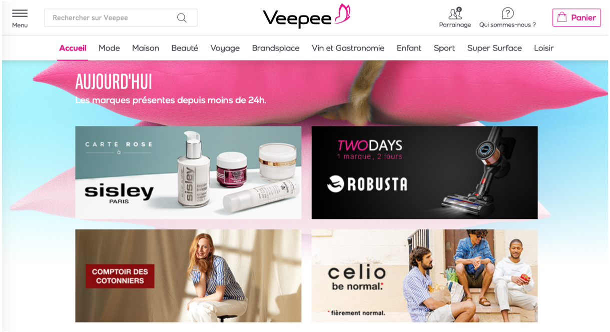
A Streaming Algorithm for Logistic Regression

Thomas Bonald
Joint work with Till Wohlfarth (Veepee)

DIG Seminar
June 2021

Motivation

How to **rank items** according to user preferences?



Veepee website

Binary classification

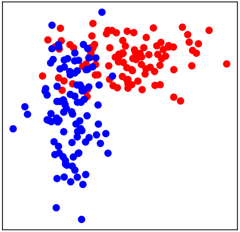
Given

- samples $x_1, \ldots, x_n \in \mathbb{R}^d$ (items)
- labels $y_1, \ldots, y_n \in \{0, 1\}$ (click / no click)

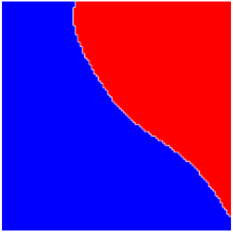
learn some mapping $f : \mathbb{R}^d \to \{0, 1\}$ such that:

$$\forall i, \quad y_i \approx f(x_i)$$

Samples



Decision



Regression

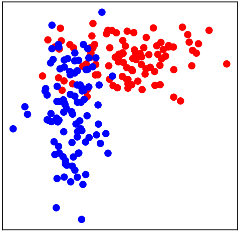
Given

- samples $x_1, \ldots, x_n \in \mathbb{R}^d$ (items)
- labels $y_1, \ldots, y_n \in \{0, 1\}$ (click / no click)

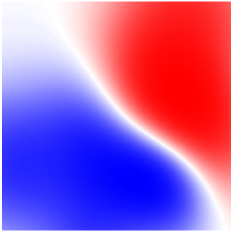
learn some mapping $f : \mathbb{R}^d \to [0, 1]$ such that:

$$\forall i, \quad y_i \approx f(x_i)$$

Samples



Decision



Logistic regression

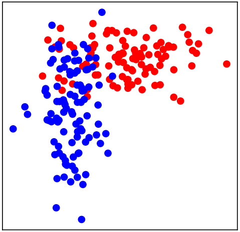
Given

- samples $x_1, \ldots, x_n \in \mathbb{R}^d$ (items)
- labels $y_1, \ldots, y_n \in \{0, 1\}$ (click / no click)

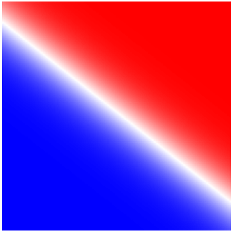
learn some parameter θ such that:

$$\forall i, \quad y_i \approx p_i(\theta) = \frac{1}{1 + e^{-\theta^T x_i}}$$

Samples



Decision



Loss function

Given

- samples $x_1, \ldots, x_n \in \mathbb{R}^d$ (items)
- labels $y_1, \ldots, y_n \in \{0, 1\}$ (click / no click)

learn some parameter θ such that:

$$\forall i, \quad y_i \approx p_i(\theta) = \frac{1}{1 + e^{-\theta^T x_i}}$$

Regularized cross-entropy

$$L(\theta) = -\sum_{i=1}^{n} \left[y_i \log p_i(\theta) + (1 - y_i) \log(1 - p_i(\theta)) \right] + \frac{\lambda}{2} ||\theta||^2$$

Outline

1. Offline learning
2. Online learning
3. Experiments
4. Summary

A convex optimization problem

Objective:

$$\arg\min_{\theta \in \mathbb{R}^d} L(\theta)$$

Gradient

$$\nabla L(\theta) = \lambda\theta + \sum_{i=1}^{n} x_i (p_i(\theta) - y_i)$$

Hessian matrix

$$\nabla^2 L(\theta) = \lambda I + \sum_{i=1}^{n} p_i(\theta)(1 - p_i(\theta)) x_i x_i^T \quad > 0$$

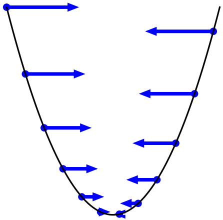
Gradient descent

Algorithm

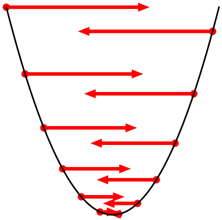
Iterate:

$$\theta \leftarrow \theta - \gamma \nabla L(\theta)$$

where γ is the learning parameter.



Small γ



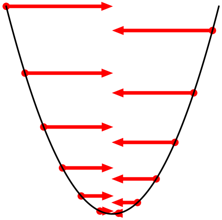
Large γ

Newton's method

Algorithm

Iterate:

$$\theta \leftarrow \theta - \nabla^2 L(\theta)^{-1} \nabla L(\theta)$$

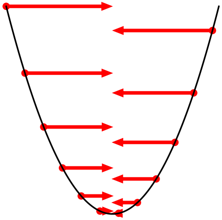


Newton's method

Algorithm

Iterate:

$$\theta \leftarrow \theta - \nabla^2 L(\theta)^{-1} \nabla L(\theta)$$



Note: Follows from the minimization in t of

$$L(\theta + t) \approx L(\theta) + t^T \nabla L(\theta) + \frac{1}{2} t^T \nabla^2 L(\theta) t$$

Outline

1. Offline learning
2. **Online learning**
3. Experiments
4. Summary

Gradient descent

From

$$\nabla L(\theta) = \lambda\theta + \underbrace{\sum_{i=1}^{n} x_i(p_i(\theta) - y_i)}_{S}$$

we get

$$\theta \leftarrow (1 - \lambda\gamma)\theta - \gamma S$$

Online algorithm

$\theta \leftarrow 0$, $S \leftarrow 0$
For each new sample (x, y):

$$p \leftarrow \frac{1}{1 + e^{-\theta^T x}}$$
$$S \leftarrow S + x(p - y)$$
$$\theta \leftarrow (1 - \lambda\gamma)\theta - \gamma S$$

Newton's method

The update is

$$\theta \leftarrow \theta - \nabla^2 L(\theta)^{-1} \nabla L(\theta)$$

with

$$\nabla L(\theta) = \lambda\theta + \sum_{i=1}^{n} x_i (p_i(\theta) - y_i)$$

$$\nabla^2 L(\theta) = \lambda I + \sum_{i=1}^{n} p_i(\theta)(1 - p_i(\theta)) x_i x_i^T$$

How to compute the **inverse** of $\nabla^2 L(\theta)$ in a streaming fashion?

The Sherman-Morrison formula

Proposition

For any invertible matrix A and vectors u, v,

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u}$$

Sherman & Morrison 1949

Newton's method

Online algorithm

$\theta \leftarrow 0$, $S \leftarrow 0$, $\Gamma \leftarrow I/\lambda$

For each new sample (x, y):

$$
\begin{aligned}
p &\leftarrow \frac{1}{1 + e^{-\theta^T x}} \\
v &\leftarrow p(1 - p) \\
S &\leftarrow S + x(y - p + v\theta^T x) \\
\Gamma &\leftarrow \Gamma - \frac{v\Gamma x x^T \Gamma}{1 + v x^T \Gamma x} \\
\theta &\leftarrow \Gamma S
\end{aligned}
$$

Outline

1. Offline learning
2. Online learning
3. **Experiments**
4. Summary

Experiments

Classification

$$y, \hat{y} \in \{0, 1\}$$

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}}$$

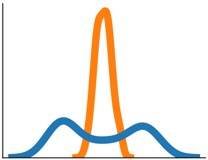
OpenML

Ranking

$$p, \hat{p} \in [0, 1]$$

$$\text{FCP} \propto \sum_{i,j} 1_{(p_i - p_j)(\hat{p}_i - \hat{p}_j) > 0}$$

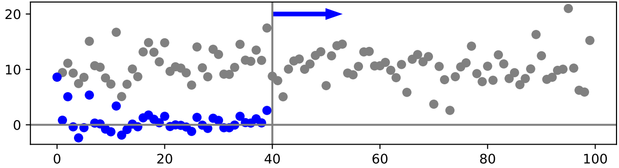
(Fraction of Concordant Pairs)



Preprocessing

Samples are **centered** and **rescaled** (in stream) before learning

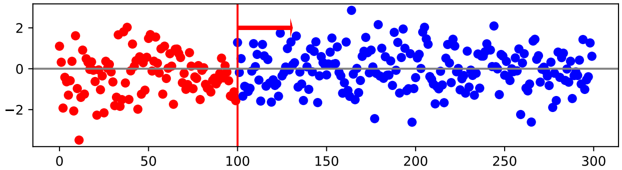
Standard scaler



Scenarios

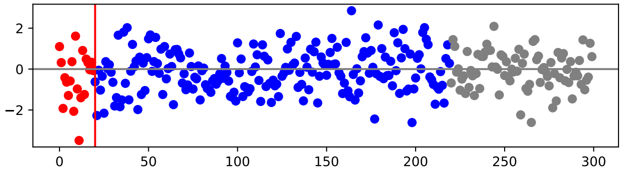
1. **All samples**

At each time t, predict sample x_t then learn using label y_t



2. **Cold start**

Learn until time $t = 20$ and test over the next 200 samples

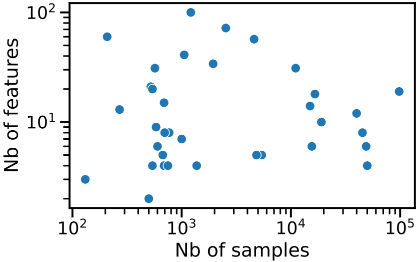


OpenML datasets

All OpenML datasets with:

- binary labels
- no missing data
- $2 \leq d \leq 100$ features
- $n \geq 100$ samples
- accuracy < 0.99 (offline)

$\rightarrow$ **36 datasets**

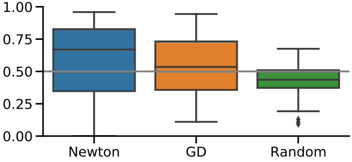


See `www.openml.org`

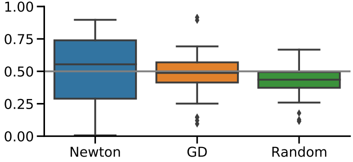
Classification

Distribution of **F1 score** over the 36 OpenML datasets

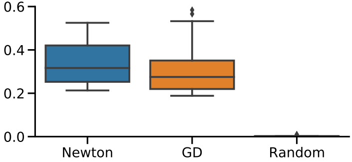
All samples



Cold start



Running time for learning (ms)



Synthetic data

Let $\mathcal{S}$ be the unit sphere in dimension d

Model

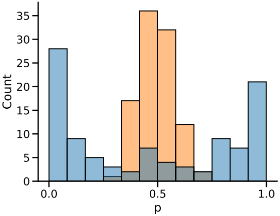
- $\theta \sim \mathcal{U}(\mathcal{S})$
- $x \sim \mathcal{U}(\mathcal{S})$
- $y \sim \mathcal{B}(p)$ with

$$p = \frac{1}{1 + e^{-\alpha \theta^T x}}$$

Example

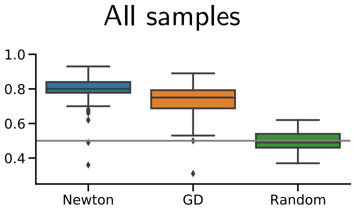
Distribution of p

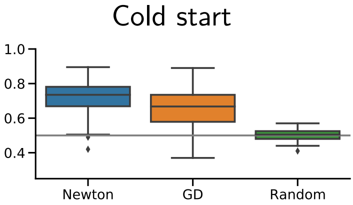
$n = 100, d = 10, \alpha \in \{1, 10\}$



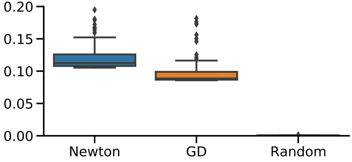
Ranking ($\alpha = 10$)

Distribution of **FCP** over 100 independent instances



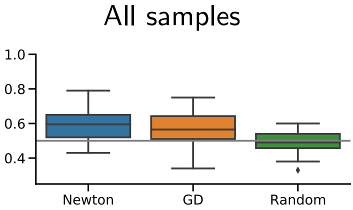


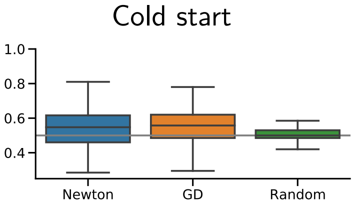
Running time for learning (ms)



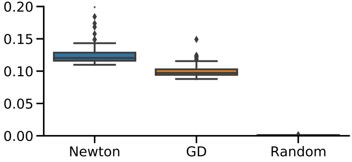
Ranking ($\alpha = 1$)

Distribution of **FCP** over 100 independent instances





Running time for learning (ms)

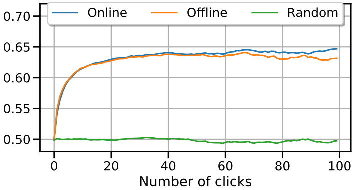


Application to Veepee data

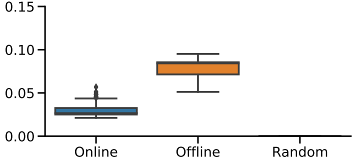
Data collected in April 2021:

- 10 days
- 105,170 new users
- $d = 132$ binary features

FCP



Running time (ms)



Summary

A novel streaming algorithm for logistic regression, based on

- **Newton's method**
- the **Sherman-Morrison** formula

Efficient and **parameter-free**

Future work:

- Benchmarking
- Convergence
- High dimension
- Control (bandit algorithm)

River