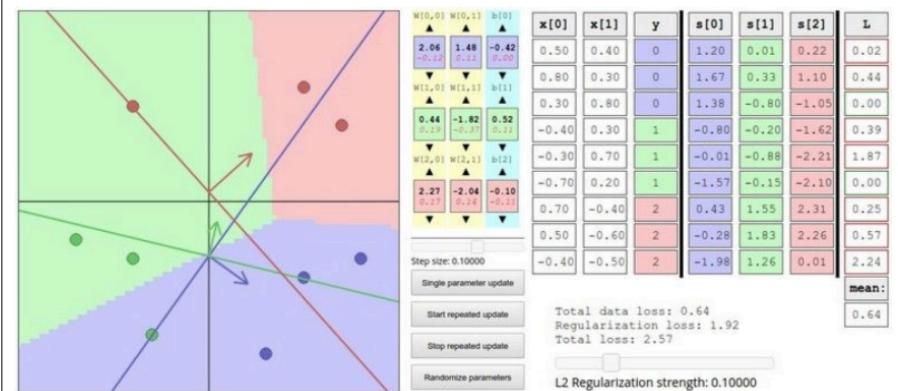
<https://cs231n.stanford.edu/schedule.html>

Interpreting a Linear Classifier: Geometric Viewpoint



<https://cs231n.stanford.edu/schedule.html>

Interactive web demo



<http://vision.stanford.edu/teaching/cs231n-demos/linear-classify/>