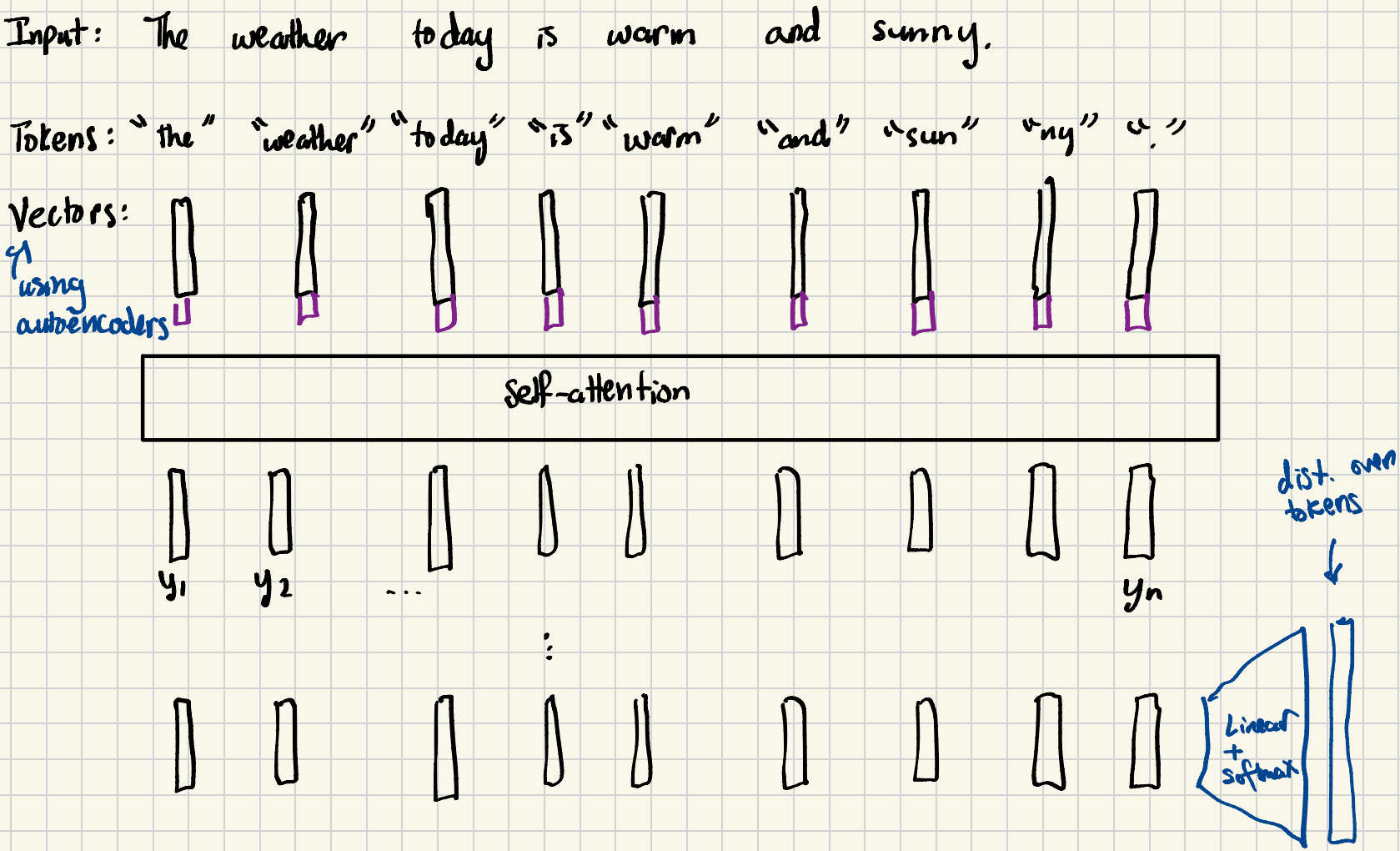
Tuesday, March 24

Welcome back!!

Plan

- Positional encoding
- Decision trees

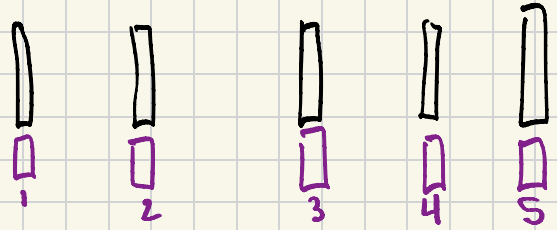

Input: The weather today is warm and sunny.
Tokens: "the" "weather" "today" "is" "warm" "and" "sun" "ny" "."
Vectors:
using autoencoders
Self-attention
y_1
y_2
...
y_n
dist. over tokens
Linear + softmax

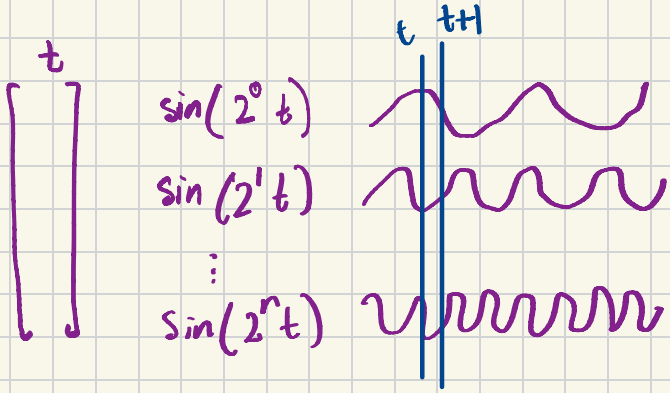
Positional Encoding

"Jack gave water to Jill"

vs.

"Jill gave water to Jack"





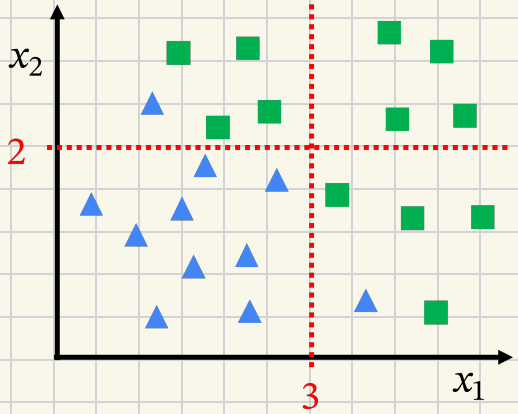
Demo on CNN and transformer ت

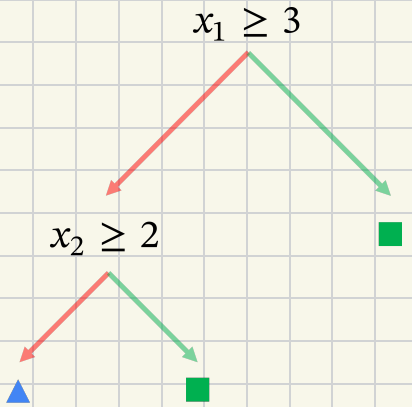
Decision Trees

↳ Remarkably accurate and interpretable

↳ works for classification and regression

↳ Idea: Recursively partition data into homogeneous regions





Greedy Approach for Classification

For each leaf, choose the feature and split that best separates classes

Let l be candidate leaf and $p_c^{(l)}$ the proportion of points of class c in leaf l.

WLOG, $c=0$ is the majority class

Goal: Find l to minimize

$$\mathbb{E}_l(\mathcal{L}(l)) = \sum_l \mathcal{L}(l)\, p^{(l)}$$

$p^{(l)}$: proportion of points in leaf l

$\mathcal{L}(l)$: How should we define loss?

Error Rate of Majority Prediction:

$$\mathcal{L}_{error}(\ell) = 1 - p_0^{(\ell)}$$

Gini Impurity: Probability of misclassifying point if we randomly predict based on proportion of class

$$\mathcal{L}_{Gini}(\ell) = \sum_c p_c^{(\ell)} \left(1 - p_c^{(\ell)}\right)$$

Entropy: like cross-entropy, measure "spread" of distribution

$$\mathcal{L}_{entropy}(\ell) = -\sum_c p_c^{(\ell)} \log\left(p_c^{(\ell)}\right)$$

$$\operatorname*{argmin}_\ell \mathcal{L}_{entropy}(\ell) = \operatorname*{argmax}_\ell \prod_c p_c^{(\ell)}$$

Previously we updated parameters to minimize loss;

For decision trees, we search over splits in data to minimize loss.

Time complexity to find best split in feature x_i?

Q: How do we adapt for regression?

Gradient Boosting

On their own, trees are not very expressive But we can "boost" them into very strong ensembles.

Let $f_t : \mathbb{R}^d \to \{-1, 1\}$ is the weak learner from iteration t

$$F_t(x) = \alpha_1 f_1(x) + \ldots + \alpha_{t-1} f_{t-1}(x) + \alpha_t f_t(x) = F_{t-1}(x) + \alpha_t f_t(x).$$

Goal: Given F_{t-1}, find f_t to minimize:

$$\underset{f_t}{\operatorname{argmin}} \sum_{i=1}^{n} \mathcal{L}\left(y^{(i)}, F_{t-1}(x^{(i)}) + \alpha f_t(x^{(i)})\right)$$

Goal: Given F_{t-1}, find f_t to minimize:

$$\underset{f_t}{\operatorname{argmin}} \sum_{i=1}^{n} \mathcal{L}\left(y^{(i)}, F_{t-1}(x^{(i)}) + \alpha f_t(x^{(i)})\right)$$

Idea: Gradient descent on predictions rather than parameters.

$$F_t(x^{(i)}) \leftarrow F_{t-1}(x^{(i)}) - \alpha \frac{\partial \mathcal{L}(y^{(i)}, F_{t-1}(x^{(i)}))}{\partial F_{t-1}(x^{(i)})}$$

Train f_t so that

$$f_t(x^{(i)}) \approx -\frac{\partial \mathcal{L}(y^{(i)}, F_{t-1}(x^{(i)}))}{\partial F_{t-1}(x^{(i)})} \quad \text{for all } i$$

When loss is MSE: $\mathcal{L}(y, F_{t-1}(x)) = \frac{1}{2}(F_{t-1}(x) - y)^2$, then

$$-\frac{\partial \mathcal{L}(y^{(i)}, F_{t-1}(x^{(i)}))}{\partial F_{t-1}(x^{(i)})} = y - F_{t-1}(x) = \text{residual error of } F_{t-1}$$

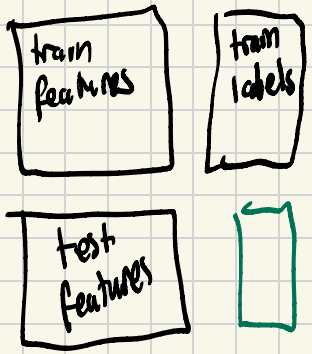
Choose learning rate, when $\mathcal{L}$ convex,

$$\alpha_t = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} \mathcal{L}(y^{(i)}, F_{t-1}(x^{(i)}) + \alpha_t f_t(x^{(i)}))$$

by setting derivative to 0 and solving for α

XGBoost was SOTA on tabular data from 2014-2025

TabPFN (Transformer for tabular data) is now SOTA



↳ in context learning