TensorFlow

创建Tensor

主讲人：龙良曲

Outline

- from numpy, list
- zeros, ones
- fill
- random
- constant
- Application

From Numpy, List

```
In [41]: tf.convert_to_tensor(np.ones([2,3]))
<tf.Tensor: id=42, shape=(2, 3), dtype=float64, numpy=
array([[1., 1., 1.],
       [1., 1., 1.]])>

In [42]: tf.convert_to_tensor(np.zeros([2,3]))
<tf.Tensor: id=44, shape=(2, 3), dtype=float64, numpy=
array([[0., 0., 0.],
       [0., 0., 0.]])>

In [44]: tf.convert_to_tensor([1, 2])
Out[44]: <tf.Tensor: id=46, shape=(2,), dtype=int32, numpy=array([1, 2], dtype=int32)>

In [45]: tf.convert_to_tensor([1, 2.])
Out[45]: <tf.Tensor: id=48, shape=(2,), dtype=float32, numpy=array([1., 2.], dtype=float32)>

In [46]: tf.convert_to_tensor([[1], [2.]])
<tf.Tensor: id=50, shape=(2, 1), dtype=float32, numpy=
array([[1.],
       [2.]], dtype=float32)>
```

tf.zeros

```
In [6]: tf.zeros([])
Out[6]: <tf.Tensor: id=2, shape=(), dtype=float32, numpy=0.0>

In [7]: tf.zeros([1])
Out[7]: <tf.Tensor: id=6, shape=(1,), dtype=float32, numpy=array([0.], dtype=float32)>

In [8]: tf.zeros([2,2])
<tf.Tensor: id=10, shape=(2, 2), dtype=float32, numpy=
array([[0., 0.],
       [0., 0.]], dtype=float32)>

In [9]: tf.zeros([2,3,3])
<tf.Tensor: id=14, shape=(2, 3, 3), dtype=float32, numpy=
array([[[0., 0., 0.],
        ...
        [0., 0., 0.]]], dtype=float32)>
```

tf.zeros_like

```
In [10]: a=tf.zeros([2,3,3])

In [11]: tf.zeros_like(a)
Out[11]:
<tf.Tensor: id=19, shape=(2, 3, 3), dtype=float32, numpy=
array([[[0., 0., 0.],
        ...
        [0., 0., 0.]]], dtype=float32)>

In [12]: tf.zeros(a.shape)
Out[12]:
<tf.Tensor: id=23, shape=(2, 3, 3), dtype=float32, numpy=
array([[[0., 0., 0.],
        ...
        [0., 0., 0.]]], dtype=float32)>
```

tf.ones

```
In [13]: tf.ones(1)
Out[13]: <tf.Tensor: id=27, shape=(1,), dtype=float32, numpy=array([1.], dtype=float32)>
In [17]: tf.ones([])
Out[17]: <tf.Tensor: id=41, shape=(), dtype=float32, numpy=1.0>

In [14]: tf.ones([2])
Out[14]: <tf.Tensor: id=31, shape=(2,), dtype=float32, numpy=array([1., 1.],
dtype=float32)>

In [15]: tf.ones([2,3])
<tf.Tensor: id=35, shape=(2, 3), dtype=float32, numpy=
array([[1., 1., 1.],
       [1., 1., 1.]], dtype=float32)>

In [16]: tf.ones_like(a)
<tf.Tensor: id=39, shape=(2, 3, 3), dtype=float32, numpy=
array([[[1., 1., 1.],
        ...
        [1., 1., 1.]]], dtype=float32)>
```

Fill

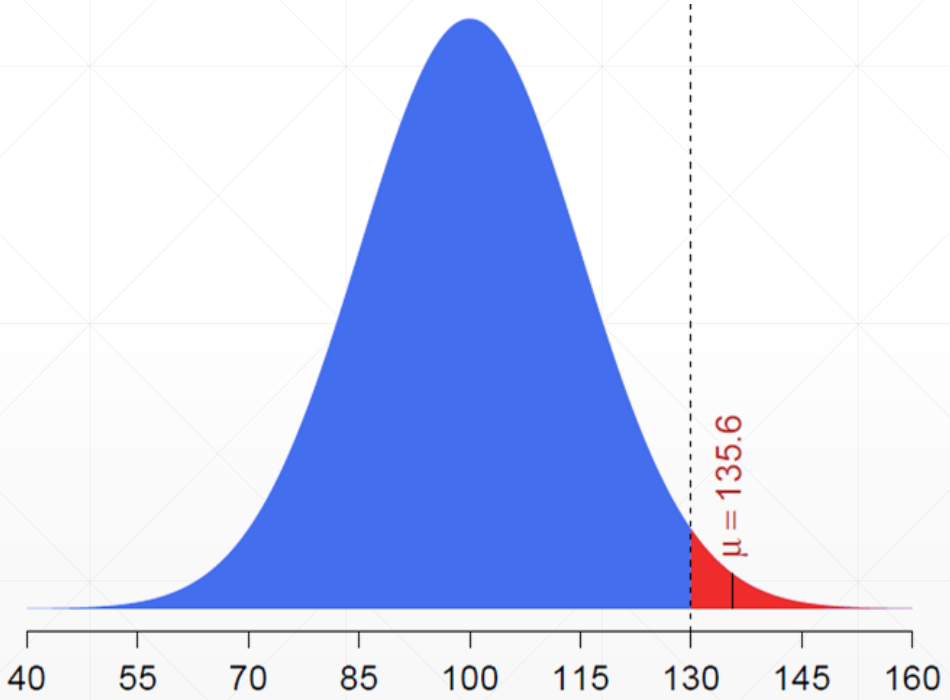
```
In [52]: tf.fill([2,2],0)
<tf.Tensor: id=91, shape=(2, 2), dtype=int32, numpy=
array([[0, 0],
       [0, 0]], dtype=int32)>

In [53]: tf.fill([2,2],0.)
<tf.Tensor: id=95, shape=(2, 2), dtype=float32, numpy=
array([[0., 0.],
       [0., 0.]], dtype=float32)>

In [54]: tf.fill([2,2],1)
<tf.Tensor: id=99, shape=(2, 2), dtype=int32, numpy=
array([[1, 1],
       [1, 1]], dtype=int32)>

In [55]: tf.fill([2,2],9)
<tf.Tensor: id=103, shape=(2, 2), dtype=int32, numpy=
array([[9, 9],
       [9, 9]], dtype=int32)>
```

Normal

```
In [47]: tf.random.normal([2,2],mean=1, stddev=1)
Out[47]:
<tf.Tensor: id=57, shape=(2, 2), dtype=float32, numpy=
array([[2.3931956 , 0.33781087],
       [1.0709286 , 1.1542176 ]], dtype=float32)>

In [48]: tf.random.normal([2,2])
Out[48]:
<tf.Tensor: id=64, shape=(2, 2), dtype=float32, numpy=
array([[1.2966702 , 0.00877222],
       [1.7096056 , 0.69879115]], dtype=float32)>

In [49]: tf.random.truncated_normal([2,2],mean=0,stddev=1)
Out[49]:
<tf.Tensor: id=71, shape=(2, 2), dtype=float32, numpy=
array([[-0.56123465,  1.3936464 ],
       [-1.1112062 , -0.76954645]], dtype=float32)>
```

Truncated Distribution

- Gradient Vanish



https://assessingpsyche.wordpress.com/2014/06/04/using-the-truncated-normal-distribution/

Uniform

```
In [50]: tf.random.uniform([2,2],minval=0, maxval=1)
Out[50]:
<tf.Tensor: id=79, shape=(2, 2), dtype=float32, numpy=
array([[0.1432991 , 0.0267868 ],
       [0.08979011, 0.8807217 ]], dtype=float32)>

In [51]: tf.random.uniform([2,2],minval=0, maxval=100)
Out[51]:
<tf.Tensor: id=87, shape=(2, 2), dtype=float32, numpy=
array([[ 9.66835 , 34.826958],
       [69.1662  , 33.22189 ]], dtype=float32)>
```

Random Permutation

```
In [24]: idx=tf.range(10)
In [26]: idx=tf.random.shuffle(idx)
Out[27]: <tf.Tensor: id=67, shape=(10,), dtype=int32, numpy=array([2, 1, 9, 3, 8,
7, 0, 5, 4, 6])>

In [29]: a=tf.random.normal([10,784])
In [31]: b=tf.random.uniform([10], maxval=10, dtype=tf.int32)
Out[32]: <tf.Tensor: id=78, shape=(10,), dtype=int32, numpy=array([1, 4, 6, 9, 4,
9, 3, 2, 0, 5])>

In [33]: a=tf.gather(a,idx)
In [34]: b=tf.gather(b,idx)
Out[35]: <tf.Tensor: id=83, shape=(10,), dtype=int32, numpy=array([6, 4, 5, 9, 0,
2, 1, 9, 4, 3])>
```

tf.constant

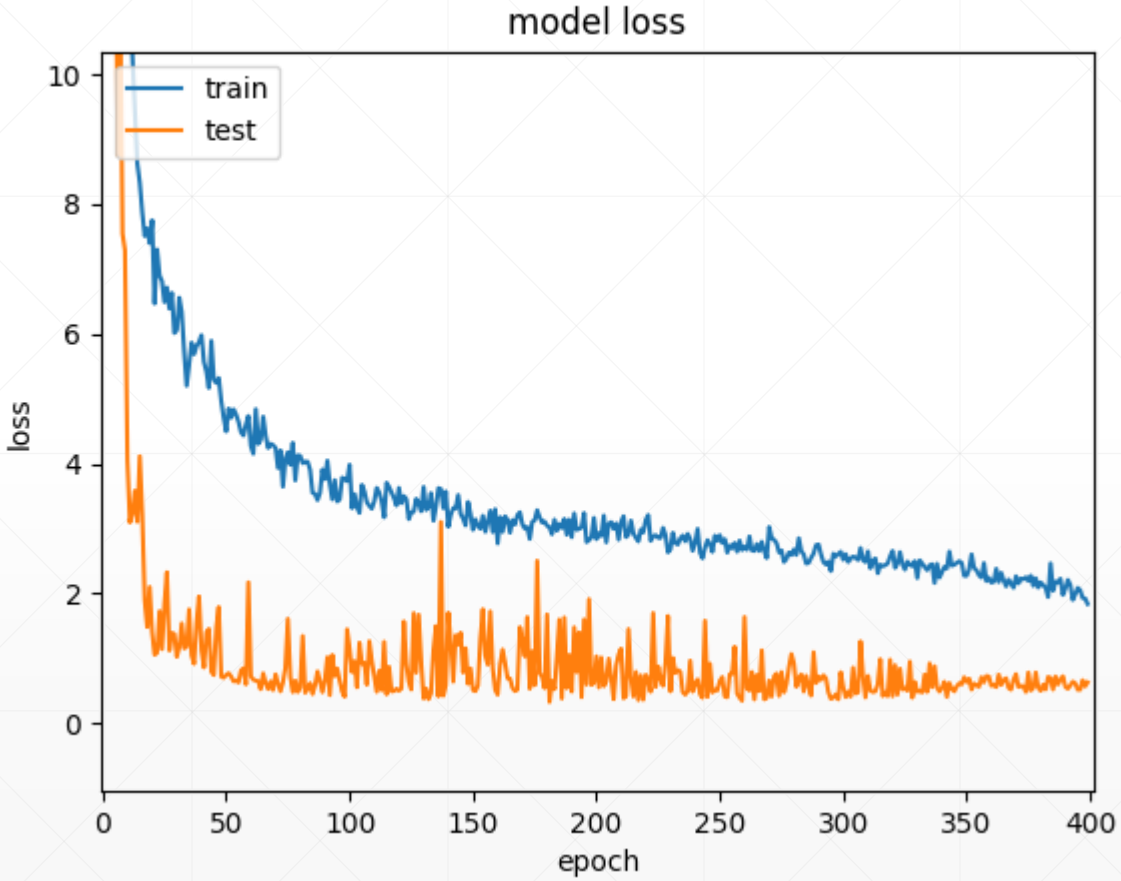
```
In [56]: tf.constant(1)
Out[56]: <tf.Tensor: id=105, shape=(), dtype=int32, numpy=1>

In [57]: tf.constant([1])
Out[57]: <tf.Tensor: id=107, shape=(1,), dtype=int32, numpy=array([1], dtype=int32)>

In [58]: tf.constant([1,2.])
Out[58]: <tf.Tensor: id=109, shape=(2,), dtype=float32, numpy=array([1., 2.], dtype=float32)>

In [59]: tf.constant([[1,2.],[3.]])
---------------------------------------------------------------------------
ValueError: Can't convert non-rectangular Python sequence to Tensor.
```

Typical Dim Data

- []
- [d]
- [h, w]
- [b, len, vec]
- [b, h, w, c]
- [t, b, h, w, c]
- ...

Scalar

- []
  - 0, 1., 2.2...
- loss = mse(out, y)
- accuracy

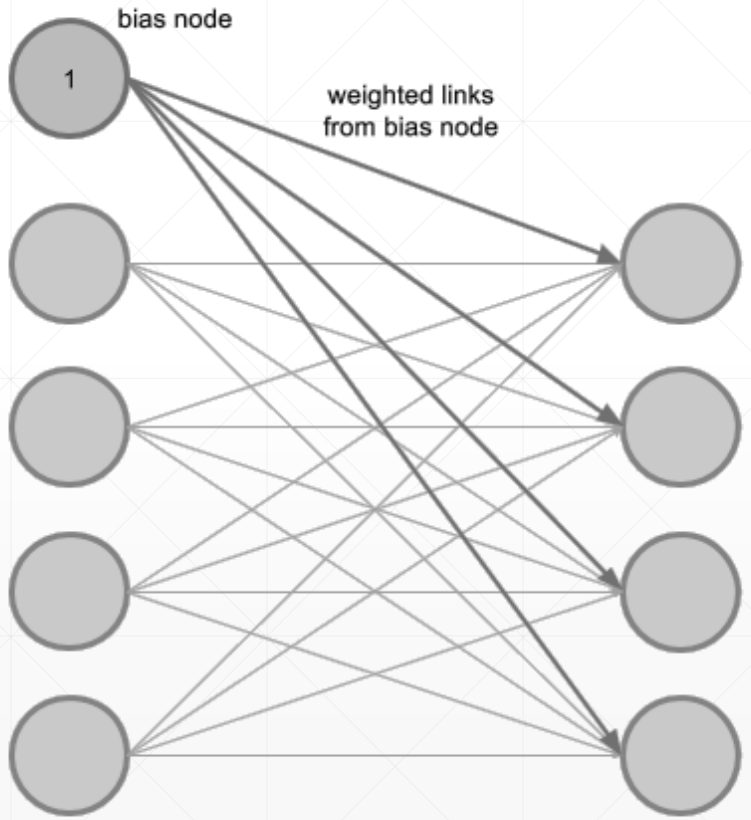


Loss

```
In [24]: out=tf.random.uniform([4,10])
<tf.Tensor: id=78, shape=(4, 10), dtype=float32, numpy=
array([[0.11917067, 0.4999225 , 0.5349126 , 0.12400055, 0.5961182 , ...
        0.48180282, 0.47066116, 0.96908057, 0.6354896 , 0.44619405]],
      dtype=float32)>

In [28]: y=tf.range(4)
In [29]: y=tf.one_hot(y,depth=10)
<tf.Tensor: id=95, shape=(4, 10), dtype=float32, numpy=
array([[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.], ...
       [0., 0., 0., 1., 0., 0., 0., 0., 0., 0.]], dtype=float32)>

In [31]: loss=tf.keras.losses.mse(y, out)
Out[32]: <tf.Tensor: id=99, shape=(4,), dtype=float32, numpy=array([0.37688   , 0.27573627,
0.5074741 , 0.3111775 ], dtype=float32)>

In [33]: loss=tf.reduce_mean(loss)
Out[34]: <tf.Tensor: id=102, shape=(), dtype=float32, numpy=0.36781698>
```

Vector

- Bias
  - [out_dim]



Vector

```
In [20]: net=layers.Dense(10)
In [21]: net.build((4,8))
In [22]: net.kernel
<tf.Variable 'kernel:0' shape=(8, 10) dtype=float32, numpy=
array([[ 0.19603091, -0.2846143 ,  0.14747244, -0.03926206, -0.02218038,
         0.02417278, -0.31929022, -0.07324755,  0.17448658,  0.06289428]],
      dtype=float32)>

In [23]: net.bias
Out[23]: <tf.Variable 'bias:0' shape=(10,) dtype=float32, numpy=array([0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.], dtype=float32)>
```

Matrix

- input x: [b, vec_dim]
- weight: [input_dim, output_dim]

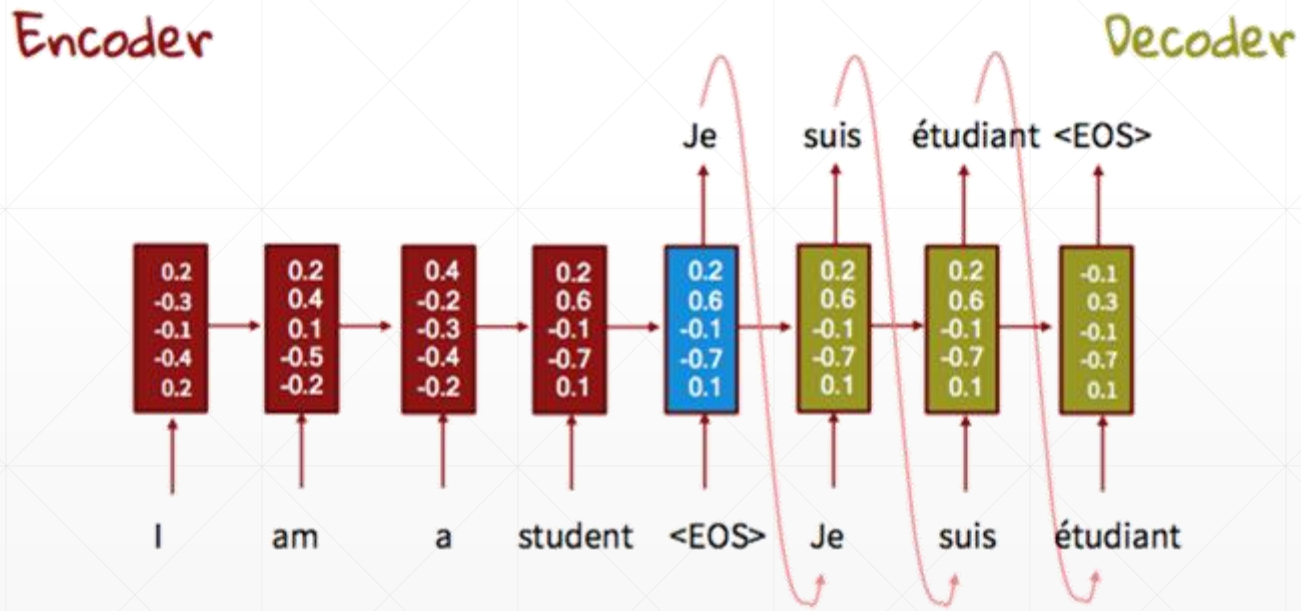
$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ & & & \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

Matrix

```
In [35]: x = tf.random.normal([4,784])
Out[36]: TensorShape([4, 784])

In [38]: net=layers.Dense(10)
In [39]: net.build((4,784))

In [40]: net(x).shape
Out[40]: TensorShape([4, 10])

In [41]: net.kernel.shape
Out[41]: TensorShape([784, 10])

In [42]: net.bias.shape
Out[42]: TensorShape([10])
```

Dim=3 Tensor

- x: [b, seq_len, word_dim]
  - [b, 5, 5]

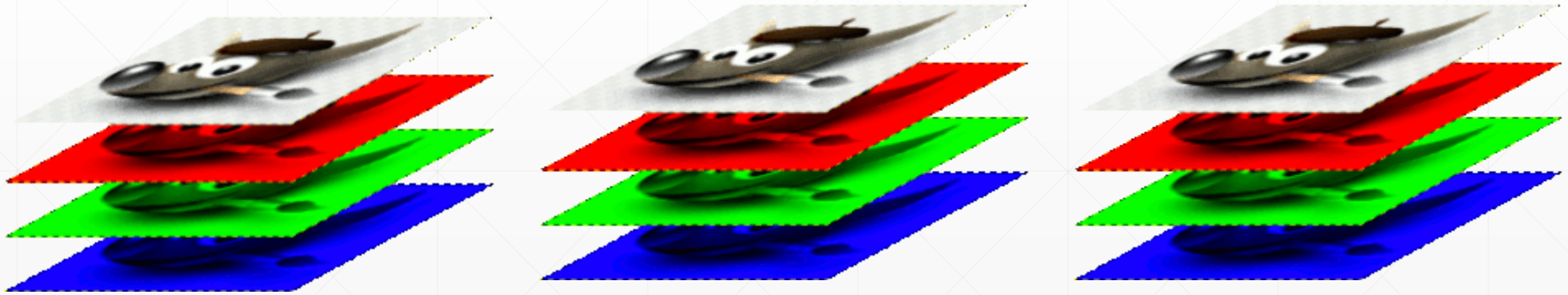


https://medium.com/syncedreview/representations-for-language-from-word-embeddings-to-sentence-meanings-3fae81b2379a

```
In [4]: (X_train, y_train), (X_test, y_test) =
keras.datasets.imdb.load_data(num_words=10000)
In [5]: x_train = keras.preprocessing.sequence.pad_sequences(X_train, maxlen=80)
In [14]: x_train.shape
Out[14]: TensorShape([25000, 80])

In [12]: emb = embedding(x_train)
In [13]: emb.shape
Out[13]: TensorShape([25000, 80, 100])

In [19]: out=rnn(emb[:4])
In [20]: out.shape
Out[20]: TensorShape([4, 256])
```

Dim=4 Tensor

- Image: [b, h, w, 3]
- feature maps: [b, h, w, c]



https://docs.gimp.org/2.4/en/gimp-images-in.html

```
In [4]: x=tf.random.normal((4,32,32,3))

In [5]: net=layers.Conv2D(16,kernel_size=3)

In [6]: net(x)
# [4, 32, 32, 16]
```

Dim=5 Tensor

- Single task: [b, h, w, 3]
- meta-learning:
  - [task_b, b, h, w, 3]

Learning
How to Learn

https://www.coursera.org/learn/learning-how-to-learn

下一课时

索引与切片

Thank You.