The perceptron algorithm

Christos Dimitrakakis

October 10, 2023

Outline

The perceptron algorithm

Input

- Feature space $X \subset \mathbb{R}^n$.
- Label space $Y = \{-1, 1\}$.
- Data (x_t, y_t), $t \in [T]$, with $x_t \in X, y_t \in Y$.

Algorithm

- $w_1 = w_0$.
- For $t = 1, \ldots, T$.
  - $a_t = \text{sgn}(w_t^\top x_t)$.
  - If $a_t \neq y_t$
    - $w_{t+1} = w_t + y_t x_t$
  - Else
    - $w_{t+1} = w_t$
  - EndIf
- Return w_{T+1}

Perceptron examples

Example 1: One-dimensional data

- Done on the board
- Shows how the algorithm works.
- Demonstrates the idea of a margin

Example 2: Two-dimensional data

- See in-class programming exercise

The Perceptron Theorem

The number of mistakes made by the perceptron algorithm is bounded by ρ^{-2}, where $\|x_t\| \leq 1$, $\rho \leq y_t(x_t^t opw^*)$ for some margin ρ and hyperplane w^* with $\|w^*\| = 1$.

Hyperplane w^*

Separates the examples

Margin ρ

The minimum distance ρ between the hyperplane and any point.

Simple proof

- Scale data: $\|x\| \leq 1$
- Separating plane: $y_t(x_t^\top w^*) > 0$, $\|w^*\| = 1$.
- When we make an update: $y_t(x_t^\top w_t) \leq 0$.
- At each mistake, $w^\top w^*$ grows by at least ρ

$$(w + yx)^\top w^* = w^\top w^* + y(x^\top w^*) \geq w^\top w^* + \rho$$

- At each mistake, $w^\top w$ grows by at most 1.

$$(w + yx)^\top (w + yx) = w^\top w + 2y(w^\top x) + y^2(x^\top x) \leq w^\top w + 1$$

Putting it together

After M mistakes:

- $w^\top w^* \geq M\rho$
- $w^\top w \leq M$

So $M\rho \leq w^\top w^* \leq \|w\| = \sqrt{w^\top w} \leq \sqrt{M}$.

- Thus, $M \leq \rho^{-2}$.

Why doesn't the perceptron always work?

- Classes must be linearly separable

Example: XOR

The gradient method

- Function to minimise $f(\theta)$.
- Derivative $\nabla_\theta f(\theta)$.

Gradient descent algorithm

- Input: initial value θ_0, learning rate schedule α_t
- For $t = 1, \dots, T$
  - $\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta f(\theta_t)$
- Return θ_T

Properties

- If $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$, it finds a local minimum θ_T, i.e. there is $\epsilon > 0$ so that

$$f(\theta_T) < f(\theta), \forall \theta : \|\theta_T - \theta\| < \epsilon.$$

Stochastic gradient method

This is the same as the gradient method, but with added noise:

- $\theta_{t+1} = \theta_t - \alpha_t[\nabla_\theta f(\theta_t) + \omega_t]$
- $\mathbb{E}[\omega_t] = 0$ is sufficient for convergence.

Example: When the cost is an expectation

In machine learning, the cost is frequently an expectation of some function ℓ,

$$f(\theta) = \int_X dP(x)\ell(x, \theta)$$

This can be approximated with a sample

$$f(\theta) \approx \frac{1}{T}\sum_t \ell(x_t, \theta)$$

The same holds for the gradient:

$$\nabla_\theta f(\theta) = \int_X dP(x)\nabla_\theta \ell(x, \theta) \approx \frac{1}{T}\sum_t \nabla_\theta \ell(x_t, \theta)$$

Gradient methods for expected value

Estimate the expected value

$x_t \sim P$ with $\mathbb{E}_P[x_t] = \mu$.

Objective: mean squared error

Here $\ell(x, \theta) = (x - \theta)^2$.

$$\min_\theta \mathbb{E}_P[(x_t - \theta)^2].$$

Derivative

Idea: at the minimum the derivative should be zero.

$$d/d\theta\, \mathbb{E}_P[(x_t - \theta)^2] = \mathbb{E}_P[d/d\theta(x_t - \theta)^2] = \mathbb{E}_P[-(x_t - \theta)] = \mathbb{E}_P[x_t] - \theta.$$

Setting the derivative to 0, we have $\theta = \mathbb{E}_P[x_t]$. This is a simple solution.

Real-world setting

- The objective function does not result in a simple solution
- The distribution P is not known.
- We can sample $x \sim P$.

Stochastic gradient for mean estimation

- The gradient is zero when the parameter is the expected value

$$
\begin{aligned}
\frac{d}{d\theta}\,\mathbb{E}_P[(x-\theta)^2] &= \int_{-\infty}^{\infty} dP(x)\frac{d}{d\theta}(x-\theta)^2 \\
&= \int_{-\infty}^{\infty} dP(x)2(x-\theta) \\
&= 2\,\mathbb{E}_P[x] - 2\theta.
\end{aligned}
$$

- If we sample x we approximate the gradient:

$$
\begin{aligned}
\frac{d}{d\theta}\,\mathbb{E}_P[(x-\theta)^2] &= \int_{-\infty}^{\infty} dP(x)\frac{d}{d\theta}(x-\theta)^2 \\
&\approx \frac{1}{T}\sum_{t=1}^{T}\frac{d}{d\theta}(x_t-\theta)^2 = \frac{1}{T}\sum_{t=1}^{T}2(x_t-\theta)
\end{aligned}
$$

Perceptron algorithm as gradient descent

- Target error function $\mathbb{E}_P^w[\ell] = \int_X dP(x) \sum_y P(y|x)\ell(x, y, w)$
- Empirical error function $\frac{1}{T}\sum_{t=1}^T \ell(x_t, y_t, w)$, $x_t, y_t \sim P$.

Perceptron cost function

The cost of each example

$$\ell(x, y, w) = -\mathbb{I}\left\{y(x^\top w) < 0\right\} y(x^\top w)$$

Derivative: Chain rule

- $\nabla_w \ell(x, y, w) = -\mathbb{I}\left\{y(x^\top w) > 0\right\} \nabla_w [y(x^\top w)]$.
- $\partial w / \partial w^i [y(x_t^\top w)] = y x_{t,i}$
- Gradient update: $w_{t+1} = w_t - \nabla_w \ell(x, y, w) = w_t + y x_t$

Classification error cost function

This is not differentiable :(

Logistic regression

Output as a measure of confidence

- $P_w(y = 1|x) = \frac{1}{1+\exp(-x_t^\top w)}$

Negative Log likelihood

- $\ell(x_t, y_t, w) = -\ln P_w(y_t|x_t) = \ln(1 + \exp(-y_t x_t^\top w))$

$$
\begin{aligned}
\nabla_w \ell(x_t, y_t, w) &= \frac{1}{1 + \exp(-y x_t^\top w)} \nabla_w [1 + \exp(-y x_t^\top w)] \\
&= \frac{1}{1 + \exp(-y x_t^\top w)} \exp(-y x_t^\top w) [\nabla_w (-y_t x_t^\top w)] \\
&= -\frac{1}{1 + \exp(x_t^\top w)} (x_{t,i})_{i=1}^n e
\end{aligned}
$$

- $\mathbb{E}_P(\ell) = \int_X dP(x) \sum_{y \in Y} P(y|x) P_w(y_t + x_t)$

Lab demonstration

- How to use kNN and LogisticRegression with sklearn (and perhaps statsmodels, time permitting)
- Use an example where there is no default 'class' label

Assignment

1. Find a dataset with some categorical variable of interest that we want to predict from the UCI repository.
2. Formulate the appropriate classification problem.
3. Perform model selection through train/validate or crossvalidation to find the best model and hyperparameters
4. Measure the model's final performance on the test set.
5. Discuss anything of interest in the data such as: feature scaling/selection, missing data, outliers.