Objective

- The big picture: building a bridge from common NLP models to powerful discriminative methods available in neural network land.

Logistic Regression / “MaxEnt”

A declarative representation

\[ p(y_i | x_i, W) \]

\[ W \]

\[ y_i \sim \text{Cat}(\mathcal{S}(W x_i)) \]

\[ W \] is a \( K \times F \) matrix

\( y_i \) : product of a \( K \times 1 \) matrix and an \( F \)-dimensional vector

Result: a \( K \)-dimensional vector

\[ \mathcal{S} \] is the softmax function

\[ \delta \] maps a vector input onto a vector output \( \delta : \mathbb{R}^K \rightarrow \mathbb{R}^F \)

Training Objective: Conditional Log Likelihood of Data

\[ L(W) = \prod_{i \in N} p(y_i | x_i, W) \] (1)

\[ = \prod_{i \in N} \sigma(W x_i) \] (2)

\[ = \prod_{i \in N} \frac{\exp(W x_i)}{1 + \exp(W x_i)} \] (3)

\[ L(W) = \sum_{i \in N} w_{y_i} - \sum_{i \in N} \log \sum_{j \in \mathcal{Y}} \exp(W_{ij}) \] (4)

Maximizing the Objective

\[ \frac{\partial}{\partial W_{yj}} L(W) = \sum_{i \in \mathcal{Y}} \frac{\partial}{\partial W_{yj}} \exp(W x_i) - \sum_{i \in \mathcal{N}} \frac{\partial}{\partial W_{yj}} \sum_{k \neq y} \exp(W x_i) \] (5)

\[ - \sum_{i \in \mathcal{N}} \frac{\partial}{\partial W_{yj}} \sum_{k \neq y} \exp(W x_i) \frac{\partial}{\partial W_{yj}} \exp(W x_i) \] (6)

\[ = \sum_{i \in \mathcal{N}} \frac{1}{\exp(W x_i)} \frac{\partial}{\partial W_{yj}} \exp(W x_i) \] (7)

\[ = \sum_{i \in \mathcal{N}} \frac{1}{\exp(W x_i)} \frac{\partial}{\partial W_{yj}} \exp(W x_i) \] (8)

\[ = \sum_{i \in \mathcal{N}} \frac{1}{\exp(W x_i)} \frac{\partial}{\partial W_{yj}} \exp(W x_i) \] (9)

\[ = \sum_{i \in \mathcal{N}} \frac{1}{\exp(W x_i)} \frac{\partial}{\partial W_{yj}} \exp(W x_i) \] (10)

For (6) & (9):

\[ \frac{\partial}{\partial W_{yj}} x_i = \frac{\partial}{\partial W_{yj}} \sum_{k \neq y} \exp(W x_i) - \frac{\partial}{\partial W_{yj}} \sum_{k \neq y} \exp(W x_i) x_i + x_i \]
**Single-layer Neural Net (no hidden layers)**

A Procedural representation

\[
\begin{align*}
X_1 & \rightarrow | a_1 | \rightarrow O_1 \\
X_2 & \rightarrow | a_2 | \rightarrow O_2 \\
X_3 & \rightarrow | a_3 | \rightarrow O_3 \\
X_4 & \rightarrow | a_4 | \rightarrow O_4
\end{align*}
\]

**Equivalently**

\[
L(W) = \sum \log S(Wx_i)_{y_i}
\]

\[
\frac{\partial}{\partial W_{ij}} L(W) = \sum_i \frac{\partial}{\partial W_{ij}} \log S(Wx_i)_{y_i}
\]

\[
= \sum_i \frac{\partial}{\partial W_{ij}} \log S(Wx_i)_{y_i} = \sum_i \frac{1}{Wx_i} \frac{\partial}{\partial W_{ij}} S(Wx_i)_{y_i}
\]

\[
= \sum_i \frac{1}{Wx_i} (1 - S(Wx_i)_{y_i})x_{ij}
\]

\[
= \sum_i 2(y_i - x_{ij})x_{ij}
\]

\[
= \sum_i (o_i - y_i)x_{ij}
\]

**Different Objective: Cross Entropy**

- Common practice to train by minimizing the summed cross-entropy CE between empirical label distributions in the training set and model predictions.
- The cross entropy for instance \(i\):
  \[
  CE_i(t, \hat{p}) = \sum_k t_i(k) \log \left( \hat{p}(k|x_i, W) \right)
  \]
- The empirical label distribution is the 1-of-\(K\) encoded label:
  \[
  t_i(k) = 1(y_i = k), \quad \forall 1 \leq k \leq K
  \]
  \[
  y_i = [y_i(1), \ldots, y_i(K)]^T
  \]

**Minimizing the Objective**

\[
\frac{\partial}{\partial W_{ij}} CE(W) = \frac{\partial}{\partial W_{ij}} \sum_i CE_i(W)
\]

\[
= -\sum_i t_i(k) \frac{1}{Wx_i} \frac{\partial}{\partial W_{ij}} S(Wx_i)_{y_i}
\]

\[
= -\sum_i t_i(k) \frac{1}{Wx_i} (1 - S(Wx_i)_{y_i})x_{ij}
\]

\[
= \sum_i \frac{d(Wx_i)_{y_i}}{} x_{ij}
\]

\[
= \sum_i (o_i - y_i)x_{ij}
\]

**Punch Line**

- MaxEnt models are single-layer neural nets.

**Drawbacks and Fixes**

- The canonical XOR problem consists of the following inputs and outputs
  \[
  x \rightarrow y: \{[0, 0] \rightarrow 0, [0, 1] \rightarrow 1, [1, 0] \rightarrow 1, [1, 1] \rightarrow 0\}
  \]
- Not linearly separable!
- Fix: transform the inputs by defining a feature that only fires if \(x_1 \neq x_2\)
- This solution reflects common practice in fields like NLP and computer vision.
- Use expert knowledge to hand-craft features that make the problem at hand more amenable to simple modeling.
Why Manual?

- Rather than manually define a transform, we might learn the transform by:
  - adding an additional $H \times F$ matrix of weights $W^H$ to our model
  - then deriving an objective that includes the new weights as well
- $H$ represents the number of transformed features with may be larger or smaller than $F$
- Rename original weights: $W^O$
- Same declarative model (orig. figure), but now:

$$y_i \sim \text{Cat}(S(W^O \sigma(W^H x_i)))$$

$$\sigma: \mathbb{R}^K \rightarrow \mathbb{R}^K$$

Objectives & Gradient

This notation also underscores the fact that lower level intermediate values can be cached and used to calculate higher level values (dynamic programming).

More of Gradient

More of Gradient (cont.)

Back-propagation

- Add another transform layer with weights $W^G$

$$CE(W) = - \sum_i \sum_k t_i(k) \log S(W^O \sigma(W^H \sigma(W^G x_i)))_{y_i}$$

- The gradient of $W^H$ and $W^O$ would remain unchanged except for substituting the transformed input $u^G$ in place of $x$.

$$\frac{\partial}{\partial W_i} CE(W) = \sum_i \sum_k (u^G_k - y_i) \sigma'(\sigma(W^H \sigma(W^G x_i)))_{y_i}$$
DP

- Add yet another transform layer with weights $W^{E}$:

$$
\frac{\partial}{\partial W} C(E(W)) = \sum \sum \sum (x_{i} - \hat{x}_{i}) \sum W_{ij}^{E} (1 - \sigma'(z_{i})) \sum W_{jk}^{E} (1 - \sigma'(z_{j})) \sum W_{kl}^{E} (1 - \sigma'(z_{l})) \sum W_{ml}^{E} (1 - \sigma'(z_{m})) \sum W_{nl}^{E} (1 - \sigma'(z_{n})) \sum W_{nl}^{E} (1 - \sigma'(z_{n})) \sum W_{nl}^{E} (1 - \sigma'(z_{n})) \sum W_{nl}^{E} (1 - \sigma'(z_{n}))
$$

- The number of terms in the nested sum would grow as $O(H^2)$ where $H$ is the width of the widest layer and $L$ is the number of layers.

$$
O(H^2) \text{ per layer: } O(H^2 \cdot L)
$$

The End

- A greater understanding of the options for building discriminative models.
- Discriminative models for sequence labeling