Machine Learning as Extreme TDD: An Introduction

Srikumar Karaikudi Subramanian
http://sriku.org
sriku@imaginea.com

Director Technology @ Pramati Technologies
https://labs.imaginea.com
6 Mar 2019

Overview

Overview

- Motivation and prelude

Overview

- Motivation and prelude
- A soft problem: Classifying gender by name

Overview

- Motivation and prelude
- A soft problem: Classifying gender by name
- Testing our gender classifier

Overview

- Motivation and prelude
- A soft problem: Classifying gender by name
- Testing our gender classifier
- Writing our gender classifier

Overview

- Motivation and prelude
- A soft problem: Classifying gender by name
- Testing our gender classifier
- Writing our gender classifier
- Learners and features

Overview

- Motivation and prelude
- A soft problem: Classifying gender by name
- Testing our gender classifier
- Writing our gender classifier
- Learners and features
- The learning cycle

Overview

- Motivation and prelude
- A soft problem: Classifying gender by name
- Testing our gender classifier
- Writing our gender classifier
- Learners and features
- The learning cycle
- The road ahead

Overview

- Motivation and prelude
- A soft problem: Classifying gender by name
- Testing our gender classifier
- Writing our gender classifier
- Learners and features
- The learning cycle
- The road ahead
- Concluding notes

Overview

- Motivation and prelude
- A soft problem: Classifying gender by name
- Testing our gender classifier
- Writing our gender classifier
- Learners and features
- The learning cycle
- The road ahead
- Concluding notes

Motivation

Motivation

- Machine learning is eating software

Motivation

- Machine learning is eating software
- Non-ML folks must be able to participate

Motivation

- Machine learning is eating software
- Non-ML folks must be able to participate
- Critical thinking is valuable and available

Motivation

- Machine learning is eating software
- Non-ML folks must be able to participate
- Critical thinking is valuable and available
- Discovering ML application areas is valuable

Prelude

Prelude

- This talk is **not** for ML experts

Prelude

- This talk is **not** for ML experts
- Hoping to address devs and QA engineers

A personal anecdote

… from a long time ago
… in GWBASIC days

A personal anecdote

… from a long time ago
… in GWBASIC days

```
Enter your name: Kumar
```

A personal anecdote

… from a long time ago
… in GWBASIC days

```
Enter your name: Kumar

Hello Mr. Kumar
```

A personal anecdote

… from a long time ago
… in GWBASIC days

```
Enter your name: Kumar

Hello Mr. Kumar

Enter your name: Shobana
```

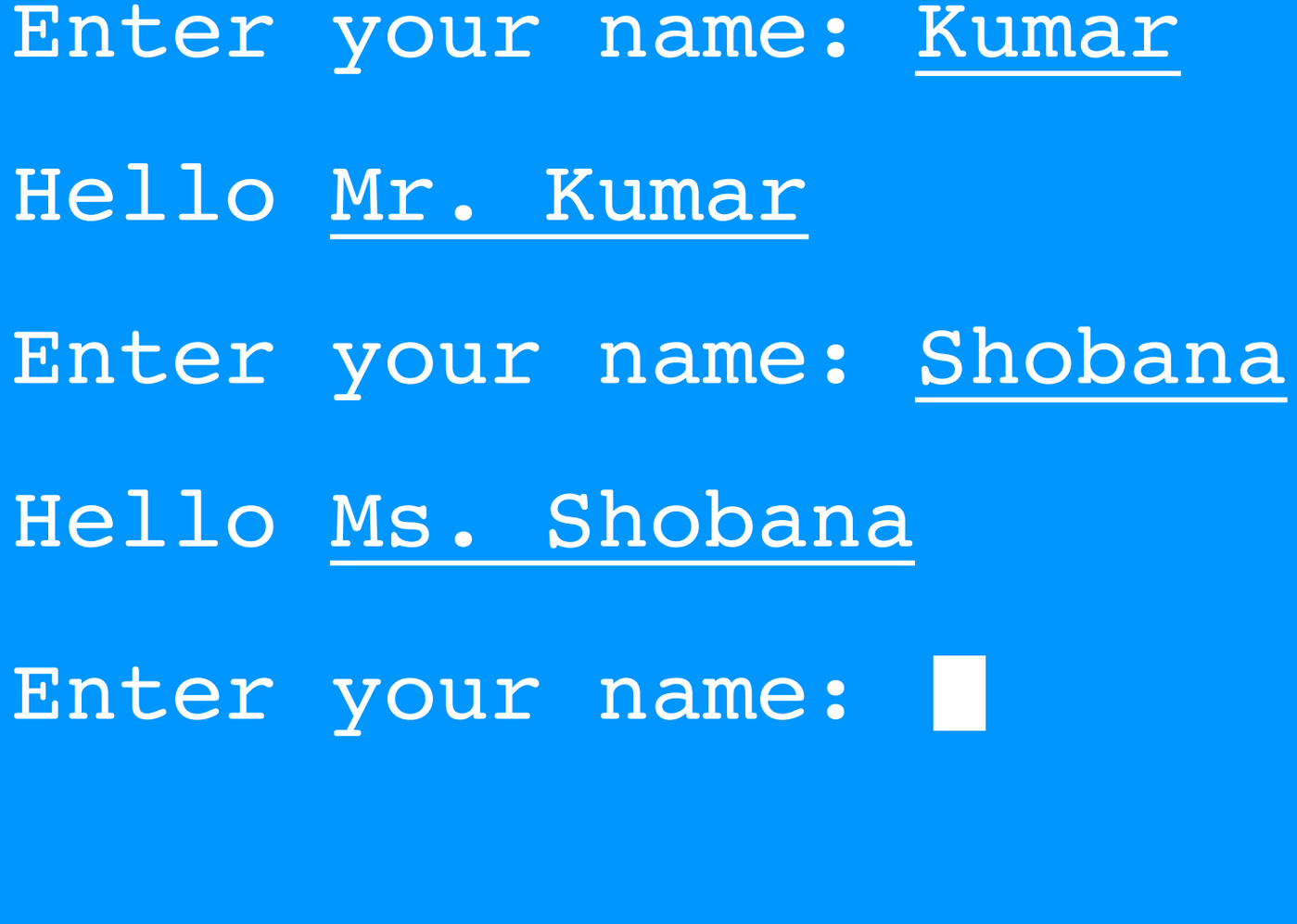
A personal anecdote

… from a long time ago
… in GWBASIC days

```
Enter your name: Kumar

Hello Mr. Kumar

Enter your name: Shobana

Hello Ms. Shobana
```

A personal anecdote

… from a long time ago
… in GWBASIC days



Guessing one's gender

Guessing one's gender

```
type GenderGuesser = String → Gender
```

Guessing one's gender

```
type GenderGuesser = String → Gender

data Gender = Man | Woman
```

Guessing one's gender

```
type GenderGuesser = String → Gender

data Gender = Man | Woman
```

How would you **test** the function?

Guessing one's gender

```
type GenderGuesser = String → Gender

data Gender = Man | Woman
```

How would you **test** the function?

(Restricting to Indian names for simplicity)

Some questions to ask

Some questions to ask

- Are “man” and “woman" the only two categories relevant to our situation?

Some questions to ask

- Are “man” and “woman" the only two categories relevant to our situation?
- How do I get a decent list of names with known gender?

Some questions to ask

- Are “man” and “woman" the only two categories relevant to our situation?
- How do I get a decent list of names with known gender?
- What anomalies exist in the set that I need to know about?

Some questions to ask

- Are “man” and “woman" the only two categories relevant to our situation?
- How do I get a decent list of names with known gender?
- What anomalies exist in the set that I need to know about?
- What do we know about the quality of the data?

Some questions to ask

- Are “man” and “woman" the only two categories relevant to our situation?
- How do I get a decent list of names with known gender?
- What anomalies exist in the set that I need to know about?
- What do we know about the quality of the data?
  - i.e How do we trust the tests we base it on?

Some questions to ask

- Are “man” and “woman" the only two categories relevant to our situation?
- How do I get a decent list of names with known gender?
- What anomalies exist in the set that I need to know about?
- What do we know about the quality of the data?
  - i.e How do we trust the tests we base it on?
- How should we deal with unisex names?

Initial take on tests

Initial take on tests

```
gender “Ram”      == Man
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
gender “Ashok”    == Man
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
gender “Ashok”    == Man
gender “Yamuna”   == Woman
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
gender “Ashok”    == Man
gender “Yamuna”   == Woman
gender “Valavan” == Man
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
gender “Ashok”    == Man
gender “Yamuna”   == Woman
gender “Valavan” == Man
gender “Aarthi”   == Woman
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
gender “Ashok”    == Man
gender “Yamuna”   == Woman
gender “Valavan” == Man
gender “Aarthi”   == Woman
gender “Valli”    == Woman
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
gender “Ashok”    == Man
gender “Yamuna”   == Woman
gender “Valavan” == Man
gender “Aarthi”   == Woman
gender “Valli”    == Woman
gender “Amjad”    == Man
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
gender “Ashok”    == Man
gender “Yamuna”   == Woman
gender “Valavan” == Man
gender “Aarthi”   == Woman
gender “Valli”    == Woman
gender “Amjad”    == Man
gender “Azma”     == Woman
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
gender “Ashok”    == Man
gender “Yamuna”   == Woman
gender “Valavan”  == Man
gender “Aarthi”   == Woman
gender “Valli”    == Woman
gender “Amjad”    == Man
gender “Azma”     == Woman
```

```
gender “Chandra” == ??
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
gender “Ashok”    == Man
gender “Yamuna”   == Woman
gender “Valavan”  == Man
gender “Aarthi”   == Woman
gender “Valli”    == Woman
gender “Amjad”    == Man
gender “Azma”     == Woman
```

```
gender “Chandra” == ??
gender “Kiran”   == ??
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
gender “Ashok”    == Man
gender “Yamuna”   == Woman
gender “Valavan” == Man
gender “Aarthi”   == Woman
gender “Valli”    == Woman
gender “Amjad”    == Man
gender “Azma”     == Woman
```

```
gender “Chandra” == ??
gender “Kiran”   == ??
gender “Rama”    == ??
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
gender “Ashok”    == Man
gender “Yamuna”   == Woman
gender “Valavan” == Man
gender “Aarthi”   == Woman
gender “Valli”    == Woman
gender “Amjad”    == Man
gender “Azma”     == Woman
```

```
gender “Chandra” == ??
gender “Kiran”   == ??
gender “Rama”    == ??
  — Is it “Ramaa” or “Raamaa”?
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
gender “Ashok”    == Man
gender “Yamuna”   == Woman
gender “Valavan” == Man
gender “Aarthi”   == Woman
gender “Valli”    == Woman
gender “Amjad”    == Man
gender “Azma”     == Woman
```

```
gender “Chandra” == ??
gender “Kiran”   == ??
gender “Rama”    == ??
  — Is it “Ramaa” or “Raamaa”?
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
gender “Ashok”    == Man
gender “Yamuna”   == Woman
gender “Valavan”  == Man
gender “Aarthi”   == Woman
gender “Valli”    == Woman
gender “Amjad”    == Man
gender “Azma”     == Woman
```

```
gender “Chandra”  == ??
gender “Kiran”    == ??
gender “Rama”     == ??
  — Is it “Ramaa” or “Raamaa”?
```

```
gender “pumpkin” == ??
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
gender “Ashok”    == Man
gender “Yamuna”   == Woman
gender “Valavan” == Man
gender “Aarthi”   == Woman
gender “Valli”    == Woman
gender “Amjad”    == Man
gender “Azma”     == Woman
```

```
gender “Chandra” == ??
gender “Kiran”    == ??
gender “Rama”     == ??
  — Is it “Ramaa” or “Raamaa”?
```

```
gender “pumpkin” == ??
gender “doormat”    == ??
```

Initial take on tests

```
gender “Ram”      == Man
gender “Sita”     == Woman
gender “Ashok”    == Man
gender “Yamuna”   == Woman
gender “Valavan” == Man
gender “Aarthi”   == Woman
gender “Valli”    == Woman
gender “Amjad”    == Man
gender “Azma”     == Woman
```

```
gender “Chandra” == ??
gender “Kiran”    == ??
gender “Rama”     == ??
  — Is it “Ramaa” or “Raamaa”?
```

```
gender “pumpkin” == ??
gender “doormat”    == ??
gender “பாத்திரம்”     == ??
```

We want a table!

Name	Gender
Valavan	Man
Azma	Woman
Kiran	Either
pumpkin	??
பாத்திரம்	??

Our first function …

Our first function …

Our first function …

Table lookup! 😎

Our first function …

Table lookup! 😎

BAD!

Our first function …

Table lookup! 😎

BAD!

- Tests will never break

Our first function …

Table lookup! 😎

BAD!

- Tests will never break
- We have to ship the table

Our first function …

Table lookup! 😎

BAD!

- Tests will never break
- We have to ship the table
- Doesn’t work for names not in the table

Our first function …

Table lookup! 😎

BAD!

- Tests will never break
- We have to ship the table
- Doesn’t work for names not in the table
- We can do better

Overfitting

Overfitting

- Means we're learning by rote. Can only answer textbook questions.

Overfitting

- Means we're learning by rote. Can only answer textbook questions.
- **Understanding** implies **compression**

Overfitting

- Means we're learning by rote. Can only answer textbook questions.
- **Understanding** implies **compression**
- Necessarily lossy - usually heavily lossy

Overfitting

- Means we're learning by rote. Can only answer textbook questions.
- **Understanding** implies **compression**
- Necessarily lossy - usually heavily lossy
- What if we say -
  "Can't use more than `MAX_MEMORY`"?

Towards generalizability

Towards generalizability

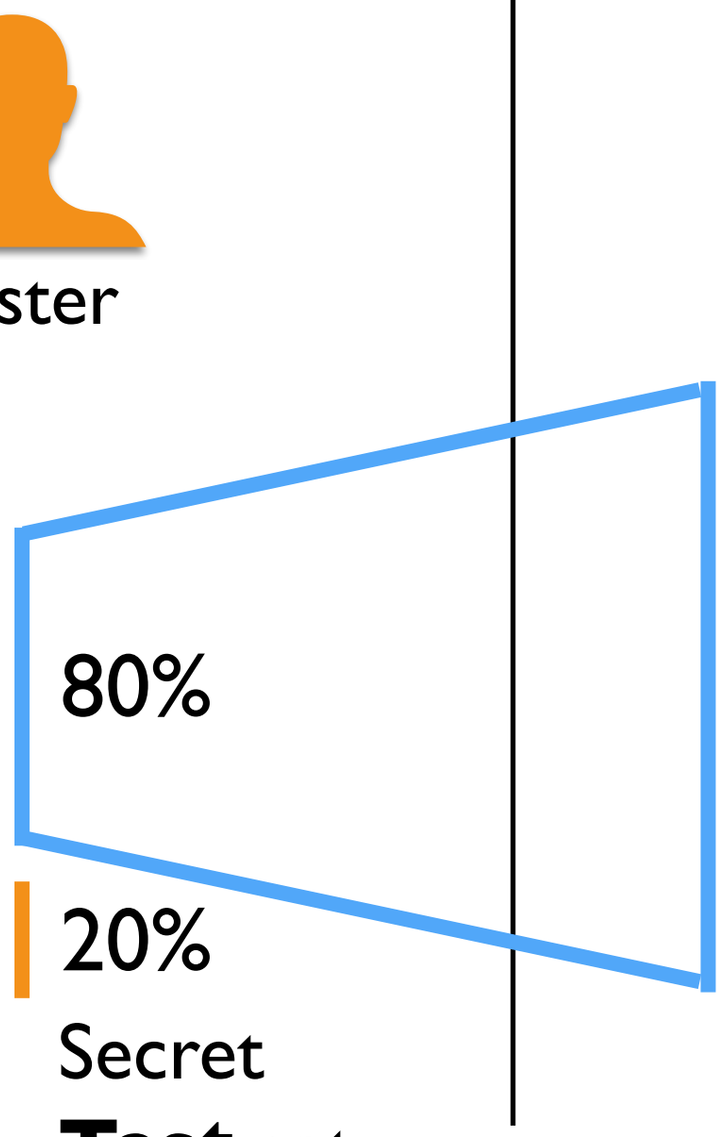
Tester

Coder

Towards generalizability

Tester

Name	Gender
Valavan	Man
Azma	Woman
Kiran	Either
pumpkin	Either
பாத்திரம்	Either



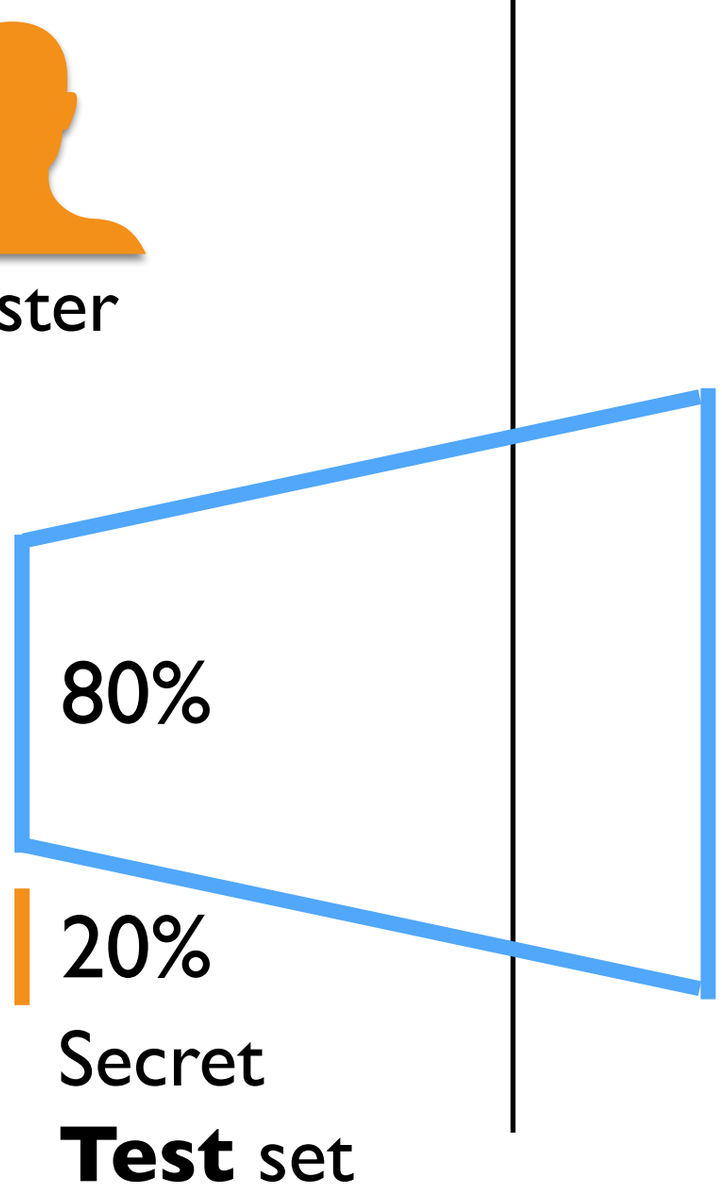
Coder

```
{
  //  ...
}
```

Towards generalizability

Tester

Name	Gender
Valavan	Man
Azma	Woman
Kiran	Either
pumpkin	Either
பாத்திரம்	Either



Coder

```
{
  //  ...
}
```

Dev set

Training set

Pause …

Pause …

- All of this has been **just** about the data and the objective

Pause …

- All of this has been **just** about the data and the objective
- Most important and usually expensive part of the process

Modeling the “learner”

Modeling the “learner”

```
type Learner = Data → (Learner, Guesser)
```

Modeling the “learner”

```
type Guesser = String → Gender

type Learner = Data → (Learner, Guesser)
```

Modeling the “learner”

```
type Data = [(String, Gender)]

type Guesser = String → Gender

type Learner = Data → (Learner, Guesser)
```

Modeling the “learner”

```
data Gender = Man | Woman

type Data = [(String, Gender)]

type Guesser = String → Gender

type Learner = Data → (Learner, Guesser)
```

Modeling the “learner”

```
data Gender = Man | Woman

type Data = [(String, Gender)]

type Guesser = String → Gender

type Learner = Data → (Learner, Guesser)
```

This is a process

Features

Features

- Are about increasing generalizability

Features

- Are about increasing generalizability
- … by reducing the space of data

Features

- Are about increasing generalizability
- … by reducing the space of data
  - The smaller the data space, the easier it is to collect **enough** data, and the easier it is to test.

Features

- Are about increasing generalizability
- … by reducing the space of data
  - The smaller the data space, the easier it is to collect **enough** data, and the easier it is to test.
  - Capture existing domain understanding

Features

- Are about increasing generalizability
- … by reducing the space of data
  - The smaller the data space, the easier it is to collect **enough** data, and the easier it is to test.
- Capture existing domain understanding

Ex: “Kum**ar**” → “**ar**”

Introducing features

Introducing features

```
type Feature = String
```

Introducing features

```
type Feature = String
type FeatureData = [(Feature, Gender)]
```

Introducing features

```
type Feature = String
type FeatureData = [(Feature, Gender)]

type FeatureExtractor = String → Feature
```

Introducing features

```
type Feature = String
type FeatureData = [(Feature, Gender)]

type FeatureExtractor = String → Feature

type GuesserF = Feature → Gender
```

Introducing features

```
type Feature = String
type FeatureData = [(Feature, Gender)]

type FeatureExtractor = String → Feature

type GuesserF = Feature → Gender

type LearnerF =
        FeatureData → (LearnerF, GuesserF)
```

Using features

Using features

```
learner :: FeatureExtractor → LearnerF → Learner
```

Using features

```
learner :: FeatureExtractor → LearnerF → Learner
learner feature learnerF data =
```

Using features

```
learner :: FeatureExtractor → LearnerF → Learner
learner feature learnerF data =
    let trans (name, g) = (feature name, g)
```

Using features

```
learner :: FeatureExtractor → LearnerF → Learner
learner feature learnerF data =
    let trans (name, g) = (feature name, g)
    (lf2, fpred) = learnerF (map trans data)
```

Using features

```
learner :: FeatureExtractor → LearnerF → Learner
learner feature learnerF data =
    let trans (name, g) = (feature name, g)
    (lf2, fpred) = learnerF (map trans data)
    pred name = fpred (feature name)
```

Using features

```
learner :: FeatureExtractor → LearnerF → Learner
learner feature learnerF data =
    let trans (name, g) = (feature name, g)
        (lf2, fpred) = learnerF (map trans data)
        pred name = fpred (feature name)
    in
```

Using features

```
learner :: FeatureExtractor → LearnerF → Learner
learner feature learnerF data =
    let trans (name, g) = (feature name, g)
        (lf2, fpred) = learnerF (map trans data)
        pred name = fpred (feature name)
    in
        (learner feature lf2, pred)
```

Code walkthrough

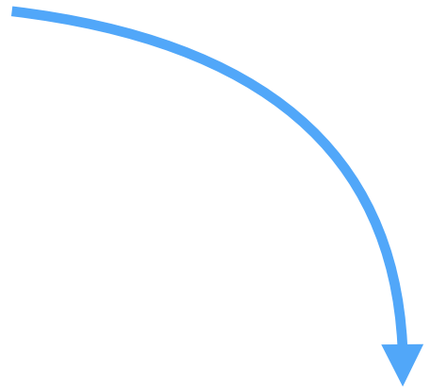
The learning cycle

The learning cycle

Try something

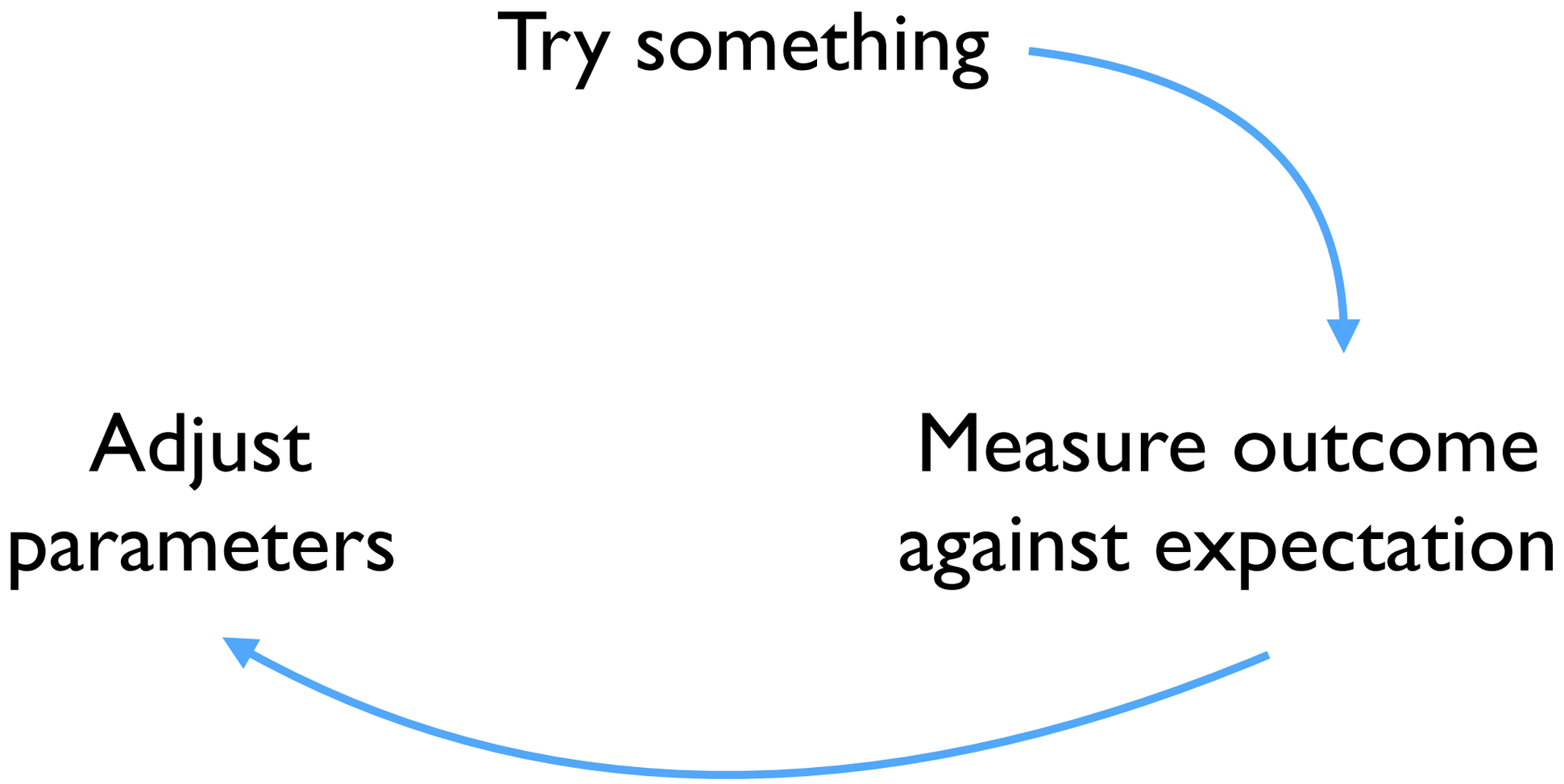
The learning cycle

Try something

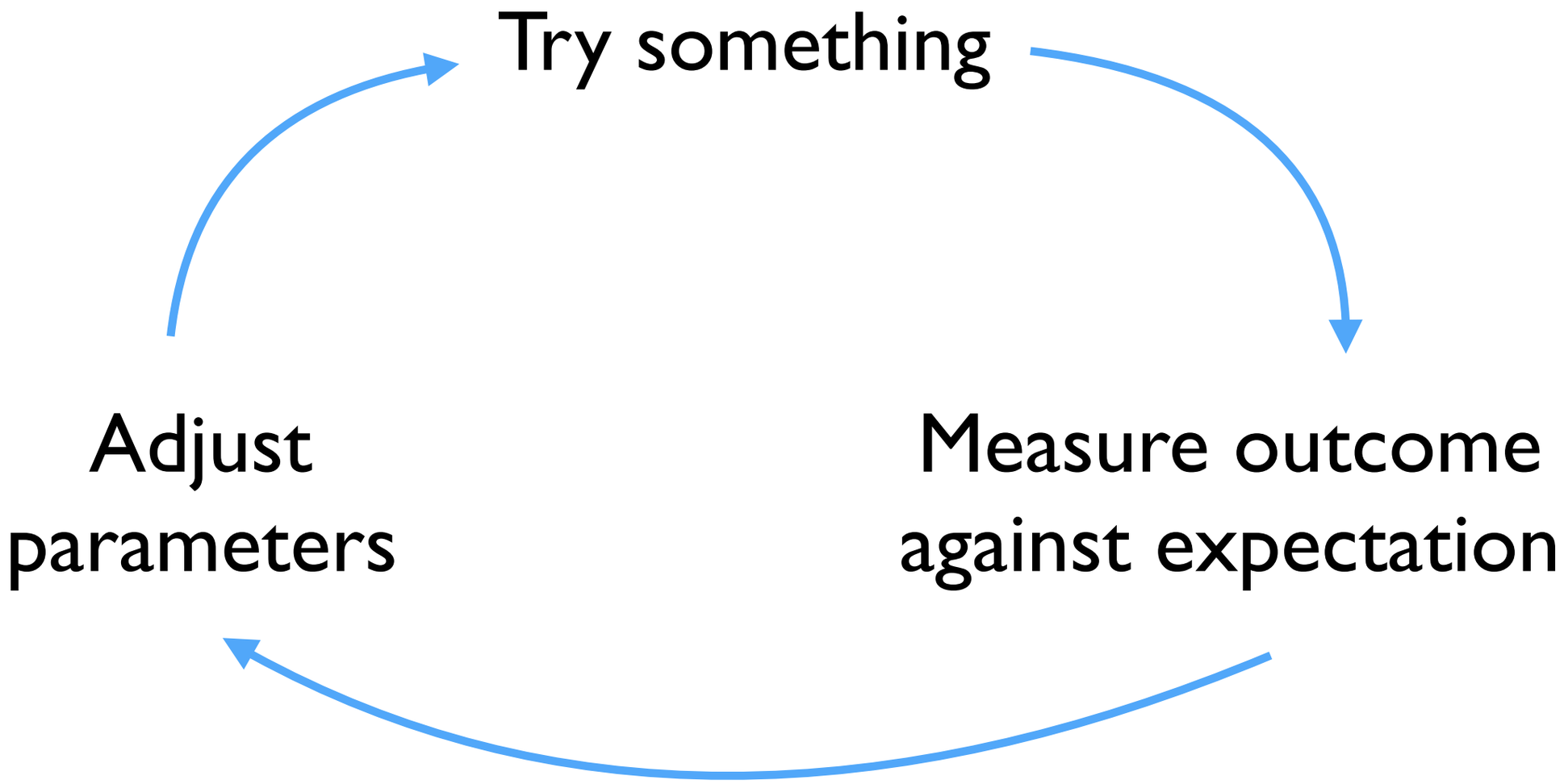


Measure outcome
against expectation

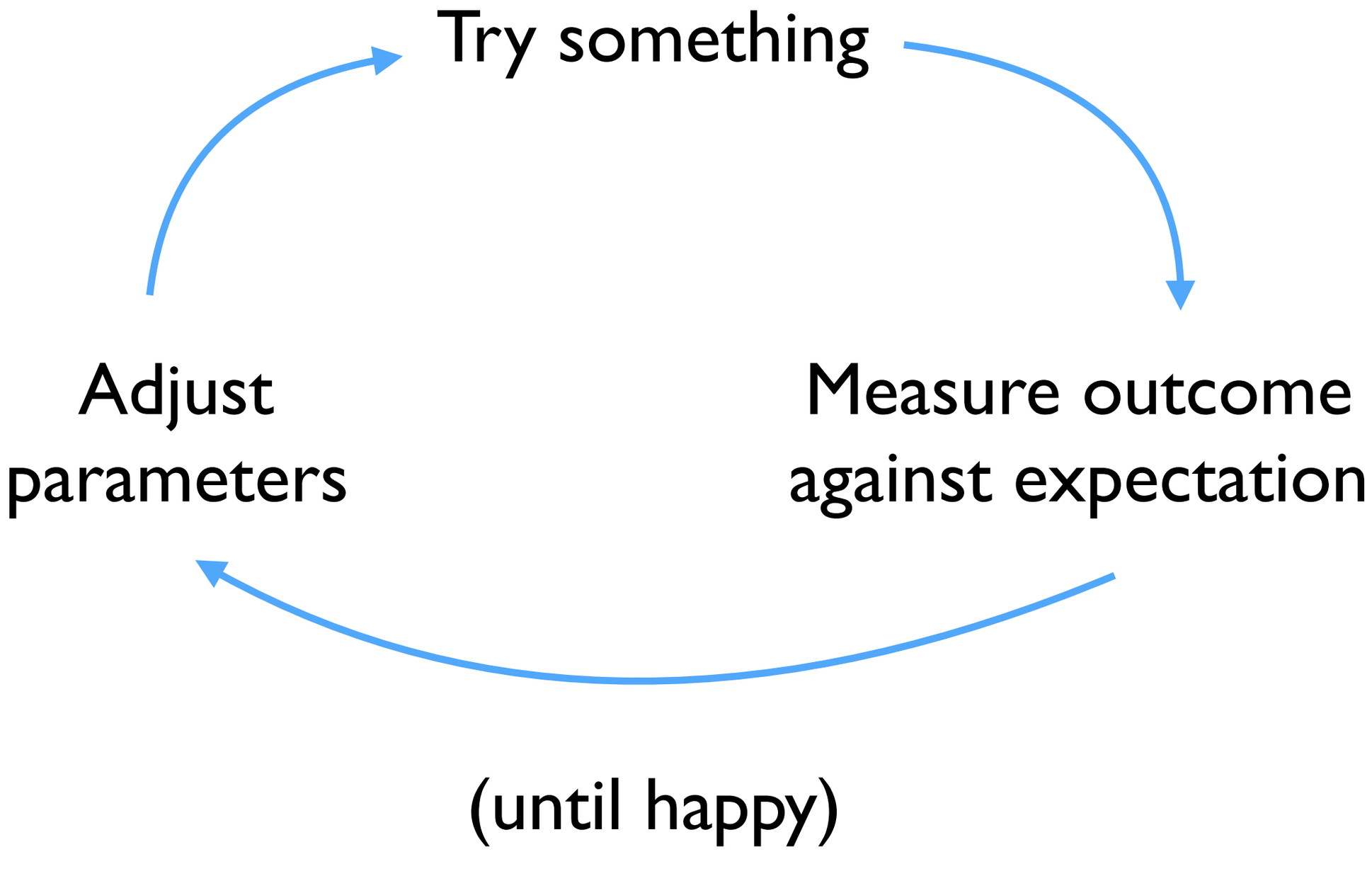
The learning cycle



The learning cycle



The learning cycle



The learning cycle

```
type LearnerP = (Params, ErrorFn, UpdateFn)
```

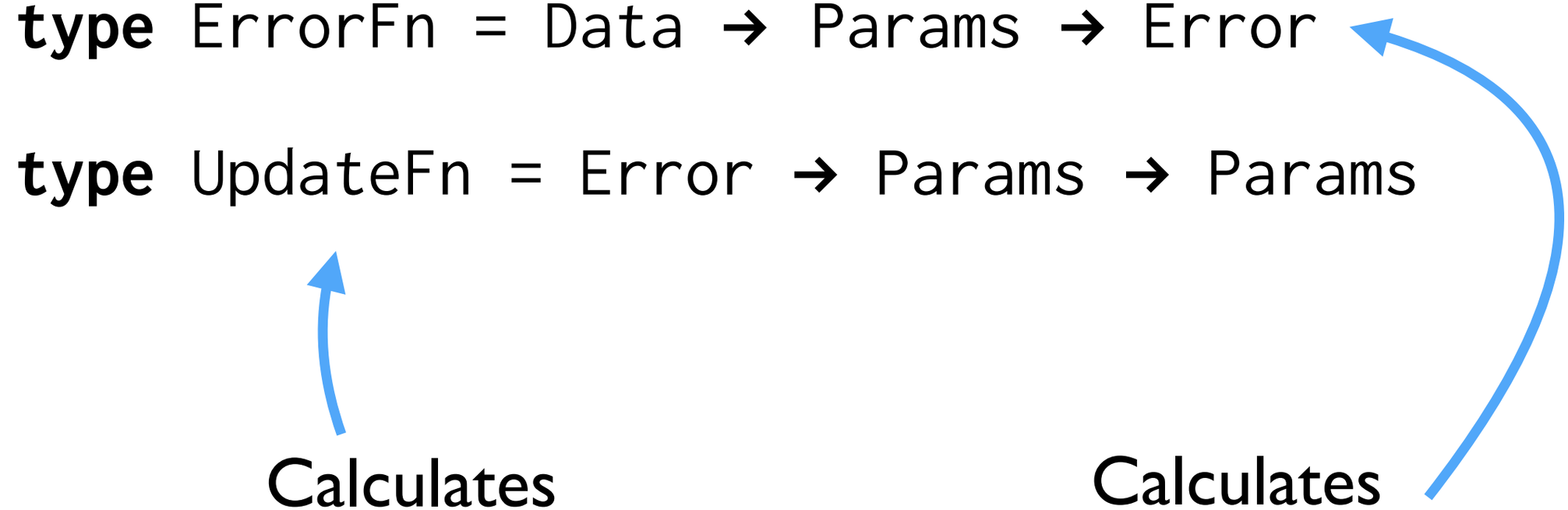
The learning cycle

```
type LearnerP = (Params, ErrorFn, UpdateFn)

type ErrorFn = Data → Params → Error
```

Calculates how we're doing

The learning cycle

```
type LearnerP = (Params, ErrorFn, UpdateFn)

type ErrorFn = Data → Params → Error

type UpdateFn = Error → Params → Params
```



The road ahead

The road ahead

"Differentiable programming"

A taste of DP …

Some takeaways

Some takeaways

- Its all about the data.
  Bad data = Bad model

Some takeaways

- Its all about the data.
  Bad data = Bad model
- Your data, training plan, tests and metrics must come first.
  Just like TDD, but extreme. Ethics must also be factored in.

Some takeaways

- Its all about the data.
  Bad data = Bad model
- Your data, training plan, tests and metrics must come first.
  Just like TDD, but extreme. Ethics must also be factored in.
- Features (esply “embeddings”) are the API between ML models

Some takeaways

- Its all about the data.
  Bad data = Bad model
- Your data, training plan, tests and metrics must come first.
  Just like TDD, but extreme. Ethics must also be factored in.
- Features (esply "embeddings") are the API between ML models
- Function factorization can help understand learning processes

Some takeaways

- Its all about the data.
  Bad data = Bad model
- Your data, training plan, tests and metrics must come first.
  Just like TDD, but extreme. Ethics must also be factored in.
- Features (esply "embeddings") are the API between ML models
- Function factorization can help understand learning processes
- Automatic differentiation is helping ML adoption, so pay attention

How my cousin did it

How my cousin did it

```
gender name =
```

How my cousin did it

```
gender name =
    if endswith “a” name
```

How my cousin did it

```
gender name =
    if endswith “a” name
    then Woman
```

How my cousin did it

```
gender name =
    if endswith “a” name
then Woman
else if endswith “i” name
```

How my cousin did it

```
gender name =
     if endswith “a” name
then Woman
else if endswith “i” name
     then Woman
```

How my cousin did it

```
gender name =
     if endswith “a” name
then Woman
else if endswith “i” name
     then Woman
     else Man
```

Thanks for listening!

References

Data sets

https://github.com/ellisbrown/name2gender

https://github.com/vsant/indian-name-classifier