DEEP LEARNING: BACKPROP

1. Basic Setup / Notation

- Input
  - $\mathbf{x} \in \mathbb{R}^n$ (continuous input) sometimes
  - $\mathbf{c} \in \{0, 1\}^n$ (discrete input) both

- Scalars: $a, b, c, i, j, k, x, y$ (small case)
- Vectors: $\mathbf{x}, \mathbf{y}, \mathbf{w}, \mathbf{h}$ (vec notation)
- Matrices: $A, B, W, X, H$ (capital case)
  - 2D objects or random variables
  - $d$: #dimension
  - $N$: #inputs

2. Output / Target / Labels

- $\mathbf{y} \in \mathbb{R}^k$ (K-dim regression)
- $\mathbf{c} \in \{0, 1\}^k$ (Binary classification)
- $\mathbf{y} \in \{1, 2, ..., K\}$ (Multi-class classification)
- $\mathbf{a} \in \{0, 1\}^k_+$ 1-hot encoding

3. Unknown Target Function

$\hat{f}: X \rightarrow Y$

Input Space \quad Output Space
Given: Database
a "Training Data"
\[ D = \{(x_i, y_i) \} \]
sample or data-point

Goal: Given D (input, output) from unknown f,
predict f(x) on NEW x

Approach: [All of Supervised Learning]

1. Pick a model class / Hypothesis Space

   \[ H = \{ g : x \rightarrow Y \} \]
   a set of mappings from \( x \rightarrow Y \)

   [a lot of ML is about what kind of \( H \) "makes sense"
   for our task]

2. Search / Optimize a loss function

   over \( H \) to find "best" \( g \in H \)

   \[ \text{learning} \equiv \arg \min_{g \in H} \text{Loss}(g; D) \]

   usually
   \[ \sum_{i=1}^{N} \text{li}(y_i, g(x_i)) \]
Artificial "Neuron"

1 = \omega_0

\omega_j

a = \sum_{j=0}^{m} w_j x_j

Activation/Response Function

\hat{y} = f(a) = f(w^T x)

Inputs
Strengths of
Connections

Many different activation functions

Linear: \( f(a) = a \)

\( \hat{y} = w^T x \) [Linear Regression]

Sigmoid: \( f(a) = \frac{1}{1 + e^{-a}} = \sigma(a) \)

\( \hat{y} = \frac{1}{1 + e^{-w^T x}} \) [Logistic Regression]

Logistic

Tanh

f(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}

For hidden units in NN, we always prefer Tanh over Sigmoid.
ReLU [Rectified Linear Unit]

\[ f(a) = \max \{0, a\} \]

In hidden layer of deep NN, this is always preferred over \( \sigma(a) \) or \( \tanh(a) \). Why??

\[ f(a) = \max \{ \mathbf{w}_1^T \mathbf{x}, \mathbf{w}_2^T \mathbf{x} \} \]

Each neuron has (say) 2 weights \( \mathbf{w}_1, \mathbf{w}_2 \).
Take max activation.
ReLU is a special case of this. How?
(3) **Loss Functions**

functions of both parameters vs training data

1. **Log-Loss / Cross-Entropy / Maximum-Likelihood / KL-Divergence**

\[ L(\theta; D) = \frac{1}{n} \sum_{i=1}^{n} L_i(\theta) \]  

Decomposable Loss

where \( L_i(\theta) = -\log p(y_i | x_i; \theta) \)

How much prob does your model assign to G, T labels?

\( - \) negative log-likelihood for this sample

Why is this called Cross-Entropy? And where is the Kullback-Leibler divergence coming in?

Consider **Multiclass classification w/ 1-HOT encoding**

\[ \hat{y} \]

\[ y \]

\[ 0 \quad 1 \]

\[ 0 \quad 1 \]

\[ 1 \times K \]

\[ p_{\text{std}}(y) \] [delta distribution]  

\[ p(y) \] [Model distribution]
\[ KL(p^\text{st} \parallel \hat{p}) = -\sum_{y=1}^{K} p^\text{st}(y) \log \hat{p}(y) \]

\[ = -\log p(y = y^\text{st} | \mathbf{x}, \mathbf{w}) \]

2. Hinge Loss [for binary classification]

\[ L_i(w) = \max \left[ 0, 1 - y_i \hat{y}_i \right] \quad \text{where} \quad y_i \in \{+1, -1\} \]
4. **Detour: Matrix/Vector differentiation**

\[
\begin{bmatrix}
S & V & M \\
\frac{dy}{dx} & \frac{dy}{dx} & \frac{dy}{dx} \\
\frac{dy}{dx} & \frac{dy}{dx} & \frac{dy}{dx} \\
\frac{dy}{dx} & \frac{dy}{dx} & \text{Tens.} \\
\end{bmatrix}
\]

\[x, y \in \mathbb{R}^1, \quad z \in \mathbb{R}^2, \quad \mathbf{g} \in \mathbb{R}^k\]

Convention:
\[
\begin{bmatrix}
\frac{dy_1}{dx} \\
\frac{dy_2}{dx} \\
\vdots \\
\frac{dy_k}{dx}
\end{bmatrix}
\]
- numerator = \(\text{dim} 1\) = \(\text{col-vector}\)
- denominator = \(\text{dim} 2\) = \(\text{row-vector}\)

\[
\frac{\partial y}{\partial x} = \begin{bmatrix}
\frac{\partial y_1}{\partial x} \\
\frac{\partial y_2}{\partial x} \\
\vdots \\
\frac{\partial y_k}{\partial x}
\end{bmatrix}
\]

\[\text{Gradient] \quad \frac{\partial y}{\partial x} = \begin{bmatrix}
\frac{\partial y_1}{\partial x} \\
\frac{\partial y_2}{\partial x} \\
\vdots \\
\frac{\partial y_k}{\partial x}
\end{bmatrix} \quad \text{denominator = dim 2} \quad = \text{row-vector}\]

\[\text{Jacobian Matrix] \quad \frac{\partial y}{\partial x} = \begin{bmatrix}
\frac{\partial y_1}{\partial x} \\
\frac{\partial y_1}{\partial y_1} \\
\vdots \\
\frac{\partial y_k}{\partial y_k}
\end{bmatrix}
\]

\[\text{Easy to prove:} \quad \frac{\partial (y^T x)}{\partial \mathbf{w}} = \begin{bmatrix}
\frac{\partial y_1}{\partial w_1} \\
\frac{\partial y_2}{\partial w_2} \\
\vdots \\
\frac{\partial y_k}{\partial w_k}
\end{bmatrix} = \mathbf{x}^T
\]

\[\frac{\partial (y^T A \mathbf{x})}{\partial \mathbf{w}} = 2y^T A\]

\[\rightarrow \quad y = A \mathbf{x} \quad \frac{\partial y}{\partial x} = A\]
5. **Chain Rule**

- **Function Composition:** \( L(x) = (f \circ g)(x) = f(g(x)) \)

**Chain Rule:**

- **[Most General Notation]**
  \[
  \frac{dx}{dx} (f \circ g) = Dg \circ f \circ Dg
  \]
  (total derivative)

- **[More concrete notation for scalars]**
  \[
  L'(x) = f'(g(x)) g'(x)
  \]

- **[With intermediate variables]**
  \[
  y = g(x) \\
  z = f(y)
  \]

  \[
  \frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}
  \]

**Example:**

\[
L_i(w) = -\log \left( \frac{1}{1+e^{-w_i}} \right) \quad \text{[For } y_i = +1]\n\]

\[
\frac{dL}{dw} = \left( -\log(\cdot) \circ \frac{1}{1+e^{-\cdot}} \circ x^T(\cdot) \right)(w)
\]

\[
\frac{dL}{dx} = \left[ \begin{array}{c} 1 \\
\frac{1}{P} \end{array} \right] \cdot \left[ \begin{array}{c} -1 \\
\frac{1}{(1+e^z)^2} \end{array} \right] \cdot x^T = (1-P) x^T
\]
Multivariate Chain Rule

$g : \mathbb{R}^k \to \mathbb{R}^m$

$f : \mathbb{R}^m \to \mathbb{R}^k$

$\nabla (\mathbf{z}) = (f \circ g)(\mathbf{x})$

$\mathbf{y} = g(\mathbf{x}) \quad \mathbf{z} = f(\mathbf{y})$

Chain Rule:

$D_{\mathbf{z}} (f \circ g) = D_{\mathbf{y}} f \circ D_{\mathbf{z}} g$

[Absent form holds?]

But what does this mean??

Visualize:

```
  x1
  |   y1
  |   |   z1
  |   |   |
  |   |   |
  |   |   |
  |   |   |
  |   |   |
  |   |   |
  |---------------------->
  y2
  |   y3
  |   y4
  |   y5
  |   y6
  |---------------------->
  ym
```

```
\[ \frac{\partial z_i}{\partial x_j} = \sum_{k} \frac{\partial z_i}{\partial y_k} \cdot \frac{\partial y_k}{\partial x_j} \]
```

All intermediate variables affect outcome!

How my "knot" affects intermediate variable.
Formally,

\[ J_{fog} = (J_{fg})^T J_g \]

\[ \begin{bmatrix} \frac{\partial z}{\partial x} \\ \frac{\partial z}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial z_i}{\partial x_k} \\ \frac{\partial z_i}{\partial y_k} \end{bmatrix} \begin{bmatrix} \frac{\partial x_k}{\partial x} \\ \frac{\partial y_k}{\partial x} \end{bmatrix} \]

→ What if my \( \mathbf{x}, \mathbf{z} \) are tensors?

→ String up into vectors & proceed

→ Matlab notation \( \times \text{vec} = \times(i) \)

→ Trust me, this is the cleanest way

→ In Neural Net \( \mathbf{z} \in \mathbb{R}^T \) \( (\log) \ L(\mathbf{w}) \)

\[ h^{(i)} = g(h^{(i-1)}, \mathbf{w}) \]

\[ L(\mathbf{w}) = f(h^{T}) \]

\[ \frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial h^{T}} \frac{\partial h^{T}}{\partial \mathbf{w}} \]

\[ \frac{\partial L}{\partial \mathbf{w}} = \langle \text{vec} \frac{\partial L}{\partial h^{T}}, \text{vec} \frac{\partial h^{T}}{\partial \mathbf{w}} \rangle \]