(つ◕ᴗ◕)つ—☆゜.*･。゜

GDG DevFest
Season 2016

Improve Your App Ratings Using Machine Learning

Gianluca Segato

founder & lead Android dev @ Uniwhere

gianluca@uniwhere.com

Agenda

stuff we’re going to talk about

Agenda

- **Context** – What you should expect
- **Stuff** you'll need

- **Problem** – What are we addressing?
- **Workflow**

- High level **solution** – The architecture

Agenda

- Let’s **code**!
  - Android
  - Python
- Conclusions & **Q&A**

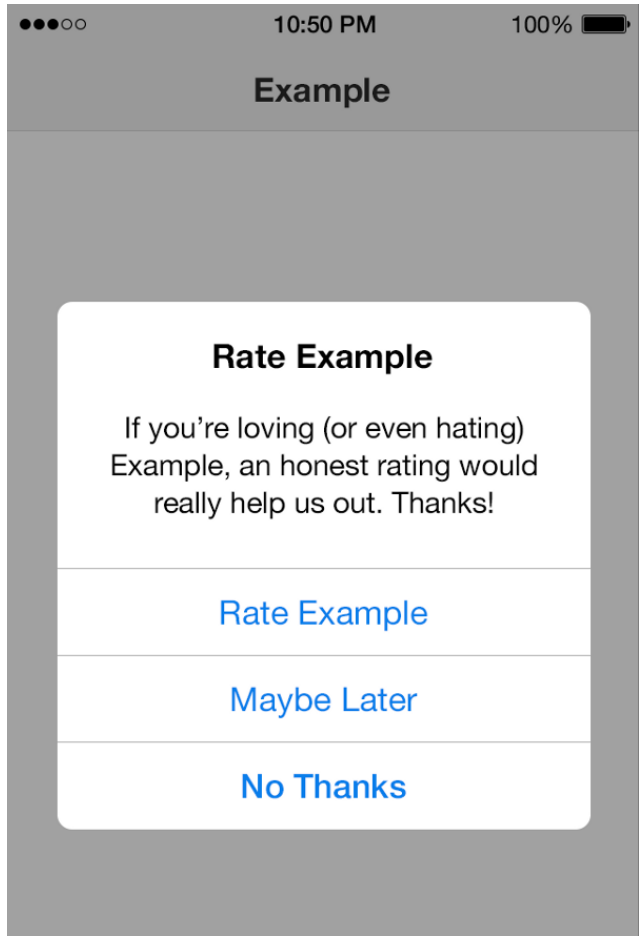
What you **should** expect

- How to deploy a ML system in the context of mobile application infrastructure (client-backend)
- An insight on how Firebase can partly get rid of a backend

What you **shouldn’t** expect

- A **production-level deliverable**: more like a MVP
- An **academic-level Machine Learning implementation**
  -> for that there’s a 2-year Master’s Degree in Statistics

Do you want to rate my app?



Prompted after:

- 5 days of usage

OR

- 12 app openings

OR

- 5 days **and** 12 app openings

Never know when/to whom you should!

I would like to ask only those people tha
I know are more likely to say yes.

In Uniwhere

- 80,000 users, 30% iOS, 65% Android, 5% WP
- Audience: University students
- Given that:
  - Bad reviews happen because of issues
  - Good reviews must be asked… otherwise people forget about it
- We wanted to improve our ratings by asking the right people at the right moment for a review

The problem

- **Whom** should I ask for a review?
- **When** should I ask for a review?

CLASSIFICATION PROBLEM

Machine Learning

- Supervised learning: using labeled data, trying to predict values
  - **Regression**: real output (ie. a price)
  - **Classification**: predict **categorical data** (ie. 0 or 1)

Output

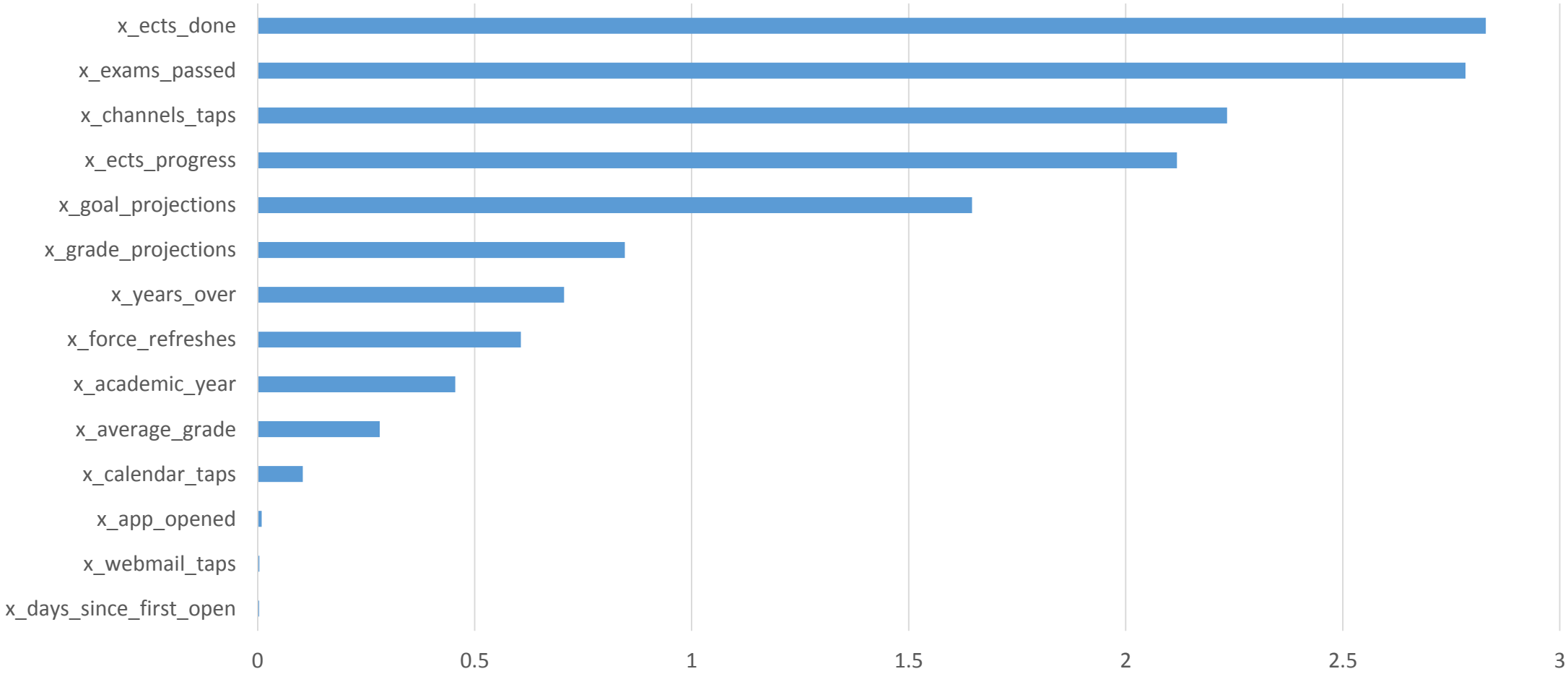
What are we going to build?

Is this going to work?

- For **Uniwhere**, after 2 weeks in production: **88% accuracy**

- The results were outstanding!
  - Timing has little to no impact
  - Usage features have a lot instead!
  - And also does user profile data!

Feature Informativeness



Formalization

We want to predict in advance whether a user will say yes to the question, based on his/her behavior and characteristics

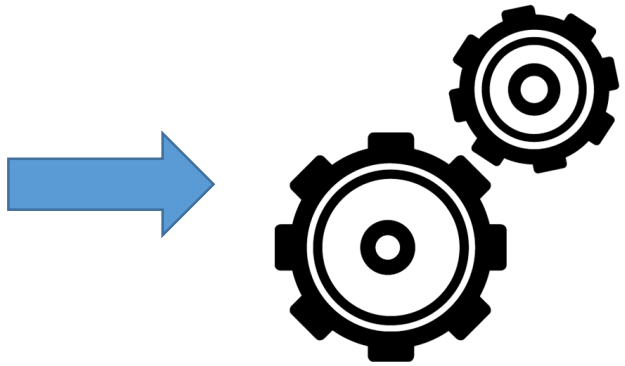
$$function(user) = \mathcal{P}(user\ says\ yes \mid user, time)$$

$$user = \begin{bmatrix} x_1 = \text{pictures shared} \\ \dots \\ x_n = \text{game score} \end{bmatrix} = \boldsymbol{x}$$

Classification Algorithms

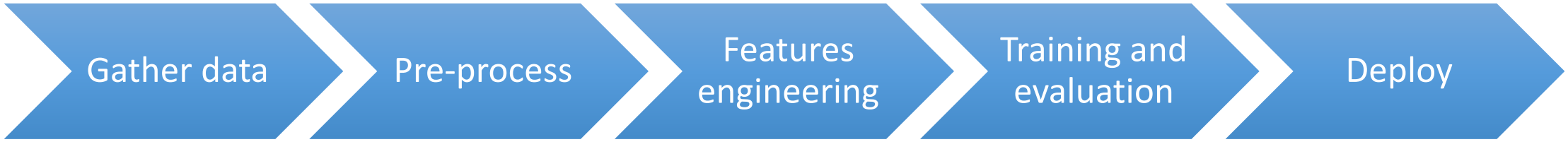
- Logistic Regression
- SVM
- Vanilla Neural Network

user behavior



$$\text{group} = \begin{cases} 0 = no \\ 1 = yes \end{cases}$$

Workflow


Gather data
Pre-process
Features engineering
Training and evaluation
Deploy

A Two-Stage Process

- **First stage**: Model training
  - gather data
  - build the model upon it
- **Second stage**: Deployment of prediction model
  - deploy the model
  - adapt online (retraining at each step)

The System

architecture and stuff

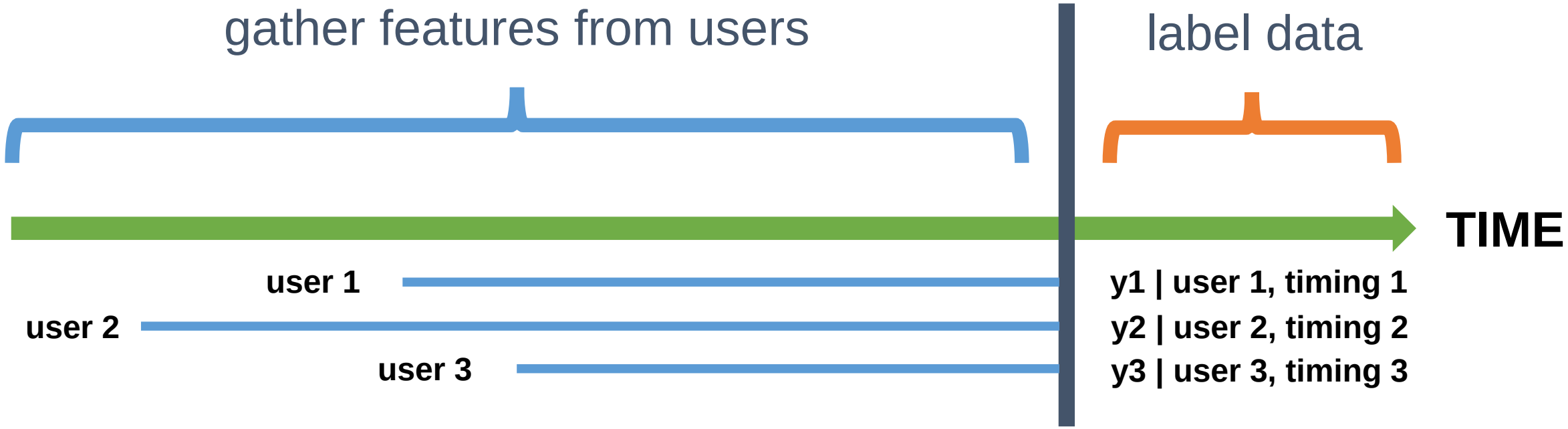
Firebase

- Store and sync data in real time
- Both a:
  - storage layer
  - communication layer
- NoSQL approach: you represent data as nodes, without tables -> **JSON**
- Free up to a (high) point, and scalable

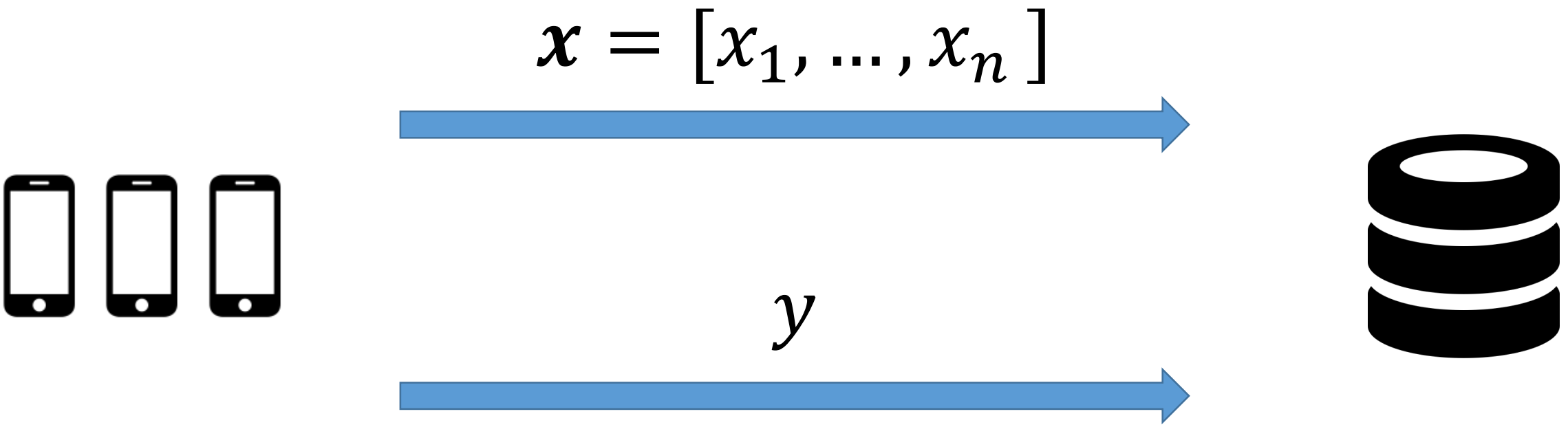
First Stage

Training

First Stage: Model Building



First Stage: Model Training



(at a certain point)

Model Building

writes features on `{userId}/x_...`

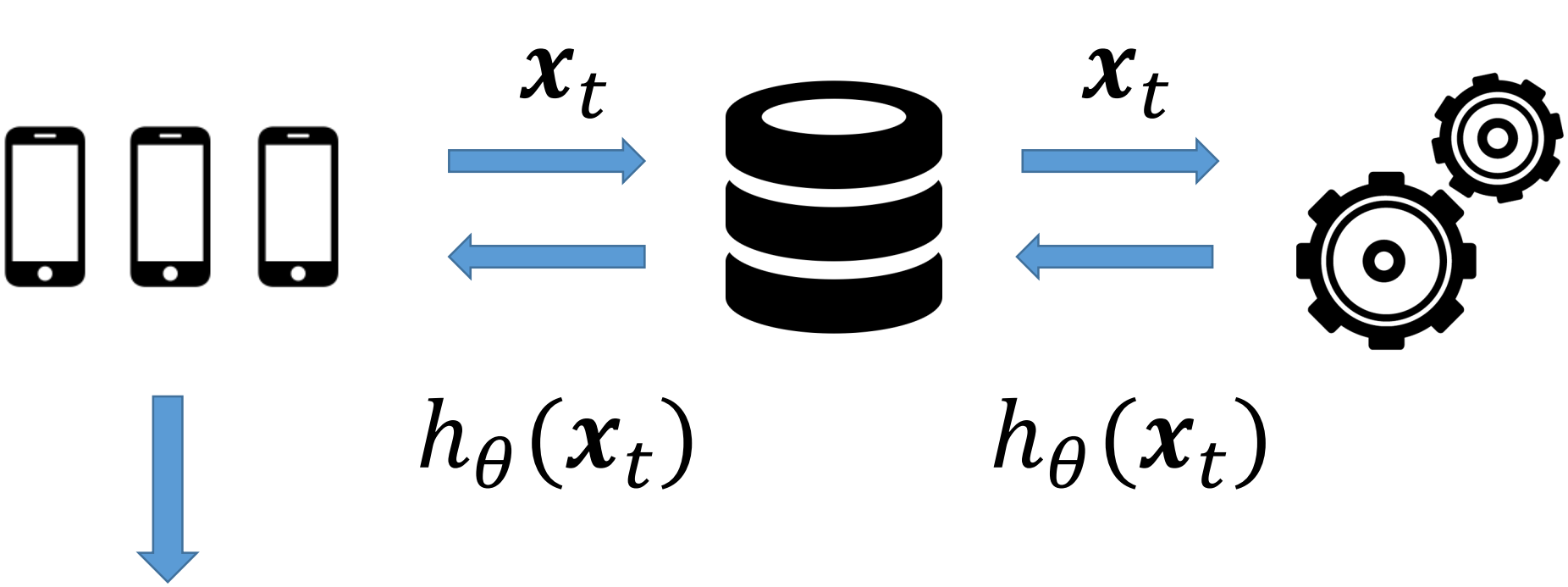
observes RemoteConfig `label_data`

if `label_data == true`, writes on `{userId}/y_observed`

Second Step

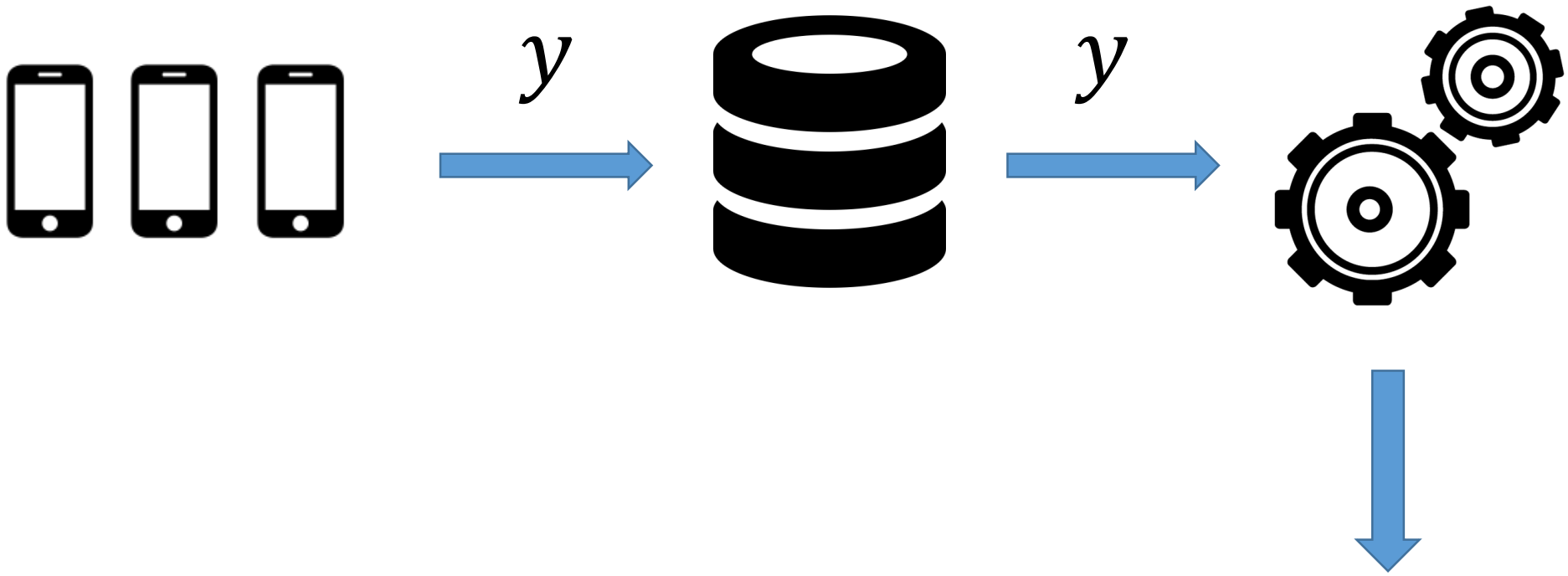
Deployed prediction model

Architecture



when $h_\theta(x_t)$
$> threshold$, **ask**

Online Prediction and Re-training



Implementation

Final System: Client Side

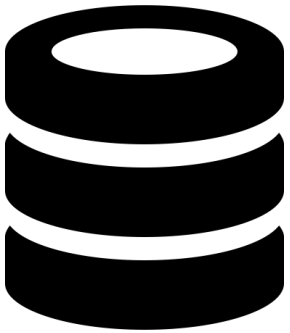
writes features on `{userId}/x_...`

observes `{userId}/action`

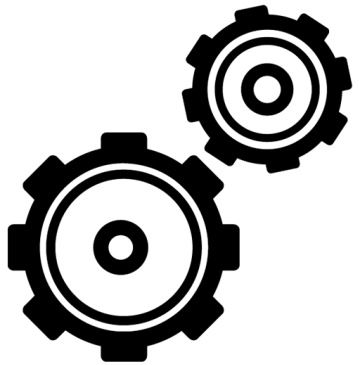
so that if `action == 1` => ask

Final System: Backend Side

observes `{userId}/x_...`



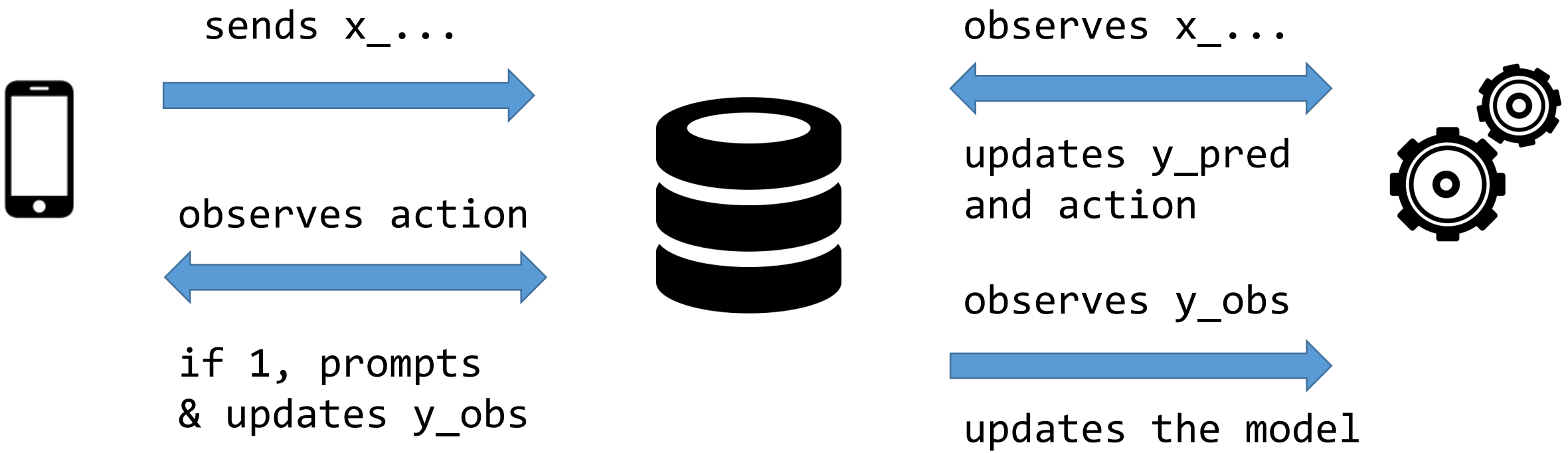
in order to update
`{userId}/y_prediction`
and `{userId}/action`



observes `{userId}/y_observed`

in order to update the model

Final System: Recap



Stuff you'll need

Starting App Rep:

`http://bit.ly/gdg-android`

Python backend Rep:

`http://bit.ly/gdg-android-backend`

https://pip.pypa.io/

```
pip install -r requirements.txt
```

Reference Slides

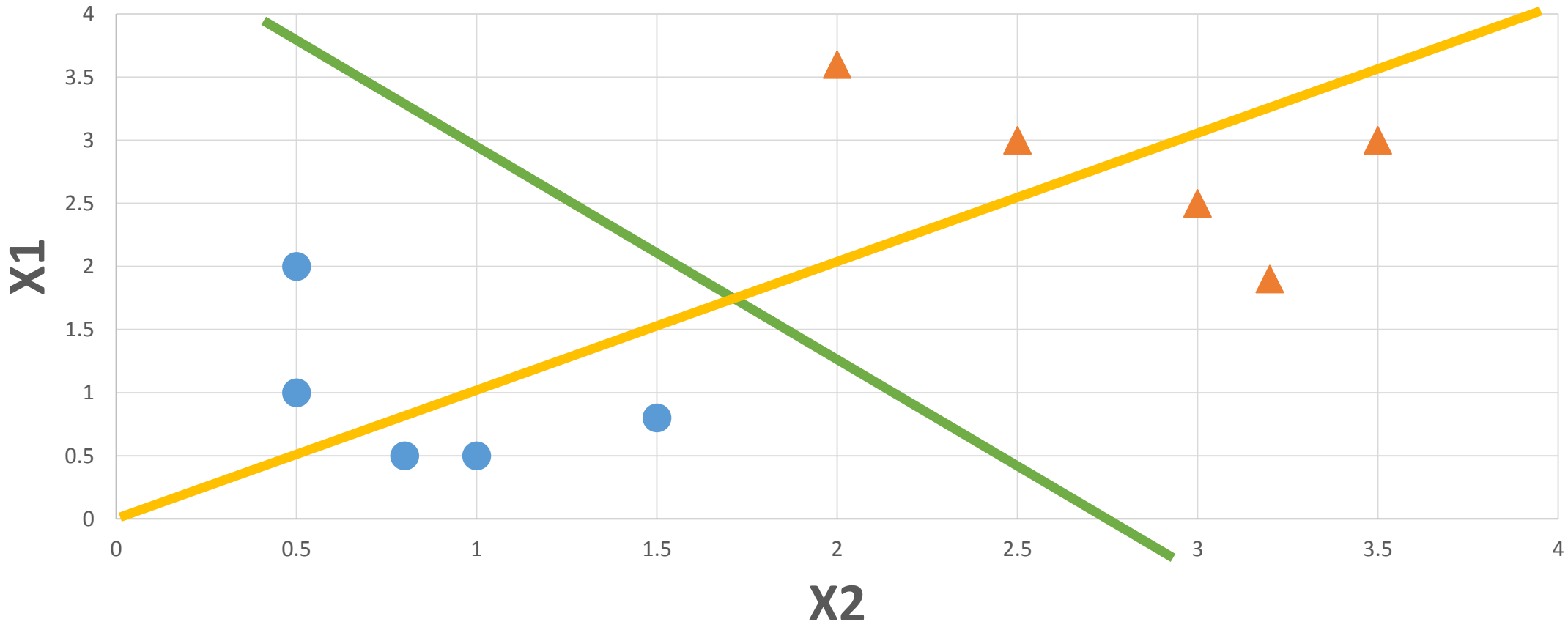
Issue: Precision and Recall

- If we take no action when `y_prediction < threshold`, then the system will never correct the model in case of false negatives (Type II error)

- In other words, the model update itself overtime just maximizing the precision by minimizing false positives (Type I errors), without touching the recall

$$\text{Precision} = \frac{tp}{tp + fp} \qquad \text{Recall} = \frac{tp}{tp + fn}$$

Formalization

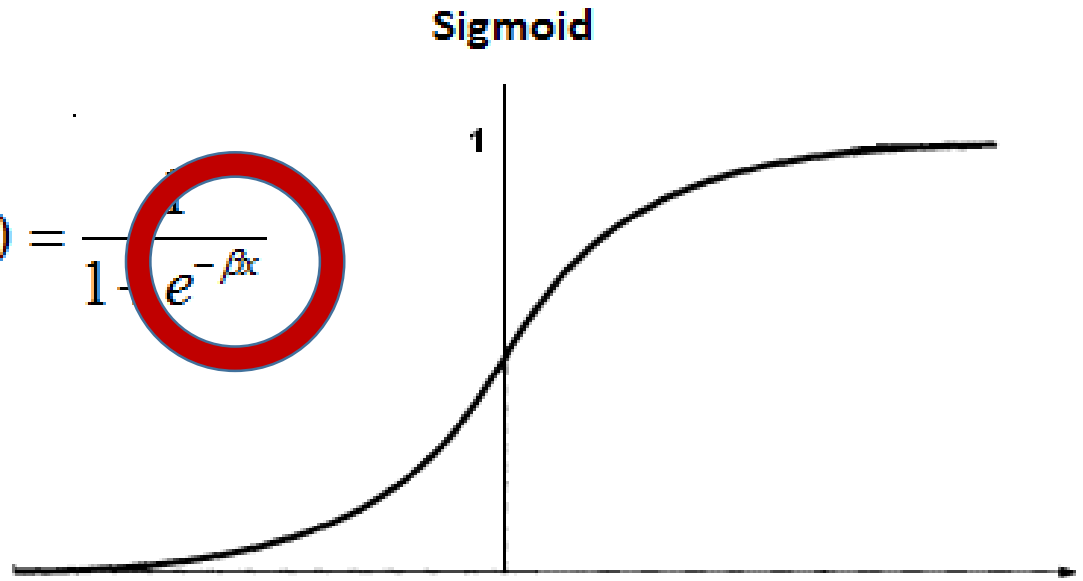


Sigmoid Function

$$h(z) = \frac{1}{1 + e^{-z}}, \qquad z = \boldsymbol{\theta}^T \boldsymbol{x}$$

Sigmoid Function

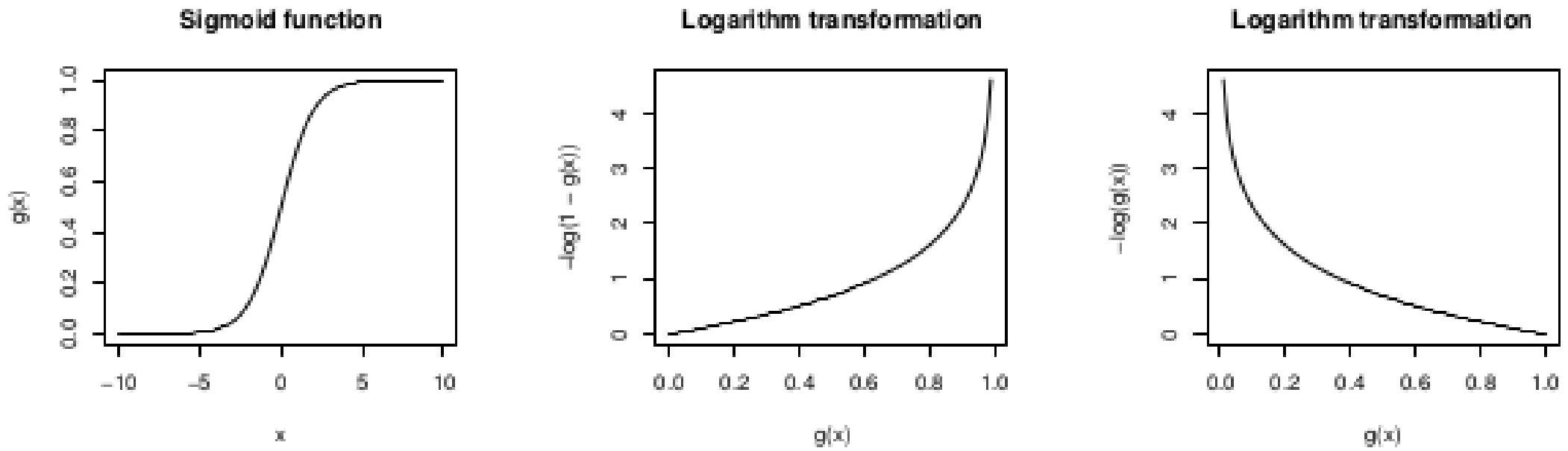
$$f(x) = \frac{1}{1 + e^{-\beta x}}$$



Sigmoid Function

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$$

$$Cost(h_\theta(x^{(i)}), y^{(i)}) = \begin{cases} -\log(h_\theta(x^{(i)})) & \text{if } y = 1 \\ -\log(1 - h_\theta(x^{(i)})) & \text{if } y = 0 \end{cases}$$

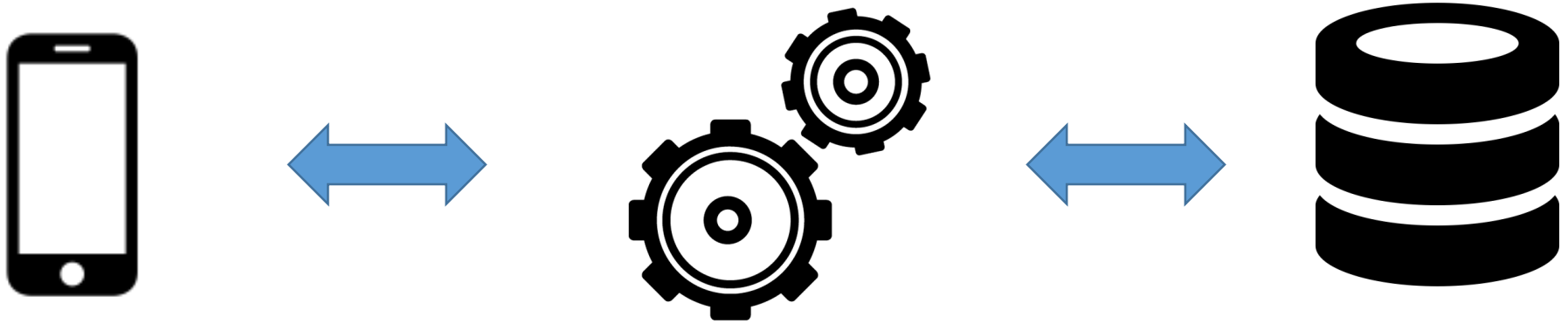


(a) *Sigmoid function.* **(b)** *Cost for* $y = 0$. **(c)** *Cost for* $y = 1$.

Figure B.1: *Logarithmic transformation of the sigmoid function.*

Firebase

It was:



Now: