1 Regression

1.1 Terminology

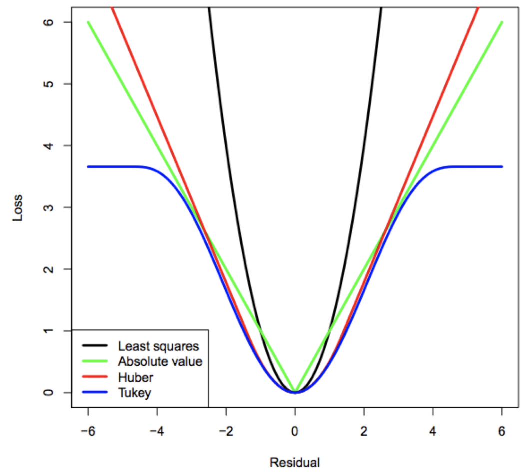
- Data consists of **pairs** $(\mathbf{x}_n, y_n)$, where y_n is the n'th output and x_n is a vector of D inputs. The number of pairs N is the data-size and D is the dimensionality.
- Two goals of regression: **prediction** and **interpretation**
- The regression function: $y_n \approx f_w(\mathbf{x}_n)\ \forall n$
- Regression finds correlation not a causal relationship.
- **Input variables** a.k.a. covariates, independent variables, explanatory variables, exogenous variables, predictors, regressors.
- **Output variables** a.k.a. target, label, response, outcome, dependent variable, endogenous variables, measured variable, regressands.

1.2 Linear Regression

- Assumes linear relationship between inputs and output.
- $y_n \approx f(\mathbf{x}_n) := w_0 + w_1 x_{n1} + \ldots + w_D x_{nD} := \tilde{\mathbf{x}}_n^T \tilde{\mathbf{w}}$ contain the additional offset term (a.k.a. bias).
- Given data we learn the weights $\mathbf{w}$ (a.k.a. estimate or fit the model)
- Overparameterisation $D > N$ eg. univariate linear regression with a single data point $y_1 \approx w_0 + w_1 x_{11}$. This makes the task under-determined (no unique solution).

1.3 Loss Functions $\mathcal{L}$

- A loss function (a.k.a. energy, cost, training objective) quantifies how well the model does (how costly its mistakes are).
- $y \in \mathbb{R} \Rightarrow$ desirable for cost to be symmetric around 0 since $\pm$ errors should be penalized equally.
- Cost function should penalize "large" mistakes and "very large" mistakes similarly to be robust to outliers.
- Mean Squared Error:
$\mathrm{MSE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}[y_n - f_{\mathrm{w}}(\mathbf{x}_n)]^2$
not robust to outliers.
- Mean Absolute Error:
$\mathrm{MAE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}|y_n - f_{\mathrm{w}}(\mathbf{x}_n)|$
- Convexity: a function is convex iff a line segment between two points on the function's graph always lies above the function.
- Convexity: a function $h(\mathbf{u}), \mathbf{u} \in \mathbb{R}^D$ is convex if $\forall\ \mathbf{u}, \mathbf{v} \in \mathbb{R}^D, 0 \le \lambda \le 1$:
$h(\lambda\mathbf{u} + (1-\lambda)\mathbf{v}) \le \lambda h(\mathbf{u}) + (1-\lambda)h(\mathbf{v})$
Strictly convex if $\le \Rightarrow <$
- Convexity, a desired computational property: A strictly convex function has a unique global minimum $\mathbf{w}^*$. For convex functions, every local minimum is a global minimum.
- Sums of convex functions are also convex $\Rightarrow$ MSE combined with a linear model is convex in $\mathbf{w}$.
- Proof of convexity for MAE:
$\mathrm{MAE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}_n(\mathbf{w}), \mathcal{L}_n(\mathbf{w}) = |y_n - f_{\mathrm{w}}(\mathbf{x}_n)|$
$\mathcal{L}_n(\lambda w_1 + (1-\lambda)w_2) \le \lambda\mathcal{L}_n(w_1) + (1-\lambda)\mathcal{L}_n(w_2)$
$|y_n - x_n^T(\lambda w_1 + (1-\lambda)w_2)| \le \lambda|y_n - x_n^T w_1| + (1-\lambda)|y_n - x_n^T w_2|$
$(1-\lambda) \ge 0 \Rightarrow (1-\lambda)|y_n - x_n^T w_2| = |(1-\lambda)y_n - (1-\lambda)x_n^T w_2|$
$a = \lambda y_n - \lambda x_n^T w_1, b = (1-\lambda)y_n - (1-\lambda)x_n^T w_2$
$a + b = y_n - x_n^T(\lambda w_1 + (1-\lambda)w_2)$
$|a+b| \le |a| + |b| \Rightarrow \mathcal{L}_n(\mathbf{w})$ convex $\Rightarrow \mathrm{MAE}(\mathbf{w})$ convex
- Huber loss:
$\mathcal{H}uber(e) := \begin{cases} \frac{1}{2}e^2 & \text{, if } |e| \le \delta \\ \delta|e| - \frac{1}{2}\delta^2 & \text{, if } |e| > \delta \end{cases}$ convex, differentiable, and robust to outliers but setting δ is not easy.
- Tukey's bisquare loss:
$\frac{\partial\mathcal{L}}{\partial e} := \begin{cases} e\{1 - e^2/\delta^2\}^2 & \text{, if } |e| \le \delta \\ 0 & \text{, if } |e| > \delta \end{cases}$ non-convex, but robust to outliers.



2 Optimisation

- Given $\mathcal{L}(\mathbf{w})$ we want $\mathbf{w}^* \in \mathbb{R}^D$ which minimises the cost: $\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \rightarrow$ formulated as an optimisation problem
- Local minimum $\mathbf{w}^* \Rightarrow \exists\epsilon > 0$ s.t.
$\mathcal{L}(\mathbf{w}^*) \le \mathcal{L}(\mathbf{w})\ \forall\mathbf{w}$ with $\|\mathbf{w} - \mathbf{w}^*\| < \epsilon$
- Global minimum $\mathbf{w}^*$, $\mathcal{L}(\mathbf{w}^*) \le \mathcal{L}(\mathbf{w})\ \forall\mathbf{w} \in \mathbb{R}^D$

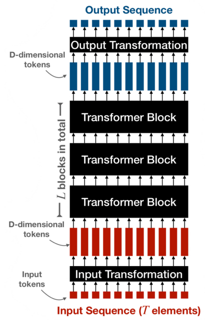
2.1 Smooth Optimisation

3 Transformers

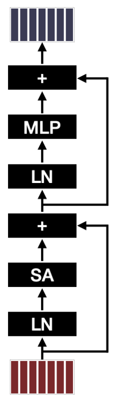
- A transformer is a neural network that iteratively transforms a sequence to another sequence and mixes the information between the sequence elements via self-attention.

3.1 Architecture

- Self-Attention (SA): mixes information between tokens
- Multi-Layer Perceptron (MLP): mixes information within each token
- Skip connections are widely used
- Layer normalization (LN) is usually placed at the start of a residual branch



3.2 Text Token Embeddings



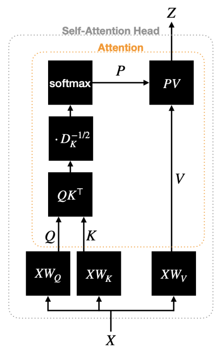
- Tokenization: split the input text into a sequence of input tokens (typically word fragments + some special symbols) according to some pre-defined tokenizer procedure:
- Convert each token ID $i \in \{1, \ldots, N_{vocab}\}$ into a real-valued vector $\mathbf{w}_i \in \mathbb{R}^D$
- This can be seen as a matrix multiplication $\mathbf{W} \cdot \mathbf{e}_i = \mathbf{W}_{:,i} = \mathbf{w}_i$ (with $\mathbf{W} \in \mathbb{R}^{\mathrm{D} \times \mathrm{N_{vocab}}}$)
- $\mathbf{W}$ is learned via backpropagation, along with all other transformer parameters (however, the tokenizer procedure is typically fixed in advance and not learned)
- The whole input sequence of T tokens leads to an input matrix $X \in \mathbb{R}^{T \times D}$

3.3 Attention

- Attention is a function that transforms a sequence of tokens to a new sequence of tokens using a learned input-dependent weighted average
- Input tokens : $V \in \mathbb{R}^{T_{in} \times D}$
- Output tokens : $Z \in \mathbb{R}^{T_{out} \times D}$
- Output tokens are simply a weighted average of the input tokens: $z_i = \sum_{j=1}^{T_i} p_{ij} v_j$ i.e. $Z = PV$
- Weighting coefficients $\mathcal{P} \in [0,1]^{T_{out} \times T_{in}}$ form valid probability distributions over the input tokens $\sum_{j=1}^{T_{in}} p_{ij} = 1$
- Query tokens : $Q \in \mathbb{R}^{T_{out} \times D_K}$
- Key tokens : $K \in \mathbb{R}^{T_{in} \times D_K}$
- Determine weight $p_{i,j}$ based on how simmilar q_i and k_j are.
- Use inner product to obtain raw similarity scores.
- Normalize with softmax (scaled the temperature by $\sqrt{D_K}$) to obtain a probability distribution.
- $P = \mathrm{softmax}\left(\frac{QK^{\mathsf{T}}}{\sqrt{D_K}}\right)$ The softmax is applied on each row independently. Scaling ensures uniformity at initialization and faster convergence

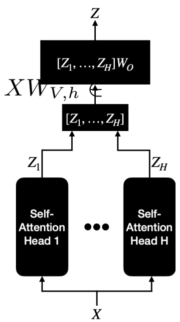
3.4 Self-Attention

- V, K, Q are all derived from the same input token sequence $X \in \mathbb{R}^{T \times D}$
- Values : $V = XW_V \in \mathbb{R}^{T \times D}$, $W_V \in \mathbb{R}^{D \times D}$
- Keys : $K = XW_K \in \mathbb{R}^{T \times D_K}$, $W_K \in \mathbb{R}^{D \times D_K}$
- Queries : $Q = XW_Q \in \mathbb{R}^{T \times D_K}, W_Q \in \mathbb{R}^{D \times D_K}$
- W_Q, W_V, W_K are learned parameters.
- $Z = \mathrm{softmax}\left(\frac{XW_Q W_K^T X^T}{\sqrt{D_K}}\right) XW_V$



3.4.1 Multi-Head Self-Attention

- Run H Self-Attention "heads" in parallel
- $Z_h = \mathrm{softmax}\left(\frac{XW_{Q,h}W_{K,h}^{\mathsf{T}}X^{\mathsf{T}}}{\sqrt{D_K}}\right) XW_{V,h} \in \mathbb{R}^{T \times D_V}$
- $W_{V,h} \in \mathbb{R}^{D \times D_V}$, $W_{K,h} \in \mathbb{R}^{D \times D_K}$, $W_{Q,h} \in \mathbb{R}^{D \times D_K}$
- The final output is obtained by concatenating head-outputs and applying a linear transformation $Z = [Z_1, \ldots, Z_H]W_o$ where $W_O \in \mathbb{R}^{HD_V \times D}$ is learned via backpropagation



3.5 Positional Information

- Attention by itself does not account for the order of input
- incorporate a positional encoding in the network which is a function from the position to a feature vector $pos : \{1, \ldots, T\} \rightarrow \mathbb{R}^D$
- The most basic choice is to add a positional embedding W_{pos} corresponding to each token's position t to the input embedding. $W_{pos} \in \mathbb{R}^{D \times T}$ is learned via backpropagation along with the other parameters

3.6 MLP

- Mixing Information within Tokens
- Apply the same transformation to each token independently: $MLP(X) = \varphi(XW_1)W_2$
- $W_1, W_2 \in \mathbb{R}^{D \times D}$ learned via backprop

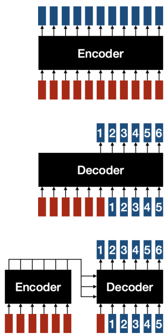
3.7 Output Transformations

- typically simple: linear transformation or a small MLP
- dependent on the task: Single output (e.g.,sequence-level classification): apply an output transformation to a special taskspecific input token or to the average of all tokens. Multiple outputs (e.g., per-token classification): apply an output transformation to each token independently

3.8 Vision Transformer Architecture

- Self-attention is more general than convolution and can potentially express it
- The receptive field is the whole image after just one self-attention layer
- ViTs require more data than CNNs due to their reduced inductive bias in extracting local features
- In many cases, the model attends to image regions that are semantically relevant for classification

3.9 Encoders & Decoders



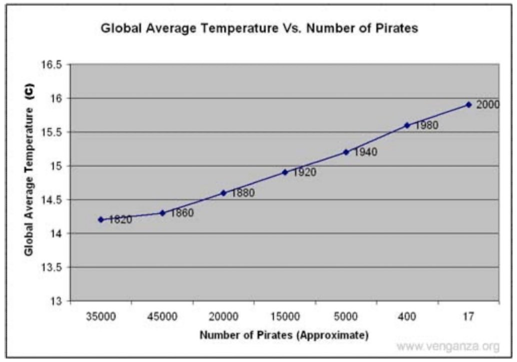
- Encoders (e.g., classification): They produce a fixed output size and process all inputs simultaneously
- Decoders (e.g., ChatGPT): Auto-regressively sample the next token as $x_{t+1} \sim sofimax(f(x_1, ..., x_t))$ and use it as new input token. Capable of generating responses of arbitrary length.
- Encoder-decoder (e.g., translation): First encode the whole input (e.g., in one language) and then decode to token by token (e.g., in a different language)

4 Adversarial ML

5 This is RGB red text.

For $x \in [r_{i-1}, r_i]$
$r(x) = \tilde{a}_1 x + \tilde{b}_1 + \sum_{j=2}^{m} \tilde{a}_j (x - \tilde{b}_j)_+$

For $x \in [r_{i-1}, r_i]$
$r(x) = \tilde{a}_1 x + \tilde{b}_1 + \sum_{j=2}^{m} \tilde{a}_j (x - \tilde{b}_j)_+$



Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

. . . .

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.