Bowl Full of Lentils

Fintan Halpenny

June 24, 2019

It Me

Figure 1: Learning about my new camera

It Me

- Fintan Halpenny

It Me

- Fintan Halpenny
- Frequently known as fintan or finto online

It Me

- Fintan Halpenny
- Frequently known as fintan or finto online
- Work at Formation

Dhall

Figure 2: https://dhall-lang.org/

Dhall

- Dhall - The non-repetitive alternative to YAML

Dhall

- Dhall - The non-repetitive alternative to YAML
- You can think of Dhall as: JSON + functions + types + imports

Dhall

- Dhall - The non-repetitive alternative to YAML
- You can think of Dhall as: JSON + functions + types + imports
- Not Turing-complete

Dhall - Features

- Built-in types

Dhall - Features

- Built-in types
- Records

Dhall - Features

- Built-in types
- Records
- Unions

Dhall - Features

- Built-in types
- Records
- Unions
- Functions

Dhall - Features

- Built-in types
- Records
- Unions
- Functions
- Let/Imports

Bool - Type

```
$ dhall type <<< "True"
Bool
```

Bool - Examples

```
$ dhall <<< "True && False"
False

$ dhall <<< "True && True"
True

$ dhall <<< "True || False"
True

$ dhall <<< "False || False"
False
```

Natural - Type

```
$ dhall type <<< "0"
Natural
```

Natural - Examples

```
$ dhall <<< "1 + 1"
2

$ dhall <<< "1 + 0"
1

$ dhall <<< "1 * 1"
1

$ dhall <<< "1 * 0"
0
```

Integer - Type

```
$ dhall type <<< "-1"
Integer
```

Integer - Examples

```
$ dhall <<< "-1"
-1

$ dhall <<< "+1"
+1

$ dhall <<< "-1 + -2"
Use "dhall --explain" for detailed errors

Error: <+> only works on <Natural>s

-1 + -2

(stdin):1:1
```

Double - Type

```
$ dhall type <<< "3.14"
Double
```

Double - Examples

```
$ dhall <<< "3.14"
3.14

$ dhall <<< "Infinity"
Infinity

$ dhall <<< "-Infinity"
-Infinity

$ dhall <<< "3.14 * (3.0 * 3.0)"
Use "dhall --explain" for detailed errors

Error: <*> only works on <Natural>s

3.14 * (3.0 * 3.0)

(stdin):1:1
```

Text - Type

```
$ dhall type <<< '"Hello, World!"'
Text
```

Text - Examples

```
$ dhall <<< '"I <3 Dhall"'
"I <3 Dhall"

$ dhall <<< '"I" ++ " <3 " ++ "NYC"'
"I <3 NYC"

$ dhall
''
    #!/bin/bash

    echo "Hi!"
''
<Ctrl-D>
''
    #!/bin/bash

    echo "Hi!"
''
```

List - Type

```
$ dhall type <<< "[1, 2, 3]"
List Natural
```

List - Examples

```
$ dhall <<< "[1, 2, 3] # [4, 5, 6]"
[ 1, 2, 3, 4, 5, 6 ]

$ dhall <<< "
List/fold
  Bool
  [True, False, True]
  Bool
  (\(x : Bool) -> \(y : Bool) -> x && y)
  True
"
False

$ dhall <<< "List/length Natural [1, 2, 3]"
3
```

Optional - Type

```
$ dhall type <<< "None Natural"
Optional Natural

$ dhall type <<< "Some 1"
Optional Natural
```

Optional - Fold

```
$ dhall <<< '
Optional/fold
  Text
  (Some "ABC")
  Text
  (\(t : Text) -> t)
  "Uhoh"
'
"ABC"
```

Optional - Fold

```
$ dhall <<< '
Optional/fold
  Text
  (None Text)
  Text
  (\(t : Text) -> t)
  "Uhoh"
'
"Uhoh"
```

Unit

```
$ dhall type <<< "{=}"
{}

$ dhall <<< "{=}"
{=}
```

Records - Type

```
dhall type <<< "
{ name : Text, age : Natural, email : Text }
"
Type
```

Records - Person

```
dhall <<< "
{ name : Text, age : Natural, email : Text }
"
{ age : Natural, email : Text, name : Text }
```

Records - Person Creation

```
dhall <<< '
{ name = "Fintan", age = 26, email = "finto@haps.com" }
'
{ age = 26, email = "finto@haps.com", name = "Fintan" }
```

Records - Person Type

```
dhall type <<< '
{ name = "Fintan", age = 26, email = "finto@haps.com" }
'
{ age : Natural, email : Text, name : Text }
```

Records - Example Type Annotation

```
dhall <<< '
{ name = "Fintan" } : { name : Text, age : Natural }
'
Error: Expression doesn\'t match annotation

{ + age : …
, …
}

{ name = "Fintan" } : { name : Text, age : Natural }
```

Records - Example Projection 1

```
dhall <<< '{ name = "Fintan", age = 26 }.age'
26

dhall <<< '{ name = "Fintan", age = 26 }.{ age, name }'
{ age = 26, name = "Fintan" }
```

Unions - Type

```
$ dhall type <<< "
< Monday
| Tuesday
| Wednesday
| Thursday
| Friday
>
"
Type
```

Unions - Example Enum

```
$ dhall <<< "
< Monday
| Tuesday
| Wednesday
| Thursday
| Friday
>
"
< Friday | Monday | Thursday | Tuesday | Wednesday >
```

Unions - Example Enum Constructor

```
$ dhall <<< "
< Monday
| Tuesday
| Wednesday
| Thursday
| Friday
>.Monday
"
<Friday | Monday | Thursday | Tuesday | Wednesday>.Monday
```

Unions - Example Eliminator

```
$ dhall <<< "
merge
{ Monday = True
, Tuesday = False
, Wednesday = False
, Thursday = False
, Friday = False
}
<Monday | Tuesday | Wednesday | Thursday | Friday>.Monday
"
True
```

Unions - Example Sum Type

```
$ dhall <<< "< IsNat : Natural | IsText : Text >"
< IsNat : Natural | IsText : Text >

$ dhall type <<< "
< IsNat : Natural | IsText : Text >.IsNat
"
  forall(IsNat : Natural)
→ < IsNat : Natural | IsText : Text >
```

Functions - Type

```
$ dhall type <<< "\(x : Natural) -> x + 1"
forall(x : Natural) → Natural
```

Functions - Example

```
$ dhall <<< "\(x : Natural) -> x + 1"
\(x : Natural) → x + 1

$ dhall <<< "(\(x : Natural) -> x + 1) 41"
42
```

Functions - Example Const

```
$ dhall <<< '
    ( \(a : Type)
  -> \(b : Type)
  -> \(x : a)
  -> \(y : b)
  -> x
    ) Text Natural "Ignored!" 17
'
"Ignored!"
```

Let/Imports - Examples

```
$ dhall <<< '
let identity = \(a : Type) -> \(x : a) -> x
in identity Text "It me!"
'
"It me!"

$ dhall <<< "
let enumerate = https://dhall/Prelude/Natural/enumerate
in enumerate 10
"
[ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ]

$ dhall <<< "let f = ../identity in f Natural 42"
42
```

Machine Learning Configuration - SVM

- Support Vector Machine (SVM)

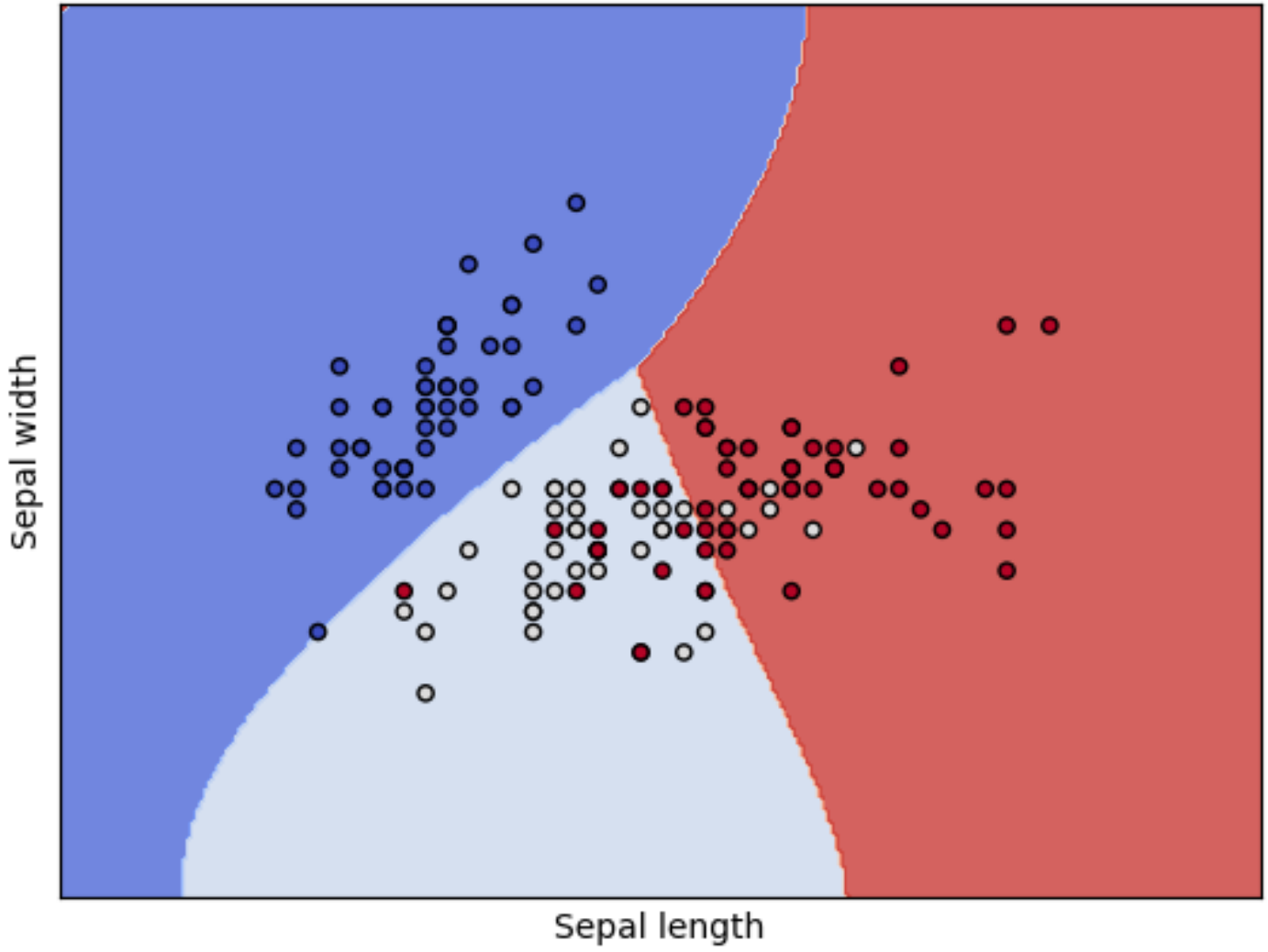
Machine Learning Configuration - SVM

- Support Vector Machine (SVM)
- Hyperparameter Learning

Machine Learning Configuration - SVM

- Support Vector Machine (SVM)
- Hyperparameter Learning
- Simple Script in Python using Scikit-Learn

Machine Learning Configuration - SVM



SVM - SVC Inputs

- C-Support Vector Classification

SVM - SVC Inputs

- C-Support Vector Classification
- `kernel`: Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used.

SVM - SVC Inputs

- C-Support Vector Classification
- `kernel`: Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used.
- `C`: Penalty parameter C of the error term.

SVM - SVC Inputs

- C-Support Vector Classification
- `kernel`: Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used.
- `C`: Penalty parameter C of the error term.
- `gamma`: Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

SVM - SVC Inputs

- C-Support Vector Classification
- `kernel`: Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used.
- `C`: Penalty parameter C of the error term.
- `gamma`: Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.
- Call `fit` on `X` and `y` training data.

Trainer - JSON

```
{ "kernel": "linear"
, "dataset": "Iris"
}
```

Trainer - JSON (But wait!)

```
{ "kernel": "rbf"
, "dataset": "Iris"
}
```

Hyper-Parameters - JSON

- We can train on the two parameters `C` and `gamma`

Hyper-Parameters - JSON

- We can train on the two parameters `C` and `gamma`
- In JSON:

```
{ "gamma": 0.1, "C": 0.1 }
```

Hyper-Parameters - JSON

- We can train on the two parameters `C` and `gamma`
- In JSON:

```
{ "gamma": 0.1, "C": 0.1 }
```

- But we need more parameters:

```
[ { "gamma": 0.1, "C": 0.1 }
, { "gamma": 0.1, "C": 1 }
, { "gamma": 0.1, "C": 10 }
, { "gamma": 0.1, "C": 100 }
, { "gamma": 0.1, "C": 1000 }
]
```

Hyper-Parameters - JSON

- We can train on the two parameters `C` and `gamma`
- In JSON:

```
{ "gamma": 0.1, "C": 0.1 }
```

- But we need more parameters:

```
[ { "gamma": 0.1, "C": 0.1 }
, { "gamma": 0.1, "C": 1 }
, { "gamma": 0.1, "C": 10 }
, { "gamma": 0.1, "C": 100 }
, { "gamma": 0.1, "C": 1000 }
]
```

- Now we want to increase `gamma` values as well, ohno!

Copy-Pasta

```
[ { "gamma": 0.1, "C": 0.1 }
, { "gamma": 0.1, "C": 1 }
, { "gamma": 0.1, "C": 10 }
, { "gamma": 0.1, "C": 100 }
, { "gamma": 0.1, "C": 1000 }

, { "gamma": 1, "C": 0.1 }
, { "gamma": 1, "C": 1 }
, { "gamma": 1, "C": 10 }
, { "gamma": 1, "C": 100 }
, { "gamma": 1, "C": 1000 }

, { "gamma": 10, "C": 0.1 }
, { "gamma": 10, "C": 1 }
, { "gamma": 10, "C": 10 }
, { "gamma": 10, "C": 100 }
, { "gamma": 10, "C": 1000 }
```

Trainer - Dhall

- `Kernel.dhall`:

```
< RBF | Linear | Poly >
```

Trainer - Dhall

- `Kernel.dhall`:

```
< RBF | Linear | Poly >
```

- `Dataset.dhall`:

```
< Iris | Wine >
```

Trainer - Dhall

- Kernel.dhall:

```
< RBF | Linear | Poly >
```

- Dataset.dhall:

```
< Iris | Wine >
```

- Trainer.dhall:

```
let Dataset = ./Dataset.dhall

let Kernel = ./Kernel.dhall

in  { dataset : Dataset, kernel : Kernel }
```

Hyper-Parameters - Dhall

- `Hyperparameters.dhall`:

```
{ gamma : Double, C : Double }
```

Hyper-Parameters - Dhall

- `Hyperparameters.dhall`:

```
{ gamma : Double, C : Double }
```

- We want a helper to create the equivalent of our set `gamma` value:

Hyper-Parameters - Dhall

- `Hyperparameters.dhall`:

```
{ gamma : Double, C : Double }
```

- We want a helper to create the equivalent of our set `gamma` value:

```
List/map
  Double
  Hyperparameters
  (\(C : Double) -> { gamma = 0.1, C = C })
  [ 0.1, 1, 10, 100, 1000 ]
```

Hyper-Parameters - Dhall

- `Hyperparameters.dhall`:

```
{ gamma : Double, C : Double }
```

- Ok, how about the equivalent our big copy-pasta monster?

Hyper-Parameters - Dhall

- `Hyperparameters.dhall`:

```
{ gamma : Double, C : Double }
```

- Ok, how about the equivalent our big copy-pasta monster?

```
List/liftA2
  Double
  Double
  Hyperparameters
  (  \(gamma : Double)
  -> \(C : Double)
  -> { gamma = gamma, C = C }
  )
  [ 0.1, 1, 10, 100 ]
  [ 0.1, 1, 10, 100, 1000 ]
```

More Tasty Meals

- https://functional.works-hub.com/learn/bowl-full-of-lentils-fcbf3
- https://functional.works-hub.com/learn/yo-yoneda-a2965
- https://kowainik.github.io/posts/2018-09-09-dhall-to-hlint.html
- https://github.com/dhall-lang

Other Links

- https://scikit-learn.org/stable/index.html
- https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769

Follow Me (But like, only if you want to)

Figure 4: For more blurry pics follow me on Instagram @fintohaps