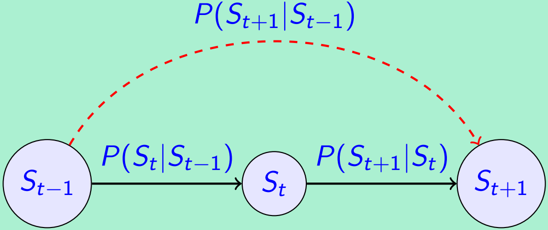
Markov Property

The next state depends only on the current state



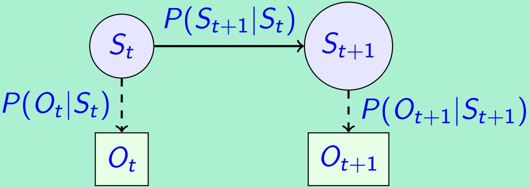
$$P(S_{t+1}|S_t, S_{t-1}, ..., S_1) = P(S_{t+1}|S_t)$$

Terms:

- S_t: State at time t
- $P(S_{t+1}|S_t)$: Transition probability from state S_t to state S_{t+1}

Hidden Markov Model

Markov Model where states are hidden but observations are visible



Terms:

- S_t: Hidden state at time t
- O_t: Observation at time t
- $P(O_t|S_t)$: Emission probability
- $P(S_{t+1}|S_t)$: Transition probability

Applications: Speech recognition, part-of-speech tagging, ...

@AIinMinutes

Cross Entropy

Measures how well a model's predicted distribution matches the true distribution

$$H(P, Q) = -\sum_i P(x_i) \log Q(x_i)$$

Terms:

- $P(x_i)$: True probability distribution
- $Q(x_i)$: Predicted probability distribution

Use-case: Loss function in classification problems, Language models

@AIinMinutes

Perplexity

Measures how confused (perplexed) your language model is.

$$PP = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 P(w_i|w_1,w_2,\ldots,w_{i-1})}$$

Terms:

- PP: Perplexity, defined as 2^{CE}
- $P(w_i \mid w_1, \ldots, w_{i-1})$: Probability of word w_i given its context
- N: Total number of words in the dataset (or sequence)

Use-case: Evaluating language models (e.g., perplexity for ARLM; pseudoperplexity for BERT)

Softmax Function

Converts logits (raw scores) into a probability distribution (multinomial)

$$P(y = i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Terms:

- z_i: Logit (raw model output) for class i
- $P(y = i)$: Probability assigned to class i

Use-case: Classification tasks, neural networks, attention mechanisms

@AIinMinutes

Temperature Scaling

Controls sharpness of the multinomial distribution

$$P(w_i) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$

Terms:

- T: Temperature parameter
- $T > 1$: Increases diversity (more randomness)
- $T < 1$: Makes output more deterministic

Use-case: Text generation (GPT), controlling randomness

@AIinMinutes

Top-p (nucleus) Sampling

Dynamically selects tokens that cover p% of the probability mass

$$V_{\text{top-p}} = \{w_i \mid \sum_{j=1}^{i} P(w_j) \leq p, \text{ sorted by } P(w_j)\}$$

Terms:

- $P(w_i)$: Probability of token w_i
- p: Cumulative probability threshold

Use-case: Text generation, reducing randomness in NLP models

@AIinMinutes

Top-k (truncated) Sampling

Samples from only the k most probable tokens in the vocabulary

$$V_{\text{top-k}} = \{w_i \mid w_i \in \text{top-}k \text{ tokens by } P(w_i)\}$$

Terms:

- k: Number of highest-probability tokens to sample from

Use-case: Controlling text generation

@AIinMinutes

Log-Sum-Exp Trick

Ensures numerical stability in softmax computation

Stable Reformulation:

$$\text{softmax}(z_i) = \frac{e^{z_i - z_{\max}}}{\sum_j e^{z_j - z_{\max}}}$$

Terms:

- z_i: Input logits (raw scores before softmax)
- $z_{\max} = \max_j z_j$: Maximum logit value

Why Use This?

- Prevents numerical overflow in exponentials

@AIinMinutes

Self-Attention Mechanism

Used to create a contextualized vector representation of a part of the input.

$$\text{Self-Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

Why divide by $\sqrt{d_k}$? Dot product values can be large when d_k is high; large values lead to extreme softmax outputs (close to 0 or 1).

Key Terms:

- $\mathbf{Q}$: Query matrix (from input)
- $\mathbf{K}$: Key matrix (from input)
- $\mathbf{V}$: Value matrix (from input)
- d_k: Dimensionality of key vectors

Use-case: Transformers of all kinds, Explainability, ...