11강 tensorflowjs vis

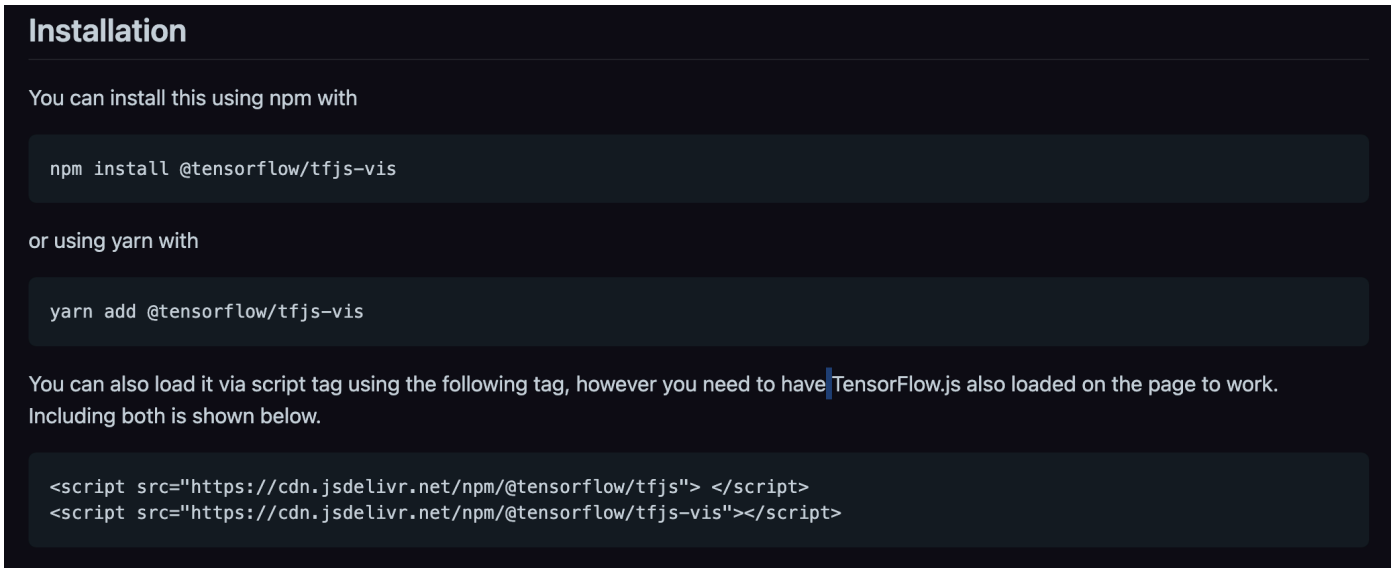
학습을 하다보면 학습을 얼마나 해야하는 가?를 고민할때가 있다

RMSE 값을 어떻게 시각적으로 볼 수 있을까?

Tensorflow.js GitHub 바로가기

시각적으로 도움을 주는 vis 라이브러리 바로가기

설치방법



도구를 사용하는방법 바로가기

tfvis.show.modelSummary (container, model) function source

Renders a summary of a tf.Model. Displays a table with layer information.

```
const model = tf.sequential({
  layers: [
    tf.layers.dense({inputShape: [784], units: 32, activation: 'relu'}),
    tf.layers.dense({units: 10, activation: 'softmax'}),
  ]
});

const surface = { name: 'Model Summary', tab: 'Model Inspection'};
tfvis.show.modelSummary(surface, model);
```

Edit Run

history

tfvis.show.history (container, history, metrics, opts?)

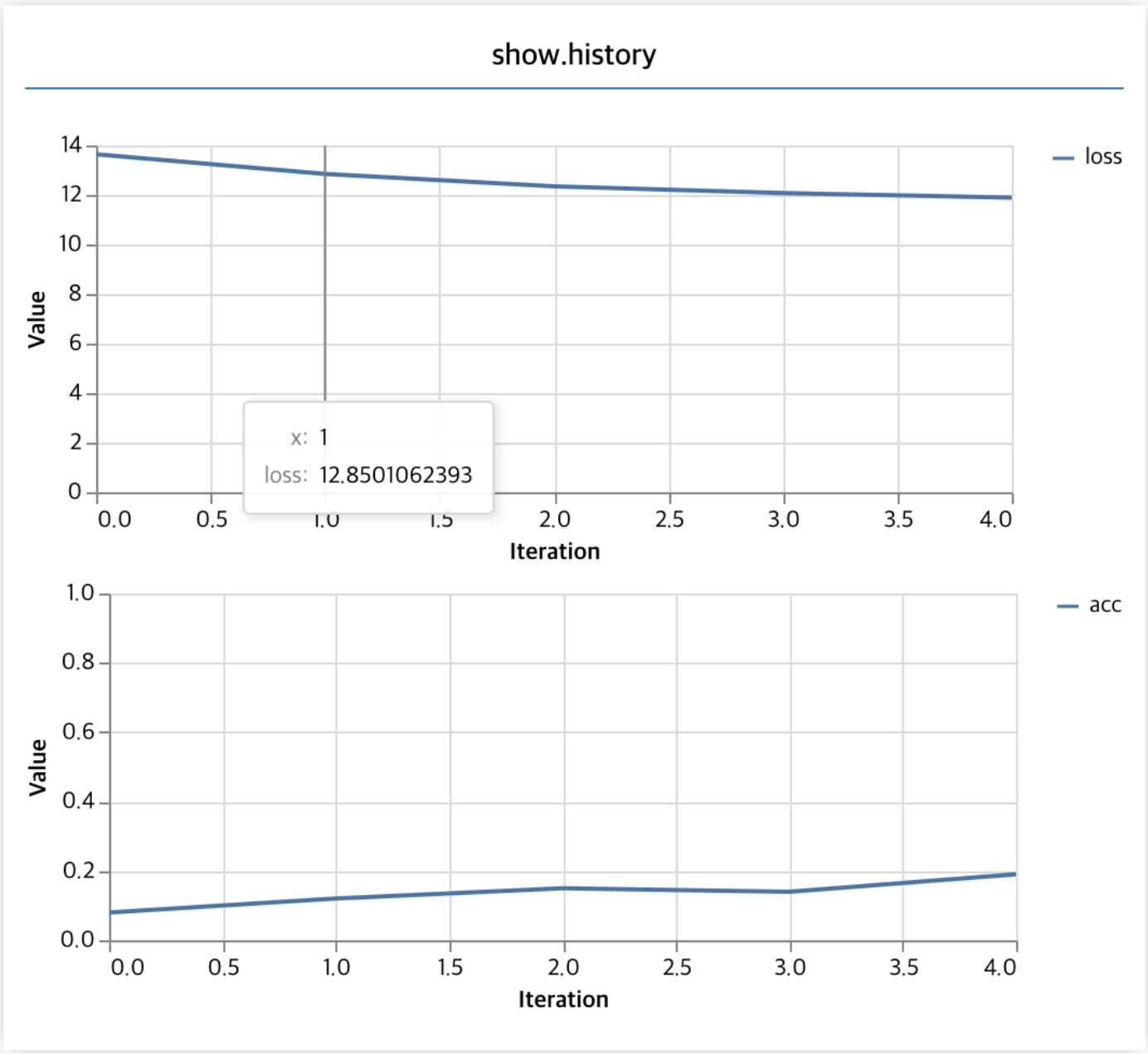
function source

Renders a tf.Model training 'History'.

```
const model = tf.sequential({
  layers: [
    tf.layers.dense({inputShape: [784], units: 32, activation: 'relu'}),
    tf.layers.dense({units: 10, activation: 'softmax'}),
  ]
});

model.compile({
   optimizer: 'sgd',
   loss: 'categoricalCrossentropy',
   metrics: ['accuracy']
});

const data = tf.randomNormal([100, 784]);
const labels = tf.randomUniform([100, 10]);

function onBatchEnd(batch, logs) {
   console.log('Accuracy', logs.acc);
}

const surface = { name: 'show.history', tab: 'Training' };
// Train for 5 epochs with batch size of 32.
const history = await model.fit(data, labels, {
    epochs: 5,
    batchSize: 32
});

tfvis.show.history(surface, history, ['loss', 'acc']);
```

Edit Run

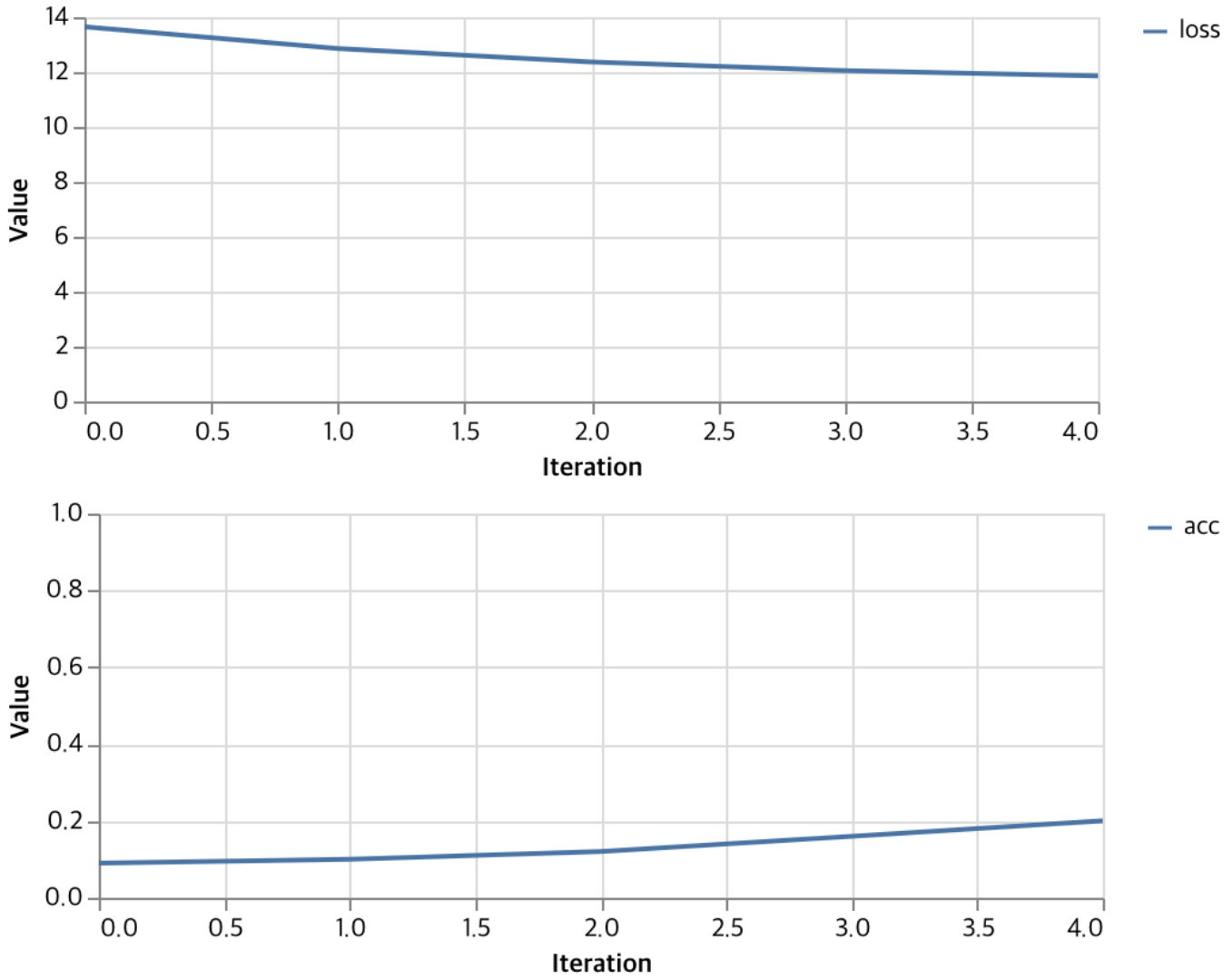
결과


show.history
loss
Value
Iteration
x: 1
loss: 12.8501062393
acc
Value
Iteration

```
const model = tf.sequential({
  layers: [
    tf.layers.dense({inputShape: [784], units: 32, activation: 'relu'}),
    tf.layers.dense({units: 10, activation: 'softmax'}),
  ]
});

model.compile({
   optimizer: 'sgd',
   loss: 'categoricalCrossentropy',
   metrics: ['accuracy']
});

const data = tf.randomNormal([100, 784]);
const labels = tf.randomUniform([100, 10]);

function onBatchEnd(batch, logs) {
   console.log('Accuracy', logs.acc);
}

const surface = { name: 'show.history live', tab: 'Training' };
// Train for 5 epochs with batch size of 32.
const history = [];
await model.fit(data, labels, {
    epochs: 5,
    batchSize: 32,
    callbacks: {
      onEpochEnd: (epoch, log) => {
        history.push(log);
        tfvis.show.history(surface, history, ['loss', 'acc']);
      }
    }
});
```

Edit Run

결과



- 움직을 보여준다

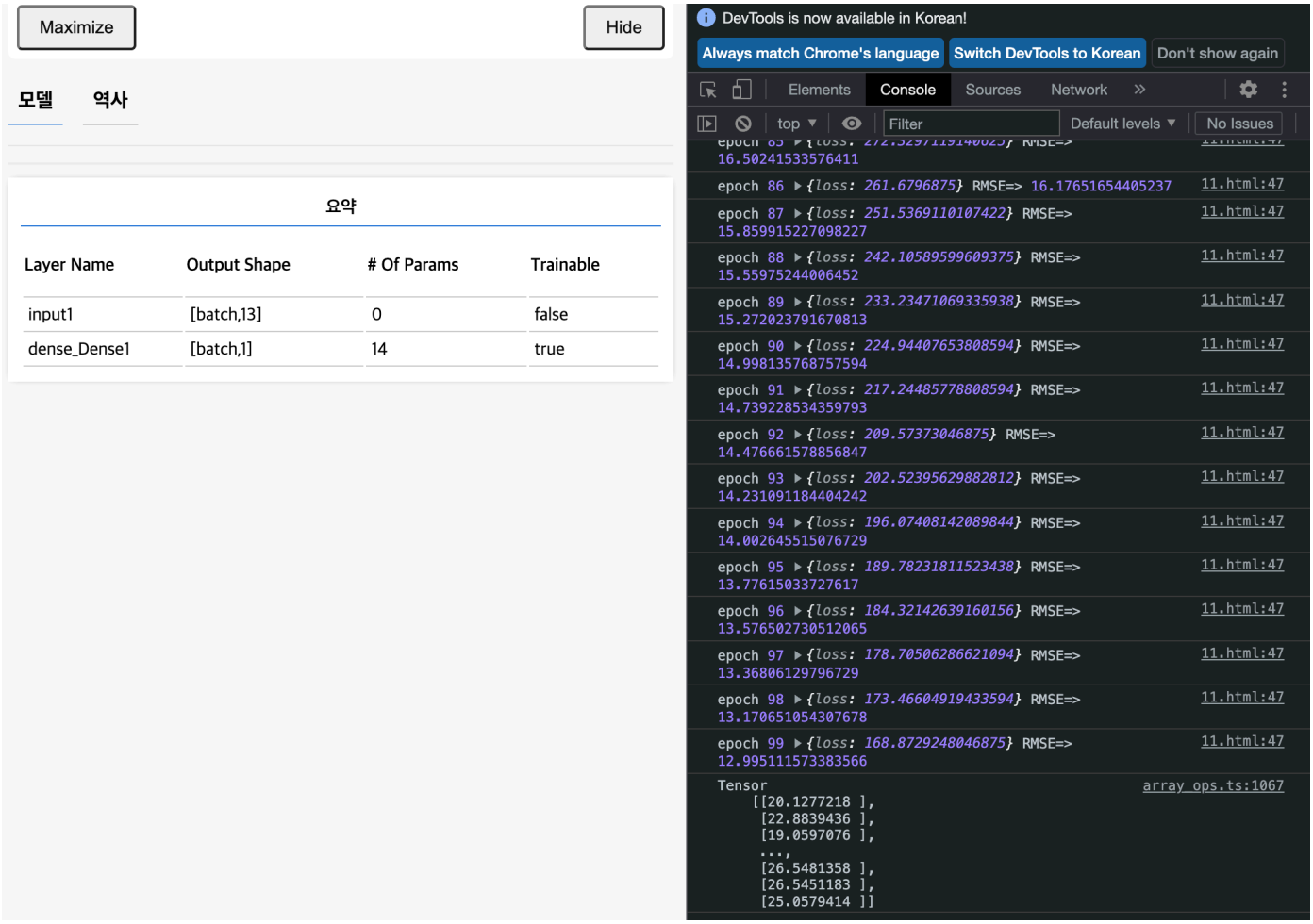
우리 코드를 수정해서 데이터의 개수와 어떻게 데이터가 변하는지 보자

```
<!DOCTYPE html>
<html>

<head>
    <title>TensorFlow.js Tutorial - boston housing </title>
```

```
    <!-- Import TensorFlow.js -->
    <script
src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.0.0/dist/tf.min.js"
></script>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs-vis">
</script>
    <script src="10.3.js"></script>
</head>

<body>
    <script>
        /*
        var 보스톤_원인 = [

[0.00632,18,2.31,0,0.538,6.575,65.2,4.09,1,296,15.3,396.9,4.98],

[0.02731,0,7.07,0,0.469,6.421,78.9,4.9671,2,242,17.8,396.9,9.14]
        ];
        var 보스톤_결과 = [
            [24],
            [21.6]
        ];
        */

        // 1. 과거의 데이터를 준비합니다.
        var 원인 = tf.tensor(보스톤_원인);
        var 결과 = tf.tensor(보스톤_결과);

        // 2. 모델의 모양을 만듭니다.
        var X = tf.input({ shape: [13] });
        var Y = tf.layers.dense({ units: 1 }).apply(X);
        var model = tf.model({ inputs: X, outputs: Y });
        var compileParam = { optimizer: tf.train.adam(), loss:
tf.losses.meanSquaredError }
        model.compile(compileParam);
        tfvis.show.modelSummary({ name: '요약', tab: '모델' }, model); //
몇개의 데이터를 받고 몇개의 데이터를 출력하는지 보여주는 것


        // 3. 데이터로 모델을 학습시킵니다.
        //         var fitParam = {epochs: 100}
        var _history = [];
        var fitParam = {
            epochs: 100,
            callbacks: {
```

```
                onEpochEnd:
                    function (epoch, logs) {
                        console.log('epoch', epoch, logs, 'RMSE=>',
Math.sqrt(logs.loss));
                        _history.push(logs);
                        tfvis.show.history({ name: 'loss', tab: '역사' },
_history, ['loss']); // 지금 값을 계속 보여주는 것
                    }
            }
        } // loss 추가 예제
        model.fit(원인, 결과, fitParam).then(function (result) {

            // 4. 모델을 이용합니다.
            // 4.1 기존의 데이터를 이용
            var 예측한결과 = model.predict(원인);
            예측한결과.print();

        });

        // 4.2 새로운 데이터를 이용
        // var 다음주온도 = [15,16,17,18,19]
        // var 다음주원인 = tf.tensor(다음주온도);
        // var 다음주결과 = model.predict(다음주원인);
        // 다음주결과.print();
    </script>
</body>

</html>
```

결과


Maximize
Hide
모델
역사
요약
Layer Name
Output Shape
Of Params
Trainable
input1
[batch,13]
0
false
dense_Dense1
[batch,1]
14
true
DevTools is now available in Korean!
Always match Chrome's language
Switch DevTools to Korean
Don't show again
Elements
Console
Sources
Network
top
Filter
Default levels
No Issues
16.50241533576411
epoch 86 {loss: 261.6796875} RMSE=> 16.17651654405237 11.html:47
epoch 87 {loss: 251.5369110107422} RMSE=> 15.859915227098227 11.html:47
epoch 88 {loss: 242.10589599609375} RMSE=> 15.55975244006452 11.html:47
epoch 89 {loss: 233.23471069335938} RMSE=> 15.272023791670813 11.html:47
epoch 90 {loss: 224.94407653808594} RMSE=> 14.998135768757594 11.html:47
epoch 91 {loss: 217.24485778808594} RMSE=> 14.739228534359793 11.html:47
epoch 92 {loss: 209.57373046875} RMSE=> 14.476661578856847 11.html:47
epoch 93 {loss: 202.52395629882812} RMSE=> 14.231091184404242 11.html:47
epoch 94 {loss: 196.07408142089844} RMSE=> 14.002645515076729 11.html:47
epoch 95 {loss: 189.78231811523438} RMSE=> 13.77615033727617 11.html:47
epoch 96 {loss: 184.32142639160156} RMSE=> 13.576502730512065 11.html:47
epoch 97 {loss: 178.70506286621094} RMSE=> 13.36806129796729 11.html:47
epoch 98 {loss: 173.46604919433594} RMSE=> 13.170651054307678 11.html:47
epoch 99 {loss: 168.8729248046875} RMSE=> 12.995111573383566 11.html:47
Tensor array_ops.ts:1067
[[20.1277218],
[22.8839436],
[19.0597076],
...,
[26.5481358],
[26.5451183],
[25.0579414]]

Maximize

Hide

모델 역사