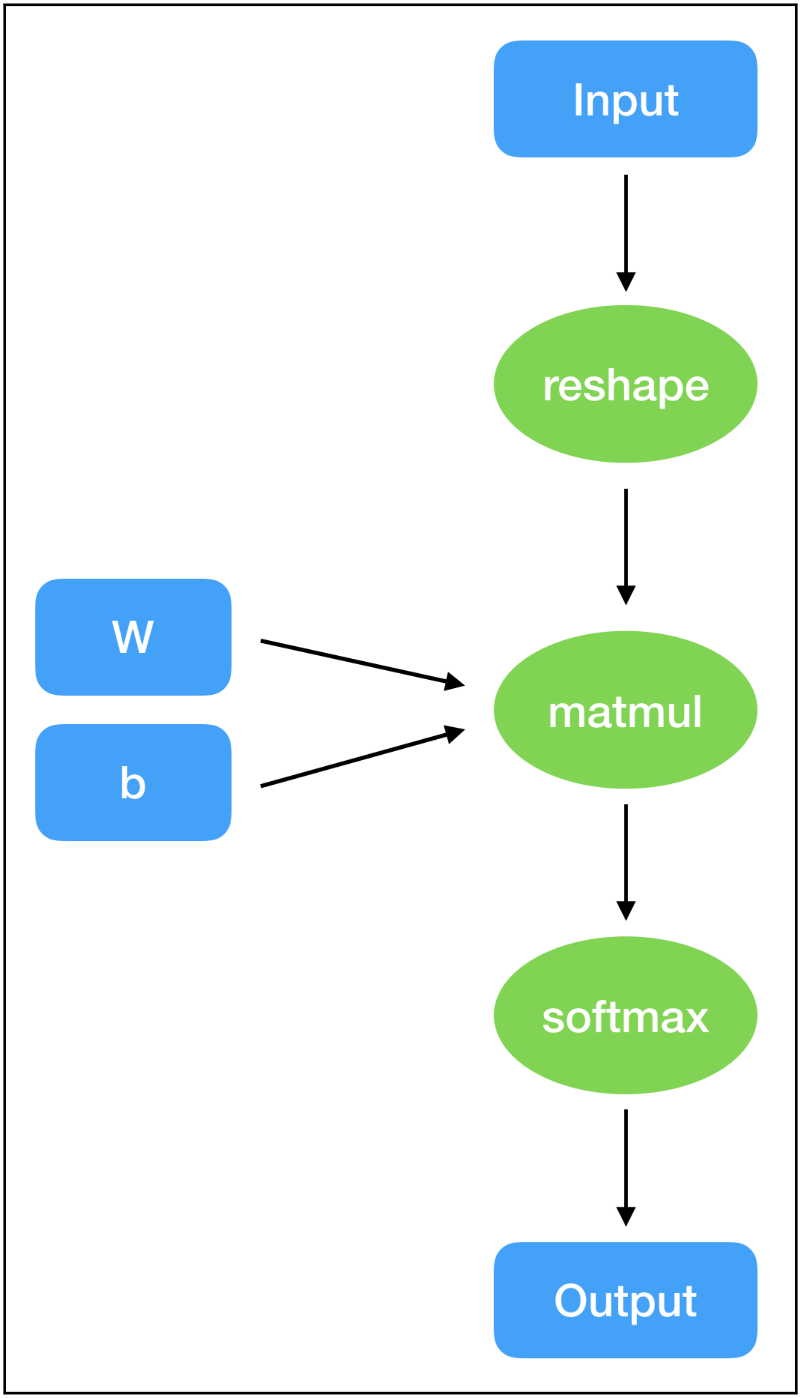
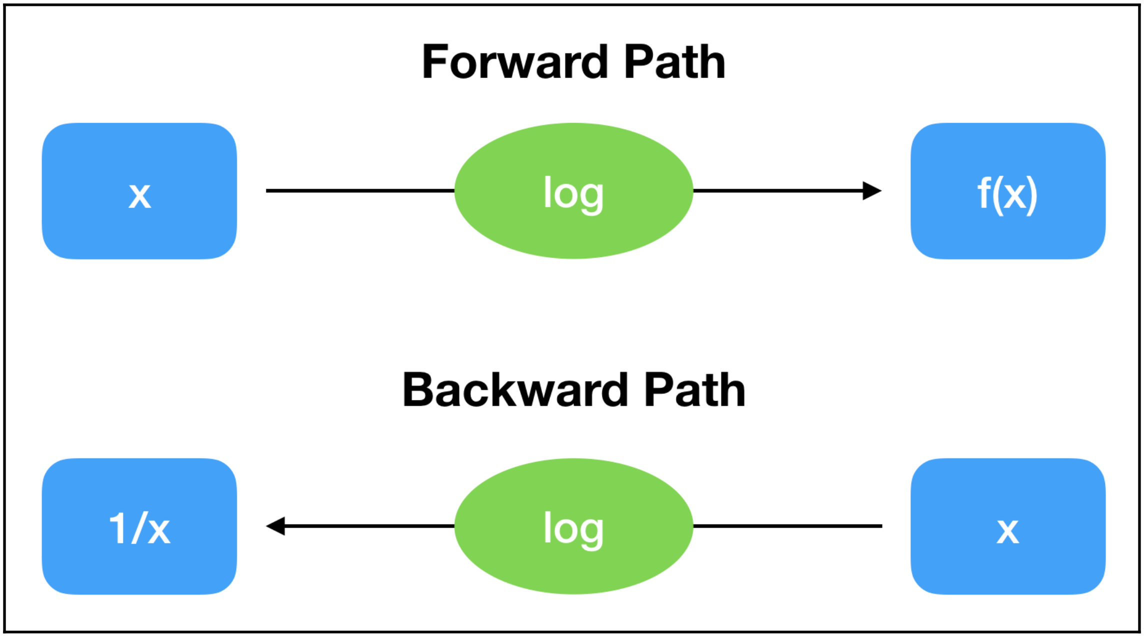
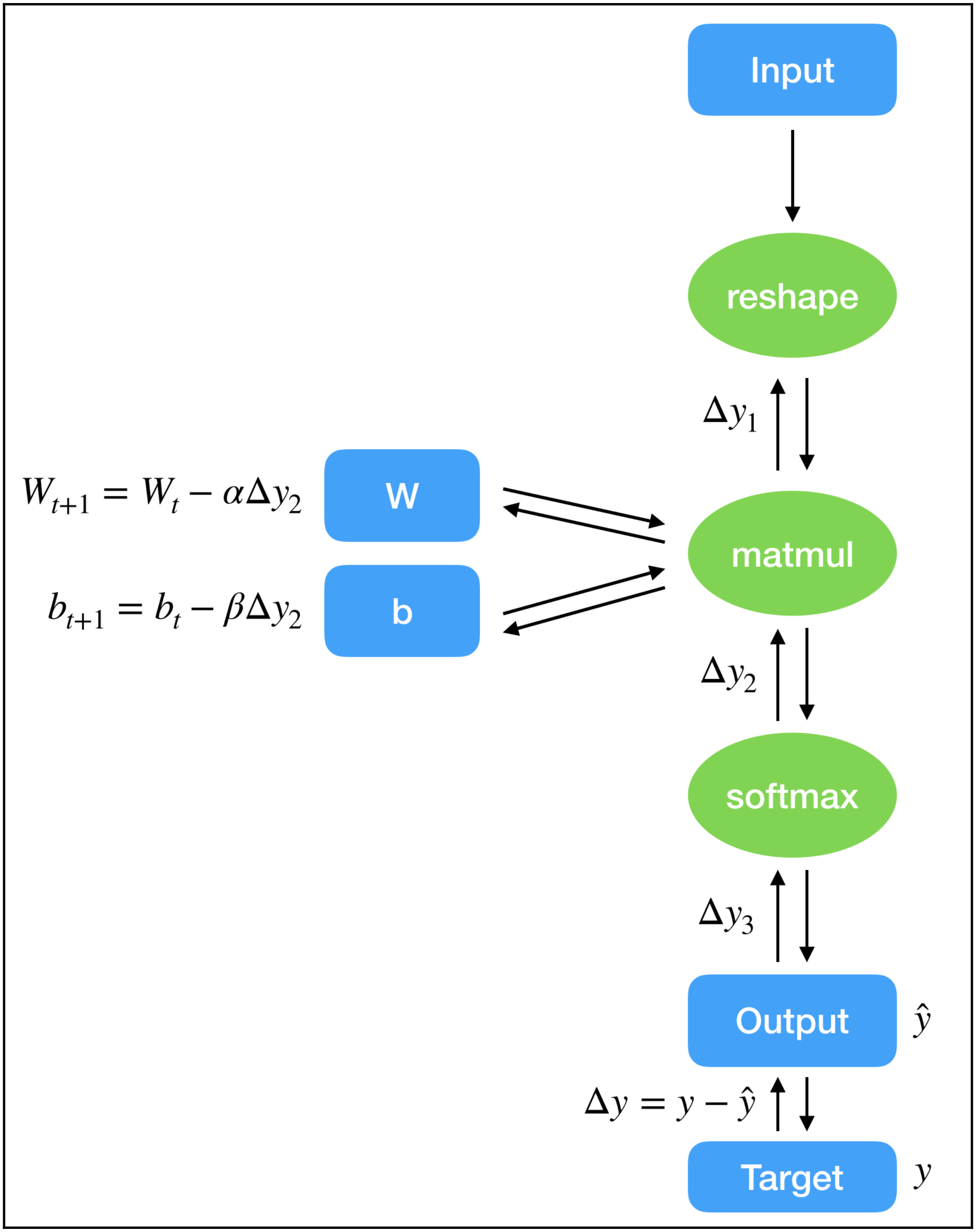
Chapter 1: Machine Learning for the Web

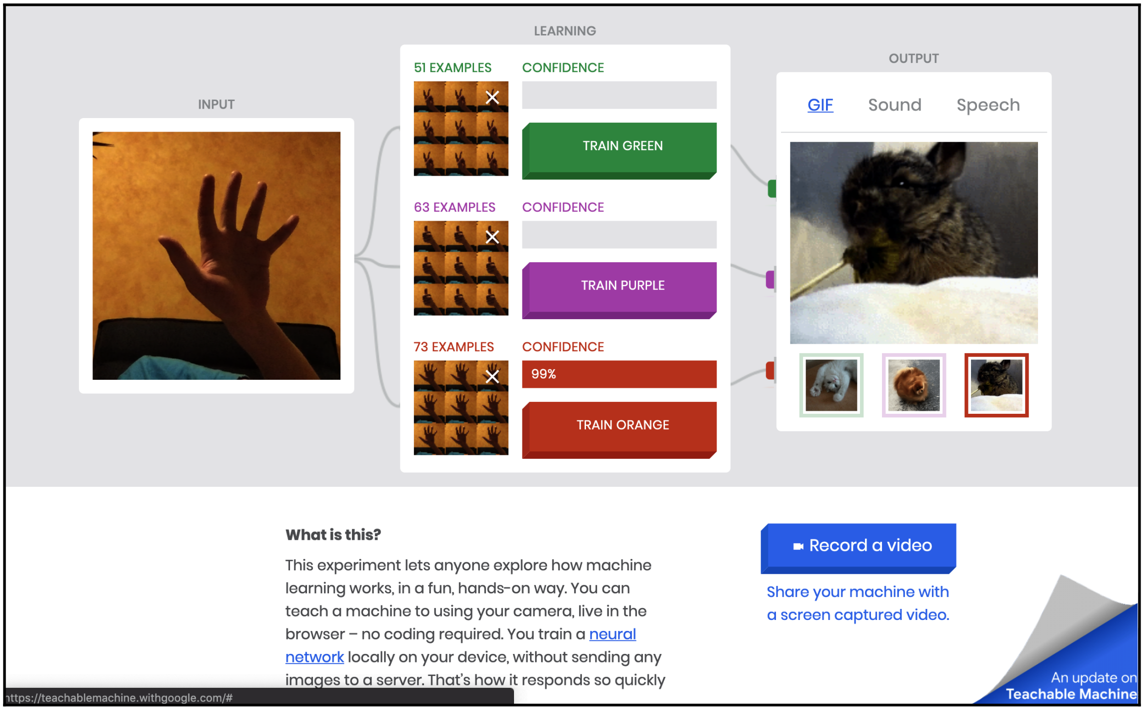

Input
reshape
W
b
matmul
softmax
Output



Forward Path

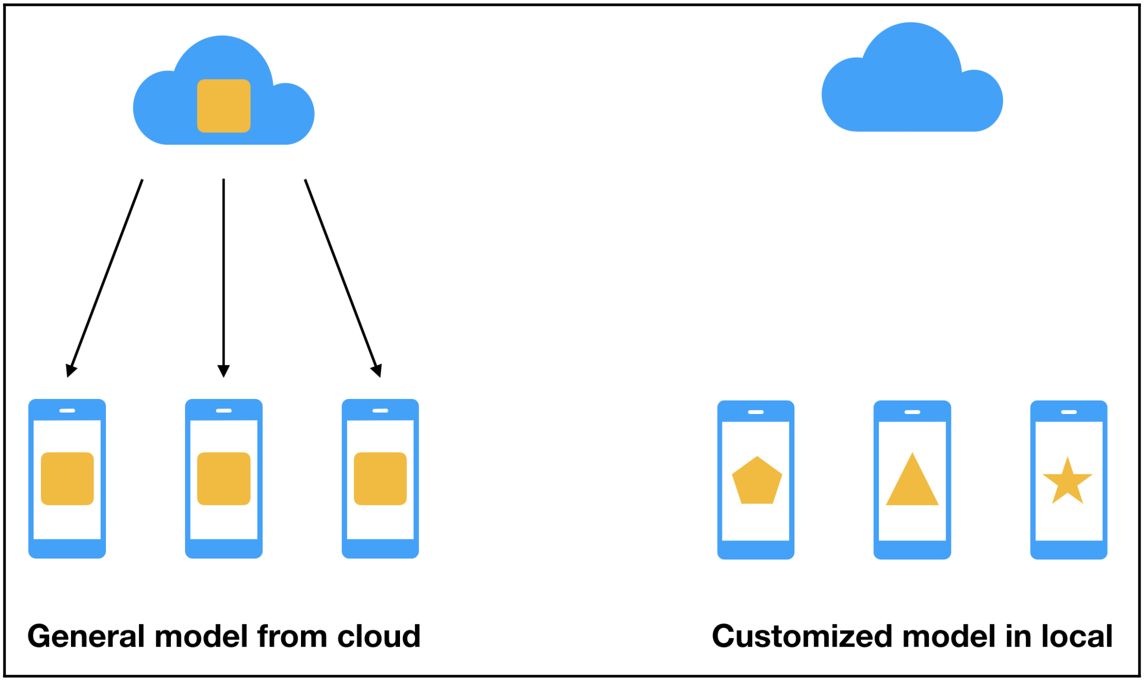
Backward Path

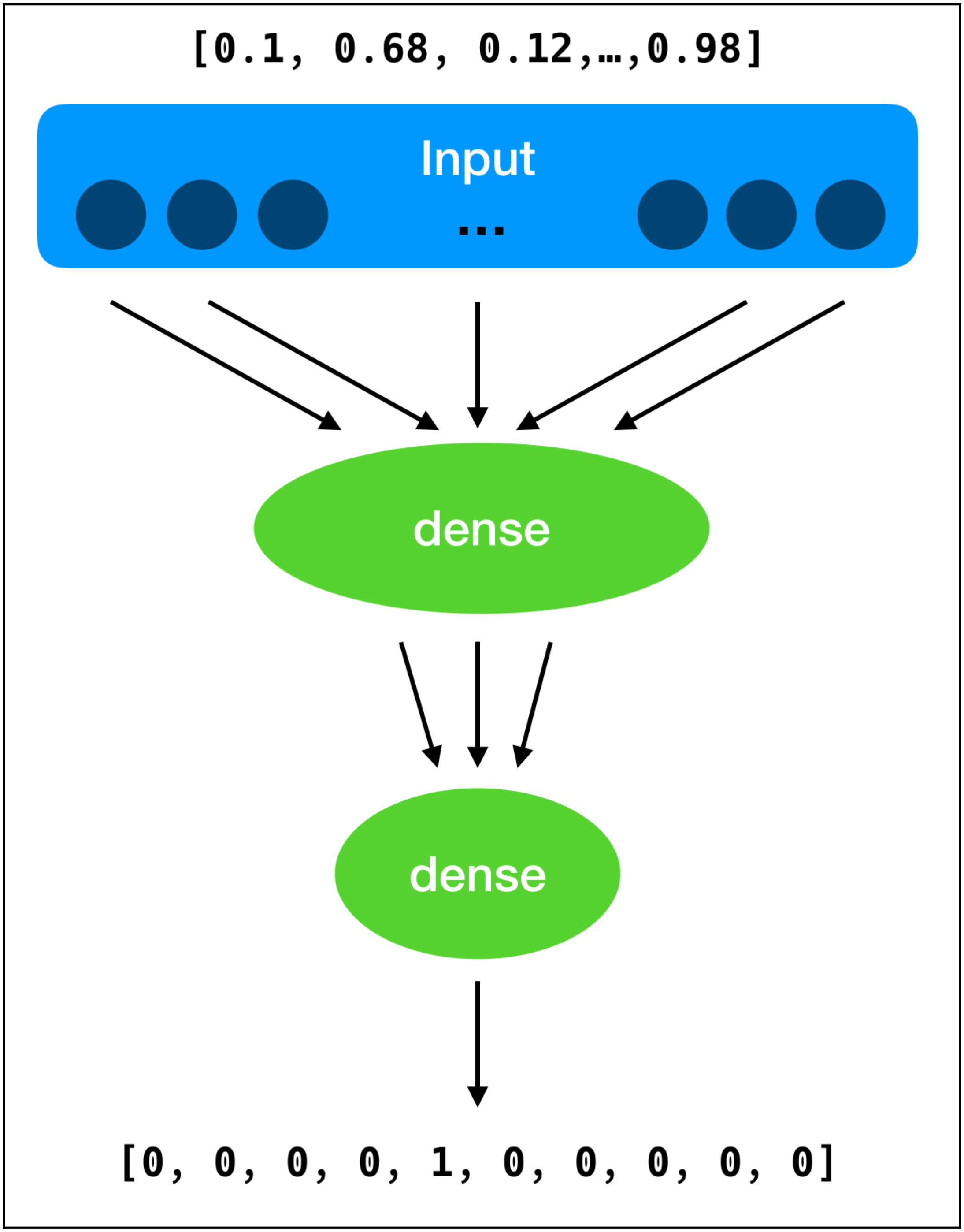

Input
reshape
Δy_1
$W_{t+1} = W_t - \alpha \Delta y_2$
W
matmul
$b_{t+1} = b_t - \beta \Delta y_2$
b
Δy_2
softmax
Δy_3
Output
$\hat{y}$
$\Delta y = y - \hat{y}$
Target
y

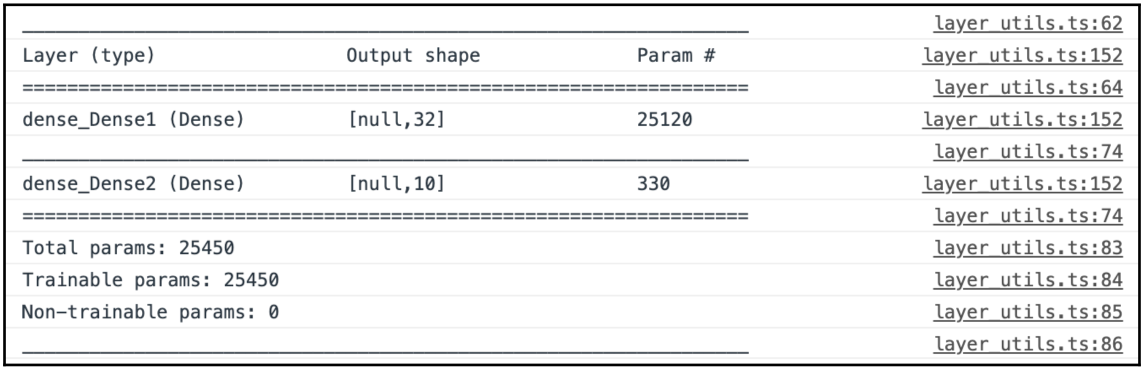


What is this?

This experiment lets anyone explore how machine learning works, in a fun, hands-on way. You can teach a machine to using your camera, live in the browser – no coding required. You train a neural network locally on your device, without sending any images to a server. That's how it responds so quickly

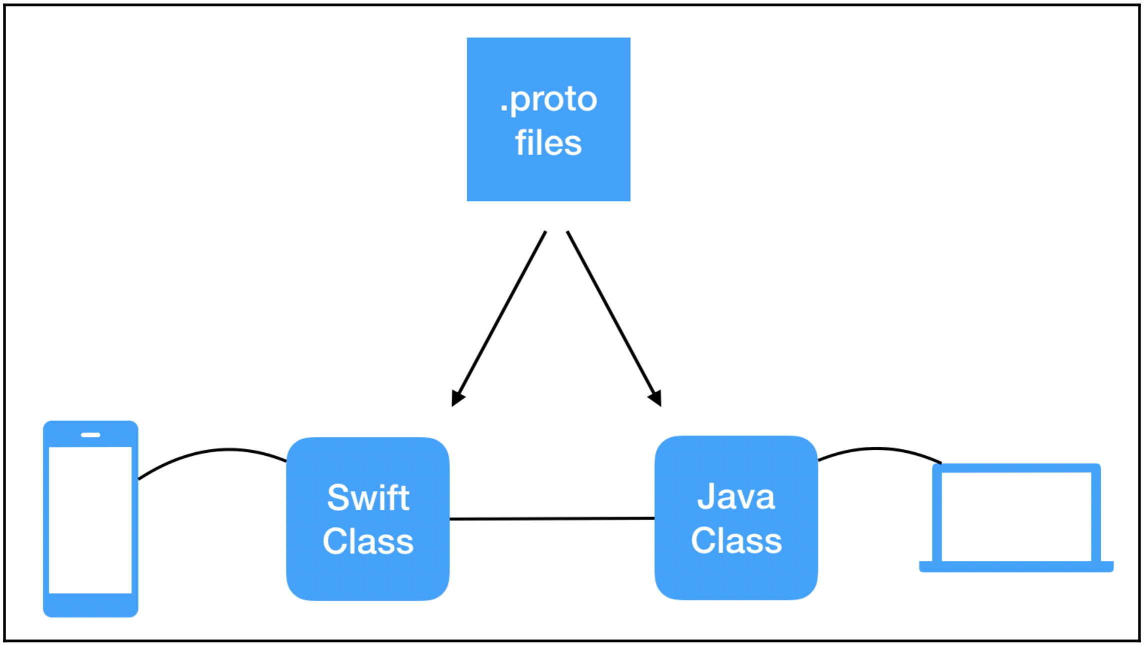

General model from cloud
Customized model in local

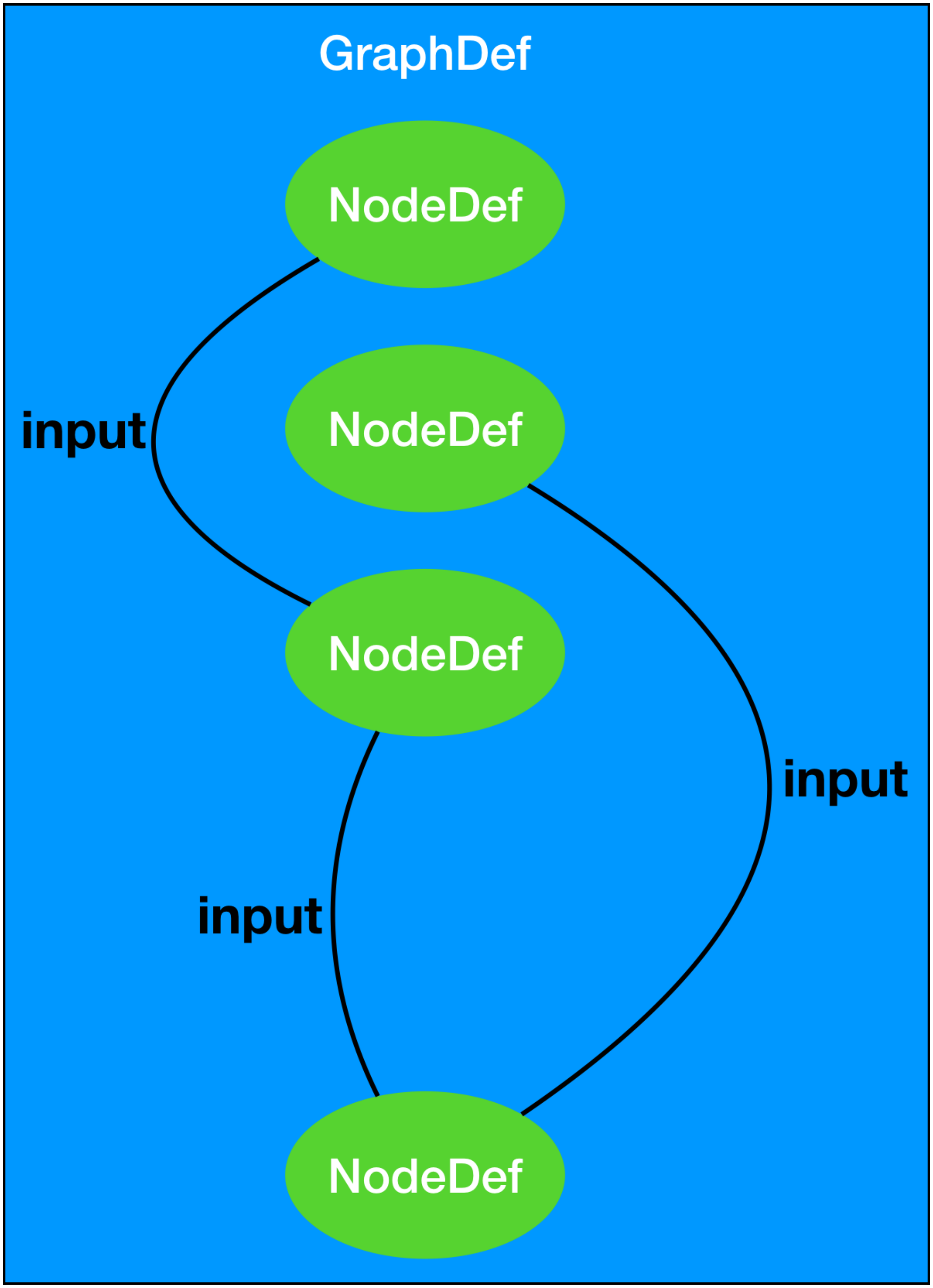

[0.1, 0.68, 0.12,…,0.98]
Input
...
dense
dense
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]

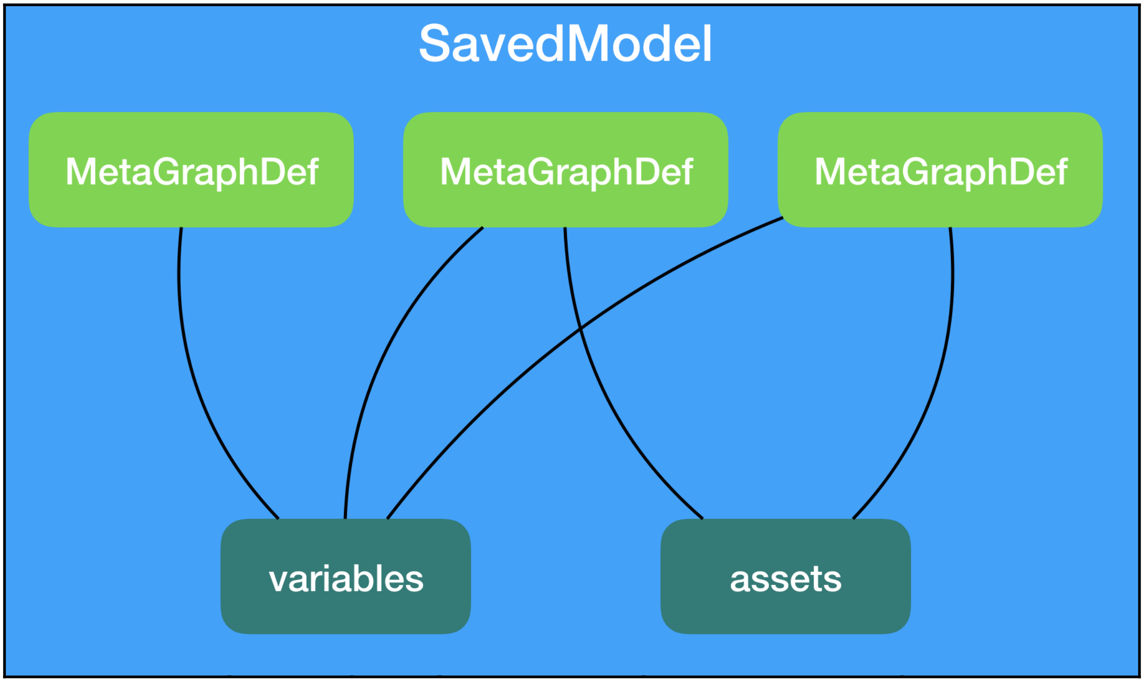


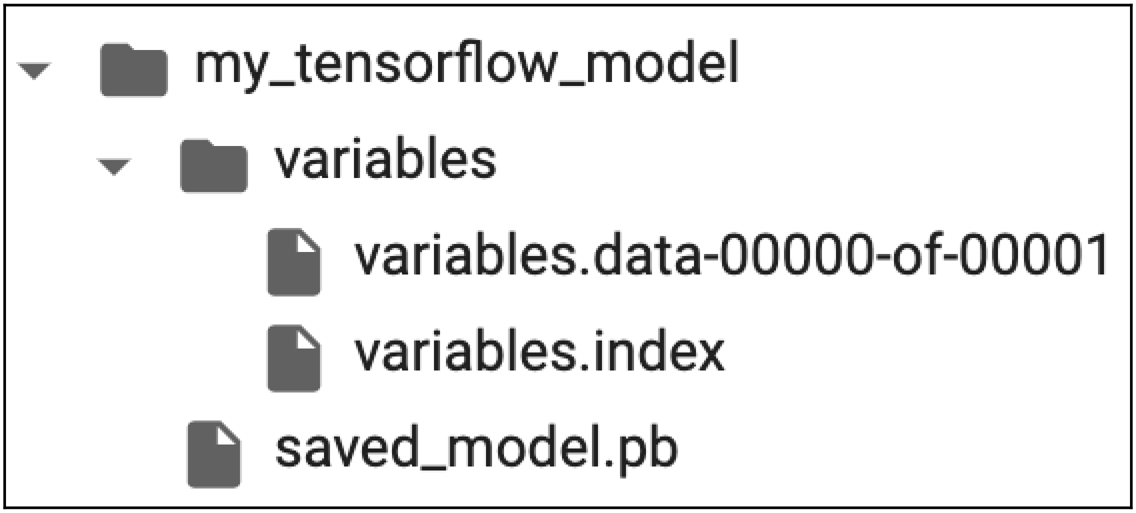
```
_________________________________________________________________          layer_utils.ts:62
Layer (type)                 Output shape              Param #             layer_utils.ts:152
=================================================================          layer_utils.ts:64
dense_Dense1 (Dense)         [null,32]                 25120               layer_utils.ts:152
_________________________________________________________________          layer_utils.ts:74
dense_Dense2 (Dense)         [null,10]                 330                 layer_utils.ts:152
=================================================================          layer_utils.ts:74
Total params: 25450                                                        layer_utils.ts:83
Trainable params: 25450                                                    layer_utils.ts:84
Non-trainable params: 0                                                    layer_utils.ts:85
_________________________________________________________________          layer_utils.ts:86
```

Chapter 2: Importing Pretrained Models into TensorFlow.js

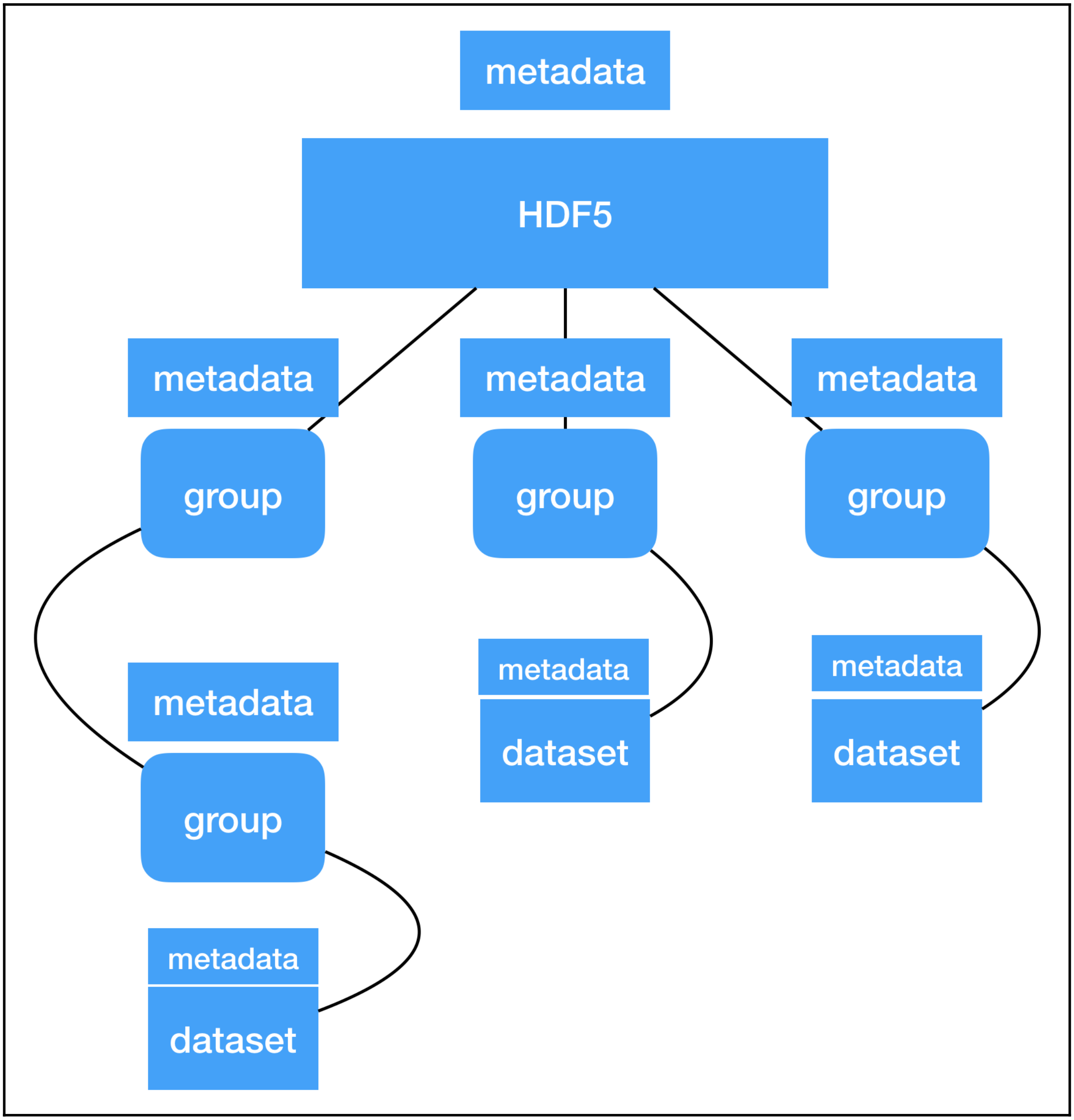



GraphDef
NodeDef
NodeDef
NodeDef
NodeDef
input
input
input

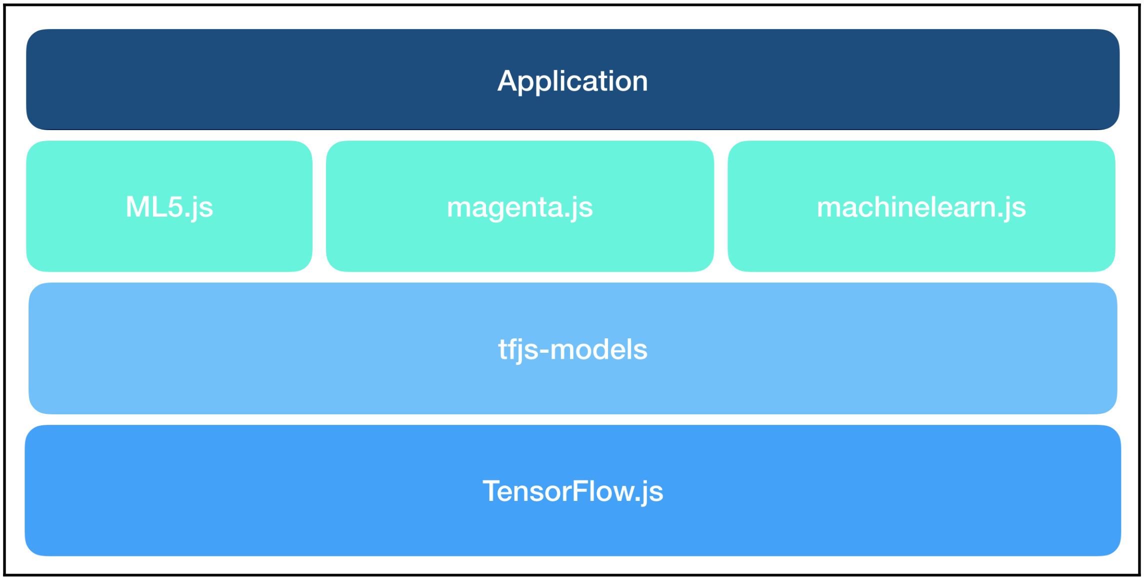




my_tensorflow_model
variables
variables.data-00000-of-00001
variables.index
saved_model.pb


metadata
HDF5
metadata
metadata
metadata
group
group
group
metadata
metadata
metadata
dataset
dataset
group
metadata
dataset

Chapter 3: TensorFlow.js Ecosystem



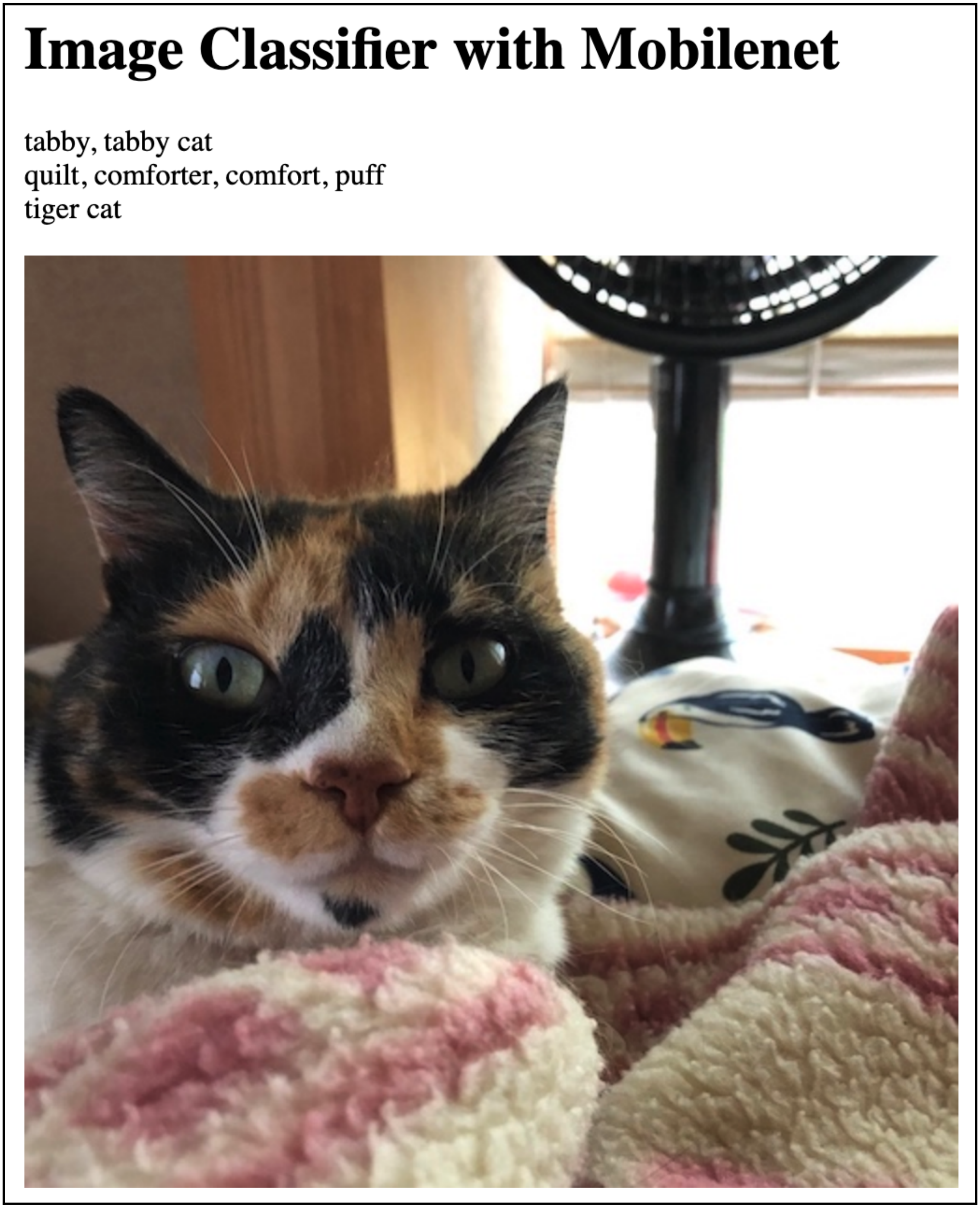
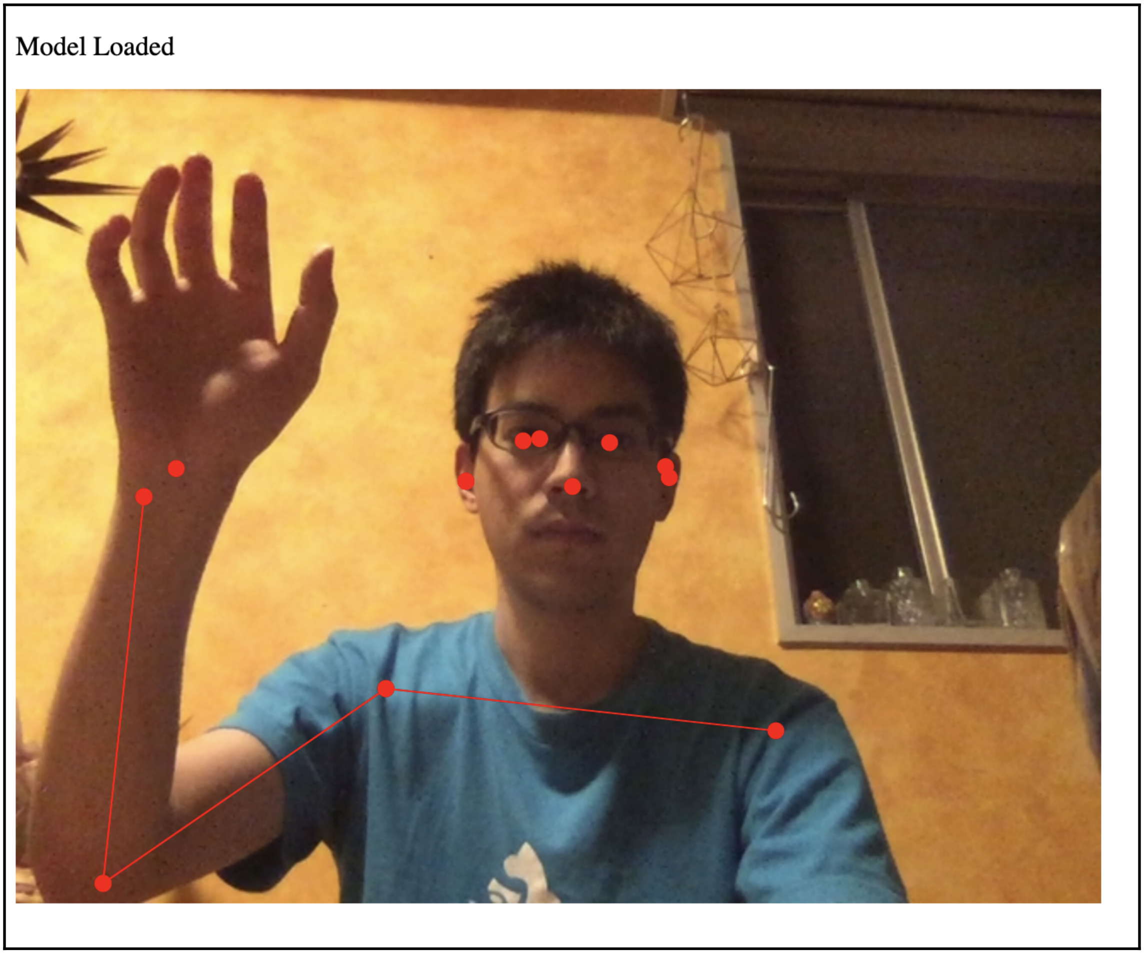
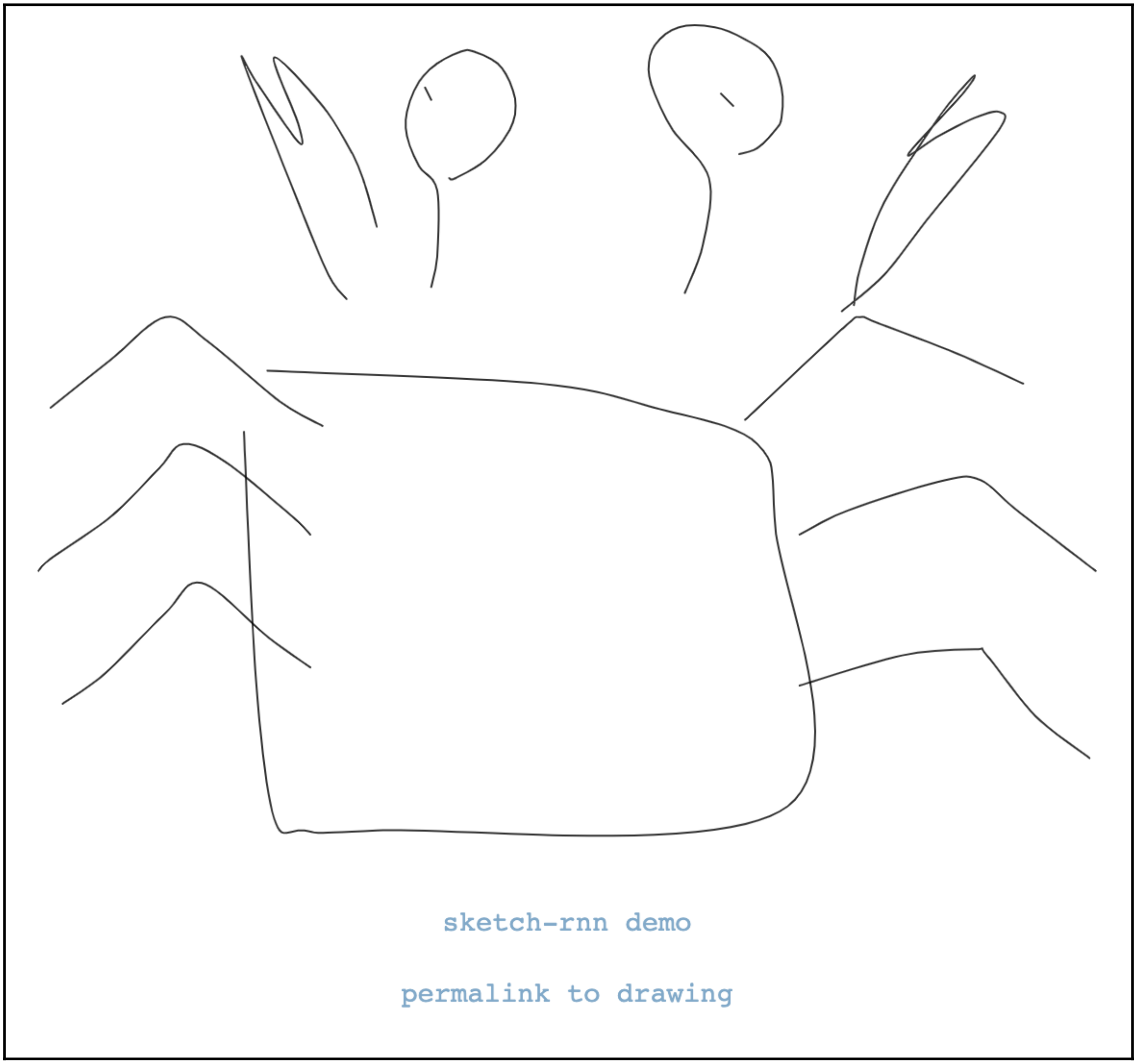


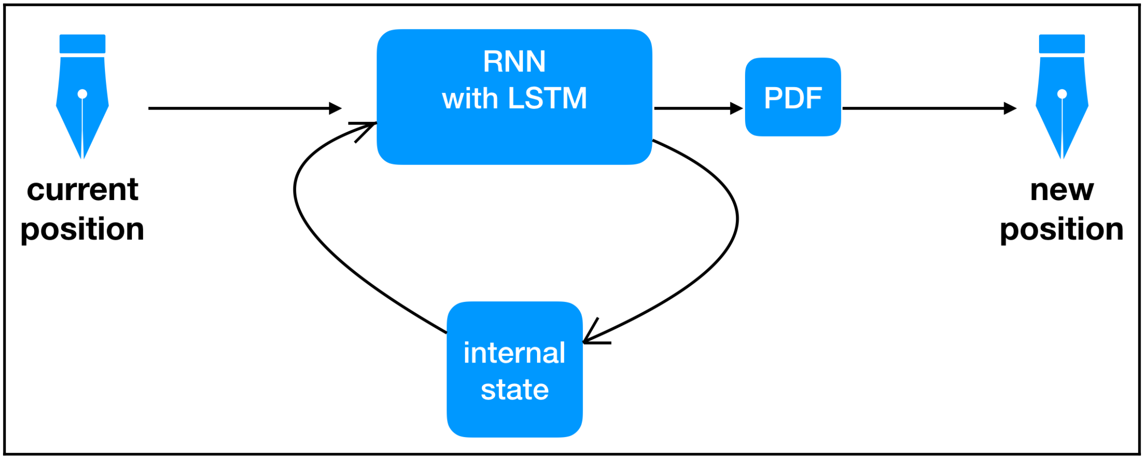
Image Classifier with Mobilenet

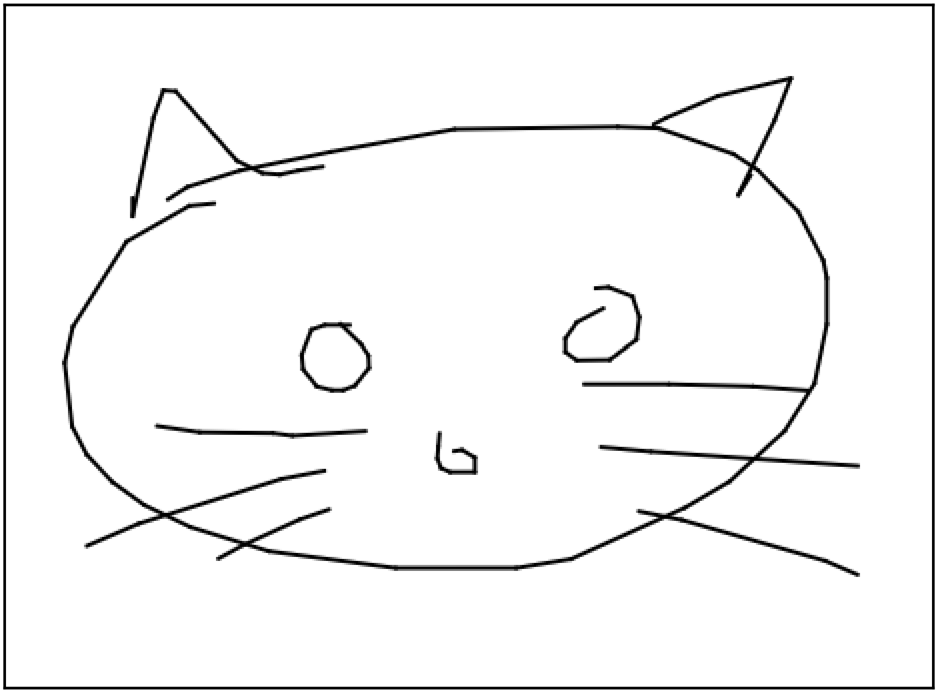
tabby, tabby cat
quilt, comforter, comfort, puff
tiger cat



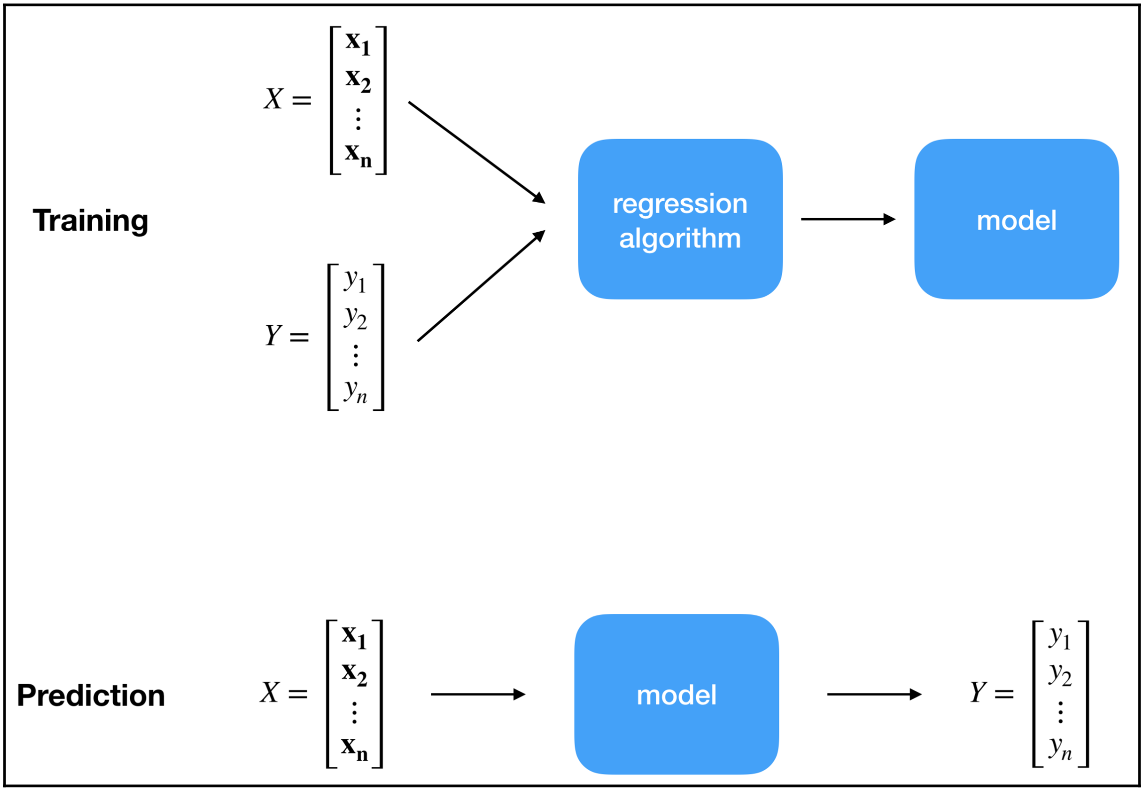
Model Loaded

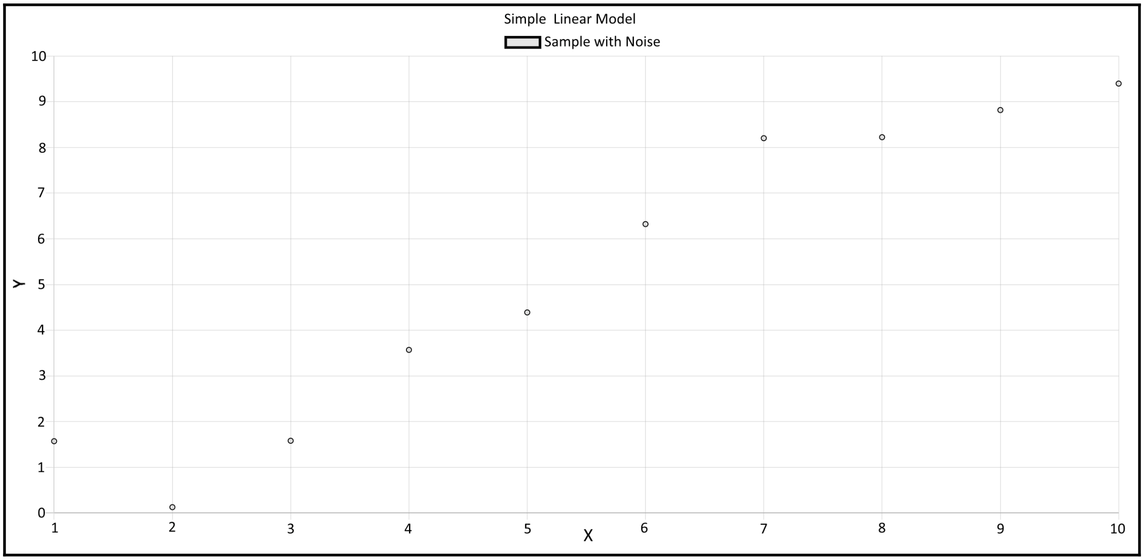

sketch-rnn demo
permalink to drawing

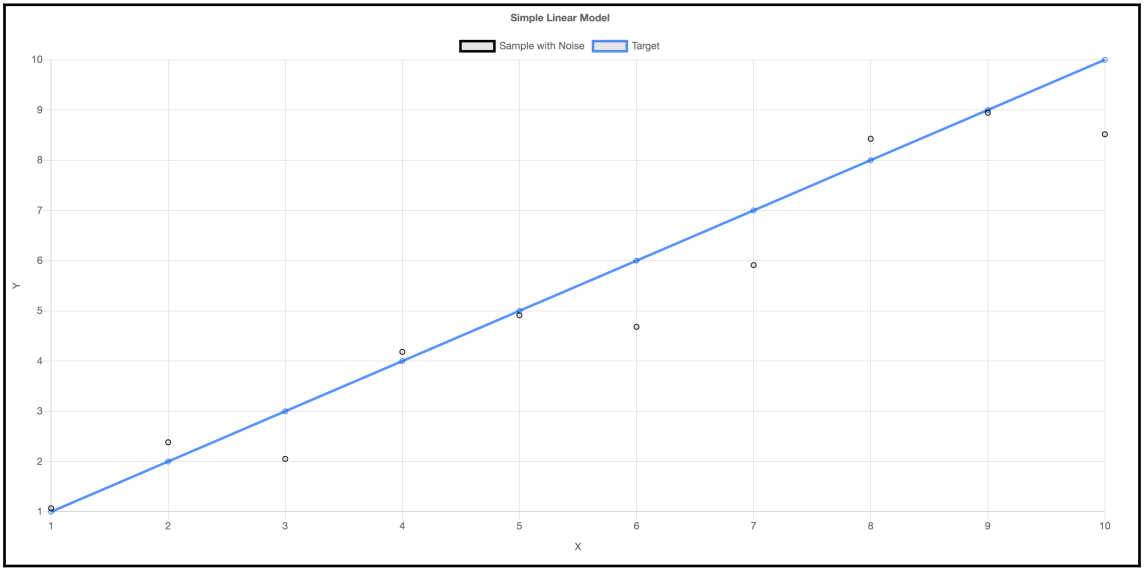

current position
RNN with LSTM
PDF
new position
internal state

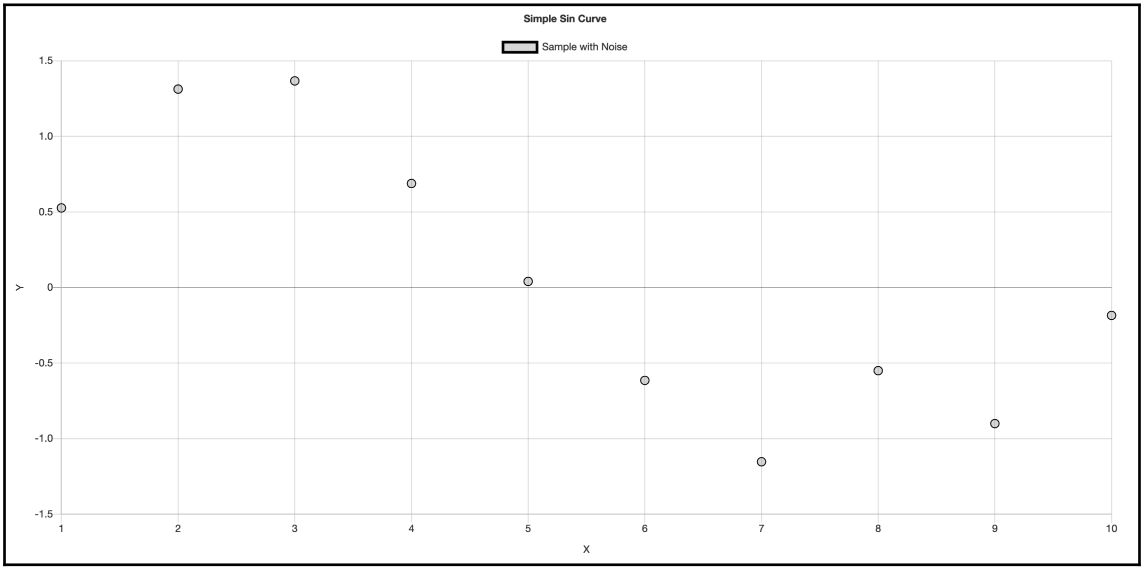


Chapter 4: Polynomial Regression

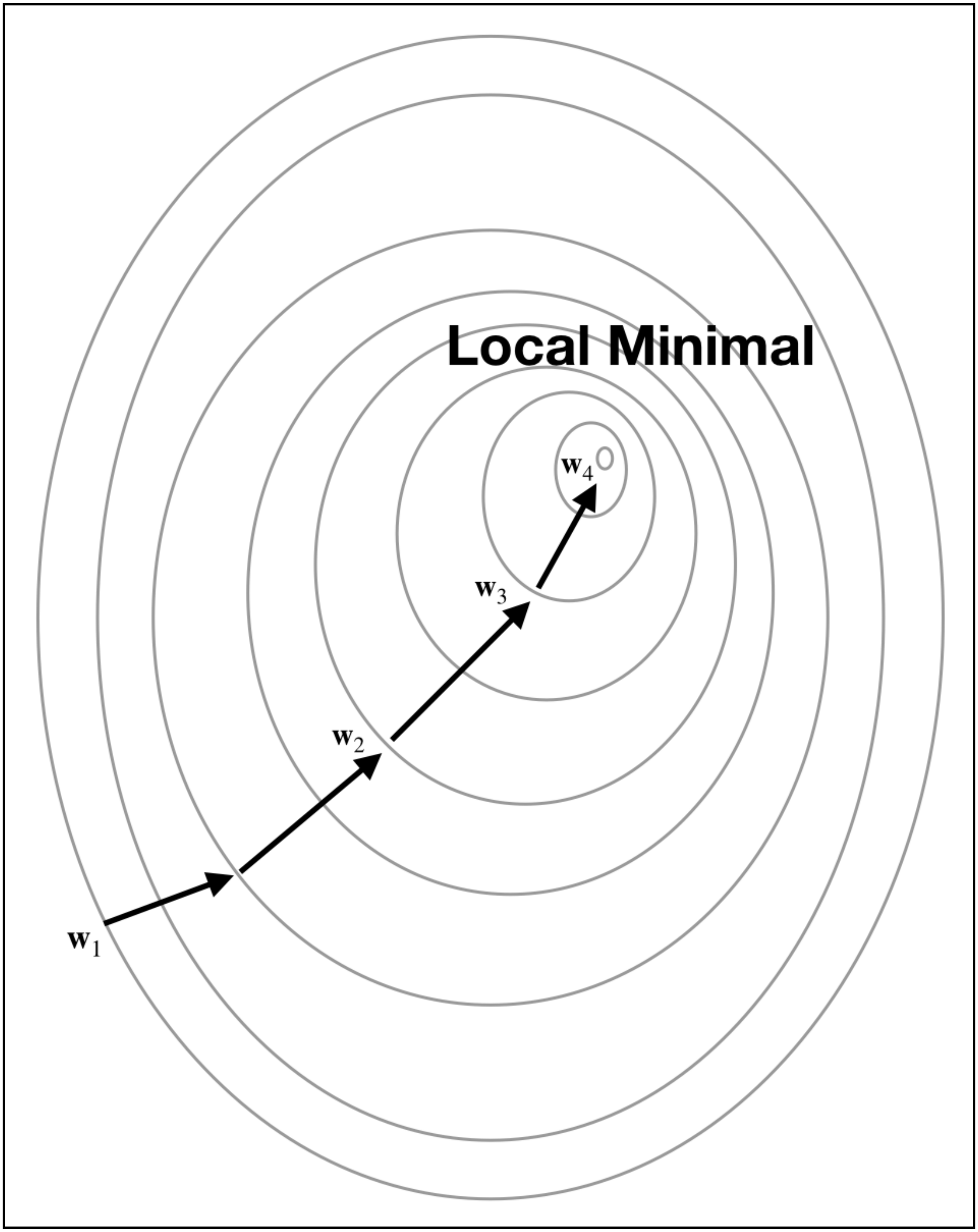

Training
$X = \begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \\ \vdots \\ \mathbf{x_n} \end{bmatrix}$
$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$
regression algorithm
model
Prediction
$X = \begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \\ \vdots \\ \mathbf{x_n} \end{bmatrix}$
model
$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

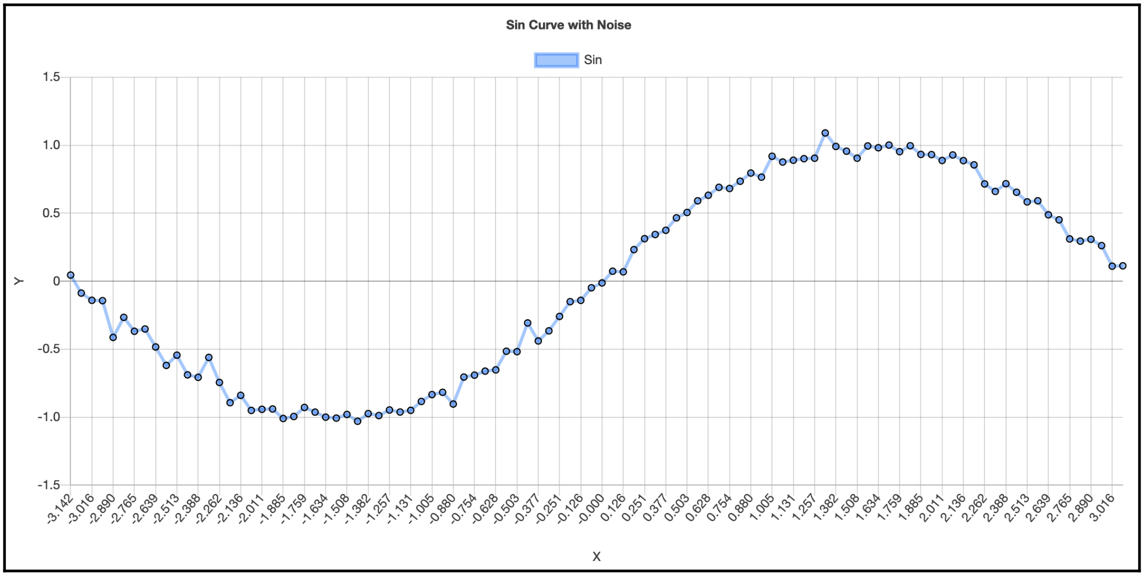

Simple Linear Model
Sample with Noise
Y
X
0
1
2
3
4
5
6
7
8
9
10
1
2
3
4
5
6
7
8
9
10

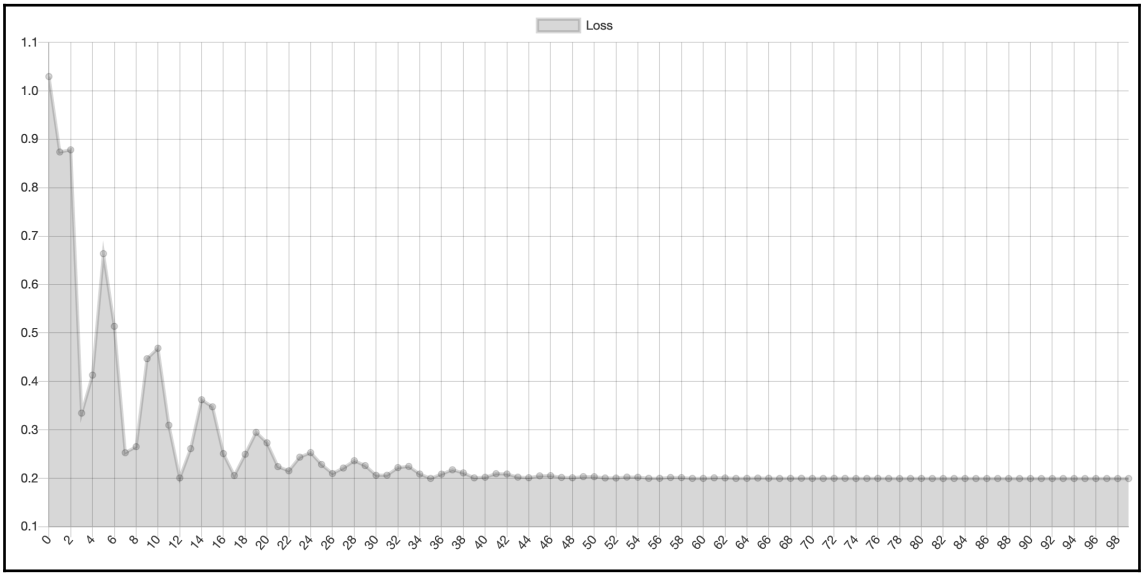

Simple Linear Model
Sample with Noise
Target
Y
X
1
2
3
4
5
6
7
8
9
10
1
2
3
4
5
6
7
8
9
10

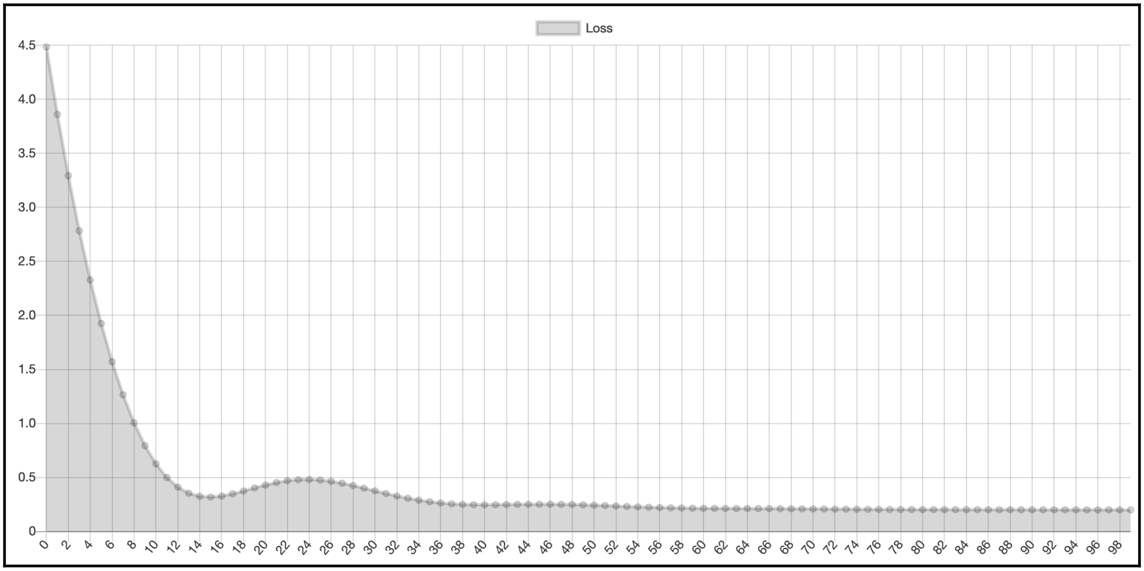


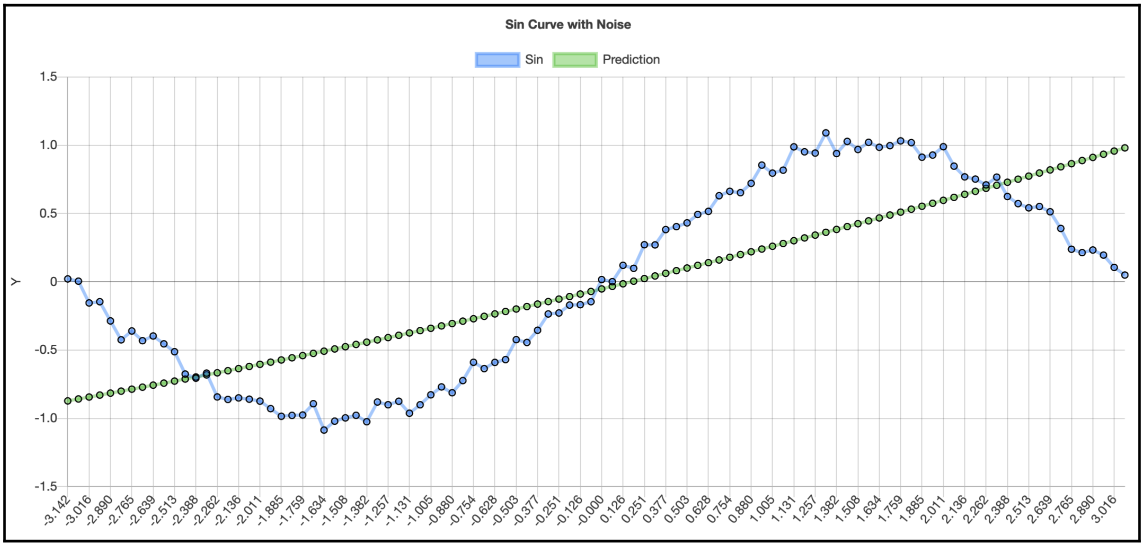
$$\begin{aligned} E(\mathbf{w}_{i+1}) - E(\mathbf{w}_i) &= E(\mathbf{w}_i - \alpha \nabla E(\mathbf{w}_i)) - E(\mathbf{w}_i) \\ &= \alpha \left(\nabla E(\mathbf{w}_i)^T (-\nabla E(\mathbf{w}_i)) + \frac{o(\alpha)}{\alpha} \right) \\ &= \alpha \left(-\left| \nabla E(\mathbf{w}_i) \right|^2 + \frac{o(\alpha)}{\alpha} \right) \end{aligned}$$

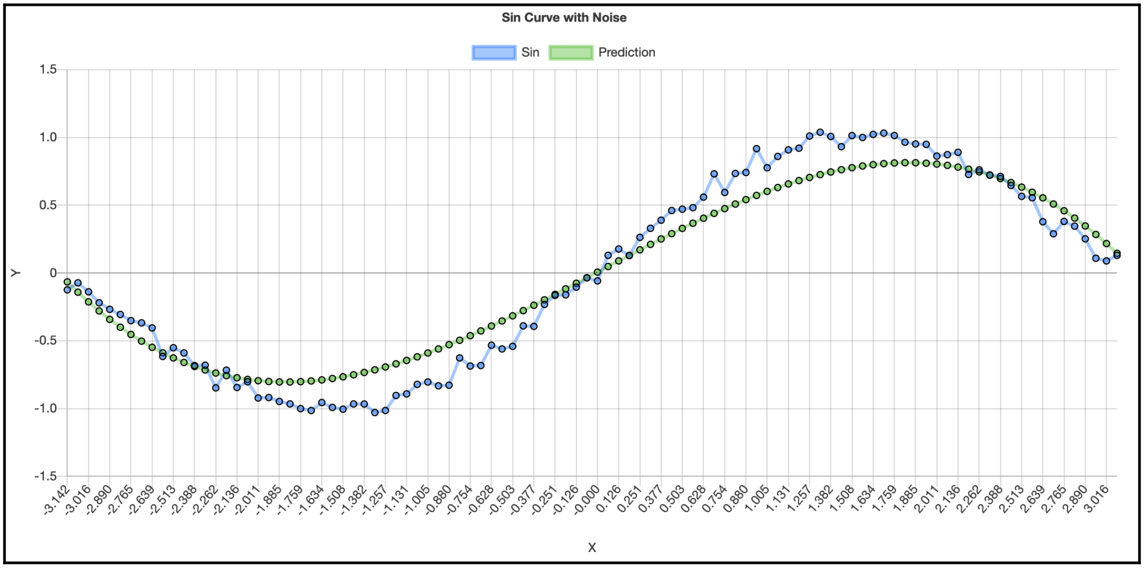

Local Minimal
$\mathbf{w}_4$
$\mathbf{w}_3$
$\mathbf{w}_2$
$\mathbf{w}_1$

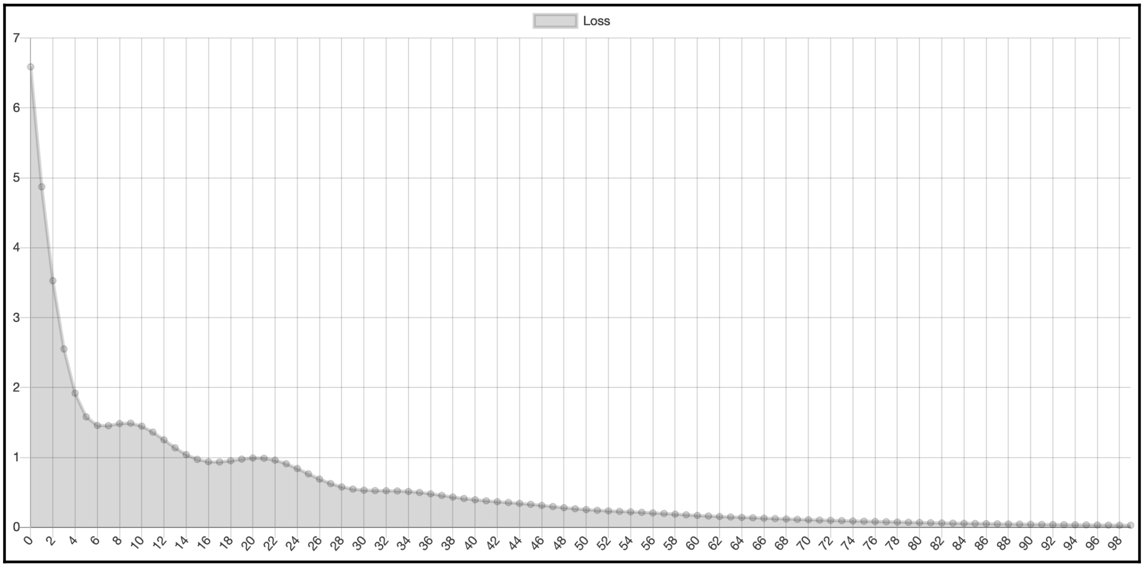

Sin Curve with Noise
Sin
Y
1.5
1.0
0.5
0
-0.5
-1.0
-1.5
-3.142 -3.016 -2.890 -2.765 -2.639 -2.513 -2.388 -2.262 -2.136 -2.011 -1.885 -1.759 -1.634 -1.508 -1.382 -1.257 -1.131 -1.005 -0.880 -0.754 -0.628 -0.503 -0.377 -0.251 -0.126 -0.000 0.126 0.251 0.377 0.503 0.628 0.754 0.880 1.005 1.131 1.257 1.382 1.508 1.634 1.759 1.885 2.011 2.136 2.262 2.388 2.513 2.639 2.765 2.890 3.016
X

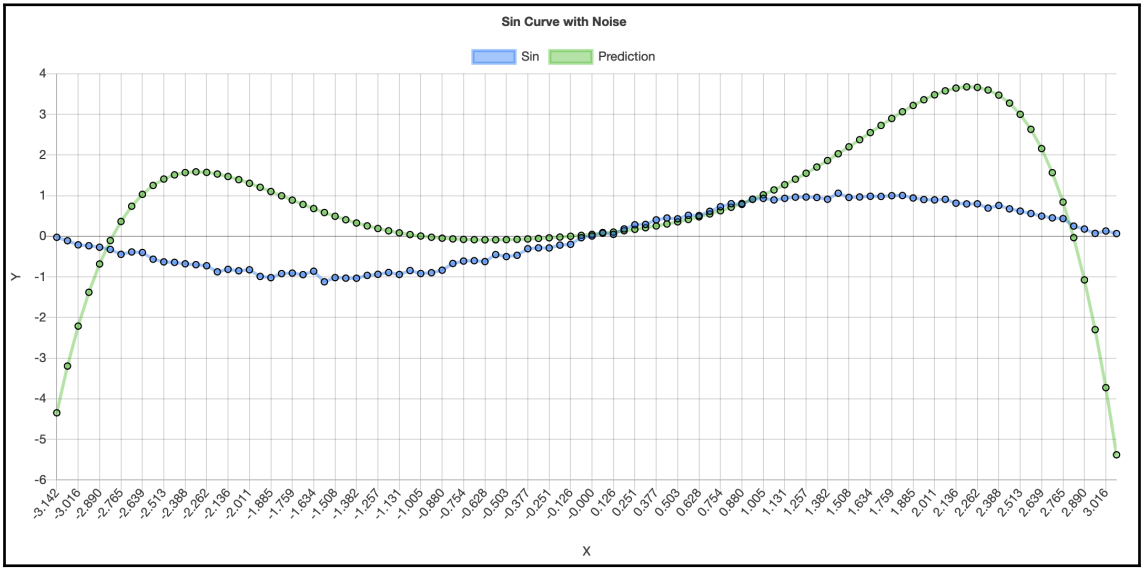

Loss
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98


Loss
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98

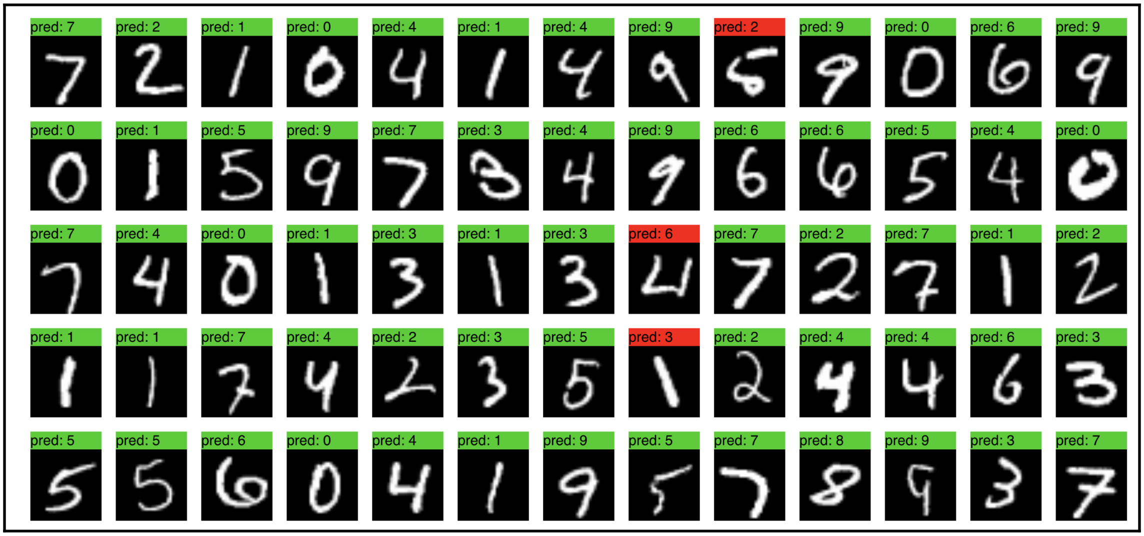

Sin Curve with Noise
Sin
Prediction
Y
1.5
1.0
0.5
0
-0.5
-1.0
-1.5
-3.142 -3.016 -2.890 -2.765 -2.639 -2.513 -2.388 -2.262 -2.136 -2.011 -1.885 -1.759 -1.634 -1.508 -1.382 -1.257 -1.131 -1.005 -0.880 -0.754 -0.628 -0.503 -0.377 -0.251 -0.126 -0.000 0.126 0.251 0.377 0.503 0.628 0.754 0.880 1.005 1.131 1.257 1.382 1.508 1.634 1.759 1.885 2.011 2.136 2.262 2.388 2.513 2.639 2.765 2.890 3.016

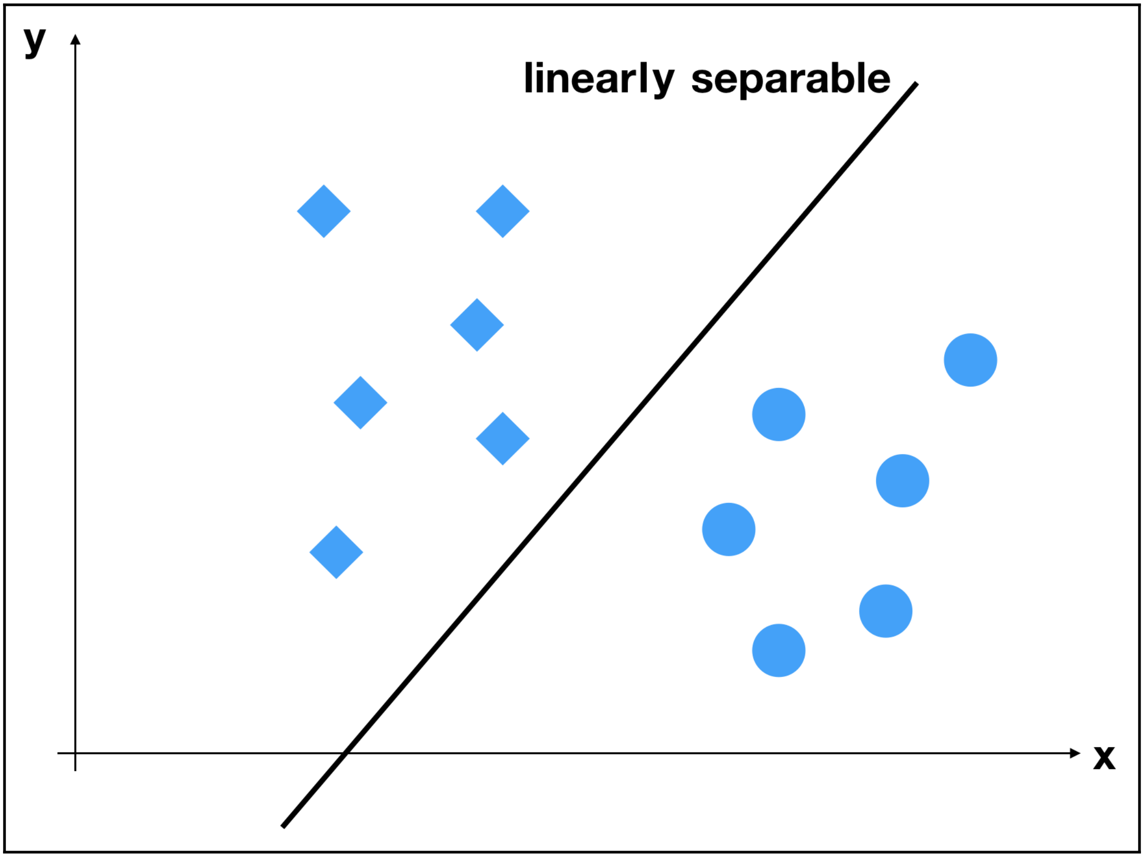

Sin Curve with Noise
Sin
Prediction
1.5
1.0
0.5
0
-0.5
-1.0
-1.5
Y
-3.142
-3.016
-2.890
-2.765
-2.639
-2.513
-2.388
-2.262
-2.136
-2.011
-1.885
-1.759
-1.634
-1.508
-1.382
-1.257
-1.131
-1.005
-0.880
-0.754
-0.628
-0.503
-0.377
-0.251
-0.126
-0.000
0.126
0.251
0.377
0.503
0.628
0.754
0.880
1.005
1.131
1.257
1.382
1.508
1.634
1.759
1.885
2.011
2.136
2.262
2.388
2.513
2.639
2.765
2.890
3.016
X

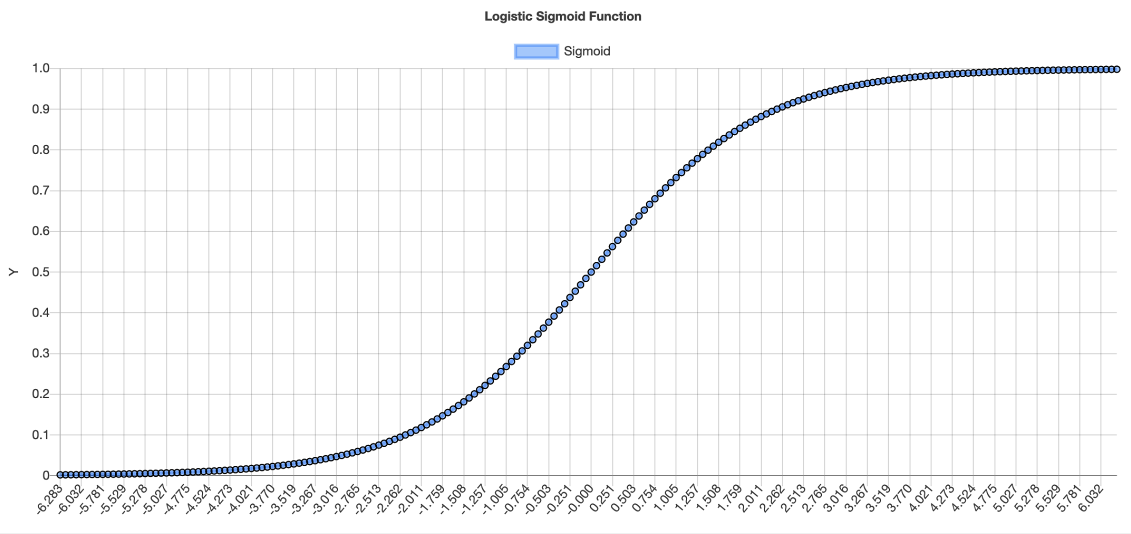

Loss
7
6
5
4
3
2
1
0
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
52
54
56
58
60
62
64
66
68
70
72
74
76
78
80
82
84
86
88
90
92
94
96
98


Sin Curve with Noise
Sin
Prediction
Y
4
3
2
1
0
-1
-2
-3
-4
-5
-6
X
-3.142
-3.016
-2.890
-2.765
-2.639
-2.513
-2.388
-2.262
-2.136
-2.011
-1.885
-1.759
-1.634
-1.508
-1.382
-1.257
-1.131
-1.005
-0.880
-0.754
-0.628
-0.503
-0.377
-0.251
-0.126
-0.000
0.126
0.251
0.377
0.503
0.628
0.754
0.880
1.005
1.131
1.257
1.382
1.508
1.634
1.759
1.885
2.011
2.136
2.262
2.388
2.513
2.639
2.765
2.890
3.016

Chapter 5: Classification with Logistic Regression



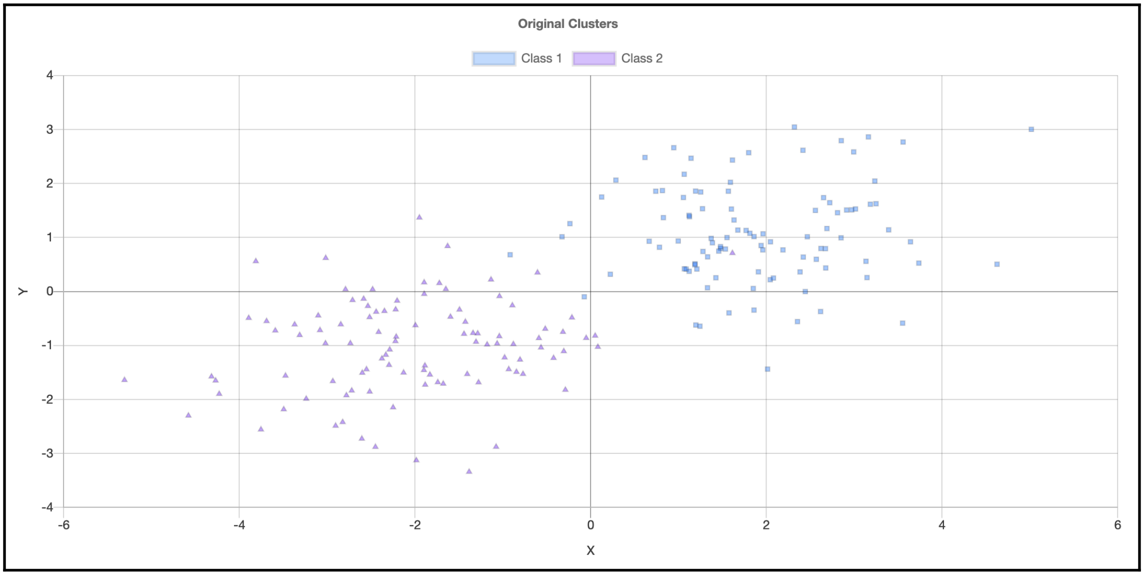

y
linearly separable
x

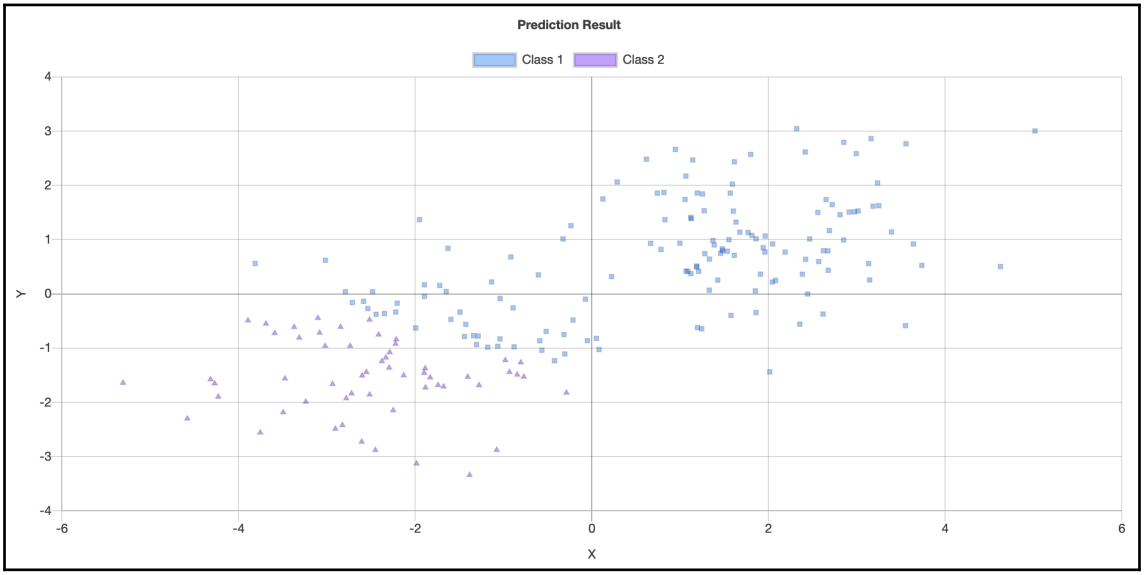


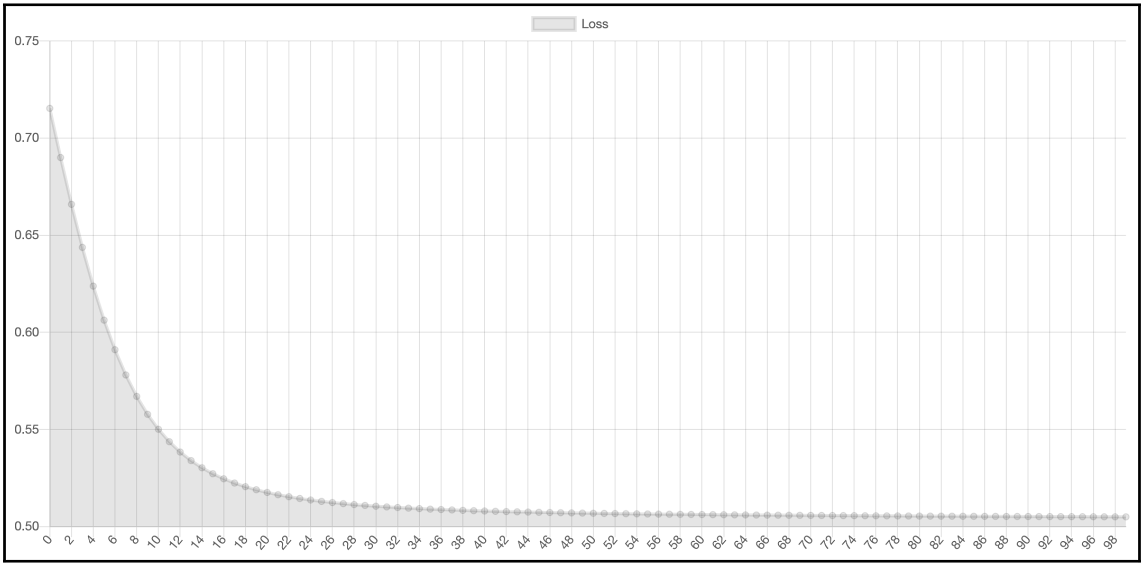
$$\begin{aligned} p(C_1|\mathbf{x}) &= \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_1)p(C_1) + p(\mathbf{x}|C_2)p(C_2)} \\ &= \frac{1}{a + \exp(-a)} \\ &= \sigma(a) \end{aligned}$$

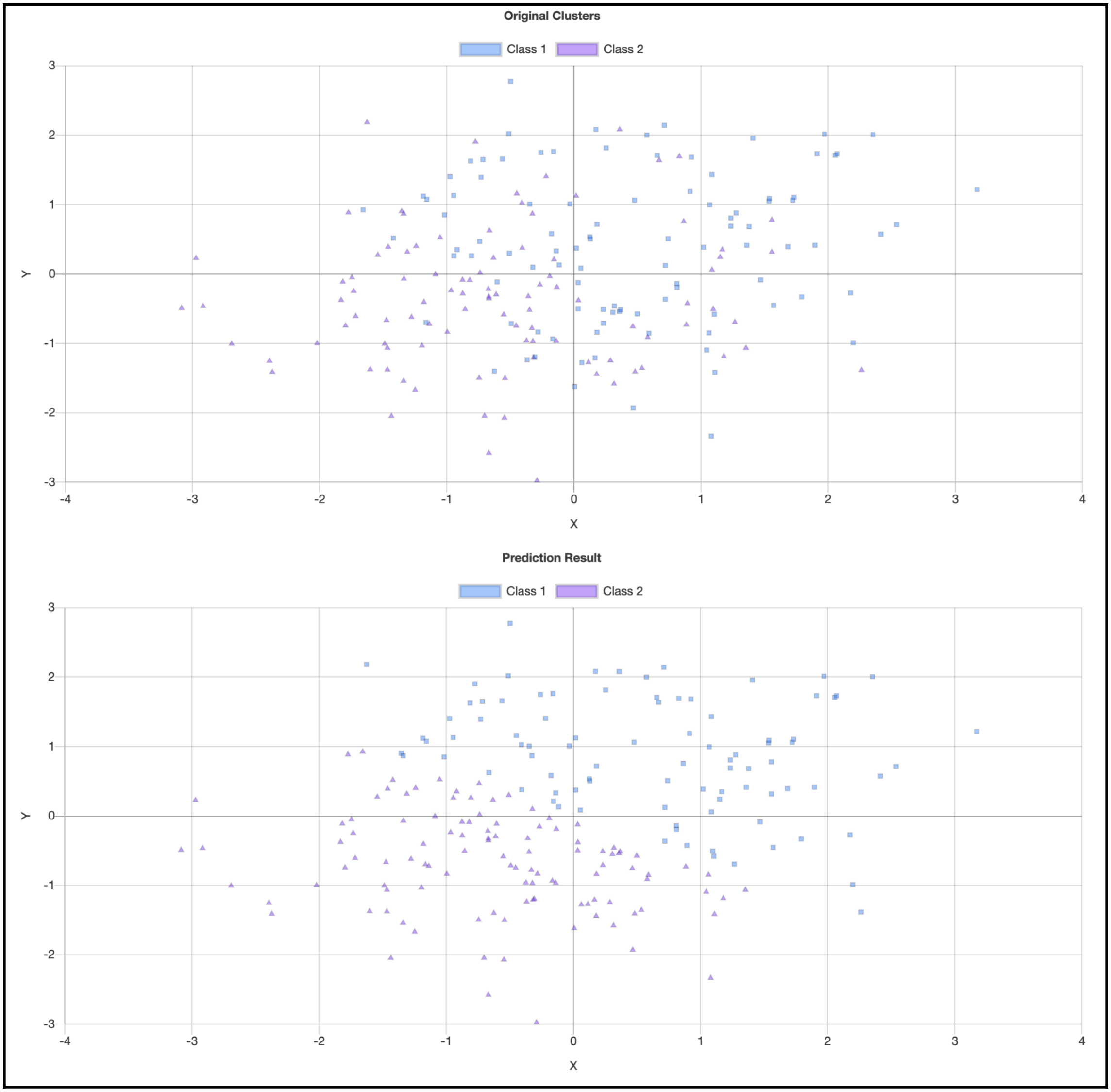
$$p(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0)$$

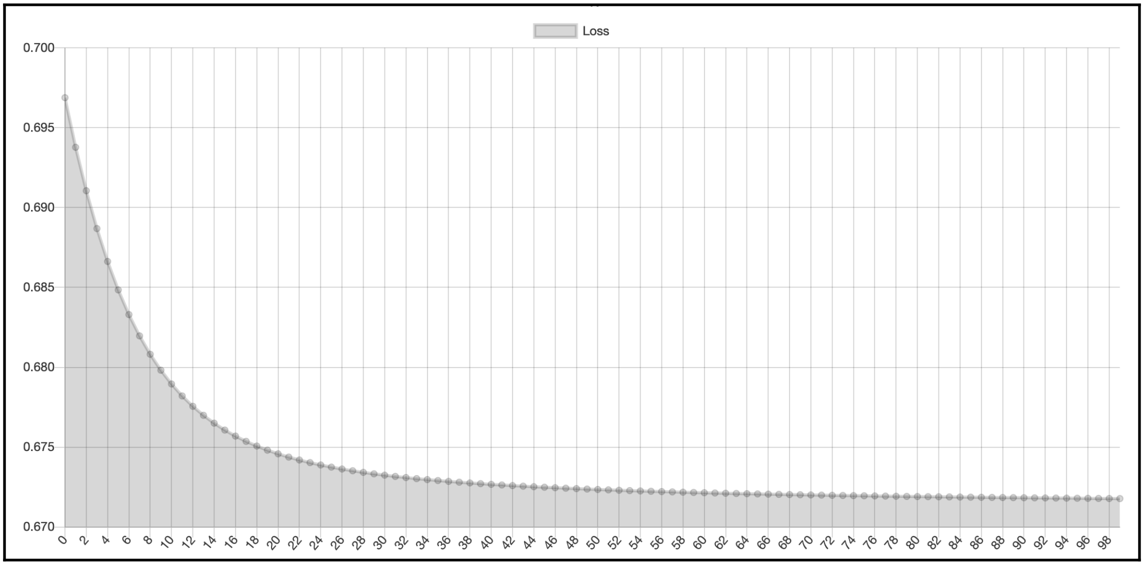
$$\begin{aligned} \mathbf{w} &= \Sigma^{-1}(\mu_1 - \mu_2) \\ w_0 &= -\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 + \ln\frac{p(C_1)}{p(C_2)} \end{aligned}$$

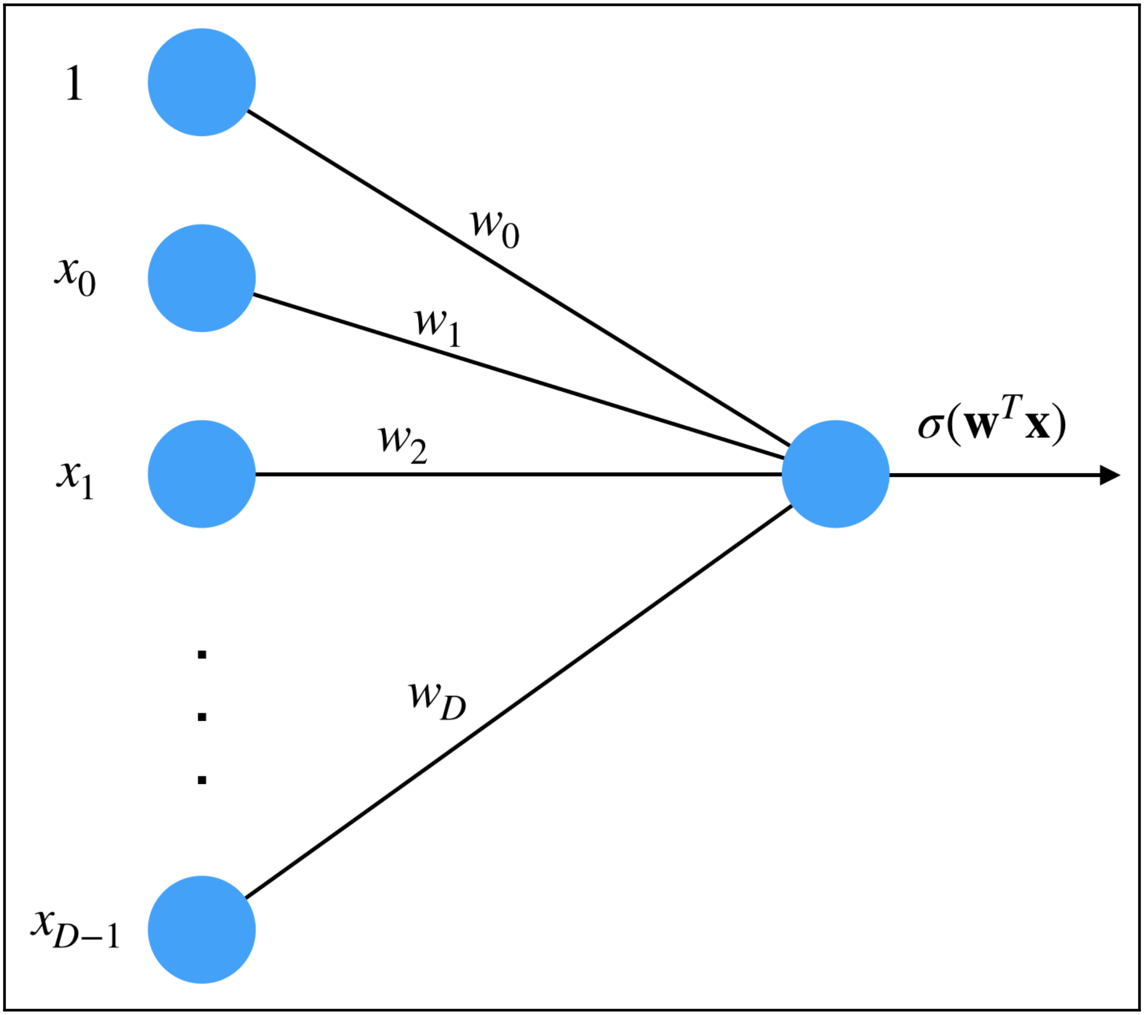

Original Clusters
Class 1
Class 2
Y
X
4
3
2
1
0
-1
-2
-3
-4
-6
-4
-2
0
2
4
6

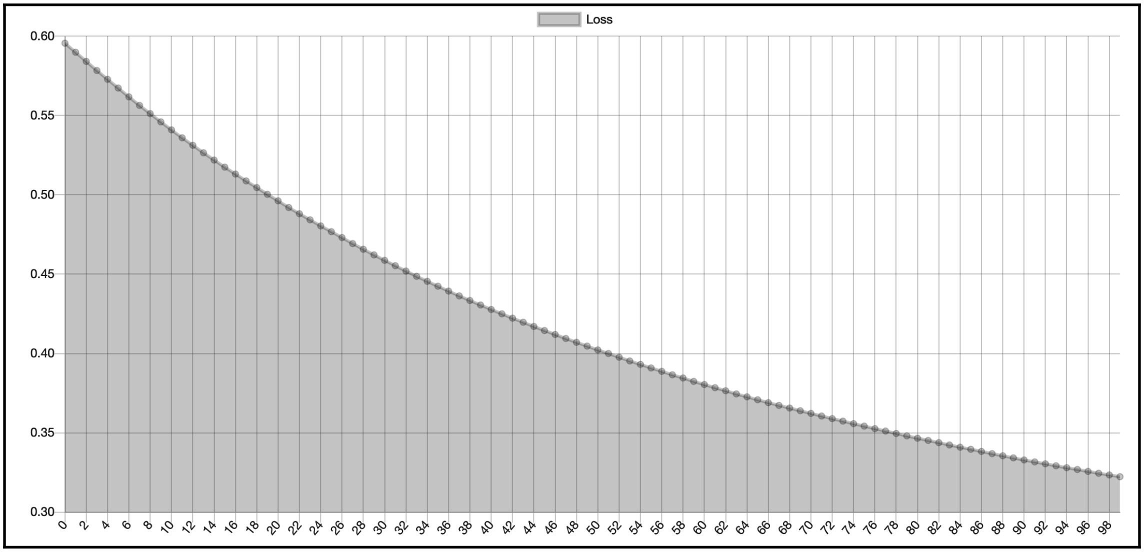

Prediction Result
Class 1
Class 2
Y
X
4
3
2
1
0
-1
-2
-3
-4
-6
-4
-2
0
2
4
6

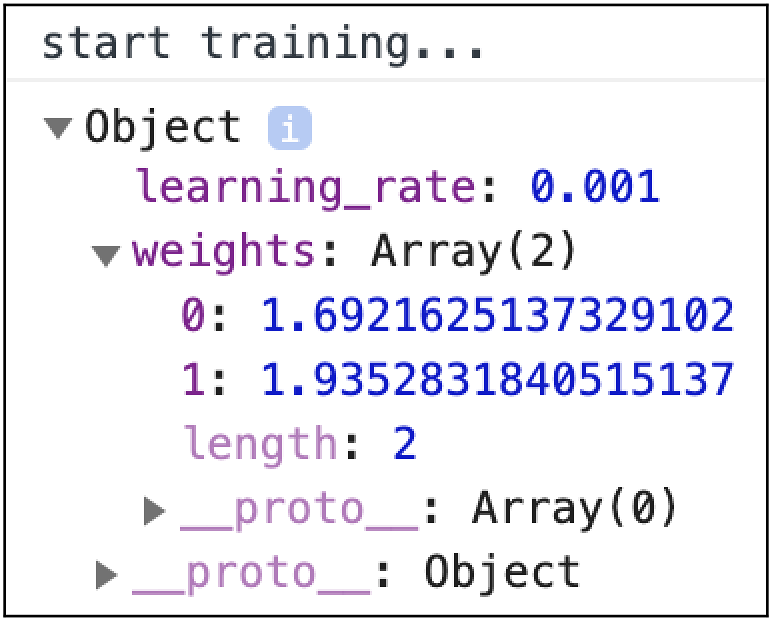

Loss
0.75
0.70
0.65
0.60
0.55
0.50
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
52
54
56
58
60
62
64
66
68
70
72
74
76
78
80
82
84
86
88
90
92
94
96
98


Original Clusters
Class 1
Class 2
3
2
1
0
-1
-2
-3
Y
-4
-3
-2
-1
0
1
2
3
4
X
Prediction Result
Class 1
Class 2
3
2
1
0
-1
-2
-3
Y
-4
-3
-2
-1
0
1
2
3
4
X


Loss
0.700
0.695
0.690
0.685
0.680
0.675
0.670
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98

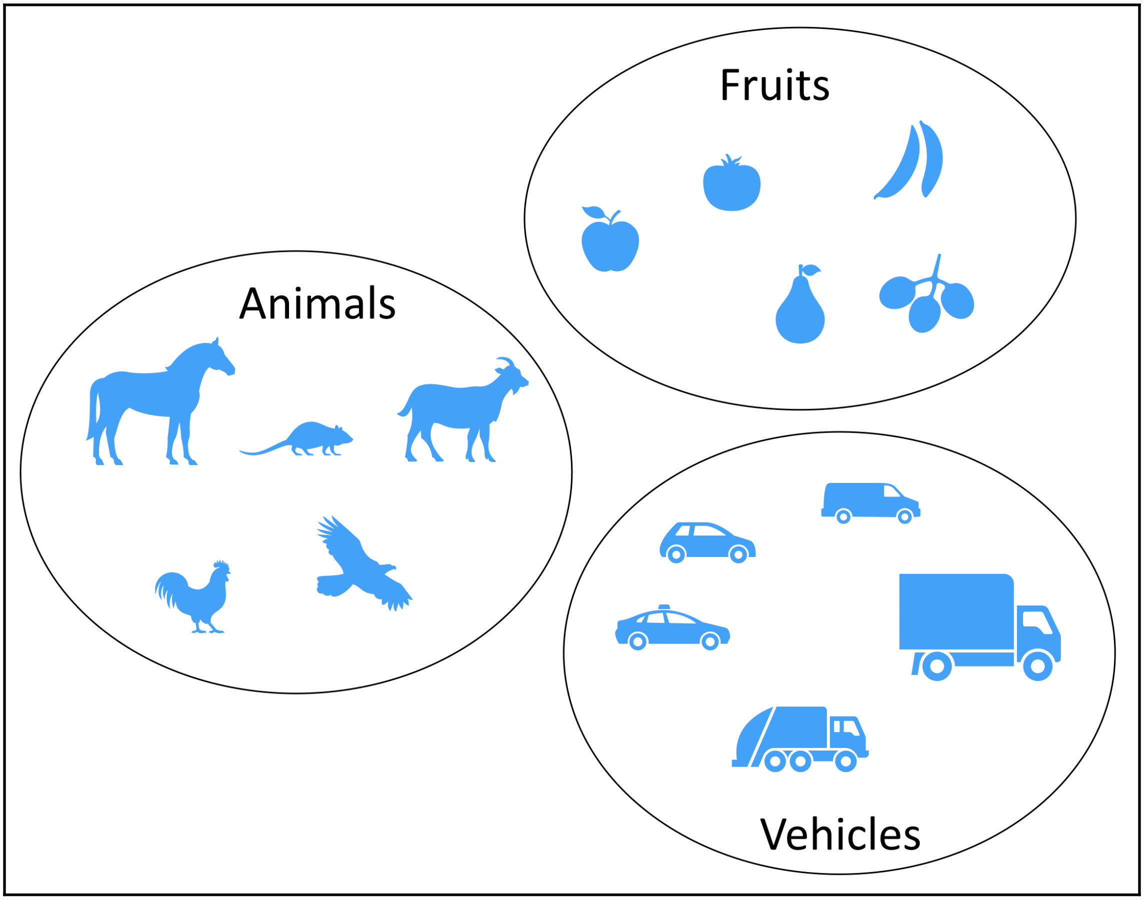

1
x_0
x_1
x_{D-1}
w_0
w_1
w_2
w_D
$\sigma(\mathbf{w}^T\mathbf{x})$

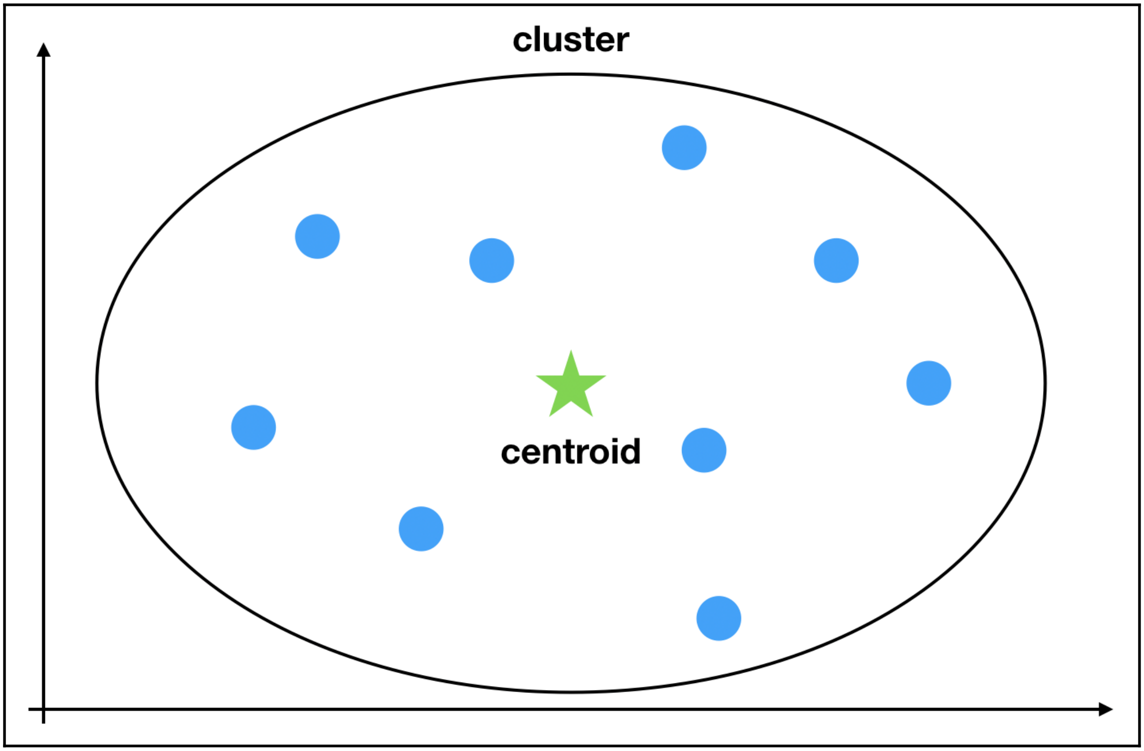


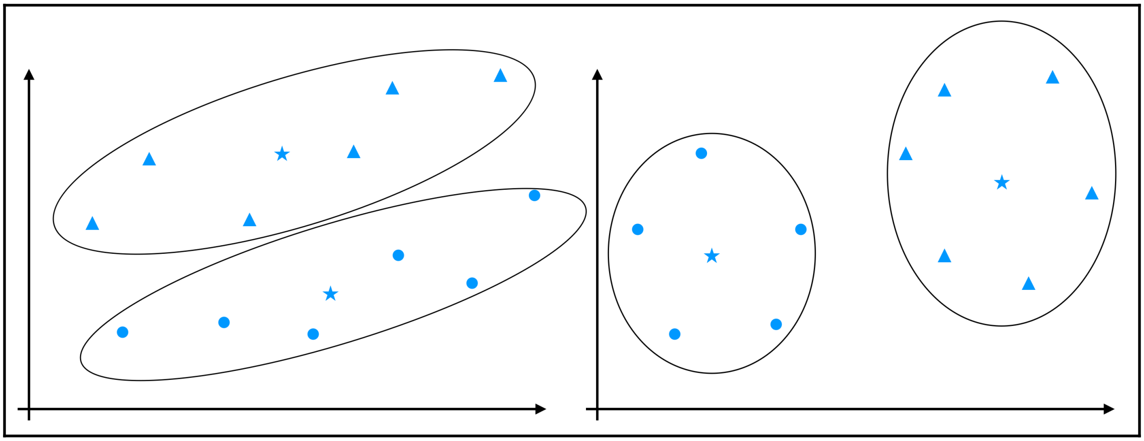


```
start training...
▼Object i
    learning_rate: 0.001
  ▼weights: Array(2)
      0: 1.6921625137329102
      1: 1.9352831840515137
      length: 2
    ▶__proto__: Array(0)
  ▶__proto__: Object
```

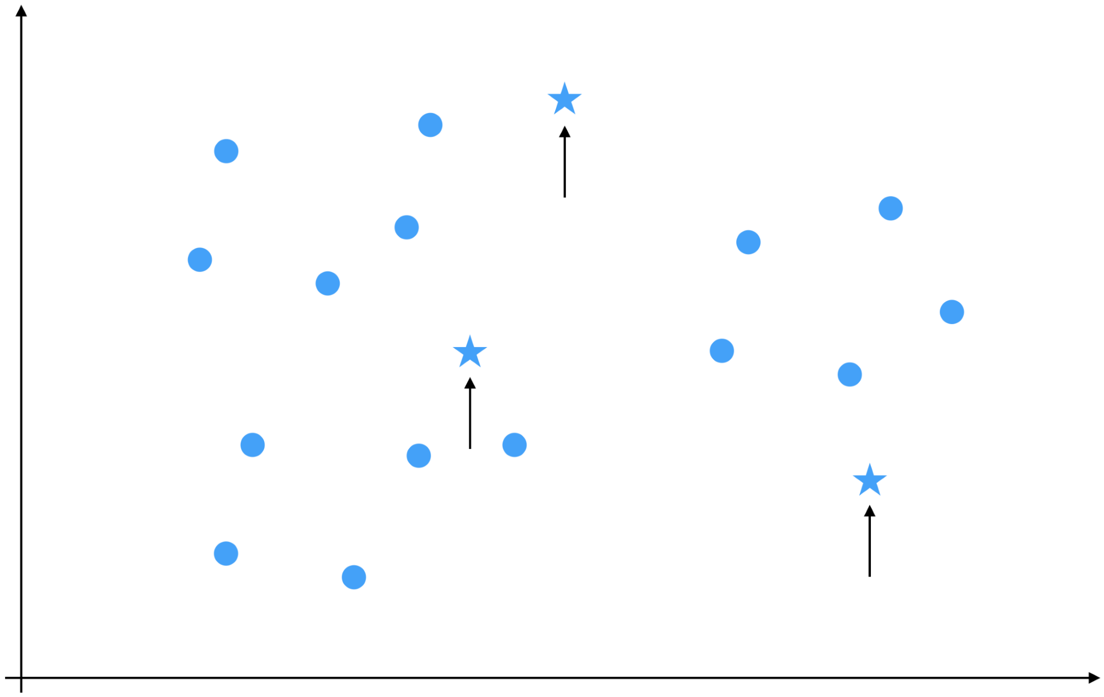
Chapter 6: Unsupervised Learning



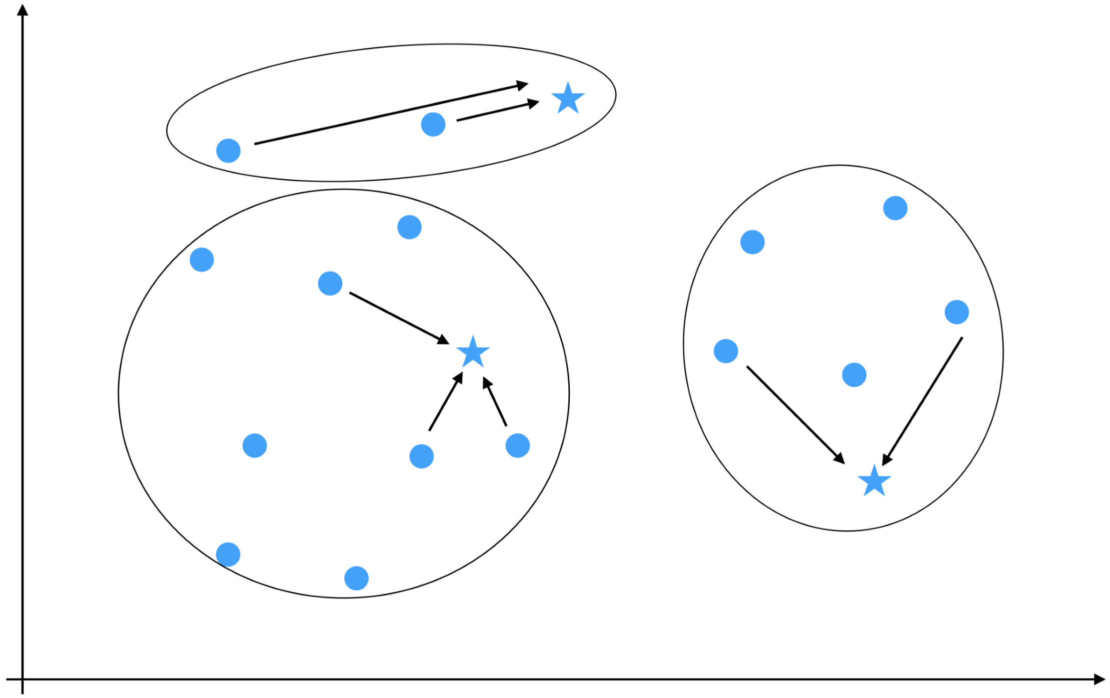

cluster
centroid



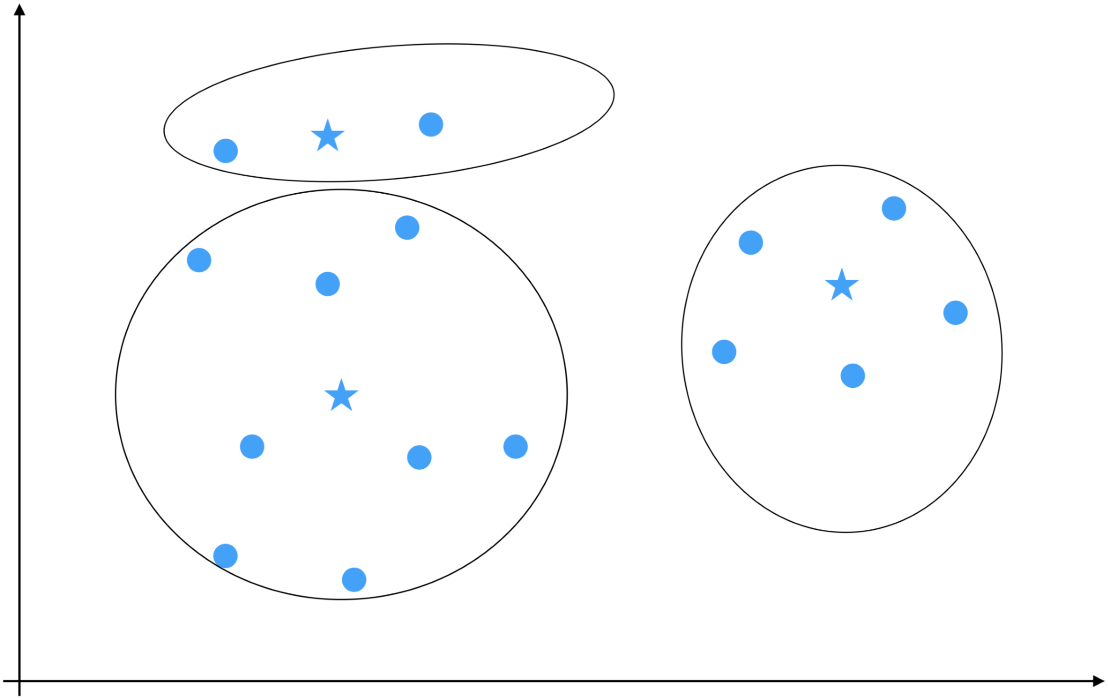
Initialize the centroids

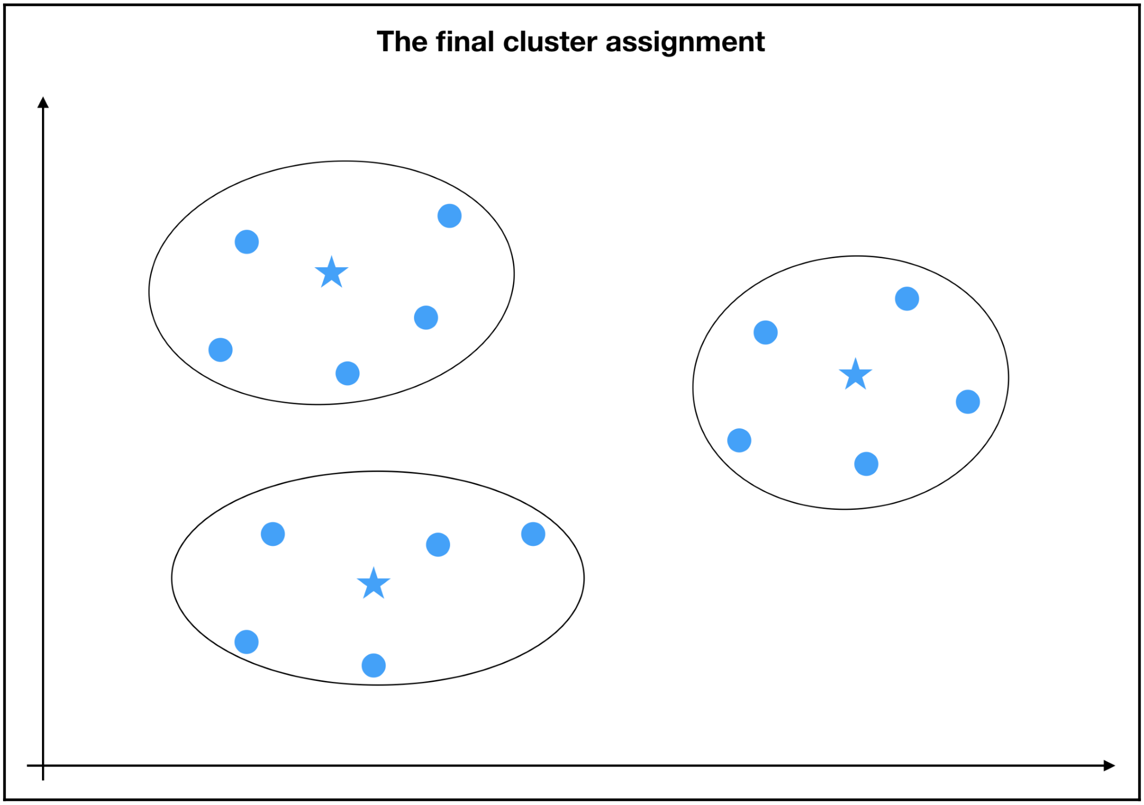


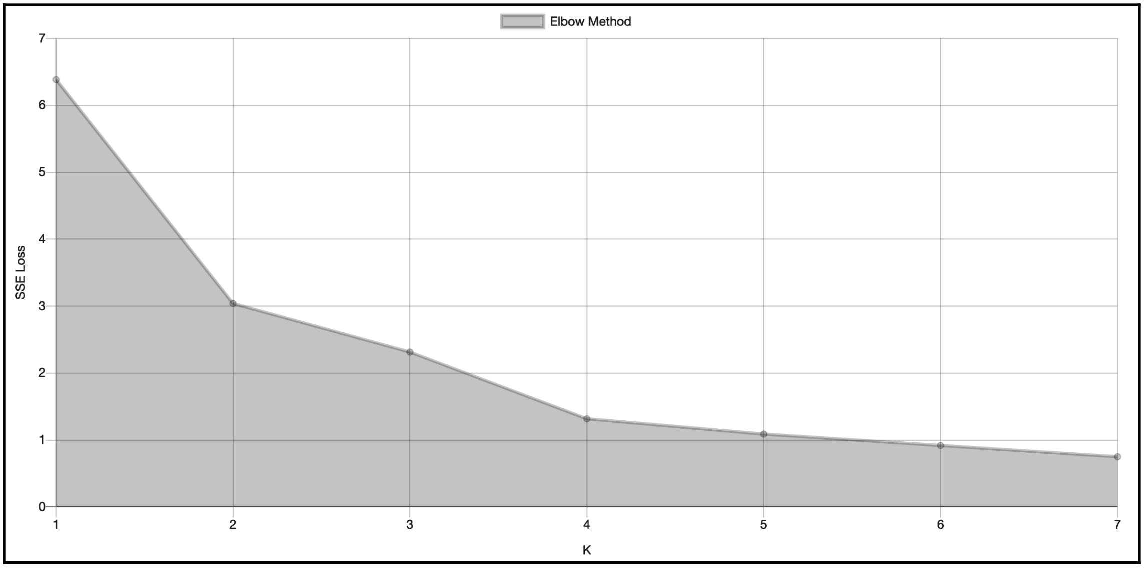
Assign points to the closest centroid



Update centroid positions




The final cluster assignment


Elbow Method
SSE Loss
0
1
2
3
4
5
6
7
K
1
2
3
4
5
6
7

$$\mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_K \end{pmatrix}$$

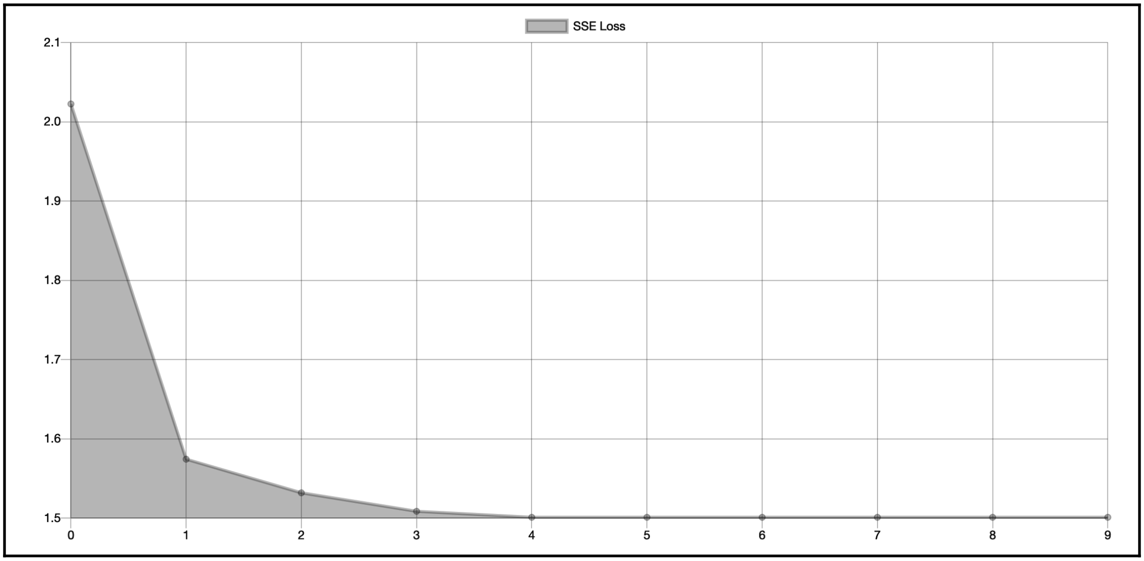
$$p(z_k = 1) = \pi_k, \text{ prior distribution}$$

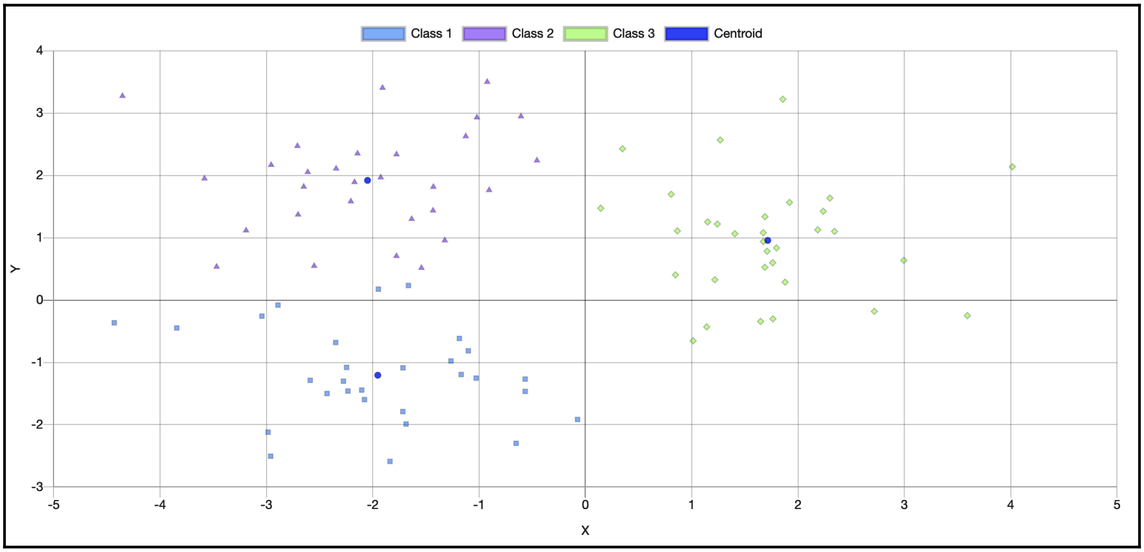
$$p(z_k = 1|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\mu_j, \Sigma_j)}, \text{posterior distribution}$$

$$p(z_k = 1|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\mu_j, \Sigma_j)}$$

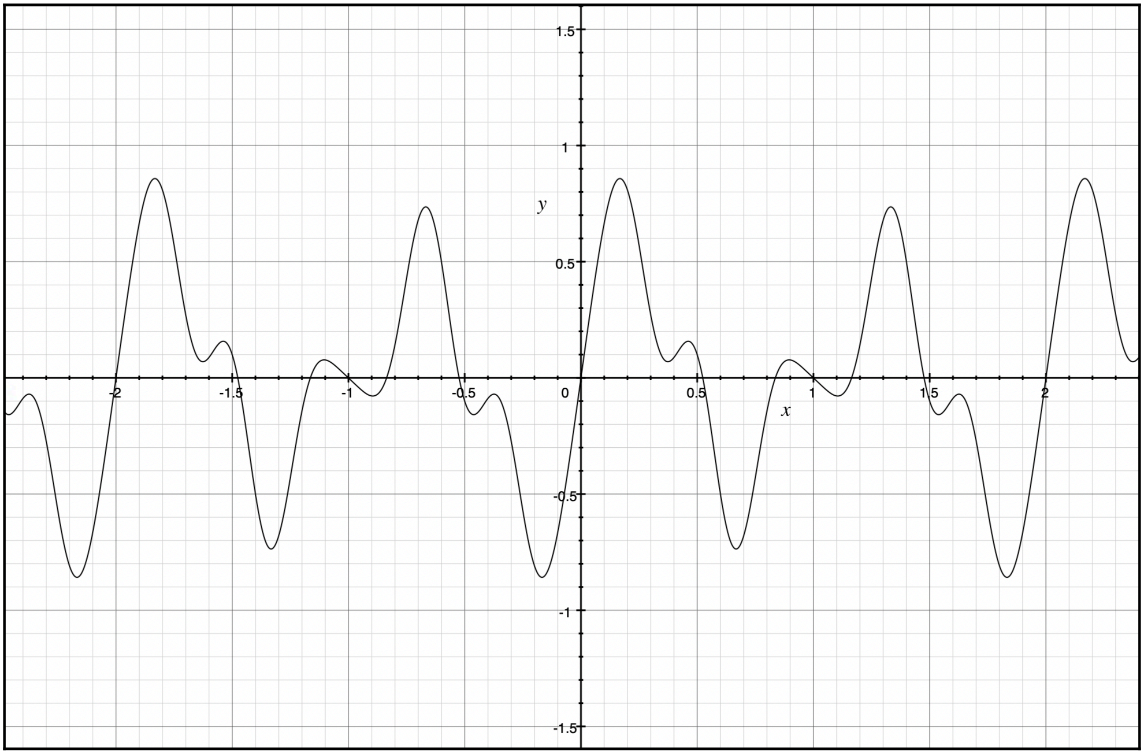
$$N_k = \sum_{n=1}^{N} p(z_k = 1|\mathbf{x}_n)$$

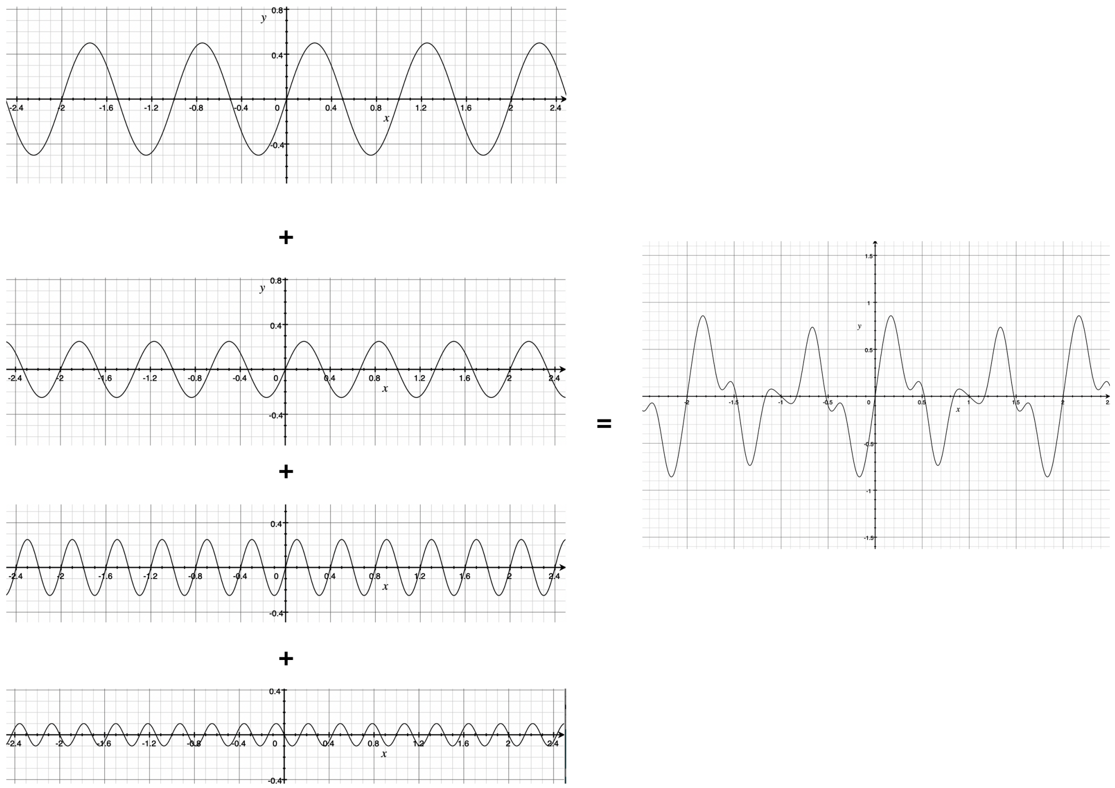
$$\pi_k = \frac{N_k}{N}$$


SSE Loss
2.1
2.0
1.9
1.8
1.7
1.6
1.5
0
1
2
3
4
5
6
7
8
9


Class 1
Class 2
Class 3
Centroid
4
3
2
1
0
-1
-2
-3
Y
-5
-4
-3
-2
-1
0
1
2
3
4
5
X

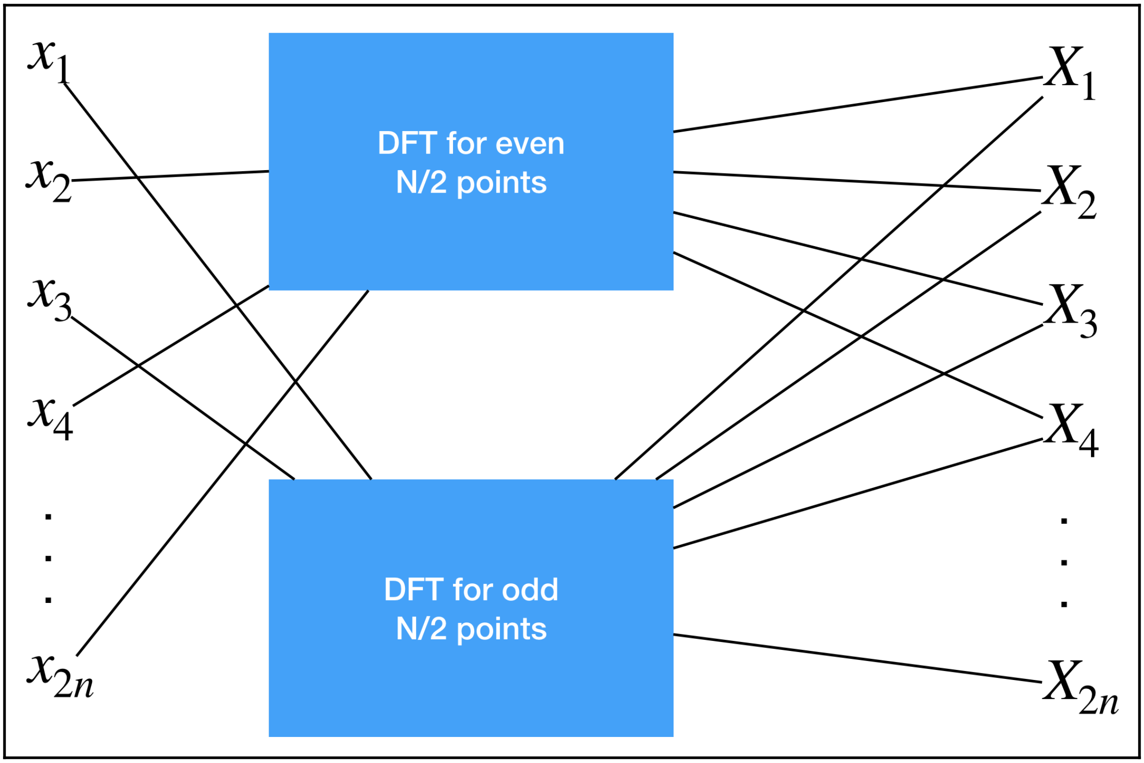
Chapter 7: Sequential Data Analysis

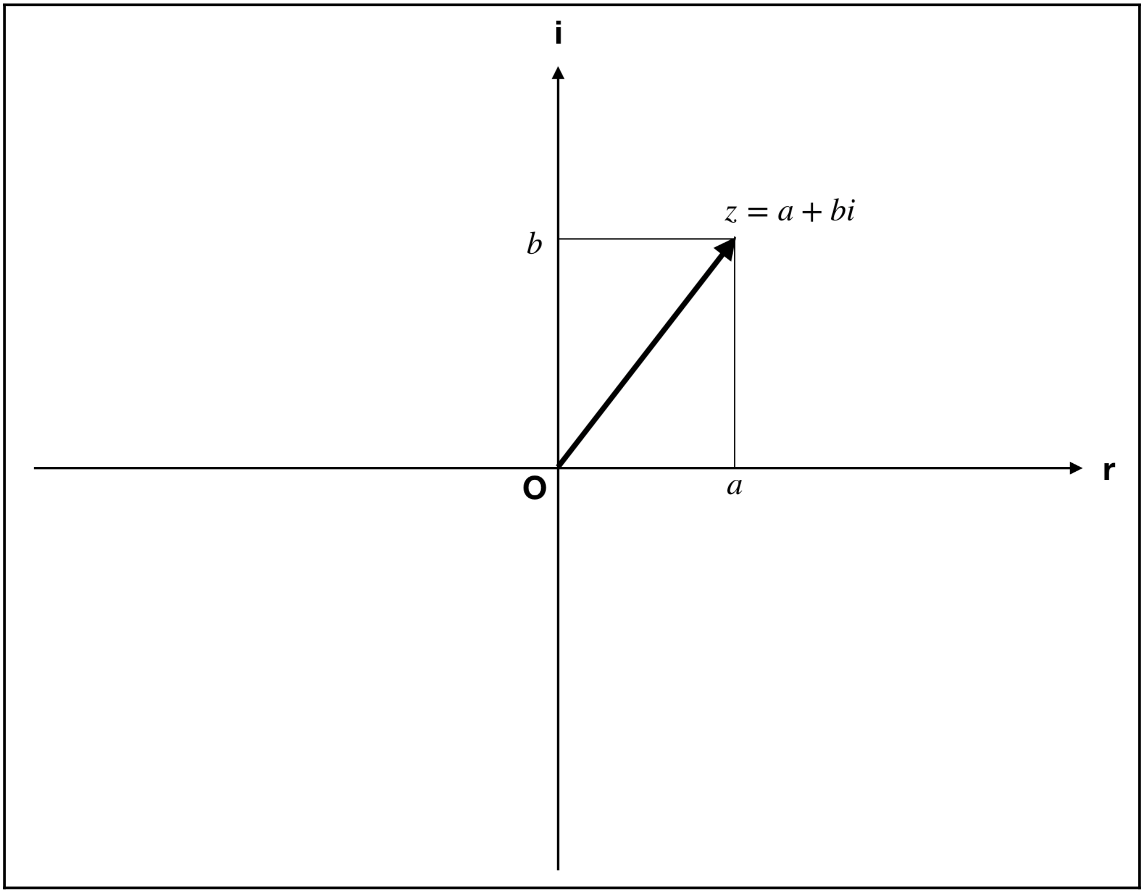




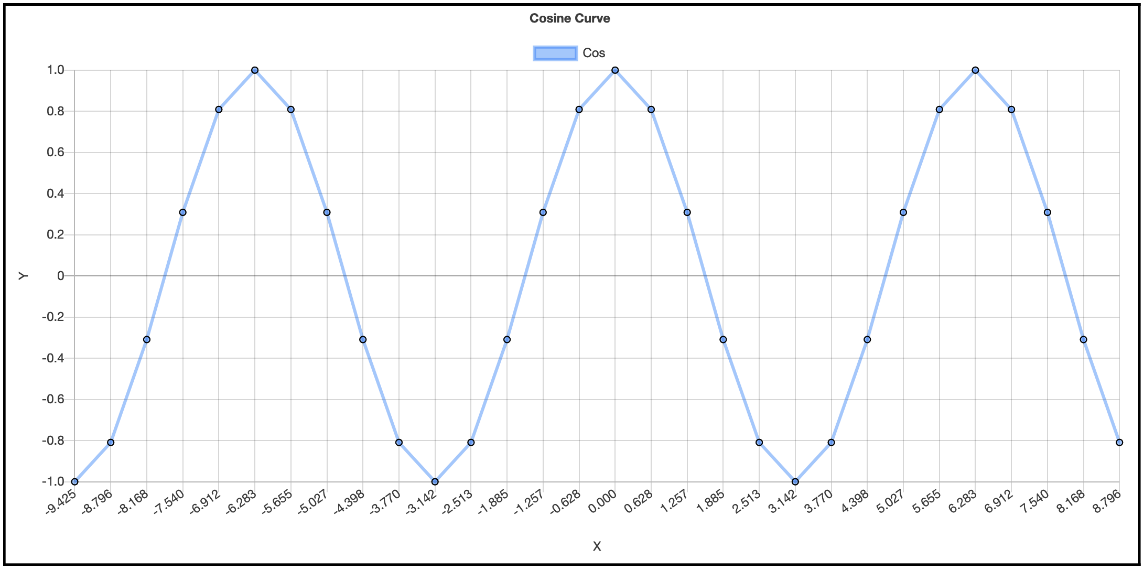
$$
\begin{aligned}
x_n &= \frac{1}{N}\sum_{k=0}^{N-1} F(\omega_k)e^{i2\pi kn/N} \\
&= \frac{1}{N}\sum_{k=0}^{N-1} F(\omega_k)(\cos(2\pi kn/N) + i\sin(2\pi kn/N)
\end{aligned}
$$

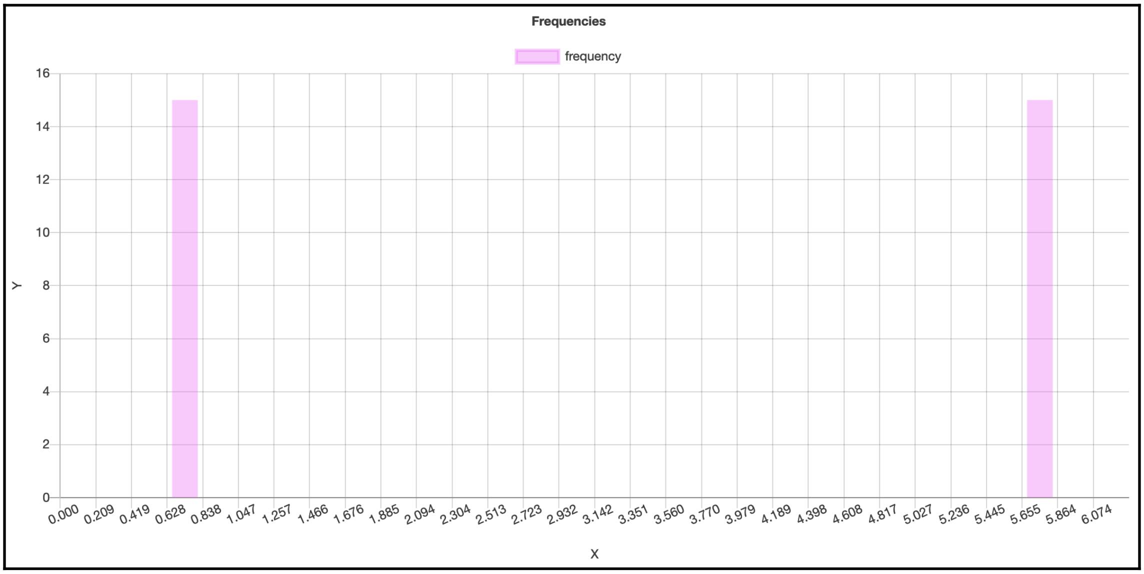
$$
\begin{aligned}
F(\omega_0) &= 2e^{-i2\pi\cdot 0\cdot 0/4} + (2+i)e^{-i2\pi\cdot 0\cdot 1/4} + (-i)e^{-i2\pi\cdot 0\cdot 2/4} + (-1+3i)e^{-i2\pi\cdot 0\cdot 3/4} \\
&= 3+3i \\
F(\omega_1) &= 2e^{-i2\pi\cdot 1\cdot 0/4} + (2+i)e^{-i2\pi\cdot 1\cdot 1/4} + (-i)e^{-i2\pi\cdot 1\cdot 2/4} + (-1+3i)e^{-i2\pi\cdot 3\cdot 1/4} \\
&= -2i \\
F(\omega_2) &= 2e^{-i2\pi\cdot 2\cdot 0/4} + (2+i)e^{-i2\pi\cdot 2\cdot 1/4} + (-i)e^{-i2\pi\cdot 2\cdot 2/4} + (-1+3i)e^{-i2\pi\cdot 2\cdot 3/4} \\
&= 1-5i \\
F(\omega_3) &= 2e^{-i2\pi\cdot 3\cdot 0/4} + (2+i)e^{-i2\pi\cdot 3\cdot 1/4} + (-i)e^{-i2\pi\cdot 3\cdot 2/4} + (-1+3i)e^{-i2\pi\cdot 3\cdot 3/4} \\
&= 4+4i
\end{aligned}
$$



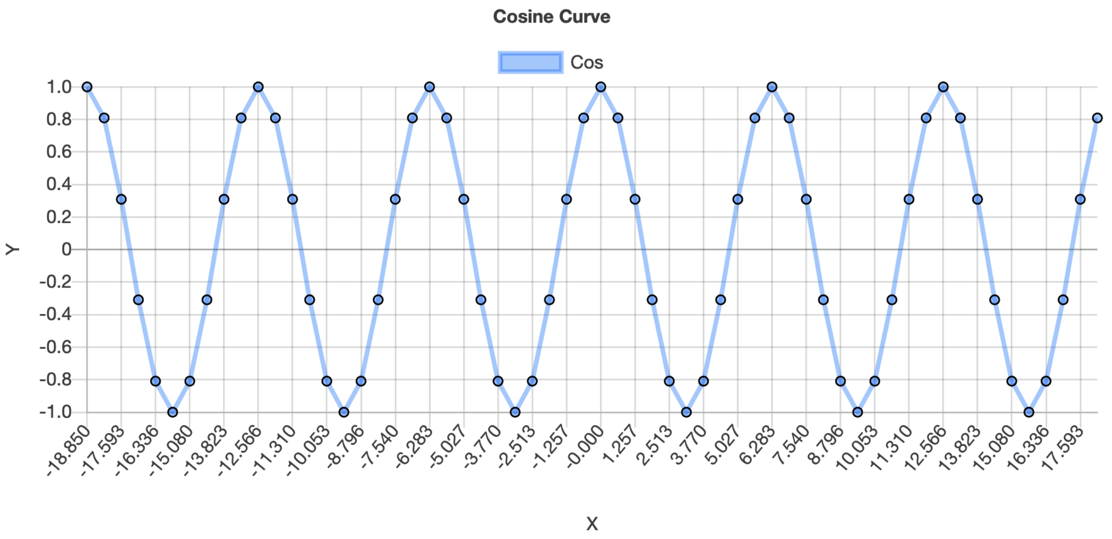


$$\begin{aligned} z_1 &= a + bi \\ z_2 &= c + di \\ z_1 + z_2 &= (a + c) + (b + d)i \\ ||z_1|| &= \sqrt{z_1 \overline{z_1}} = \sqrt{(a + bi)(a - bi)} = \sqrt{a^2 + b^2} \end{aligned}$$

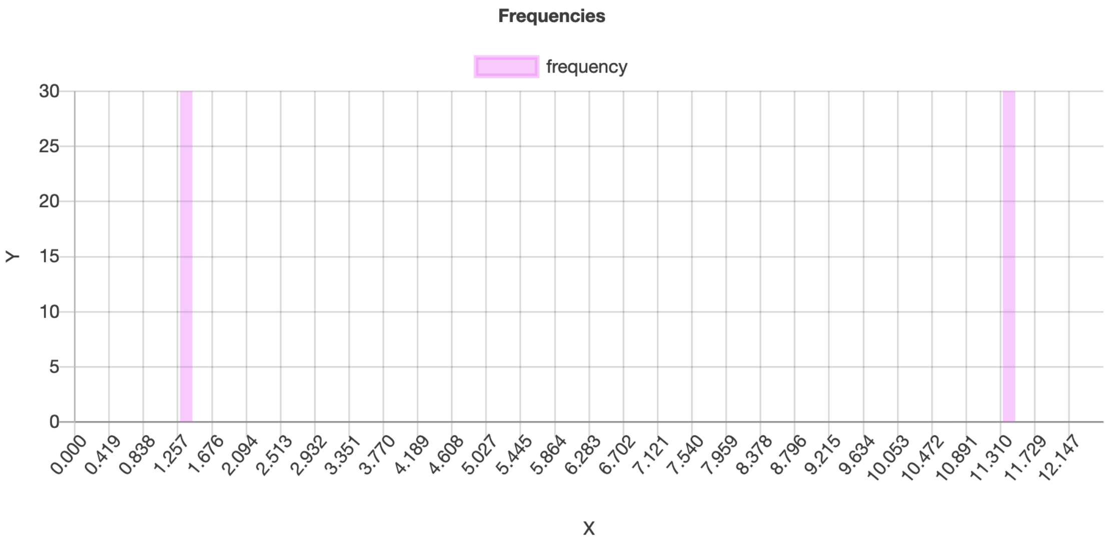

Cosine Curve
Cos
1.0
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1.0
Y
-9.425
-8.796
-8.168
-7.540
-6.912
-6.283
-5.655
-5.027
-4.398
-3.770
-3.142
-2.513
-1.885
-1.257
-0.628
0.000
0.628
1.257
1.885
2.513
3.142
3.770
4.398
5.027
5.655
6.283
6.912
7.540
8.168
8.796
X


Frequencies
frequency
16
14
12
10
8
6
4
2
0
Y
0.000
0.209
0.419
0.628
0.838
1.047
1.257
1.466
1.676
1.885
2.094
2.304
2.513
2.723
2.932
3.142
3.351
3.560
3.770
3.979
4.189
4.398
4.608
4.817
5.027
5.236
5.445
5.655
5.864
6.074
X

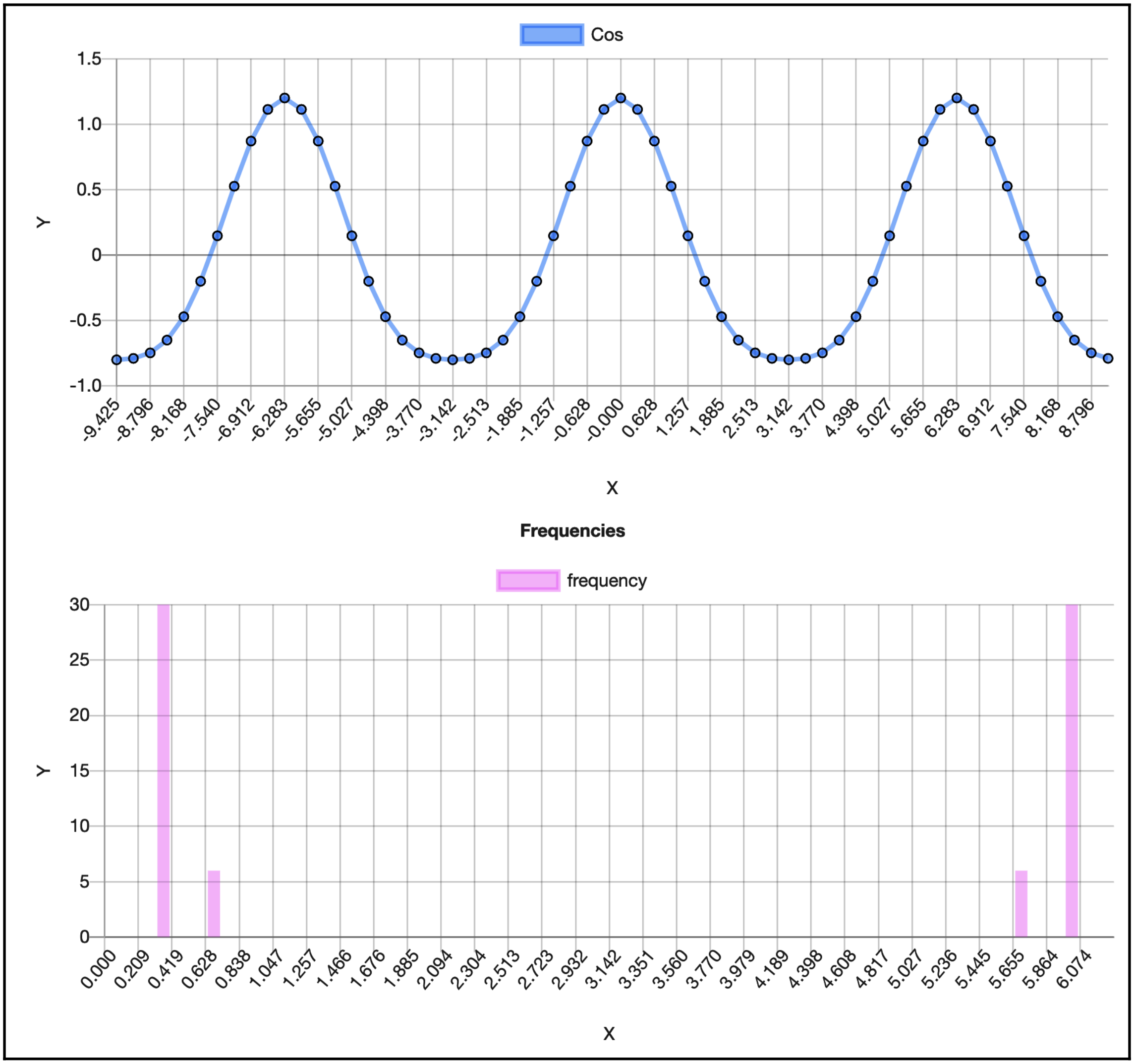
$$\boxed{\begin{pmatrix} \omega^{0\cdot 0} & \omega^{0\cdot 1} & \omega^{0\cdot 2} & \omega^{0\cdot 3} \\ \omega^{1\cdot 0} & \omega^{1\cdot 1} & \omega^{1\cdot 2} & \omega^{1\cdot 3} \\ \omega^{2\cdot 0} & \omega^{2\cdot 1} & \omega^{2\cdot 2} & \omega^{2\cdot 3} \\ \omega^{3\cdot 0} & \omega^{3\cdot 1} & \omega^{3\cdot 2} & \omega^{3\cdot 3} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix}}$$

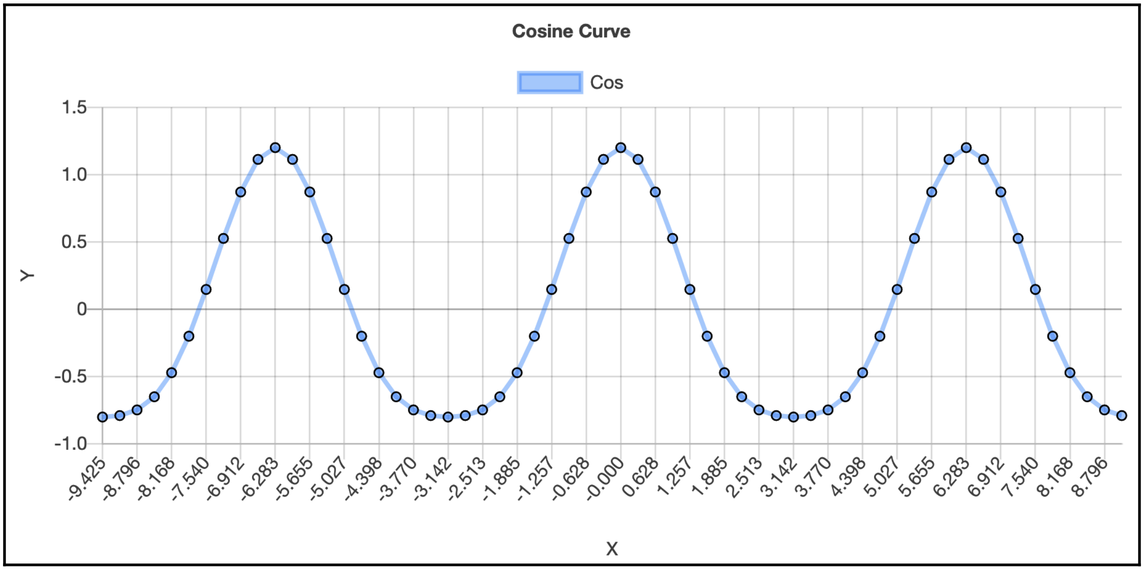


Cosine Curve

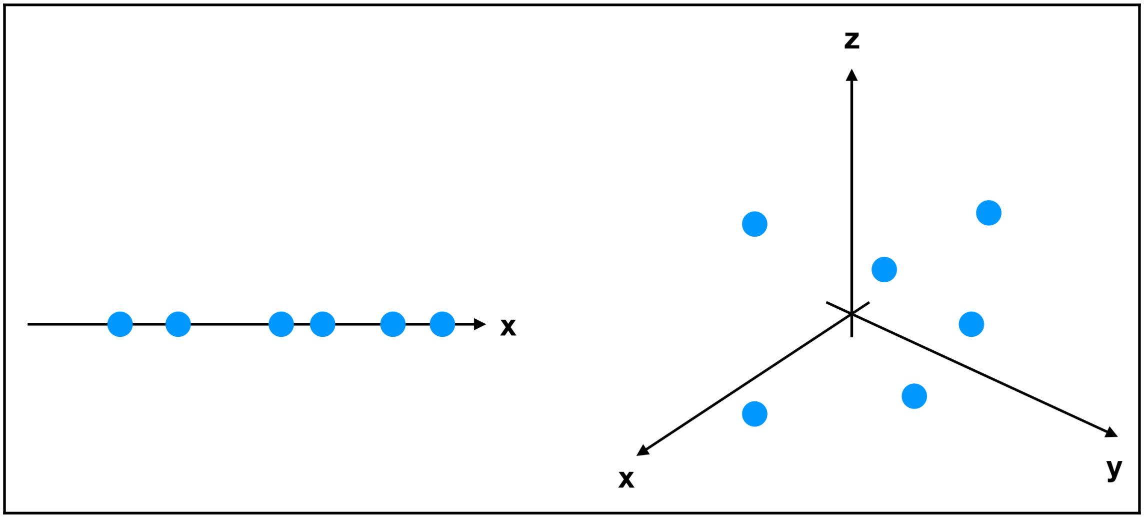


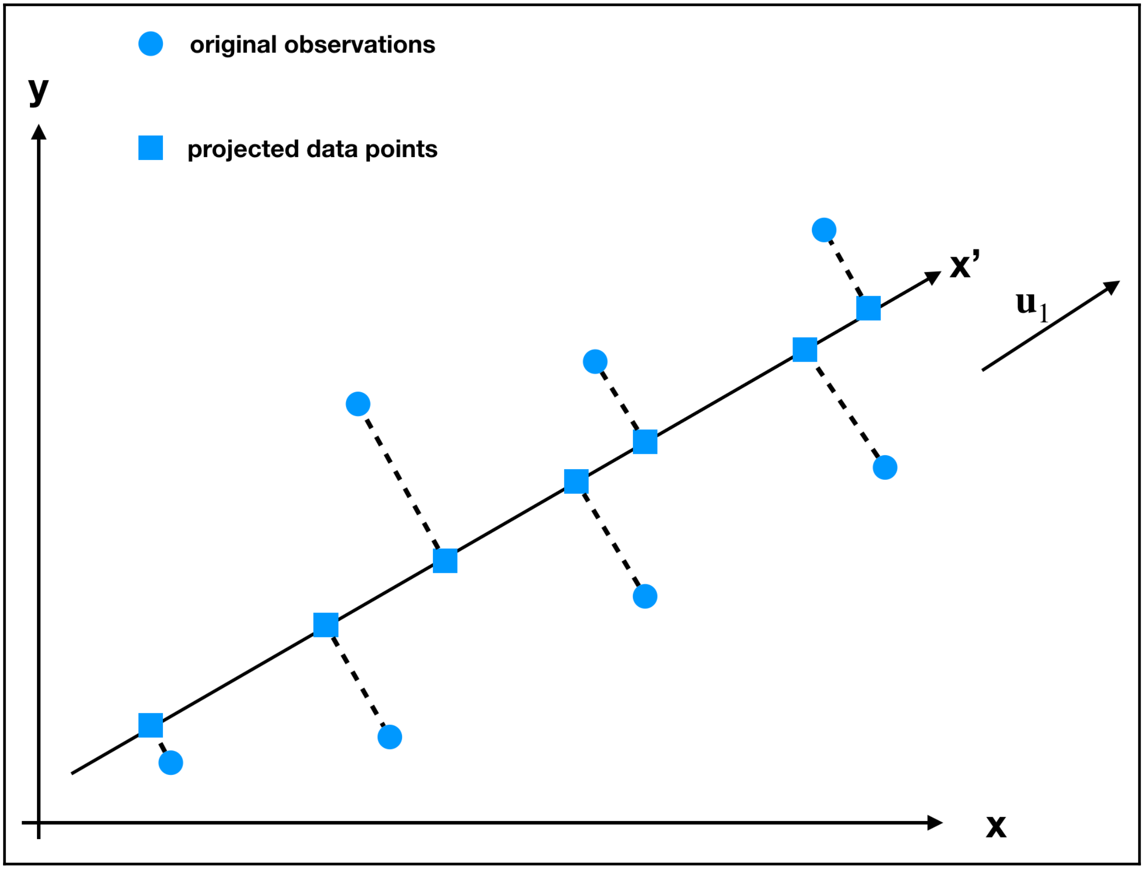
Frequencies

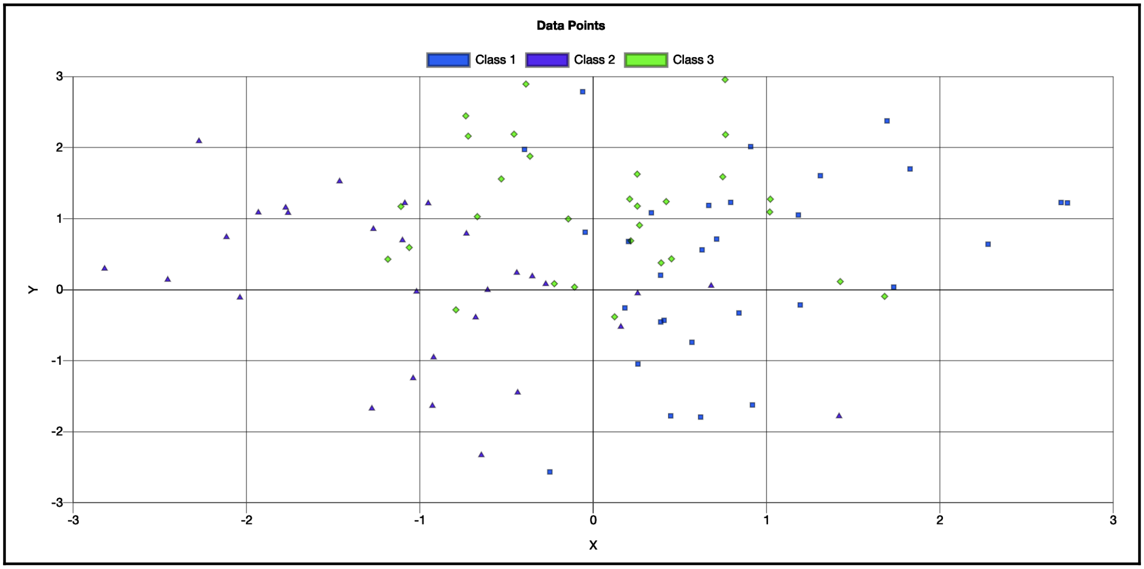

Cos
Y
1.5
1.0
0.5
0
-0.5
-1.0
-9.425
-8.796
-8.168
-7.540
-6.912
-6.283
-5.655
-5.027
-4.398
-3.770
-3.142
-2.513
-1.885
-1.257
-0.628
-0.000
0.628
1.257
1.885
2.513
3.142
3.770
4.398
5.027
5.655
6.283
6.912
7.540
8.168
8.796
X
Frequencies
frequency
Y
30
25
20
15
10
5
0
0.000
0.209
0.419
0.628
0.838
1.047
1.257
1.466
1.676
1.885
2.094
2.304
2.513
2.723
2.932
3.142
3.351
3.560
3.770
3.979
4.189
4.398
4.608
4.817
5.027
5.236
5.445
5.655
5.864
6.074
X


Cosine Curve
Cos
1.5
1.0
0.5
0
-0.5
-1.0
Y
-9.425
-8.796
-8.168
-7.540
-6.912
-6.283
-5.655
-5.027
-4.398
-3.770
-3.142
-2.513
-1.885
-1.257
-0.628
-0.000
0.628
1.257
1.885
2.513
3.142
3.770
4.398
5.027
5.655
6.283
6.912
7.540
8.168
8.796
X

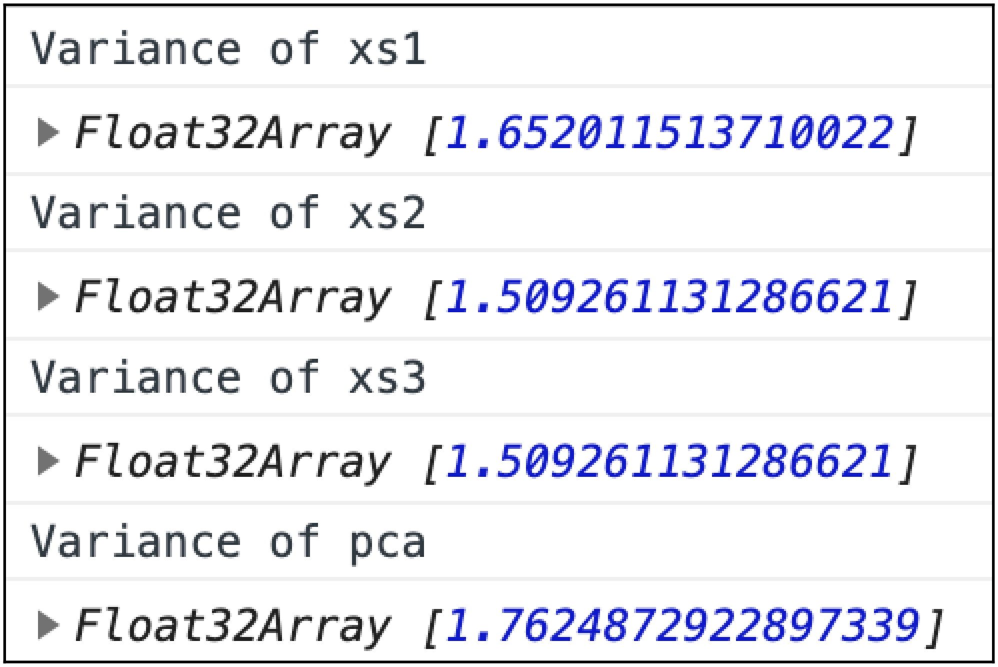
Chapter 8: Dimensionality Reduction



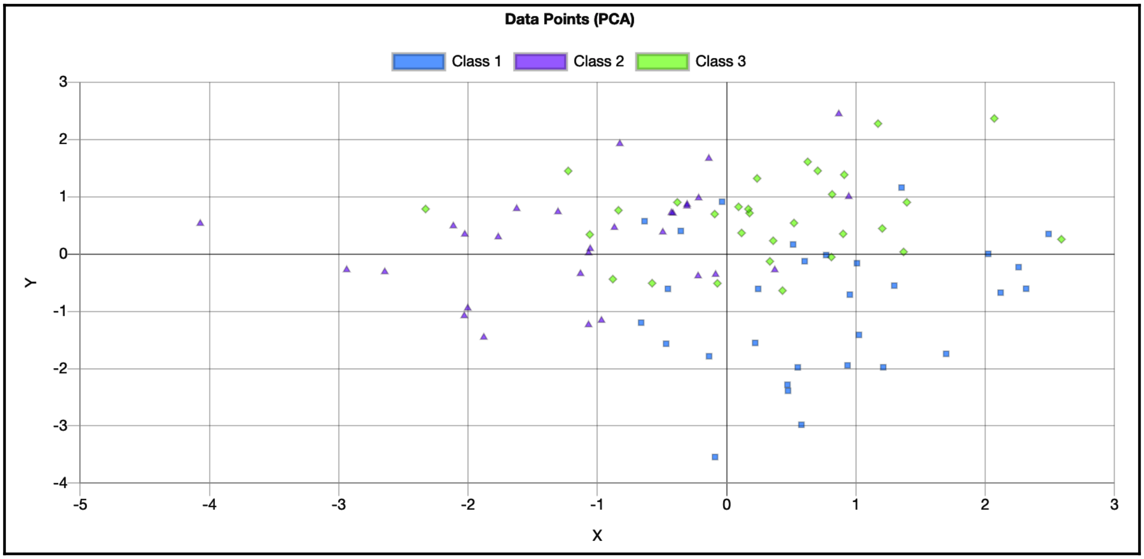

original observations
projected data points
y
x'
u₁
x

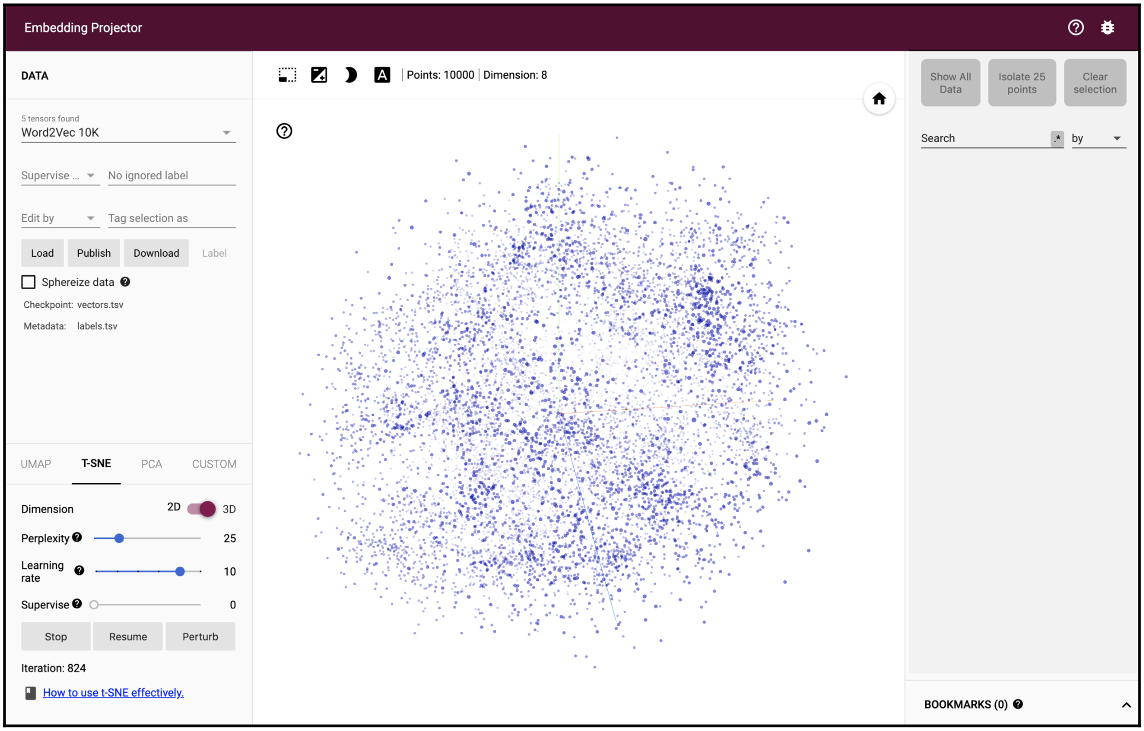

Data Points
Class 1
Class 2
Class 3
Y
X
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

$$\begin{tikzcd}
\begin{pmatrix} x_1^1 & x_2^1 & \dots & x_d^1 \\ x_1^2 & x_2^2 & \dots & x_d^2 \\ x_1^3 & x_2^3 & \dots & x_d^3 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^n & x_2^n & \dots & x_d^n \end{pmatrix} \arrow[r, "\textbf{axis=1}"] \arrow[d, "\textbf{axis=0}"'] & \begin{pmatrix} \overline{x^1} \\ \overline{x^2} \\ \overline{x^3} \\ \vdots \\ \overline{x^n} \end{pmatrix} \\
(\overline{x_1} \quad \overline{x_2} \quad \dots \quad \overline{x_d}) &
\end{tikzcd}$$



```
Variance of xs1
▶ Float32Array [1.652011513710022]
Variance of xs2
▶ Float32Array [1.509261131286621]
Variance of xs3
▶ Float32Array [1.509261131286621]
Variance of pca
▶ Float32Array [1.7624872922897339]
```





5 tensors found

Word2Vec 10K

☐ Sphereize data

Checkpoint: vectors.tsv

Metadata: labels.tsv

Points: 10000 | Dimension: 8

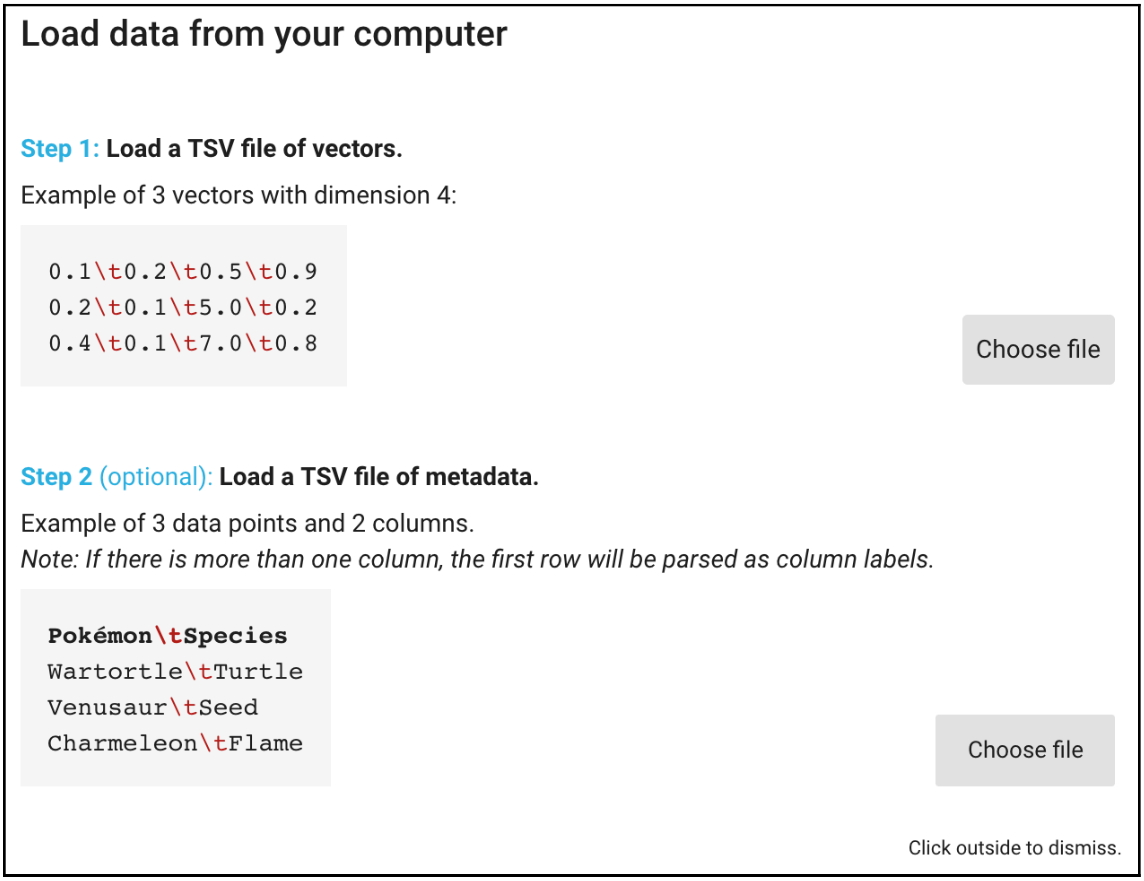
Dimension 2D 3D

Perplexity 25

Learning rate 10

Supervise 0

Iteration: 824



Load data from your computer

Step 1: Load a TSV file of vectors.

Example of 3 vectors with dimension 4:

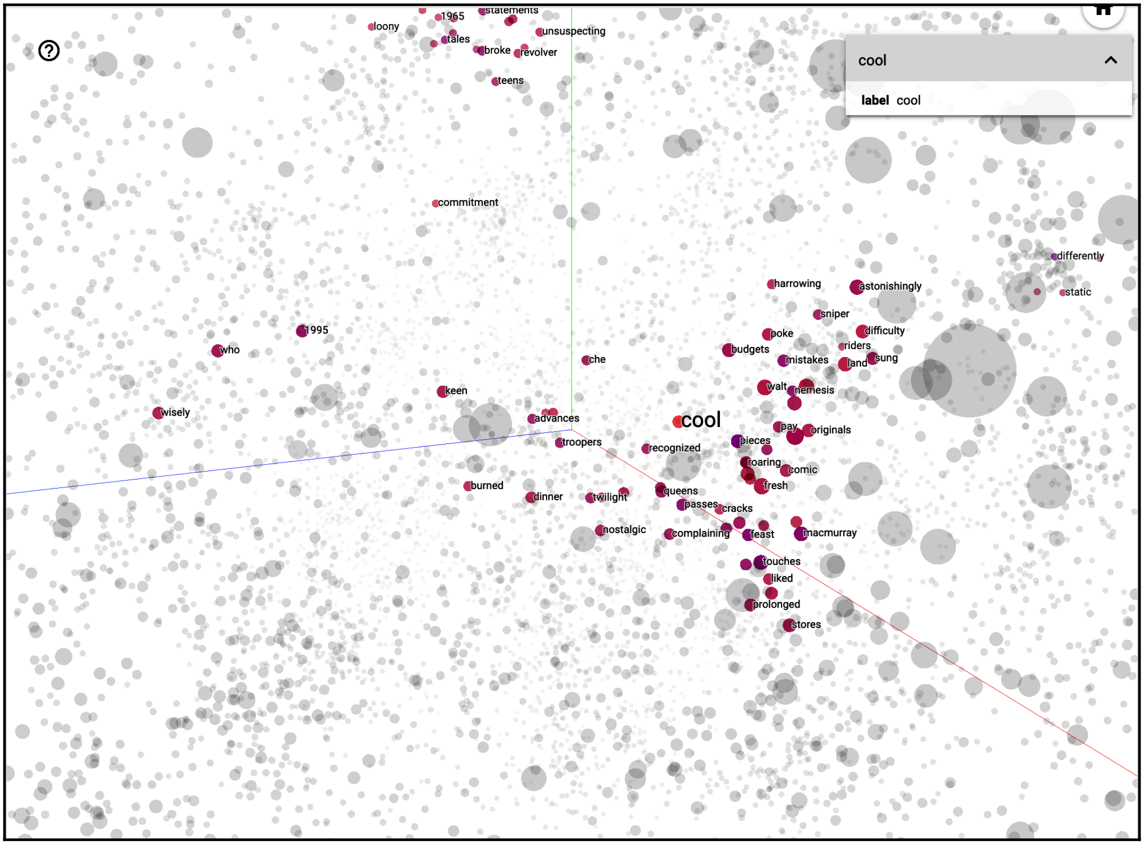
```
0.1\t0.2\t0.5\t0.9
0.2\t0.1\t5.0\t0.2
0.4\t0.1\t7.0\t0.8
```

Step 2 (optional): Load a TSV file of metadata.

Example of 3 data points and 2 columns.
Note: If there is more than one column, the first row will be parsed as column labels.

```
Pokémon\tSpecies
Wartortle\tTurtle
Venusaur\tSeed
Charmeleon\tFlame
```

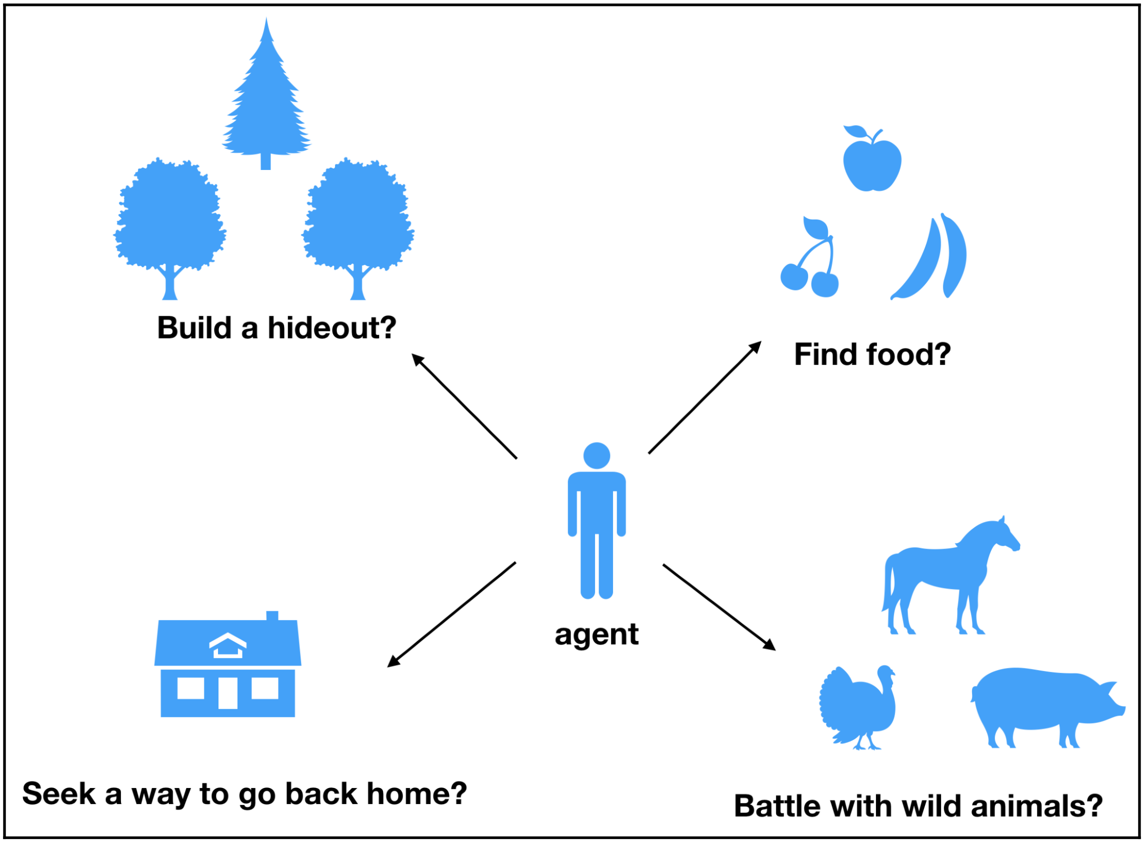
Click outside to dismiss.

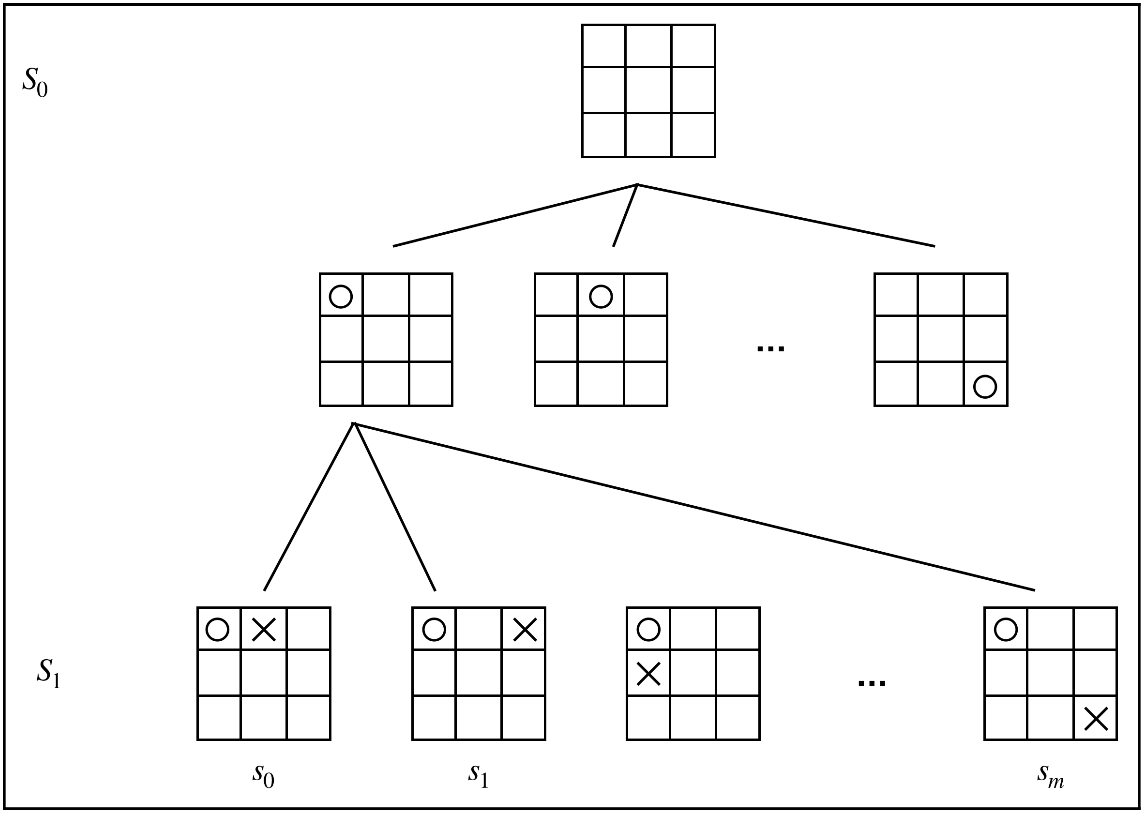


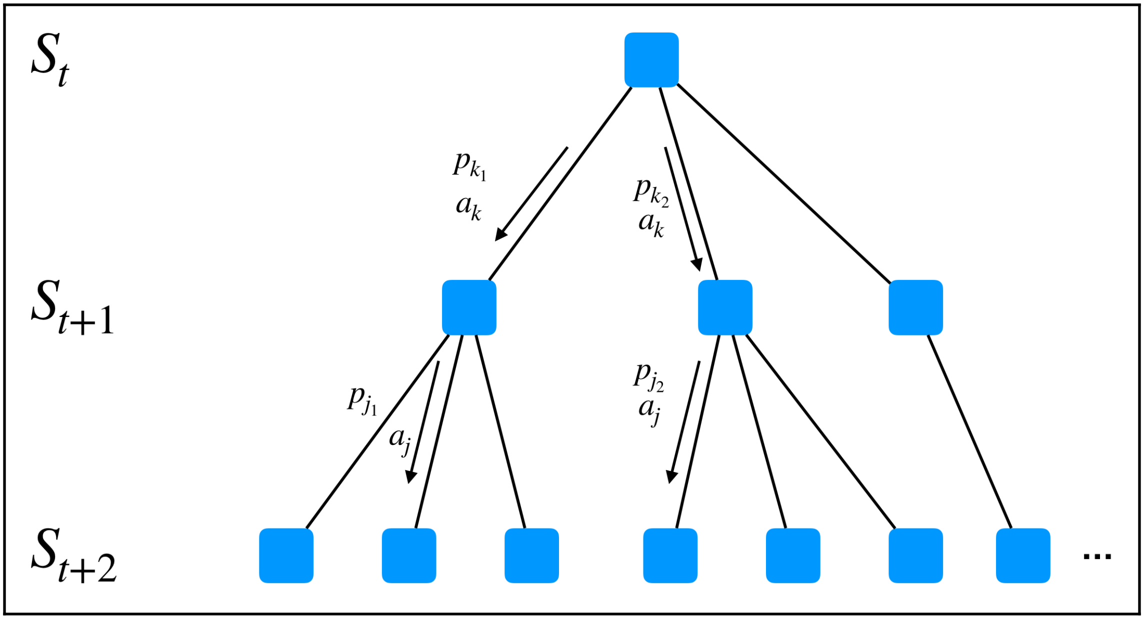
cool

label cool

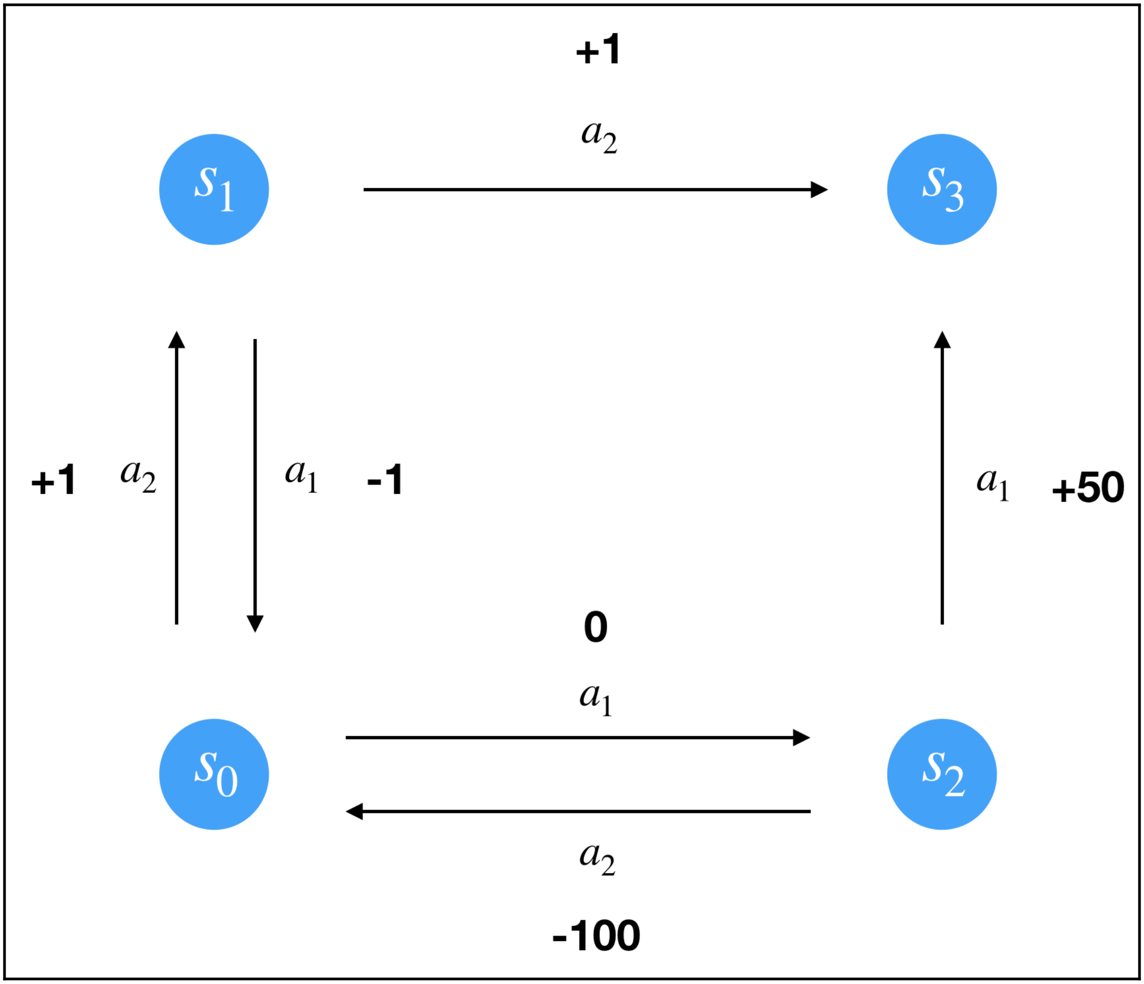
Chapter 9: Solving the Markov Decision Process

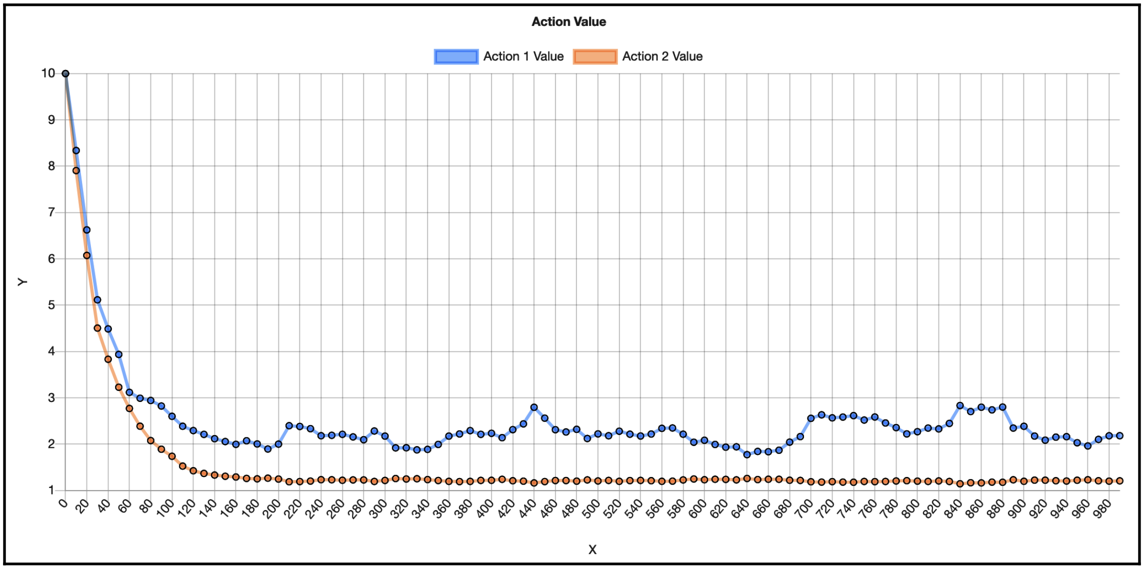


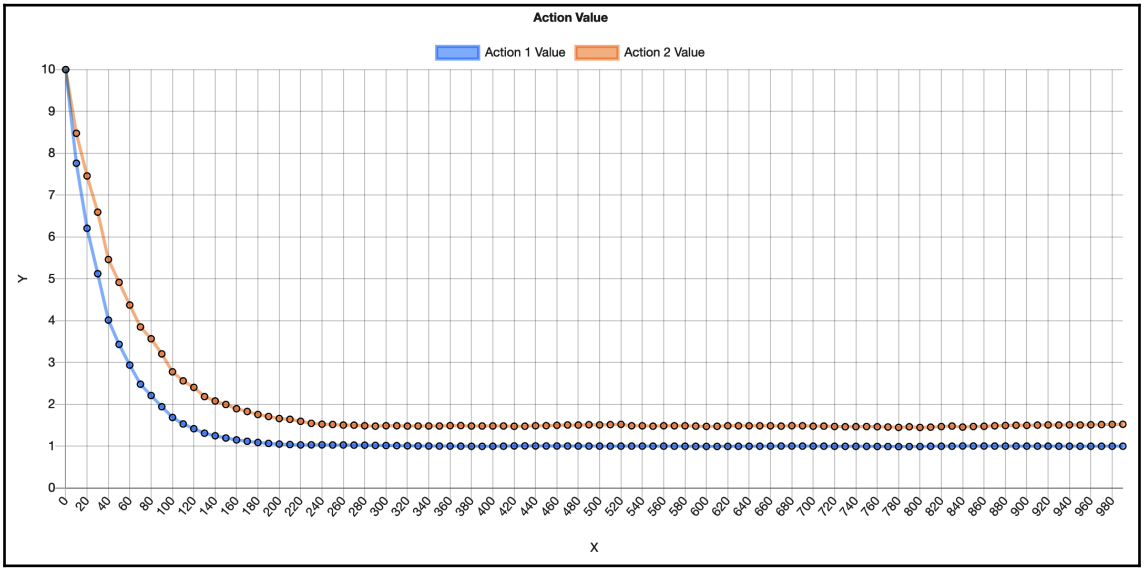

S_0
...
S_1
...
s_0
s_1
s_m

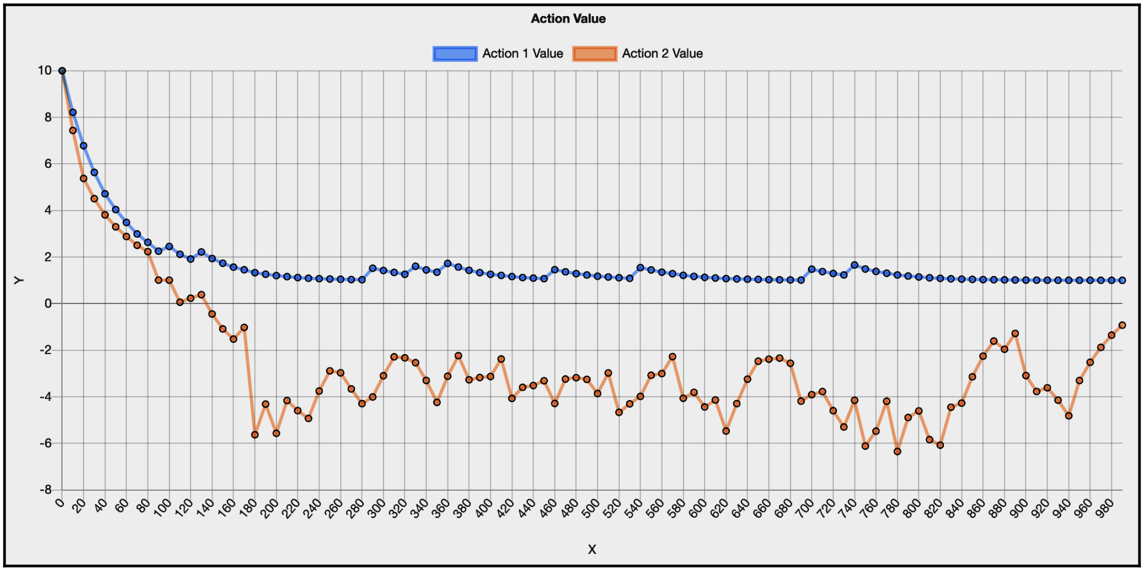


$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}^\pi[G_{t+1}|S_t = s, A_t = a] \\ &= \mathbb{E}^\pi[R_{t+1}|S_t = s, A_t = a] + \mathbb{E}^\pi[\gamma R_{t+2} + \gamma^2 Rt + 3 + \dots|S_t = s, A_t = a] \\ &= \mathbb{E}^\pi[R_{t+1}|S_t = s, A_t = a] + \gamma\mathbb{E}^\pi[R_{t+2} + \gamma Rt + 3 + \dots|S_t = s, A_t = a] \end{aligned}$$

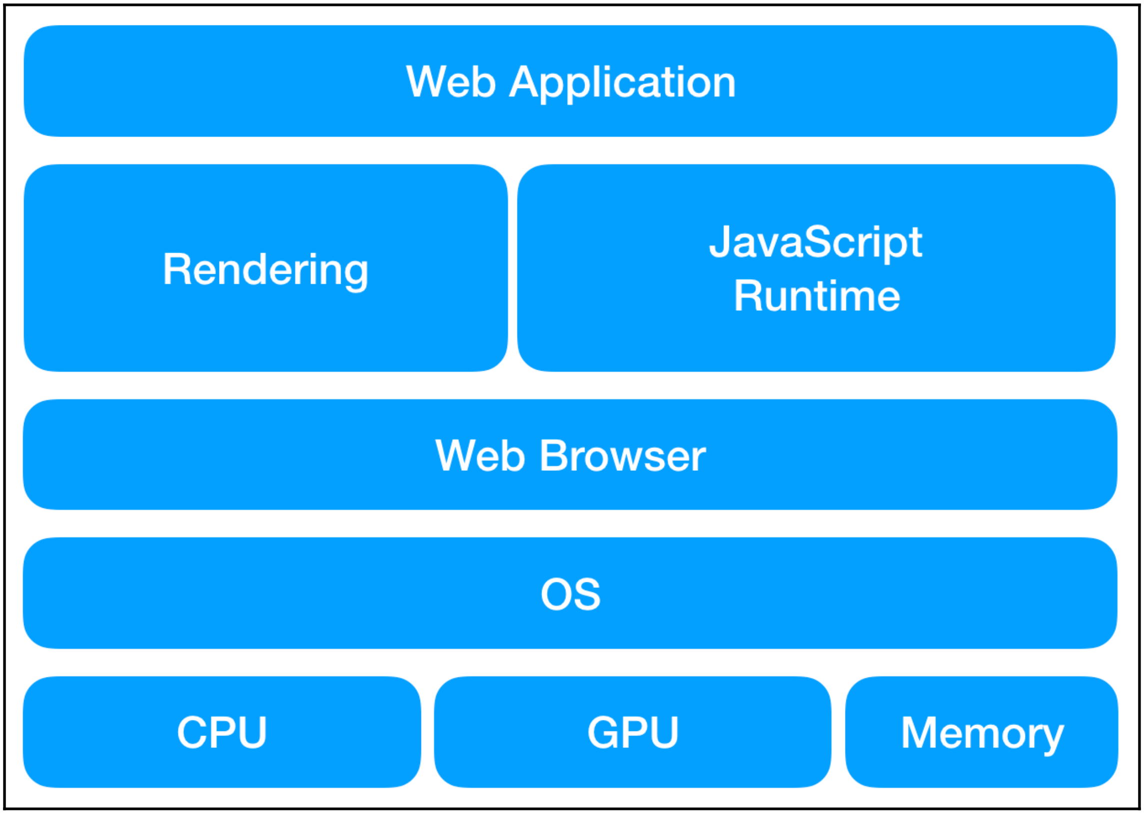

+1
a_2
s_1
s_3
+1
a_2
a_1
-1
a_1
+50
0
a_1
s_0
s_2
a_2
-100

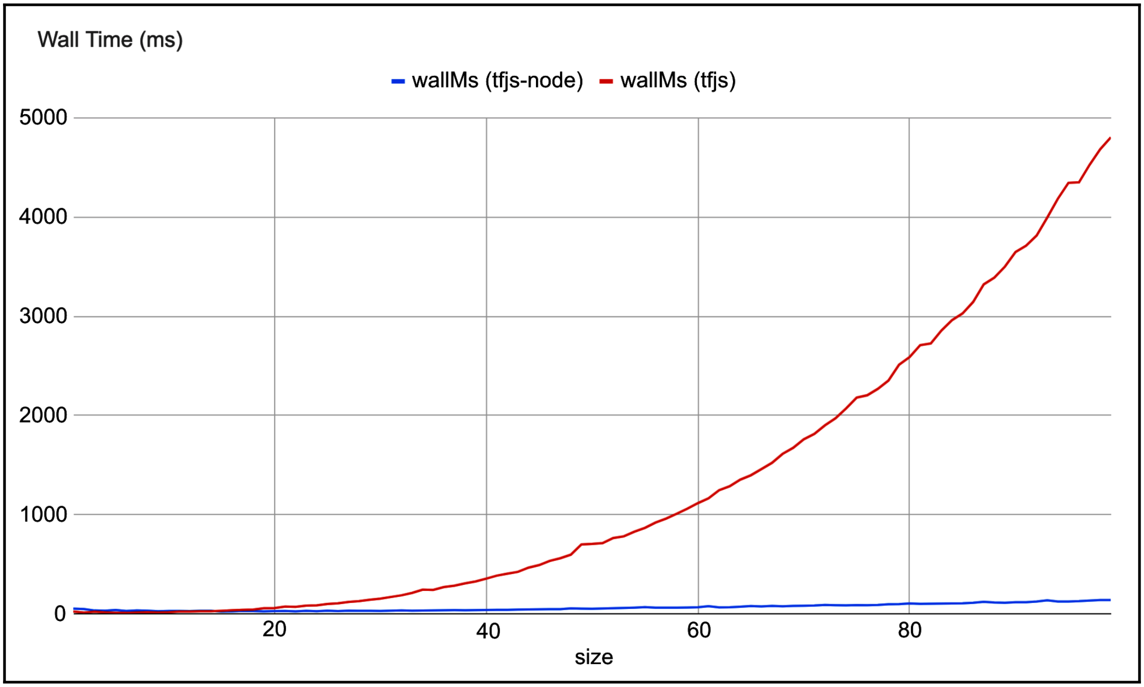

Action Value
Action 1 Value
Action 2 Value
Y
10
9
8
7
6
5
4
3
2
1
0
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
360
380
400
420
440
460
480
500
520
540
560
580
600
620
640
660
680
700
720
740
760
780
800
820
840
860
880
900
920
940
960
980
X

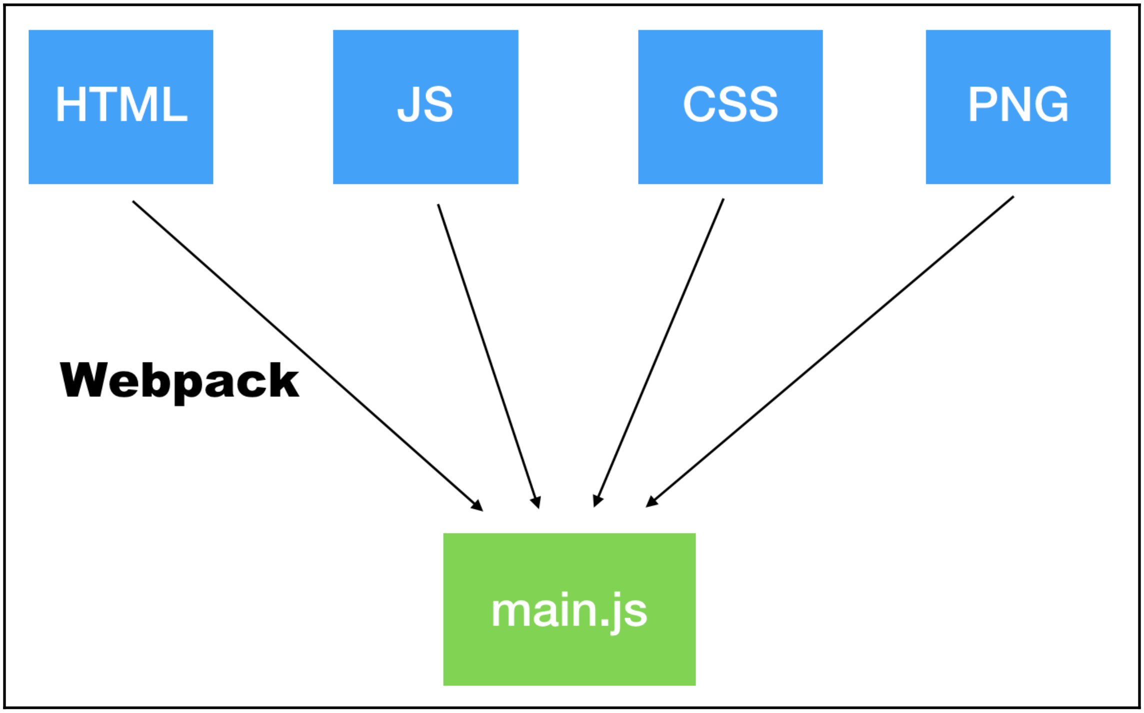

Action Value
Action 1 Value
Action 2 Value
Y
10
9
8
7
6
5
4
3
2
1
0
0
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
360
380
400
420
440
460
480
500
520
540
560
580
600
620
640
660
680
700
720
740
760
780
800
820
840
860
880
900
920
940
960
980
X

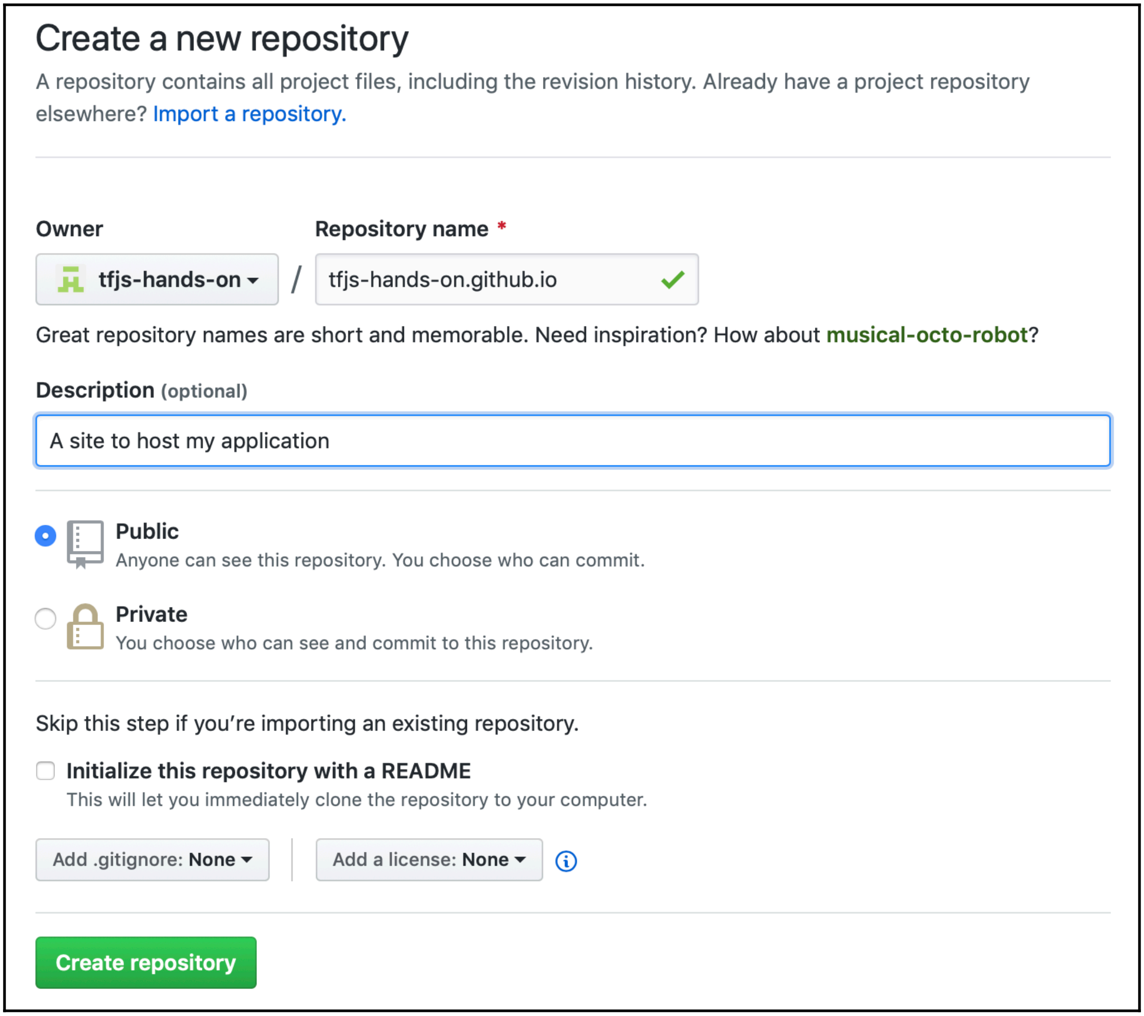

Action Value
Action 1 Value
Action 2 Value
Y
X
10
8
6
4
2
0
-2
-4
-6
-8
0
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
360
380
400
420
440
460
480
500
520
540
560
580
600
620
640
660
680
700
720
740
760
780
800
820
840
860
880
900
920
940
960
980

Chapter 10: Deploying Machine Learning Applications




Wall Time (ms)
wallMs (tfjs-node)
wallMs (tfjs)
5000
4000
3000
2000
1000
0
20
40
60
80
size


HTML
JS
CSS
PNG
Webpack
main.js



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner: tfjs-hands-on / Repository name *: tfjs-hands-on.github.io ✓

Great repository names are short and memorable. Need inspiration? How about musical-octo-robot?

Description (optional): A site to host my application

● Public
Anyone can see this repository. You choose who can commit.

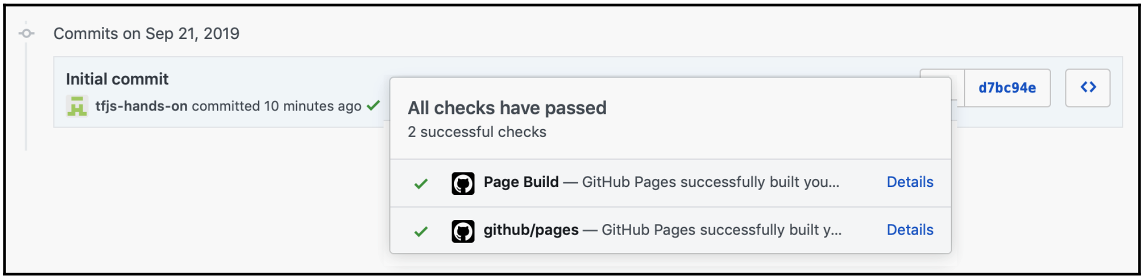
○ Private
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

□ Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: None

Add a license: None



Commits on Sep 21, 2019

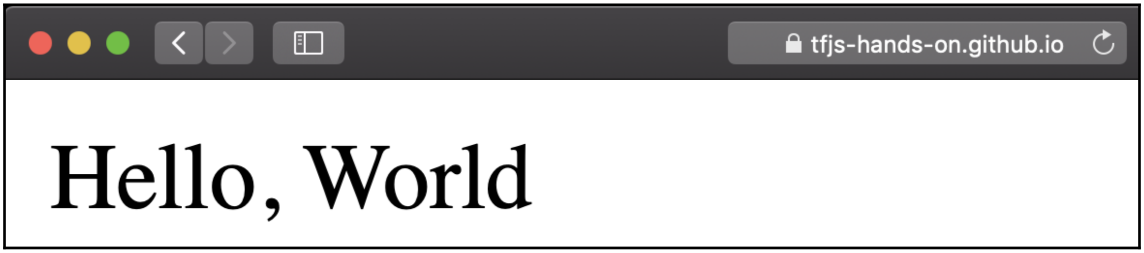
Initial commit
tfjs-hands-on committed 10 minutes ago ✓

d7bc94e

All checks have passed
2 successful checks

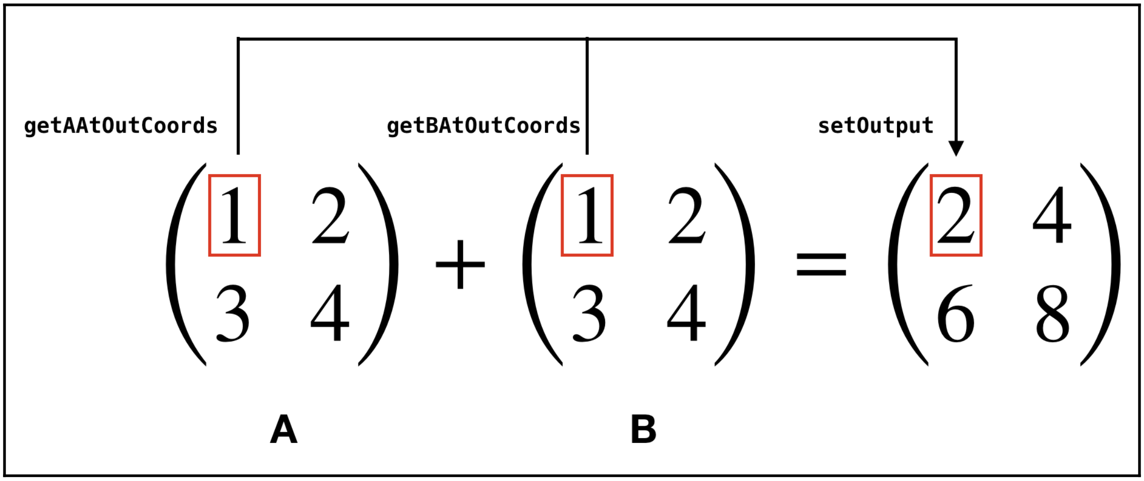
✓ Page Build — GitHub Pages successfully built you... Details

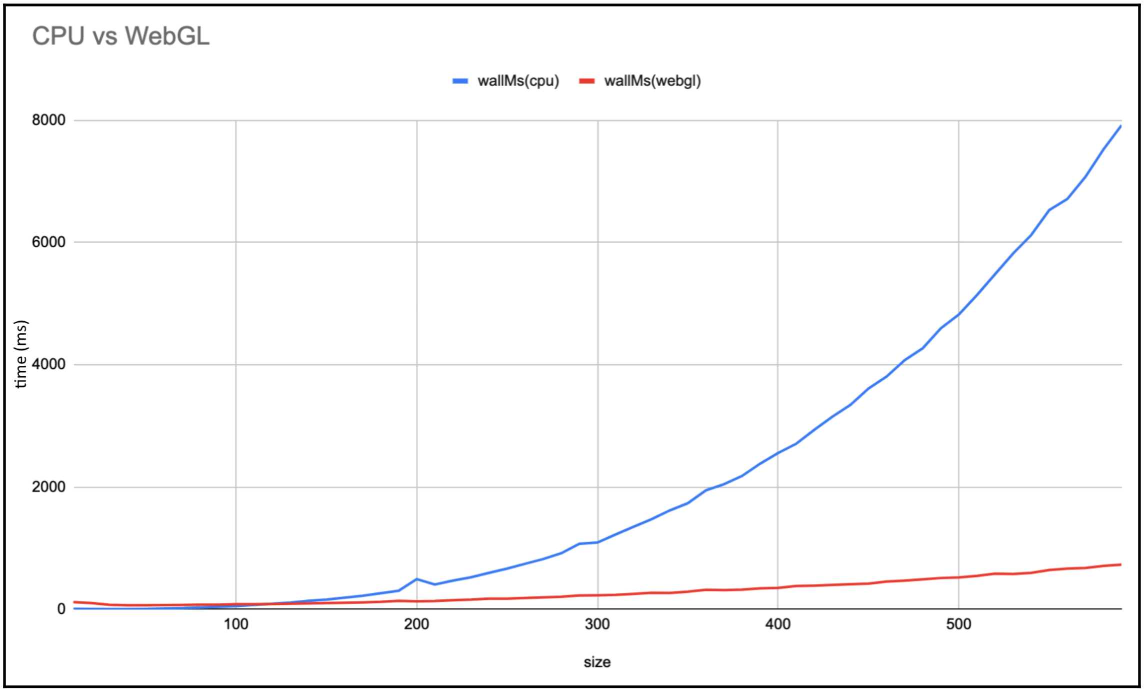
✓ github/pages — GitHub Pages successfully built y... Details

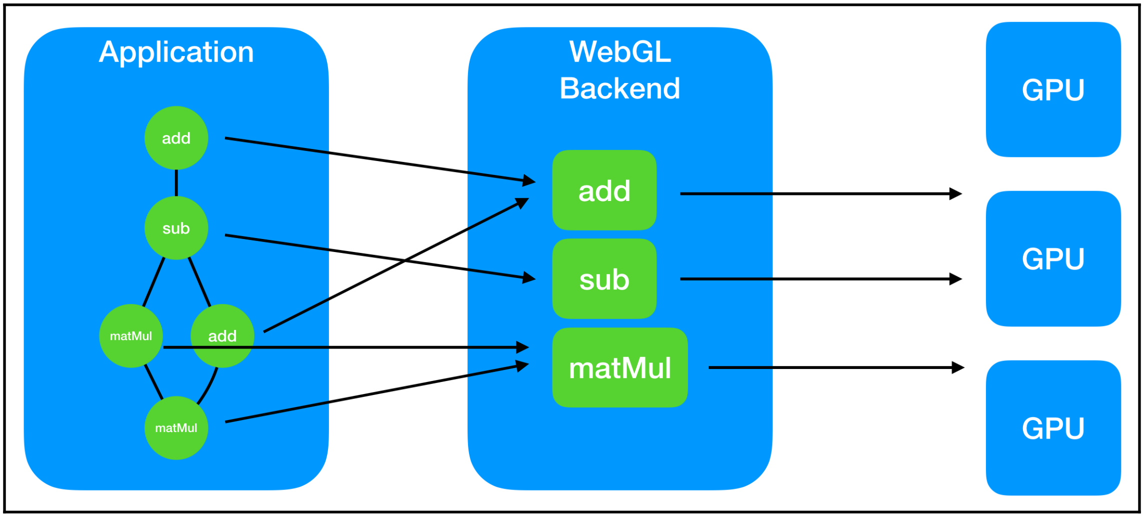


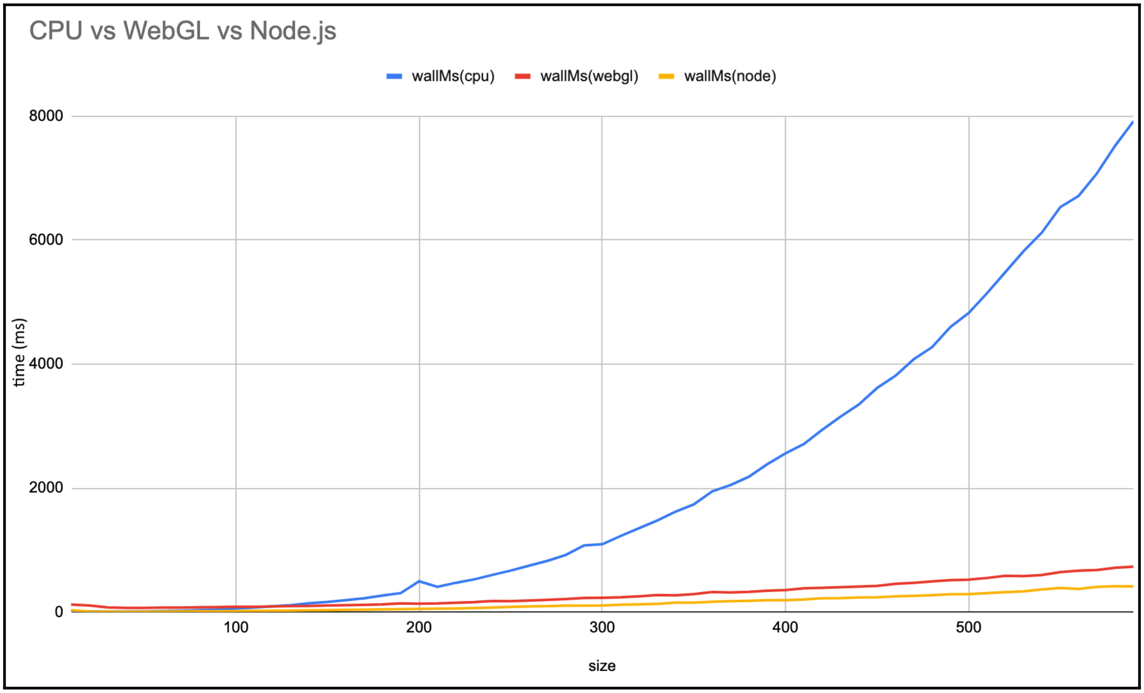
Hello, World

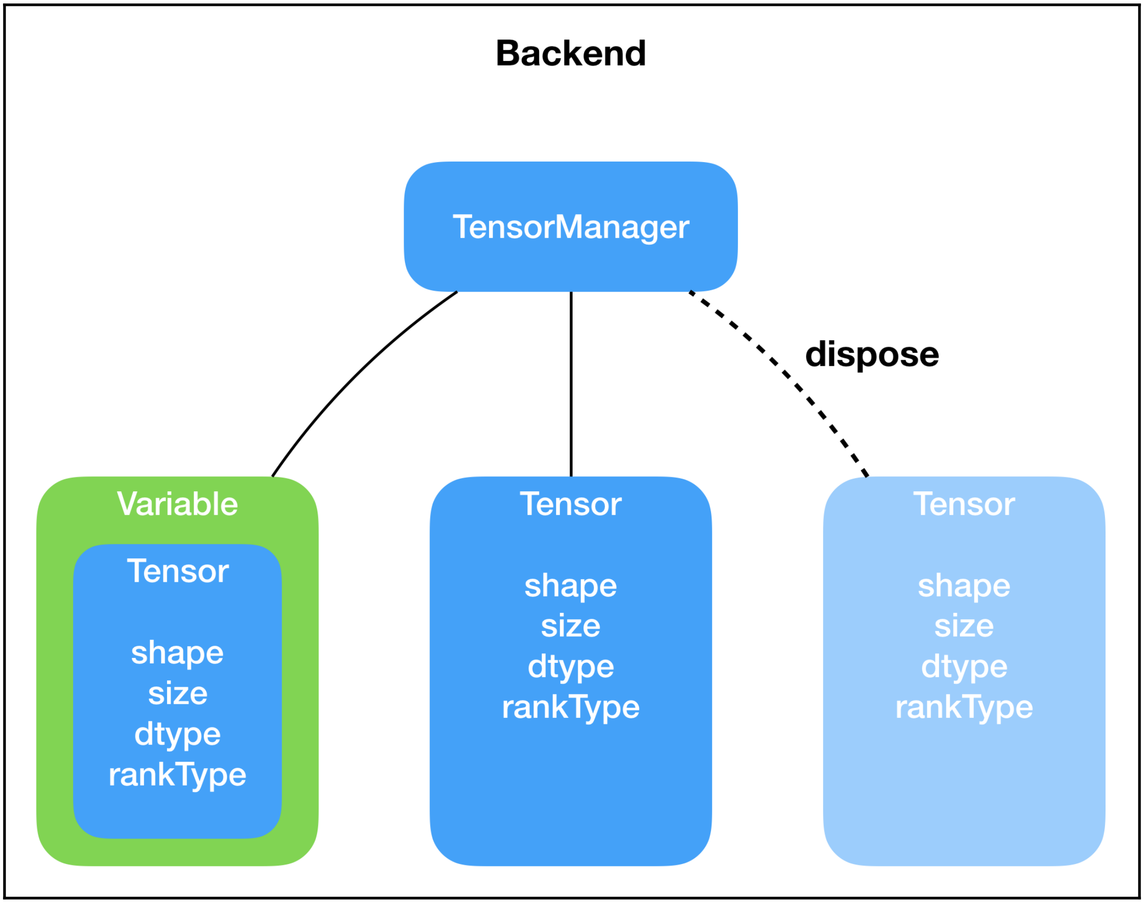
Chapter 11: Tuning Applications to Achieve High Performance

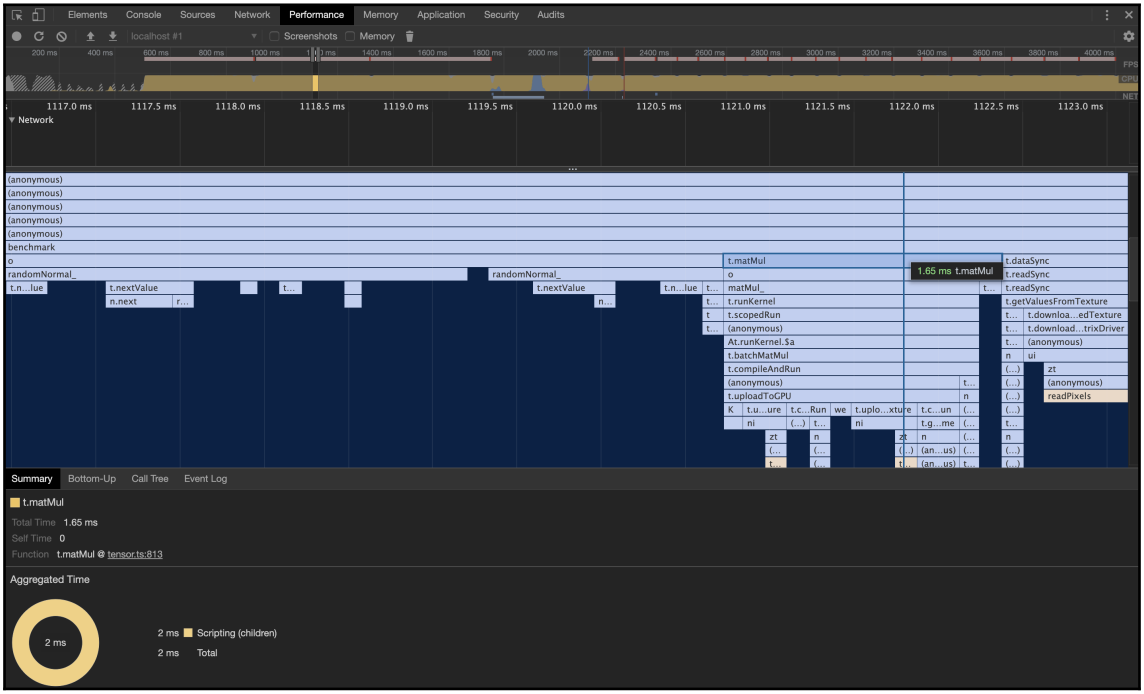





Application
add
sub
matMul
add
matMul
WebGL Backend
add
sub
matMul
GPU
GPU
GPU


CPU vs WebGL vs Node.js
wallMs(cpu)
wallMs(webgl)
wallMs(node)
8000
6000
4000
2000
0
time (ms)
100
200
300
400
500
size


Backend
TensorManager
dispose
Variable
Tensor
shape
size
dtype
rankType
Tensor
shape
size
dtype
rankType
Tensor
shape
size
dtype
rankType



t.matMul

Total Time 1.65 ms

Self Time 0

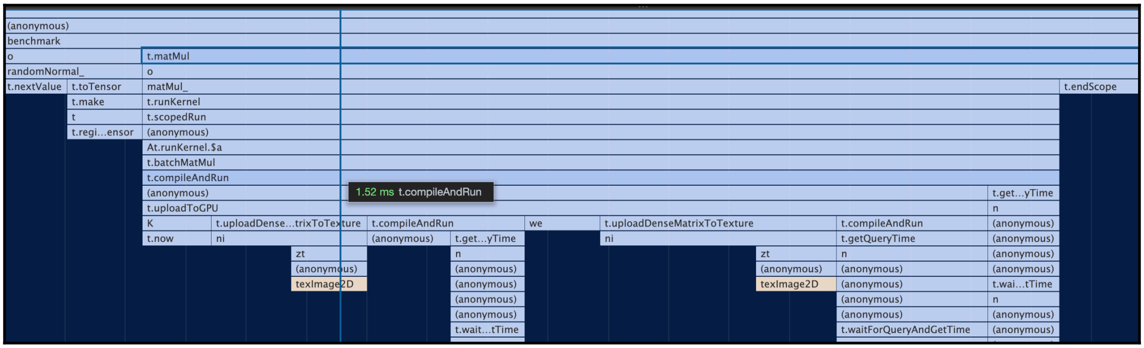
Function t.matMul @ tensor.ts:813

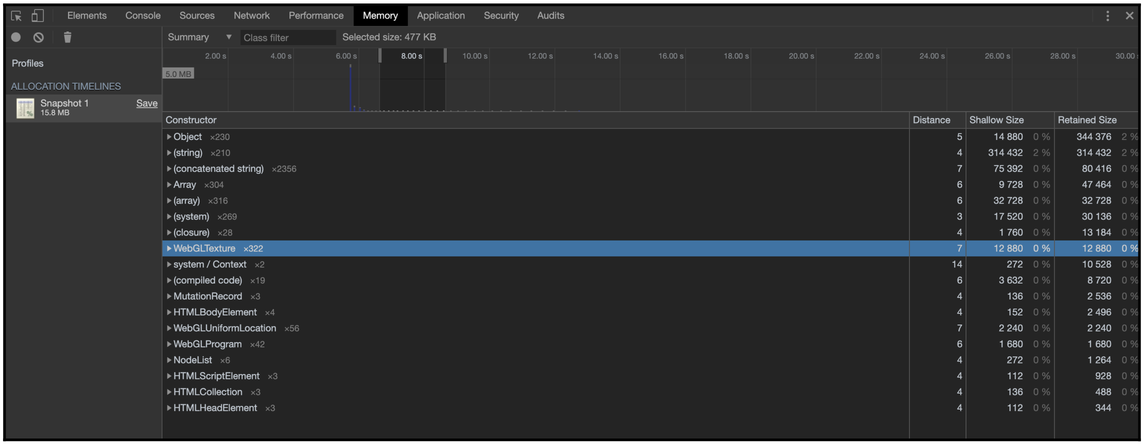
Aggregated Time

2 ms

2 ms Scripting (children)

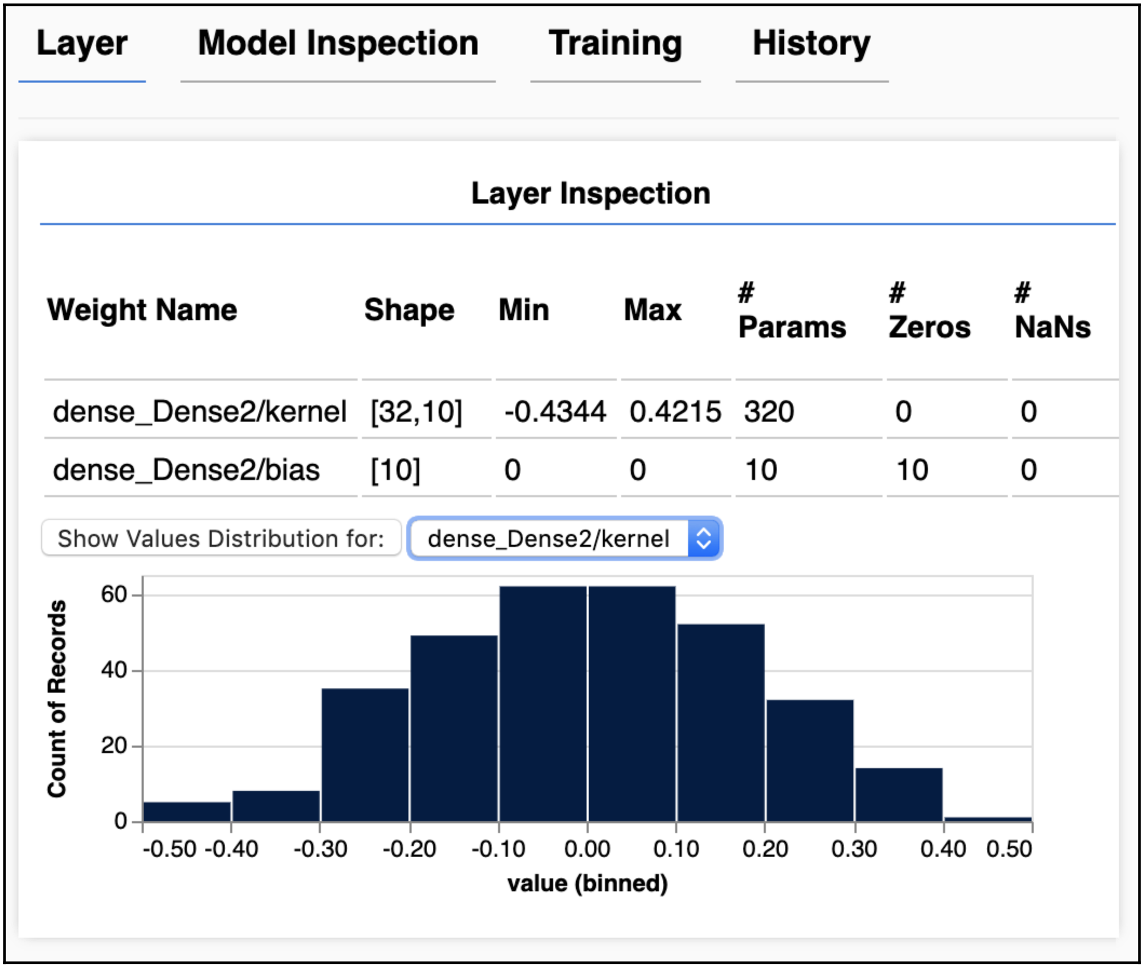
2 ms Total





Selected size: 477 KB

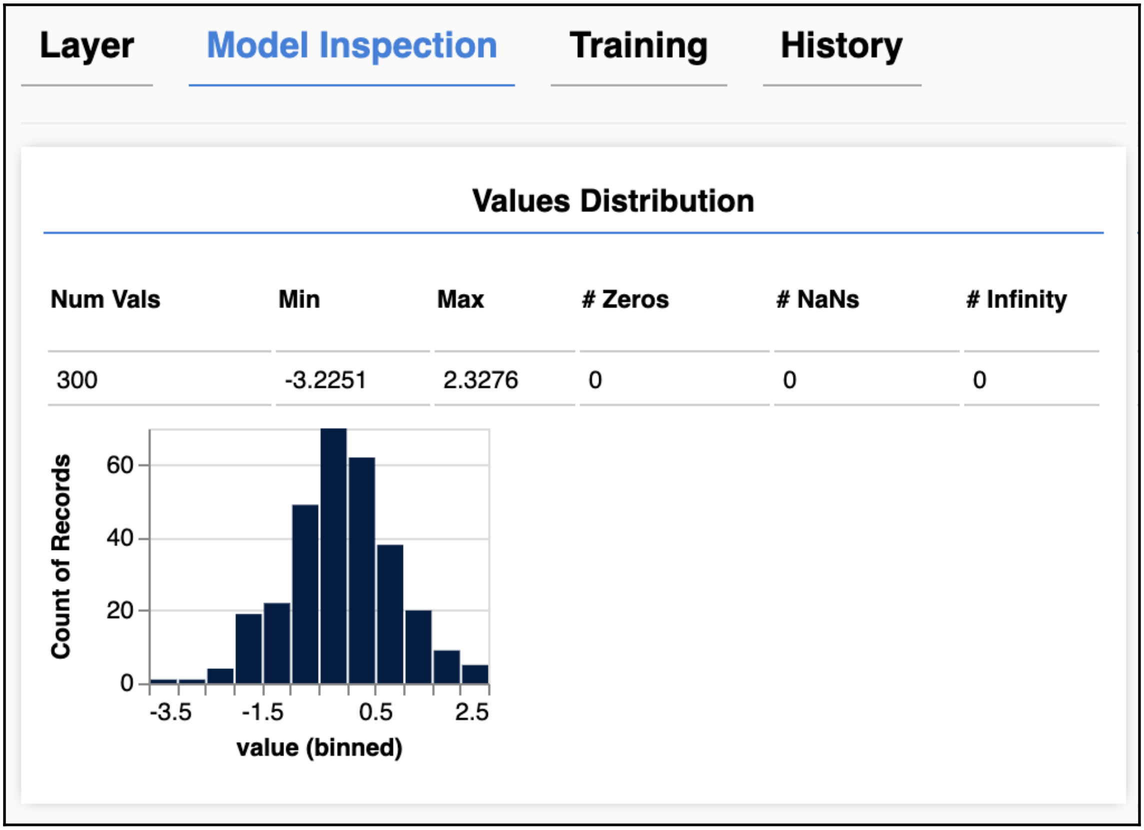
Constructor	Distance	Shallow Size		Retained Size	
Object ×230	5	14 880	0 %	344 376	2 %
(string) ×210	4	314 432	2 %	314 432	2 %
(concatenated string) ×2356	7	75 392	0 %	80 416	0 %
Array ×304	6	9 728	0 %	47 464	0 %
(array) ×316	6	32 728	0 %	32 728	0 %
(system) ×269	3	17 520	0 %	30 136	0 %
(closure) ×28	4	1 760	0 %	13 184	0 %
WebGLTexture ×322	7	12 880	0 %	12 880	0 %
system / Context ×2	14	272	0 %	10 528	0 %
(compiled code) ×19	6	3 632	0 %	8 720	0 %
MutationRecord ×3	4	136	0 %	2 536	0 %
HTMLBodyElement ×4	4	152	0 %	2 496	0 %
WebGLUniformLocation ×56	7	2 240	0 %	2 240	0 %
WebGLProgram ×42	6	1 680	0 %	1 680	0 %
NodeList ×6	4	272	0 %	1 264	0 %
HTMLScriptElement ×3	4	112	0 %	928	0 %
HTMLCollection ×3	4	136	0 %	488	0 %
HTMLHeadElement ×3	4	112	0 %	344	0 %



Layer Inspection

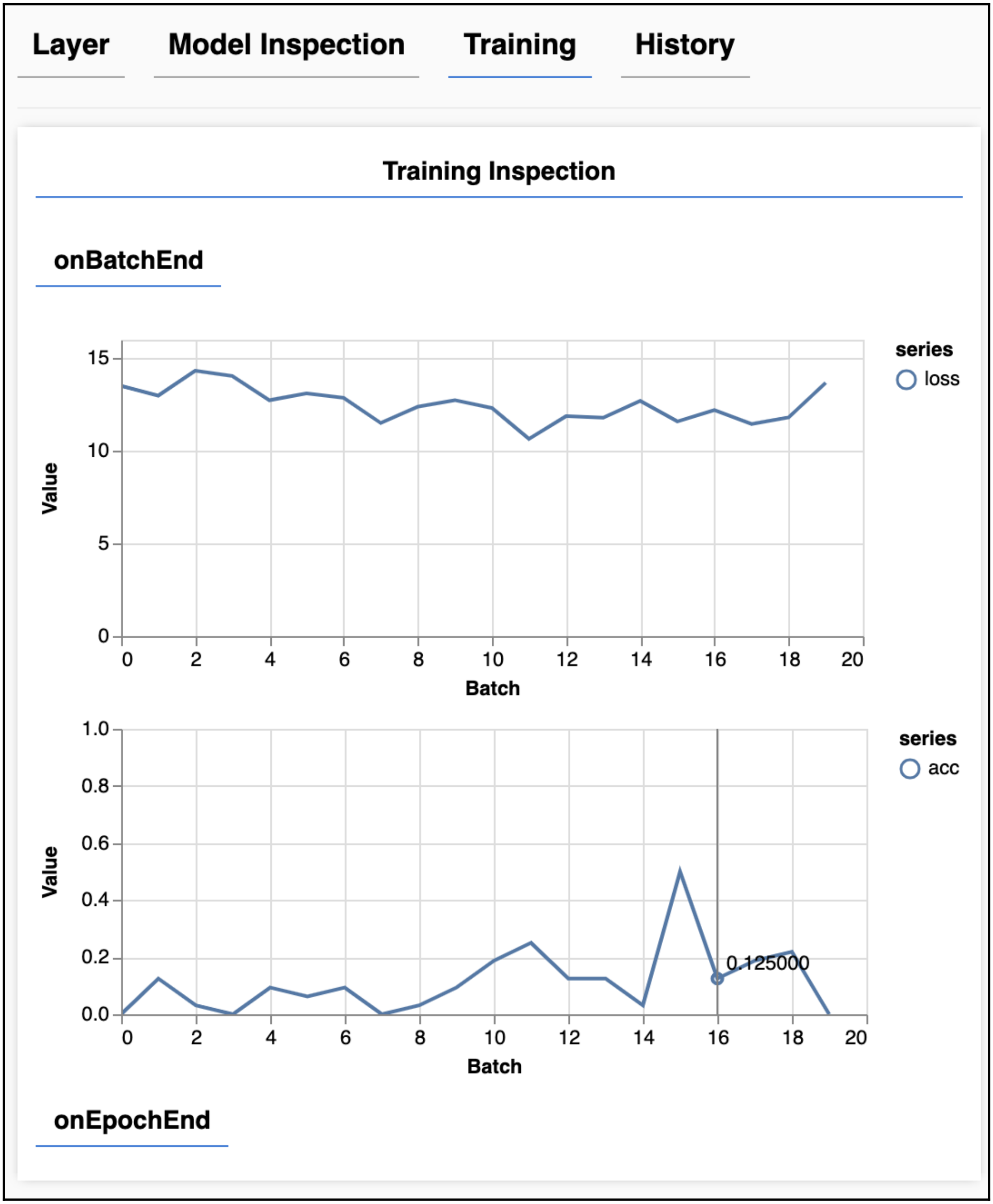
Weight Name	Shape	Min	Max	# Params	# Zeros	# NaNs
dense_Dense2/kernel	[32,10]	-0.4344	0.4215	320	0	0
dense_Dense2/bias	[10]	0	0	10	10	0

Show Values Distribution for: dense_Dense2/kernel



Values Distribution

Num Vals	Min	Max	# Zeros	# NaNs	# Infinity
300	-3.2251	2.3276	0	0	0

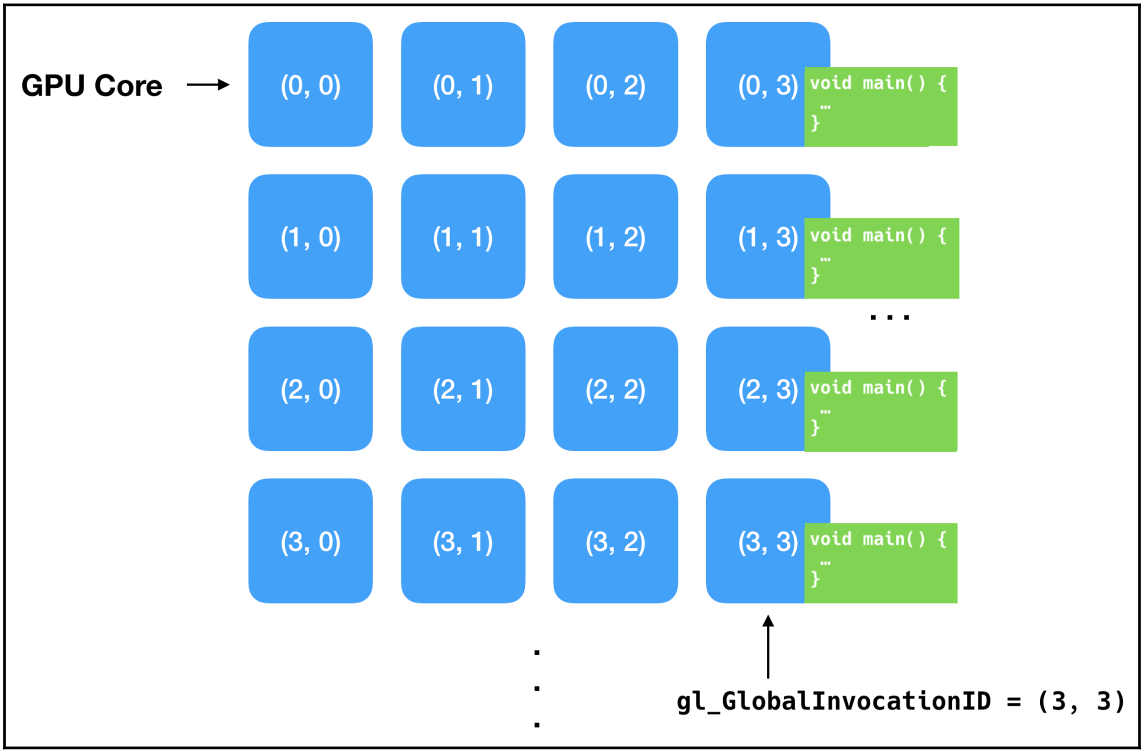


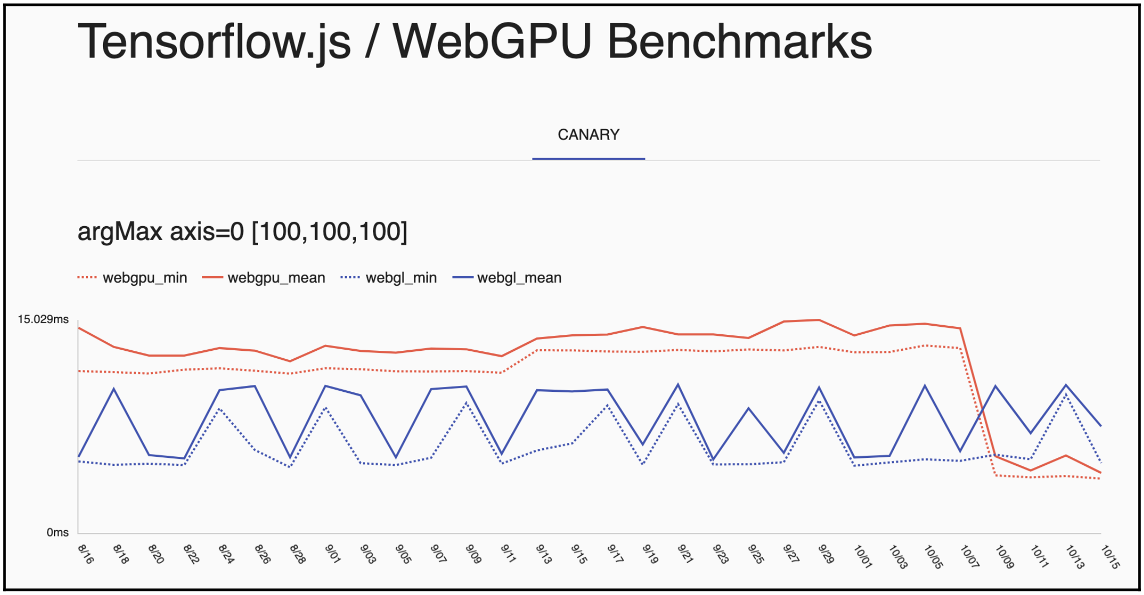
Training Inspection

onBatchEnd

onEpochEnd

Chapter 12: Future Work Around TensorFlow.js

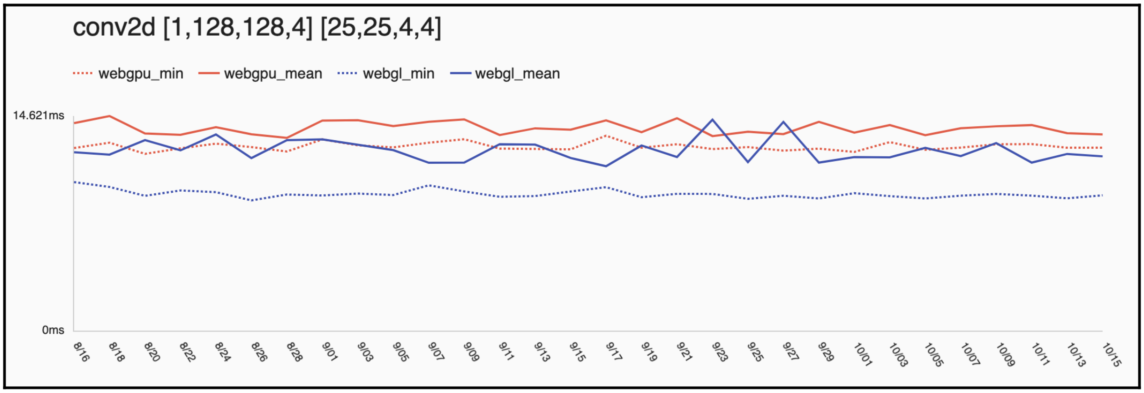




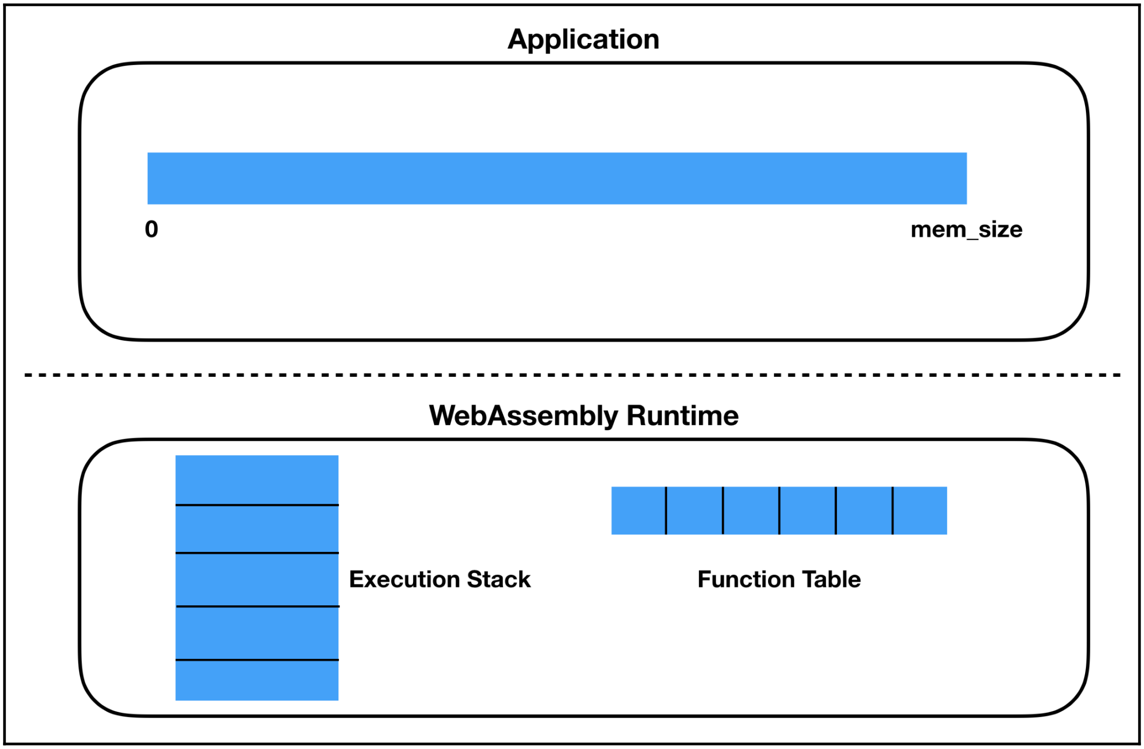
Tensorflow.js / WebGPU Benchmarks

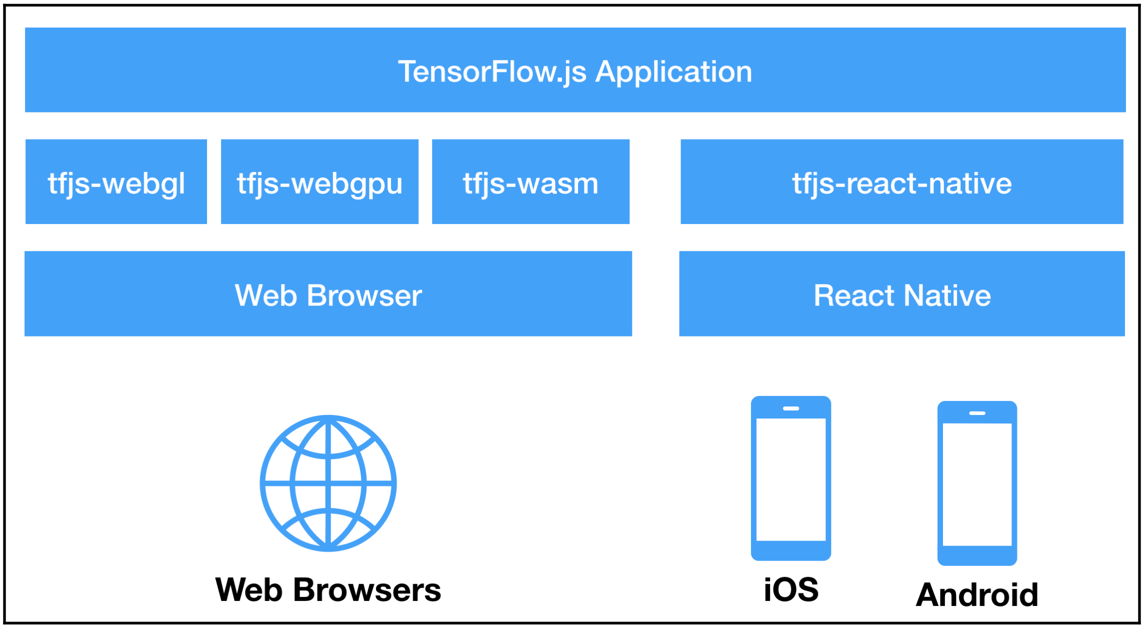
CANARY

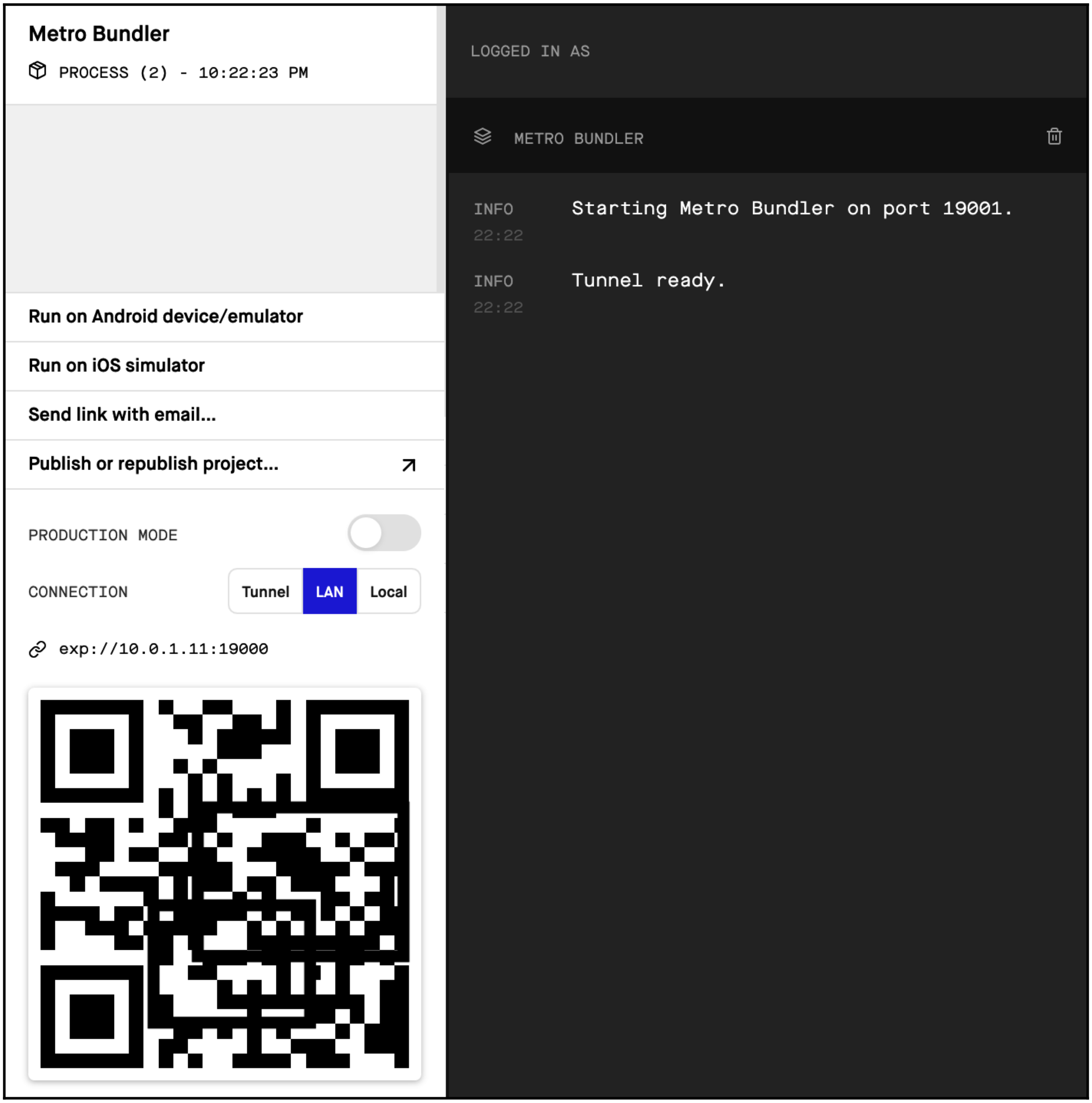
argMax axis=0 [100,100,100]



conv2d [1,128,128,4] [25,25,4,4]


Application
0
mem_size
WebAssembly Runtime
Execution Stack
Function Table


TensorFlow.js Application
tfjs-webgl
tfjs-webgpu
tfjs-wasm
tfjs-react-native
Web Browser
React Native
Web Browsers
iOS
Android



Metro Bundler

PROCESS (2) - 10:22:23 PM

Run on Android device/emulator
Run on iOS simulator
Send link with email...
Publish or republish project...
Tunnel
LAN
Local

PRODUCTION MODE ○

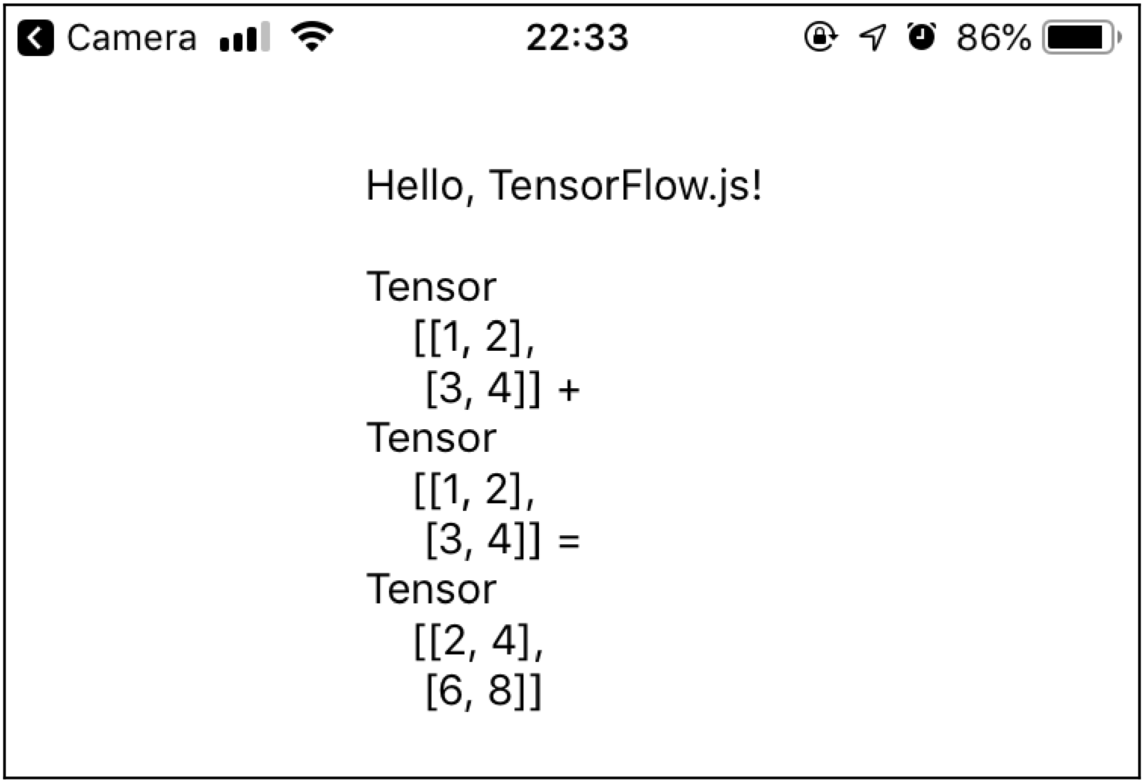
CONNECTION [✓ LAN]

exp://10.0.1.11:19000

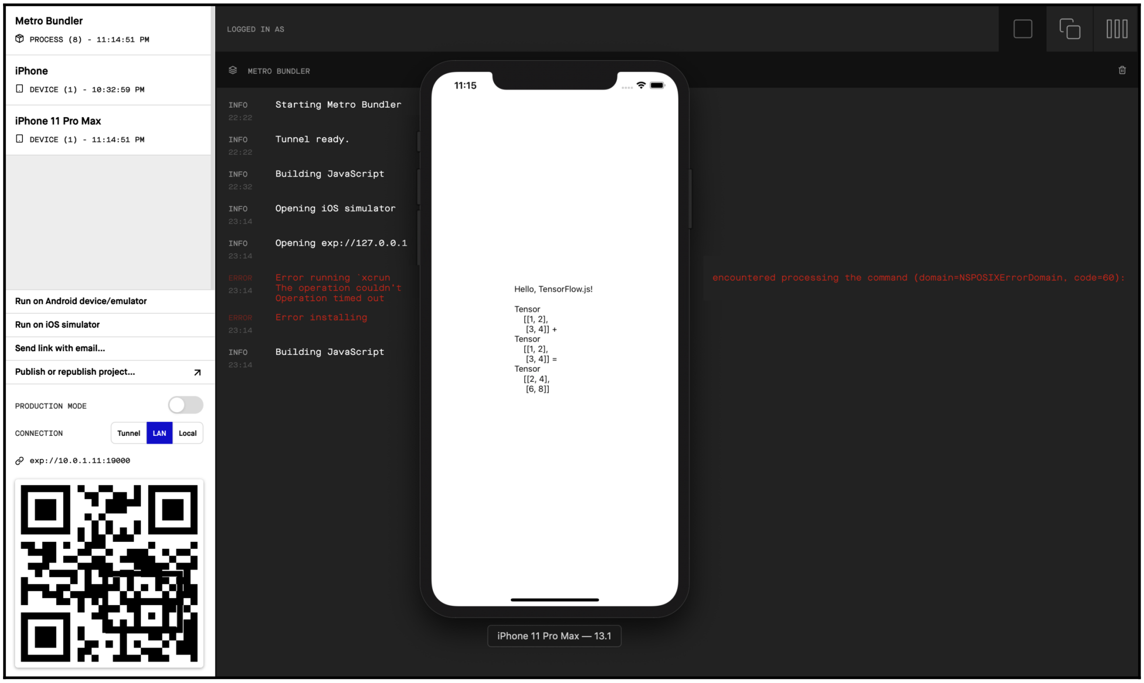
LOGGED IN AS

METRO BUNDLER

```
INFO     Starting Metro Bundler on port 19001.
22:22

INFO     Tunnel ready.
22:22
```



```
Hello, TensorFlow.js!

Tensor
    [[1, 2],
     [3, 4]] +
Tensor
    [[1, 2],
     [3, 4]] =
Tensor
    [[2, 4],
     [6, 8]]
```



Metro Bundler
PROCESS (8) - 11:14:51 PM

iPhone
DEVICE (1) - 10:32:59 PM

iPhone 11 Pro Max
DEVICE (1) - 11:14:51 PM

Run on Android device/emulator
Run on iOS simulator
Send link with email...
Publish or republish project...

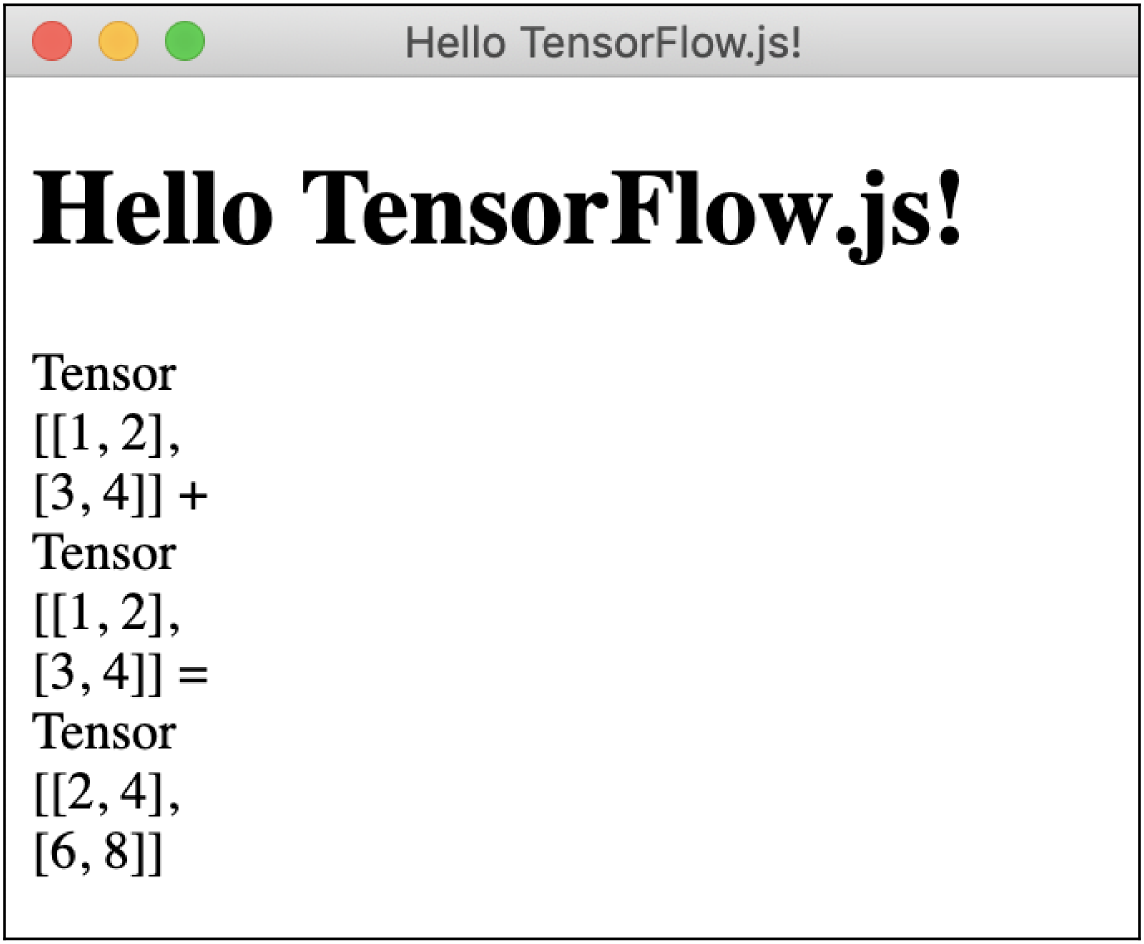
PRODUCTION MODE

CONNECTION: Tunnel [✓ LAN] Local

exp://10.0.1.11:19000

```
INFO     Starting Metro Bundler
22:22

INFO     Tunnel ready.
22:22

INFO     Building JavaScript
22:32

INFO     Opening iOS simulator
23:14

INFO     Opening exp://127.0.0.1
23:14

ERROR    Error running `xcrun
23:14    The operation couldn't
         Operation timed out                                       encountered processing the command (domain=NSPOSIXErrorDomain, code=60):

ERROR    Error installing
23:14

INFO     Building JavaScript
23:14
```

```
Hello, TensorFlow.js!

Tensor
    [[1, 2],
     [3, 4]] +
Tensor
    [[1, 2],
     [3, 4]] =
Tensor
    [[2, 4],
     [6, 8]]
```



Hello TensorFlow.js!

```
Tensor
[[1, 2],
[3, 4]] +
Tensor
[[1, 2],
[3, 4]] =
Tensor
[[2, 4],
[6, 8]]
```