CS 480/680 Winter 2024:
Lecture Notes

Lecture notes taken, unless otherwise specified, by myself during section 002 of the Winter 2024 offering of CS 480/680, taught by Hongyang Zheng.

Lectures

Chapter 1

Classic Machine Learning

1.1 Introduction

There have been three historical AI booms: *Lecture 1 Jan 9*

1. 1950s–1970s: search-based algorithms (e.g., chess), failed when they realized AI is actually a hard problem
2. 1980s–1990s: expert systems
3. 2012 – present: deep learning

Machine learning is the subset of AI where a program can learn from experience.

Major learning paradigms of machine learning:

- Supervised learning: teacher/human labels answers (e.g., classification, ranking, etc.)
- Unsupervised learning: without labels (e.g., clustering, representation, generation, etc.)
- Reinforcement learning: rewards given for actions (e.g., gaming, pricing, etc.)
- Others: semi-supervised, active learning, etc.

Active focuses in machine learning research:

- Representation: improving the encoding of data into a space
- Generalization: improving the use of the model on new distributions
- Interpretation: understanding how deep learning actually works
- Complexity: improving time/space requirements
- Efficiency: reducing the amount of samples required
- Privacy: respecting legal/ethical concerns of data sourcing
- Robustness: gracefully failing under errors or malicious attack
- Applications

A machine learning algorithm has three phases: training, prediction, and evaluation. *Lecture 2 Jan 11*

Definition 1.1.1 (dataset)

A dataset consists of a list of features $\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{x}'_1, \dots, \mathbf{x}'_m \in \mathbb{R}^d$ which are d-dimensional vectors and a label vector $\mathbf{y}^\top \in \mathbb{R}^n$.

Each training sample $\mathbf{x}_i$ is associated with a label y_i. A test sample $\mathbf{x}'_i$ may or may not be labelled.

Example 1.1.2 (email filtering)**.** Suppose we have a list D of d English words.

Define the training set $X = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ and $\mathbf{y} = [y_1, \dots, y_n] \in \{\pm 1\}^n$ such that $\mathbf{x}_{ij} = 1$ if the word $j \in D$ appears in email i (this is the bag-of-words representation):

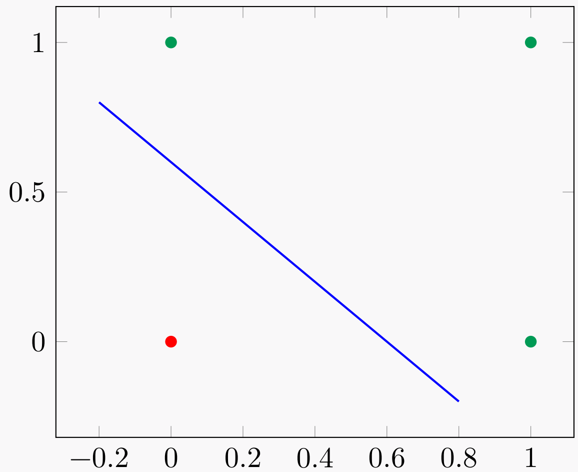
	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}'$
and	1	0	0	1	1	1	1
viagra	1	0	1	0	0	0	1
the	0	1	1	0	1	1	0
of	1	1	0	1	0	1	0
nigeria	1	0	0	0	1	0	0
y	$+$	$-$	$+$	$-$	$+$	$-$	?

Then, given a new email $\mathbf{x}'_1$, we must determine if it is spam or not.

Example 1.1.3 (OR dataset)**.** We want to train the OR function:

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$
	0	1	0	1
	0	0	1	1
y	$-$	$+$	$+$	$+$

This can be represented graphically by finding a line dividing the points:



1.2 Perceptron

Definition 1.2.1

The inner product of vectors $\langle \mathbf{a}, \mathbf{b} \rangle$ is the sum of the element-wise product $\sum_j a_j b_j$.

A linear function is a function $f : \mathbb{R}^d \to \mathbb{R}^d$ such that for all $\alpha, \beta \in \mathbb{R}$, $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d$, $f(\alpha\mathbf{x}+\beta\mathbf{z}) = \alpha f(\mathbf{x}) + \beta f(\mathbf{z})$.

Theorem 1.2.2 (linear duality)

A function is linear if and only if there exists $\mathbf{w} \in \mathbb{R}^d$ such that $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle$.

Proof. ($\Rightarrow$) Suppose f is linear. Let $\mathbf{w} := [f(\mathbf{e}_1), \dots, f(\mathbf{e}_d)]$ where $\mathbf{e}_i$ are coordinate vectors. Then:

$$\begin{aligned} f(\mathbf{x}) &= f(x_1\mathbf{e}_1 + \dots + x_d\mathbf{e}_d) \\ &= x_1 f(\mathbf{e}_1) + \dots + x_d f(\mathbf{e}_d) \\ &= \langle \mathbf{x}, \mathbf{w} \rangle \end{aligned}$$

by linearity of f.

($\Leftarrow$) Suppose there exists $\mathbf{w}$ such that $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle$. Then:

$$\begin{aligned} f(\alpha\mathbf{x} + \beta\mathbf{z}) &= \langle \alpha\mathbf{x} + \beta\mathbf{z}, \mathbf{w}, \alpha\mathbf{x} + \beta\mathbf{z}, \mathbf{w} \rangle \\ &= \alpha \langle \mathbf{x}, \mathbf{w} \rangle + \beta \langle \mathbf{x}, \mathbf{w} \rangle \\ &= \alpha f(\mathbf{x}) + \beta f(\mathbf{z}) \end{aligned}$$

since inner products are linear in the first argument. $\square$

Definition 1.2.3 (affine function)

A function $f(\mathbf{x})$ where there exist $\mathbf{w} \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$ such that $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle + b$.

Definition 1.2.4 (sign function)

$$\operatorname{sgn}(t) = \begin{cases} +1 & t > 0 \\ -1 & t \leq 0 \end{cases}$$

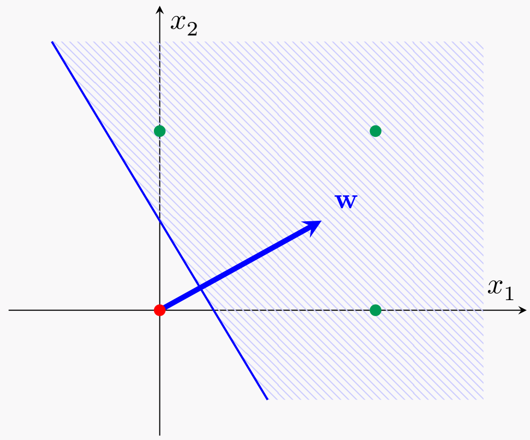
It does not matter what $\operatorname{sgn}(0)$ is defined as.

Definition 1.2.5 (linear classifier)

$\hat{y} = \operatorname{sgn}(\langle \mathbf{x}, \mathbf{w} \rangle + b)$

The parameters $\mathbf{w}$ and b will uniquely determine the linear classifier.

Example 1.2.6 (geometric interpretation)**.** We can interpret $\hat{y} > 0$ as a halfspace (see CO 250). Then, we can draw something like:



Proposition 1.2.7

The vector $\mathbf{w}$ is orthogonal to the decision boundary H.

Proof. Let $\mathbf{x}, \mathbf{x}' \in H$ be vectors on the boundary $H = \{x : \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$. Then, we must show $\mathbf{x}' - \mathbf{x} = \overrightarrow{\mathbf{x}\mathbf{x}'} \perp \mathbf{w}$.

We can calculate $\langle \mathbf{w}, \mathbf{x}' - \mathbf{x} \rangle = \langle \mathbf{w}, \mathbf{x} \rangle - \langle \mathbf{w}, \mathbf{x}' \rangle = -b - (-b) = 0$. □

Originally, the inventor of the perceptron thought it could do anything. He was (obviously) wrong.

Algorithm 1 Training Perceptron

Require: Dataset $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{\pm 1\}$, initialization $\mathbf{w}_0 \in \mathbb{R}^d$, $b_0 \in \mathbb{R}$.
Ensure: $\mathbf{w}$ and b for linear classifier $\operatorname{sgn}(\langle \mathbf{x}, \mathbf{w} \rangle + b)$

```
for t = 1, 2, ... do
    receive index I_t ∈ {1, ..., n}
    if y_{I_t}(⟨x_{I_t}, w⟩ + b) ≤ 0 then
        w ← w + y_{I_t} x_{I_t}
        b ← b + y_{I_t}
```

In a perceptron, we train by adjusting $\mathbf{w}$ and b whenever a training data feature is classified "wrong" (i.e., $\mathsf{score}_{\mathbf{w},b}(\mathbf{x}) := y\hat{y} < 0 \iff$ the signs disagree).

The perceptron solves the feasibility problem

$$\text{Find } \mathbf{w} \in \mathbb{R}^d,\ b \in \mathbb{R} \text{ such that } \forall i, y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) > 0$$

by iterating one-by-one. It will converge "faster" (with fewer t-iterations) if the data is "easy".

Consider what happens when there is a "wrong" classification. Let $w_{k+1} = w_k + yx$ and $b_{k+1} = b_k + y$.

Then, the updated score is:

$$\begin{aligned}
\mathsf{score}_{\mathbf{w}_{k+1},b_{k+1}}(\mathbf{x}) &= y \cdot (\langle \mathbf{x}, \mathbf{w}_{k+1} \rangle + b_{k+1}) \\
&= y \cdot (\langle \mathbf{x}, \mathbf{w}_k + y\mathbf{x} \rangle + b_k + y) \\
&= y \cdot (\langle \mathbf{x}, \mathbf{w}_k \rangle + b_k) + \langle \mathbf{x}, \mathbf{x} \rangle + 1 \\
&= y \cdot (\langle \mathbf{x}, \mathbf{w}_k \rangle + b_k) + \underbrace{\|\mathbf{x}\|_2^2 + 1}_{\text{always positive}}
\end{aligned}$$

which is always an increase over the previous "wrong" score.