STATS / DATA SCI 315
Lecture 05

Classification
Softmax
Cross-entropy loss

Classification Problems

Regression vs Classification

- **Regression** answers *how much?* or *how many?* questions
    - Dollar price of a home
    - # of wins of a baseball team
    - Hospitalization duration in days
- **Classification** answers *which one?* questions
    - Is this email spam or legit?
    - Will a customer sign up for a new service?
    - Which movie is a customer going to watch next (see extreme classification)?

Toy problem

- Imagine classifying 2 x 2 grey scale images into one of the 3 categories:
  - “cat” , “chicken”, “dog”
- Input image consists of just 4 features x_1, x_2, x_3, x_4
- How do we represent the label *y*?
- We could use $y \in \{1, 2, 3\}$ but this suggests an ordering in the labels
- **One-hot encoding** y would be a three-dimensional vector, with (1,0,0) corresponding to “cat”, (0,1,0) to “chicken”, and (0,0,1) to “dog”

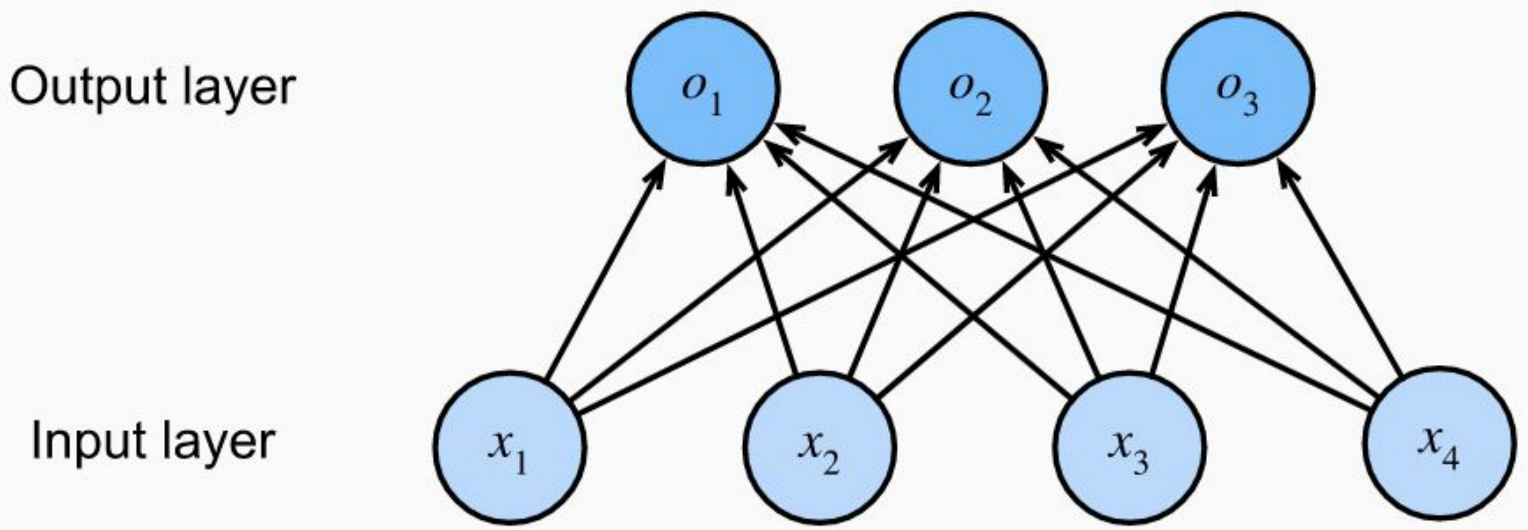
$$y \in \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}.$$

What does our model output?

- We might want to output **hard labels**, i.e., a definite assignment to one of the 3 classes
- Or we might prefer **soft labels**, i.e. probabilities
- E.g., (0.5, 0, 0.5) to 50-50 chance for cat or dog
- Need a model with multiple outputs, one per class
- If restrict ourselves to linear (actually affine) models then we need 3 affine functions
- Each affine function has 4 weights and 1 bias

Network Architecture

$$
\begin{aligned}
o_1 &= x_1 w_{11} + x_2 w_{12} + x_3 w_{13} + x_4 w_{14} + b_1,\\
o_2 &= x_1 w_{21} + x_2 w_{22} + x_3 w_{23} + x_4 w_{24} + b_2,\\
o_3 &= x_1 w_{31} + x_2 w_{32} + x_3 w_{33} + x_4 w_{34} + b_3.
\end{aligned}
$$



Compact matrix notation

- We gather all of our weights into a 3×4 matrix $\mathbf{W}$
- For features of a given data example $\mathbf{x}$, our outputs are given by: a matrix-vector product of weights with features plus biases $\mathbf{b}$
- $\mathbf{o} = \mathbf{W}\mathbf{x}+\mathbf{b}$
- Left dimension: output 3 x 1
- Right dimensions:
    - Matrix-vector product: (3 x 4) x (4 x 1)
    - Bias: 3 x 1

Parameterization Cost

Parameters in a fully connected layer

- Fully-connected layers are ubiquitous in deep learning
- Fully-connected layers have many learnable parameters
- For a f.c. layer with d inputs and q outputs, the parameterization cost is $O(dq)$

Softmax Operation

Why softmax?

- Recall linear model for probabilities
- Outputs are not necessarily positive!
- Outputs don't sum to 1!
- Violates basic probability laws

$$
\begin{aligned}
o_1 &= x_1 w_{11} + x_2 w_{12} + x_3 w_{13} + x_4 w_{14} + b_1, \\
o_2 &= x_1 w_{21} + x_2 w_{22} + x_3 w_{23} + x_4 w_{24} + b_2, \\
o_3 &= x_1 w_{31} + x_2 w_{32} + x_3 w_{33} + x_4 w_{34} + b_3.
\end{aligned}
$$

Softmax function

- We exponentiate our outputs (to get positives values)
- Then divide by the sum (to get them to sum to 1)

$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{o}) \quad \text{where} \quad \hat{y}_j = \frac{\exp(o_j)}{\sum_k \exp(o_k)}.$$

Properties of softmax function

- It produces valid probabilities
- It preserves ordering
- Our model has now become: softmax($\mathbf{W}\ \mathbf{x} + \mathbf{b}$)
- D2L book says this is still a linear model
- I think a more appropriate description is a *generalized* linear model (you're allowed to add one nonlinear function on top of a linear model)

Softmax output as conditional probabilities

- On some input **x** softmax function gives us a vector

$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{o})$$

Where

$$\mathbf{o} = \mathbf{W}\mathbf{x} + \mathbf{b},$$

we can interpret this as estimated conditional probabilities of each class given any input $\mathbf{x}$

Likelihood

Likelihood

- Suppose that the entire dataset $\{\mathbf{X}, \mathbf{Y}\}$ has n examples
- Example i consists of a feature vector $\mathbf{x}^{(i)}$ and a one-hot label vector $\mathbf{y}^{(i)}$
- Probability of actual observed class labels given features

$$P(\mathbf{Y} \mid \mathbf{X}) = \prod_{i=1}^{n} P(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)}).$$

Likelihood

- Note that $P(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)})$ can be written in terms of the $\hat{\mathbf{y}}^{(i)}$
- It is simply the product:

$$\prod_j (\hat{y}_j)^{y_j}$$

- This is a complicated way of saying that the probability of seeing an observed label is one of the components of your predicted probability vector!

Log Likelihood

$$-\log P(\mathbf{Y} \mid \mathbf{X}) = \sum_{i=1}^{n} -\log P(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)}) = \sum_{i=1}^{n} l(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)}),$$

where for any pair of label $\mathbf{y}$ and model prediction $\hat{\mathbf{y}}$ over q classes, the loss function l is

$$l(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{j=1}^{q} y_j \log \hat{y}_j.$$

Cross-entropy loss

$$l(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{j=1}^{q} y_j \log \hat{y}_j.$$

- Because of one-hot encoding, only one term survives
- You pay a high loss if an improbable label (according to your model) is seen
- Loss is always non-negative
- When is it (close to) zero?