自编码器与Neural Transfer

讲师：佟达

简介：人工智能架构师

邮箱：dtong@thoughtworks.com

学习目标：

- 优化器
- 自编码器
- 反卷积
- Live Coding - Autoencoder
- Regularization (L1/L2, Dropout, Batch Normalization)
- Neural Transfer

讲师介绍：

会coding的科学家，懂数学的工程师，常年提着酱油瓶游走在学术与工程之间，从云计算到DevOps，从微服务到人工智能，不敢止步。

回顾

卷积神经络(CNN)

- 解卷积神经 络和相关概念
- 学习如何在 TensorFlow 中创建、训练 CNN 模型

优化器（Optimizer）

随机梯度递降（SGD)

随机梯度递降（SGD)

- 采样一批次（batch）数据
- 前向通过计算图，计算Loss
- 反向计算梯度
- 更新参数

随机梯度递降（SGD)

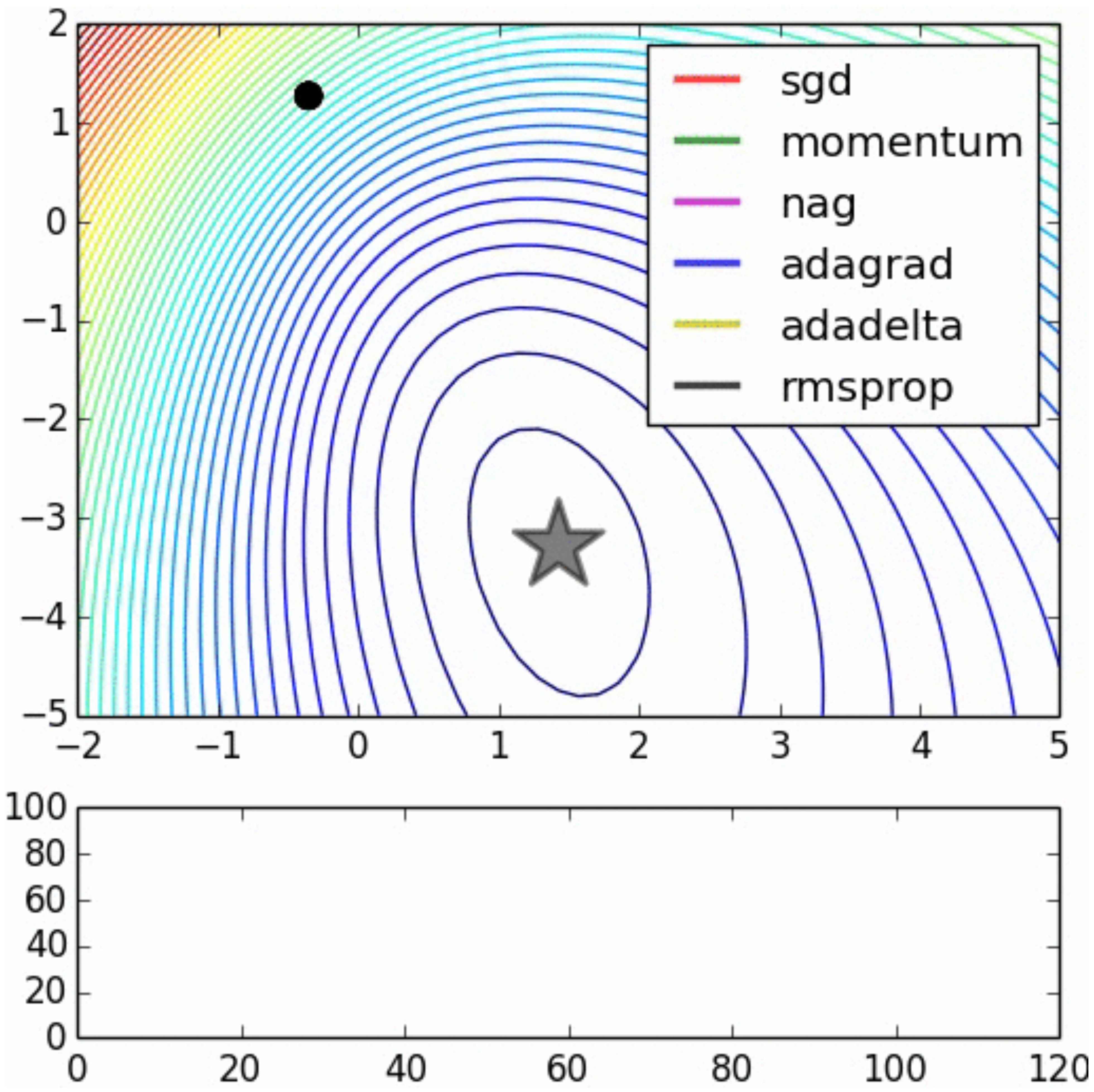
```
while True:
    data_batch = dataset.sample_data_batch()
    loss = network.forward(data_batch)
    dx = network.backward()
    x += - learning_rate * dx
```

随机梯度递降（SGD)

```
while True:
    data_batch = dataset.sample_data_batch()
    loss = network.forward(data_batch)
    dx = network.backward()
    x += - learning_rate * dx
```

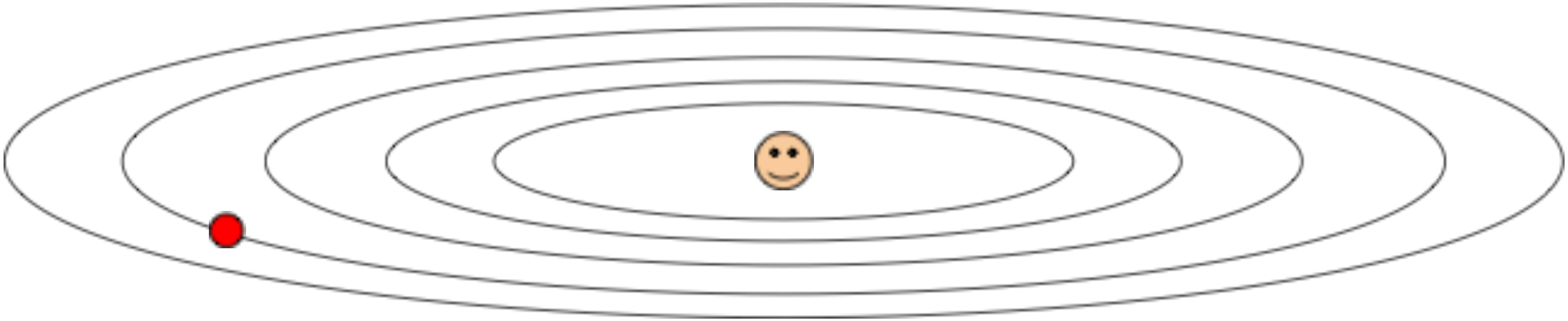
更复杂的更新策略

随机梯度递降（SGD)



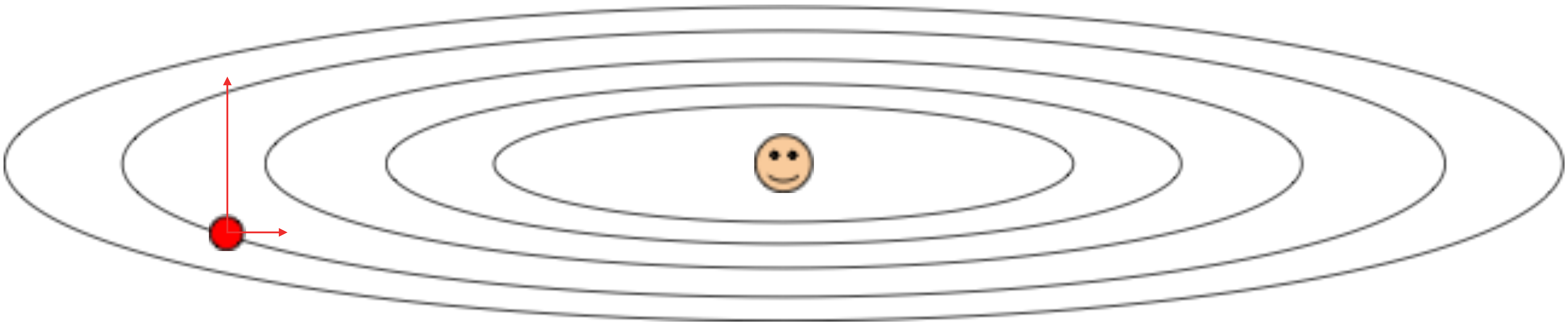
慕课学院
www.mooc.ai

随机梯度递降（SGD)



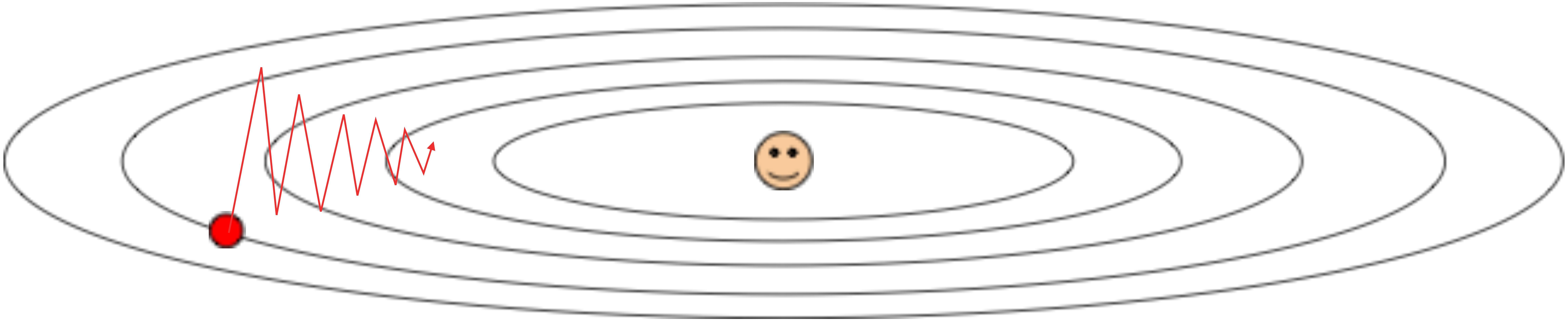
思考：当前情况下，SGD策略的更新轨迹是什么样的。

随机梯度递降（SGD)



思考：当前情况下，SGD策略的更新轨迹是什么样的。

随机梯度递降（SGD)



思考：当前情况下，SGD策略的更新轨迹是什么样的。

在比较陡峭的方向会来回抖动，在平缓的方向缓慢趋向最低点。

Momentum （动量） Update

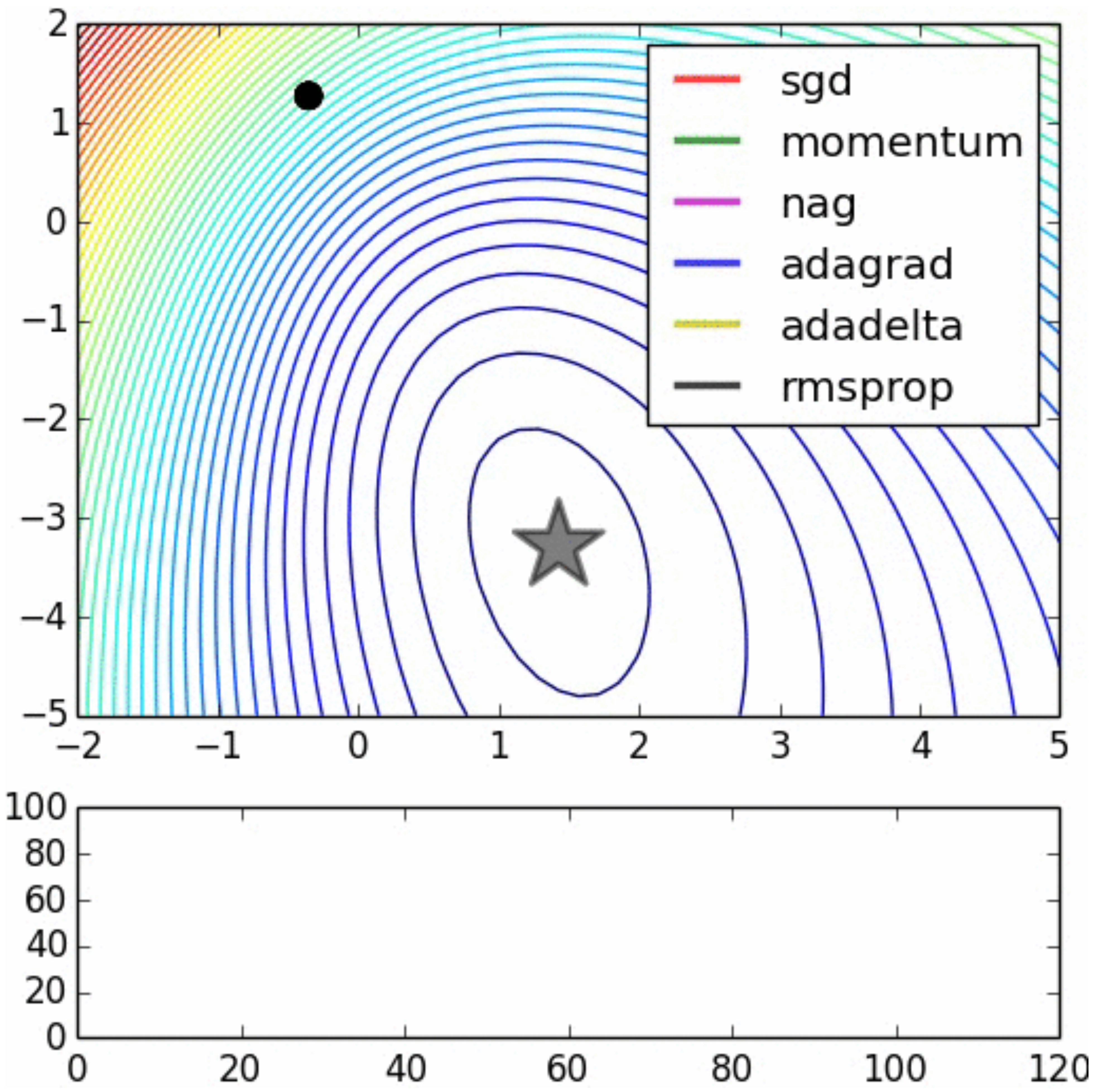
```
x += - learning_rate * dx
```

```
x = mu * v - learning_rate * dx
x += v
```

从物理的惯性角度思考
在平缓的方向上，不断累积速度，
在陡峭的方向上，由于不断变换符号，速度会被抑制。

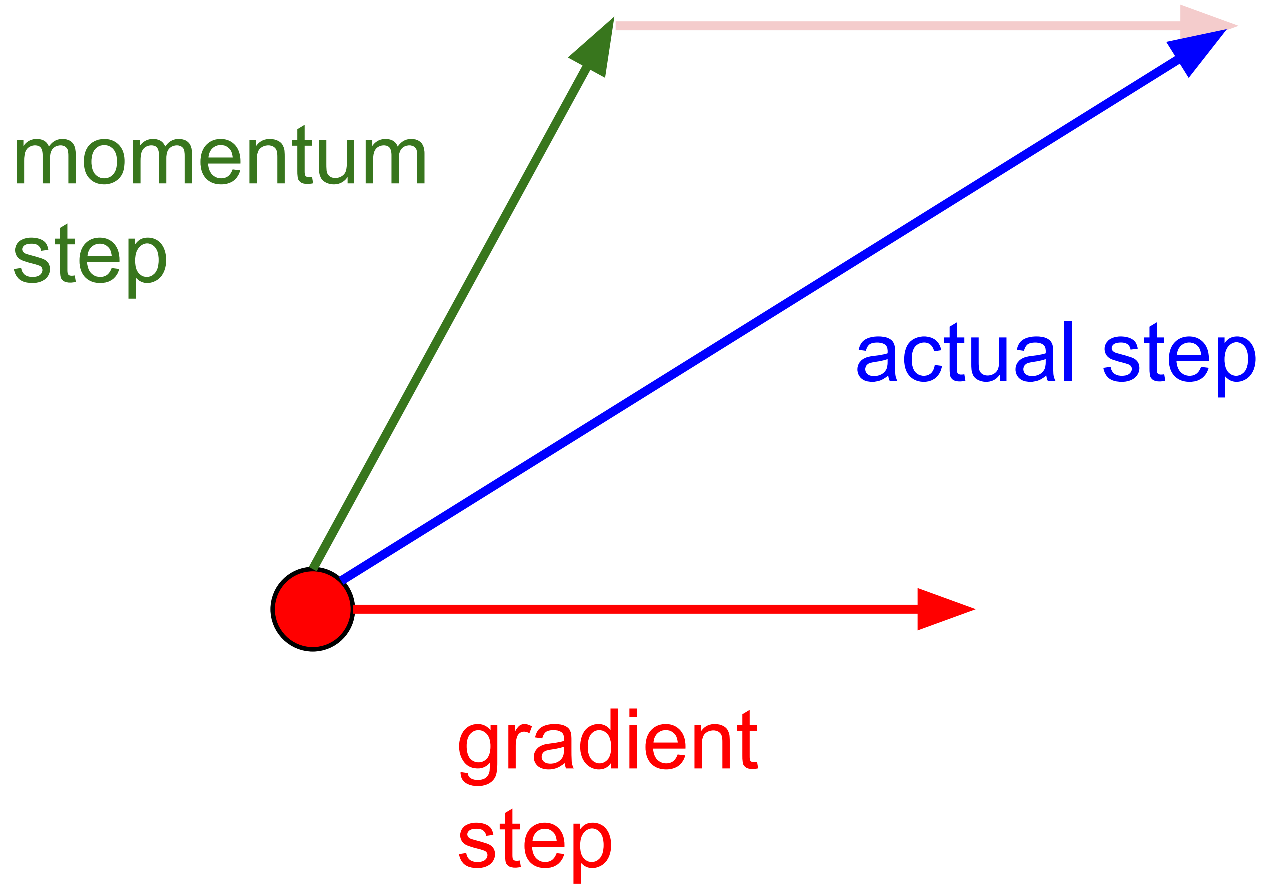
其中，mu=0.5, 0.9, 0.99

SGD vs Momentum

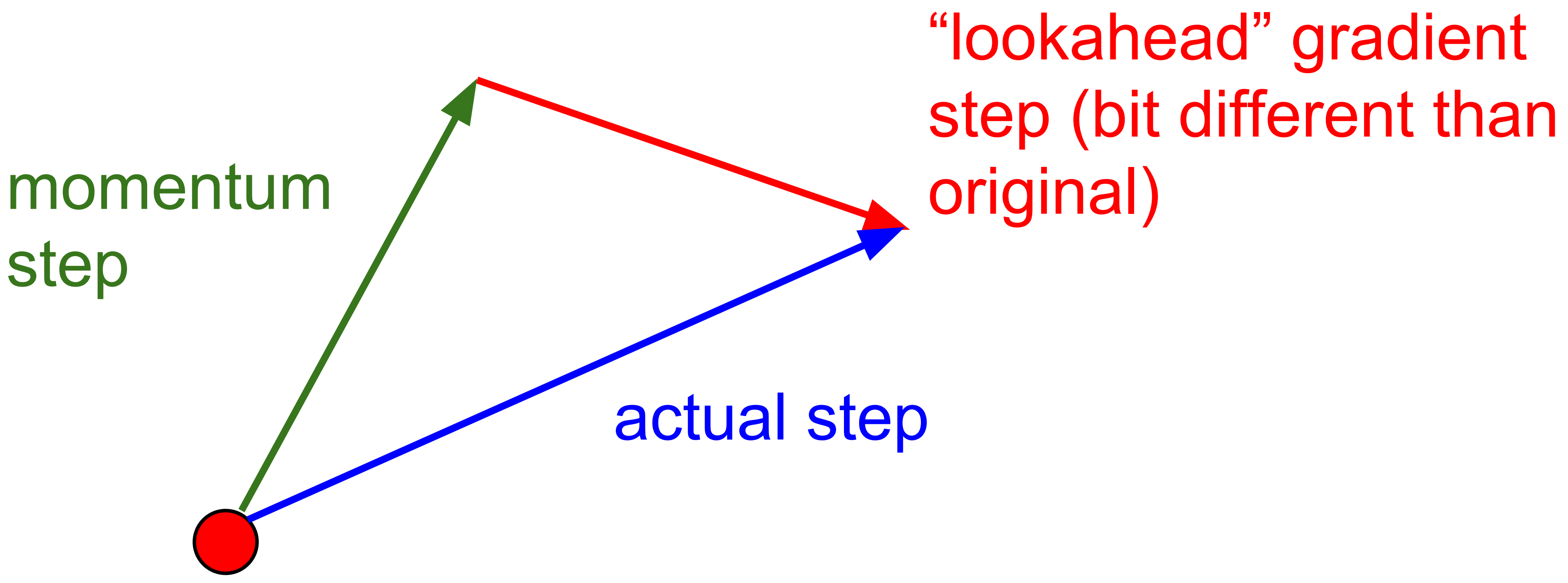


Nesterov Accelerated Gradient

Momentum update



Nesterov momentum update

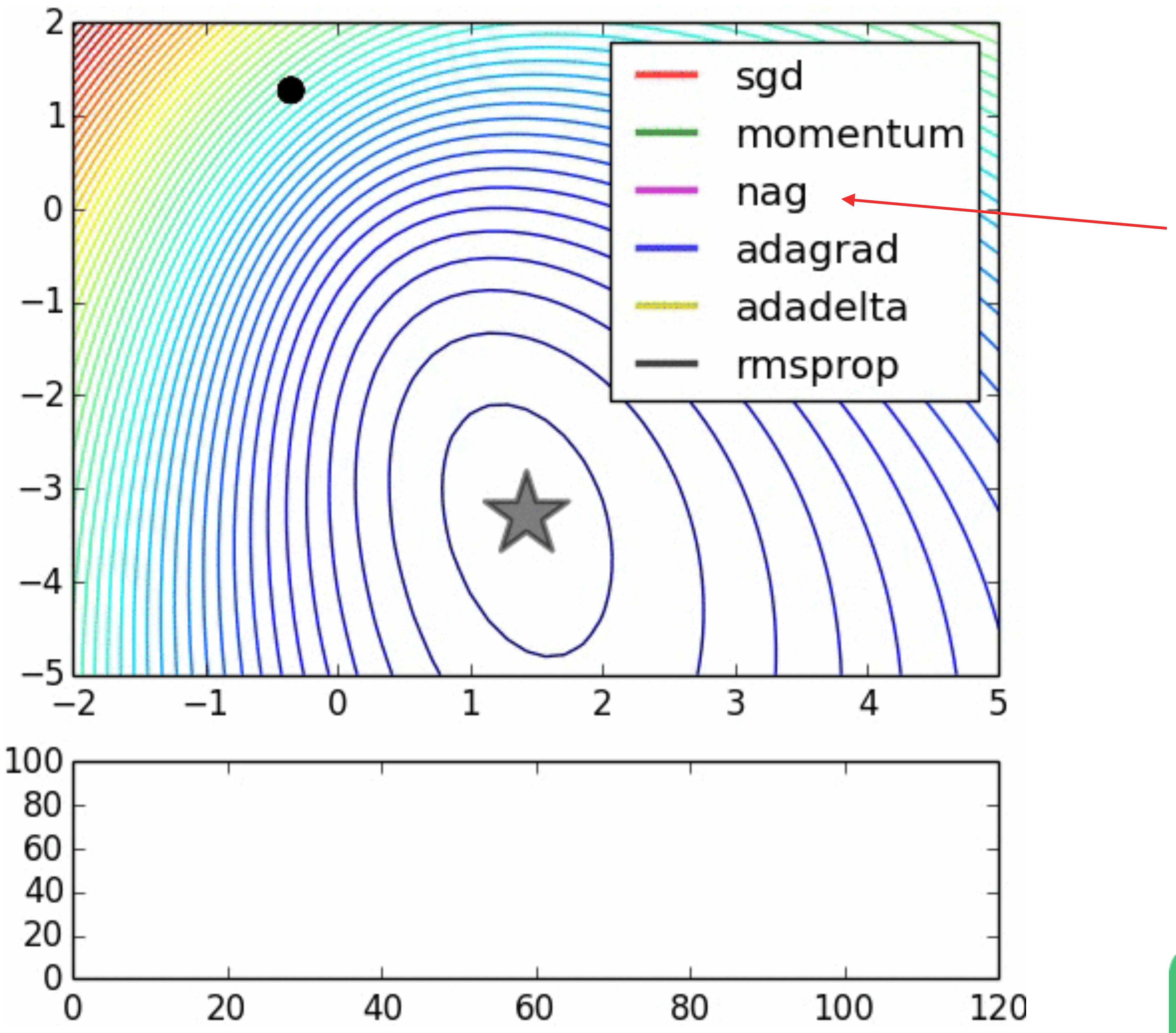


Nesterov Accelerated Gradient

```
x = mu * v - learning_rate * dx
x += v
```

```
v_prev = v
v = mu * v - learning_rate * dx
x += -mu * v_prev + (1 + mu) * v
```

NAG



慕课学院
www.mooc.ai

AdaGrad

```
cache += dx**2
x += - learning_rate * dx / (np.sqrt(cache) + 1e-7)
```

基于每个维度每一个元素的历史平方和，计算learning_rate的调整比例

AdaGrad

```
cache += dx**2
x += - learning_rate * dx / (np.sqrt(cache) + 1e-7)
```

基于每个维度每一个元素的历史平方和，计算learning_rate的调整比例

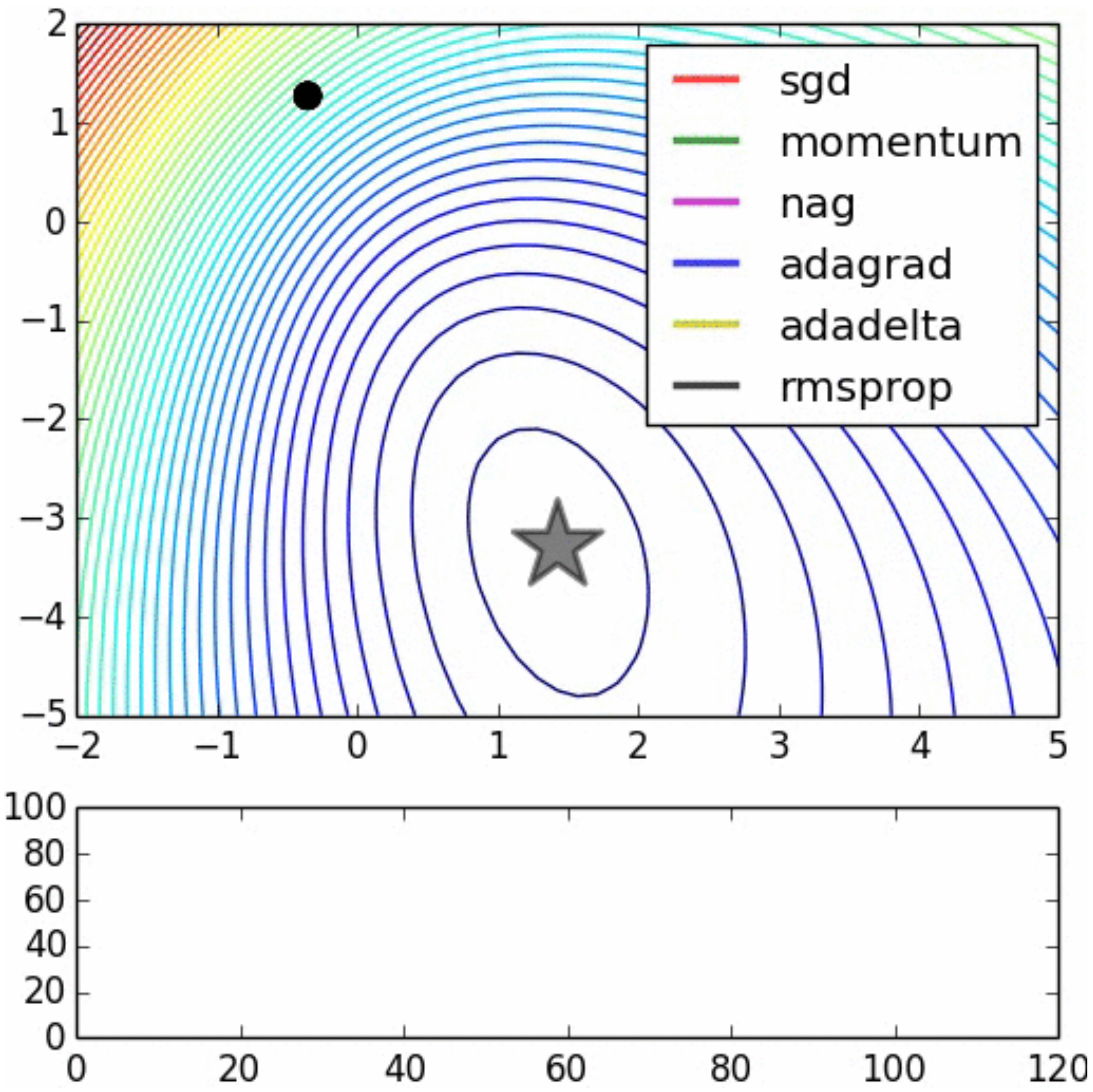
随着训练时间越来越长，每次更新的步长会越来越小。

RMSProp

```
cache += dx**2
x += - learning_rate * dx / (np.sqrt(cache) + 1e-7)
```

```
cache += decay_rate * cache + (1 - decay_rate) * dx**2
x += - learning_rate * dx / (np.sqrt(cache) + 1e-7)
```

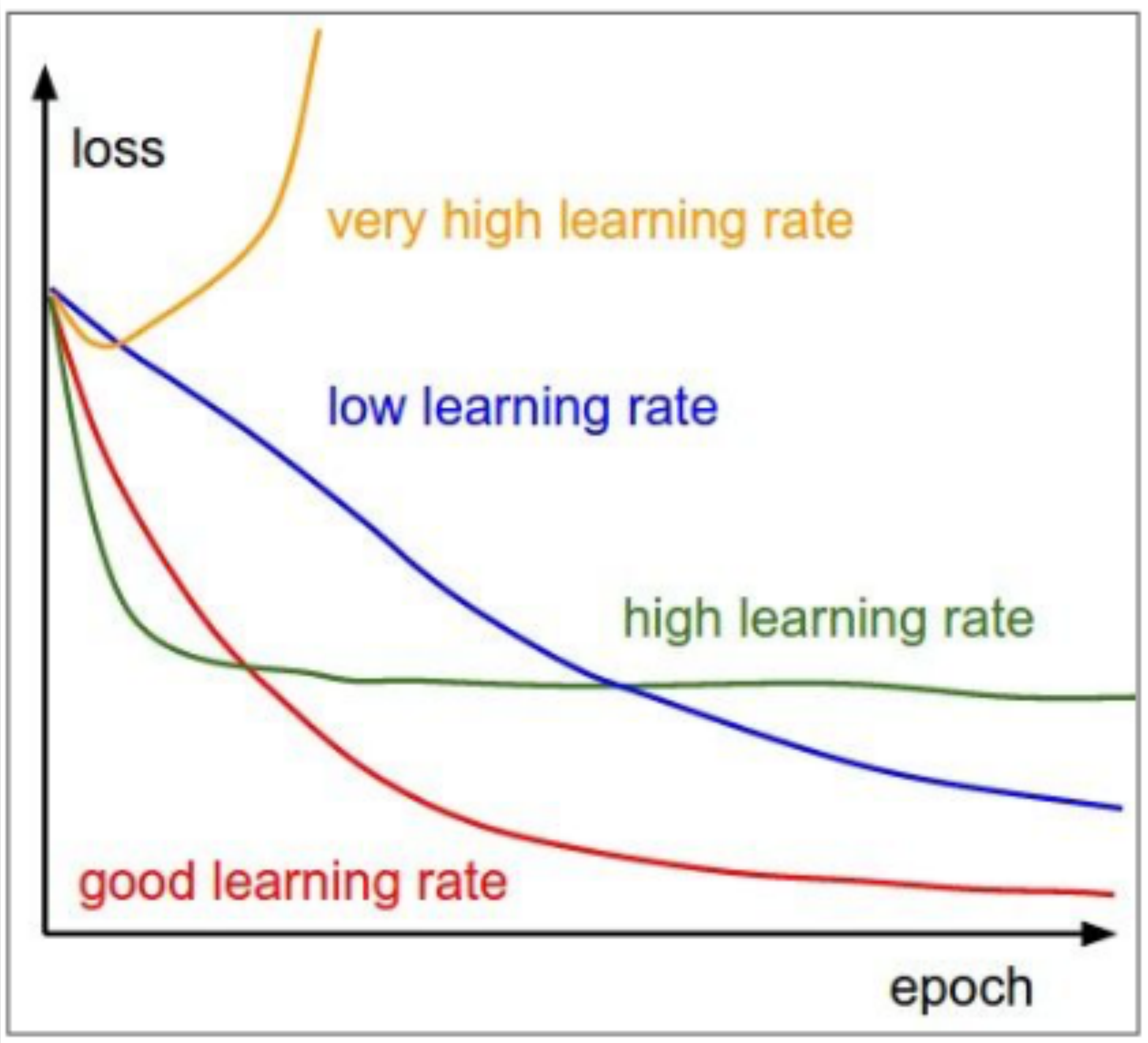
AdaGrad vs RMSProp



Adam Update

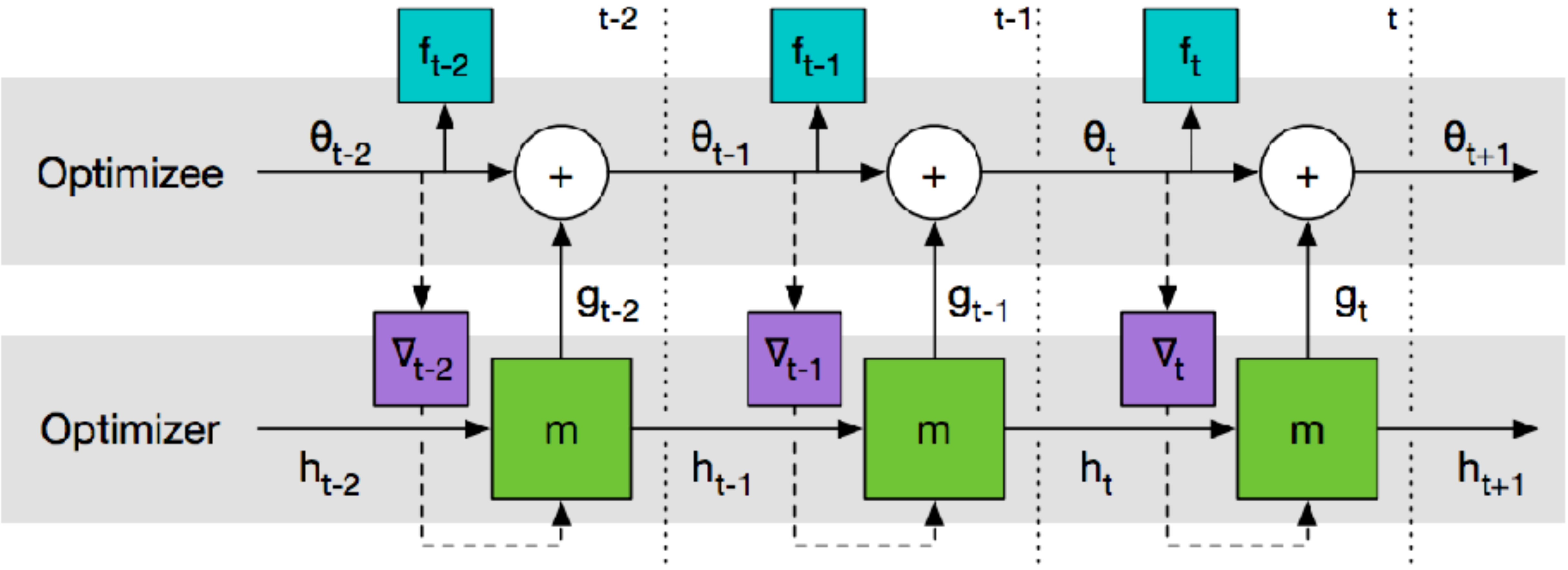
```
m = beta1 * m + (1 - beta1) * dx
v = beta2*v + (1 - beta2) * (dx**2)
mb = m / (1 - beta1**5)
vb = v / (1 - beta2**t)
x += - learning_rate * mb / (np.sqrt(vb) + 1e-7)
```

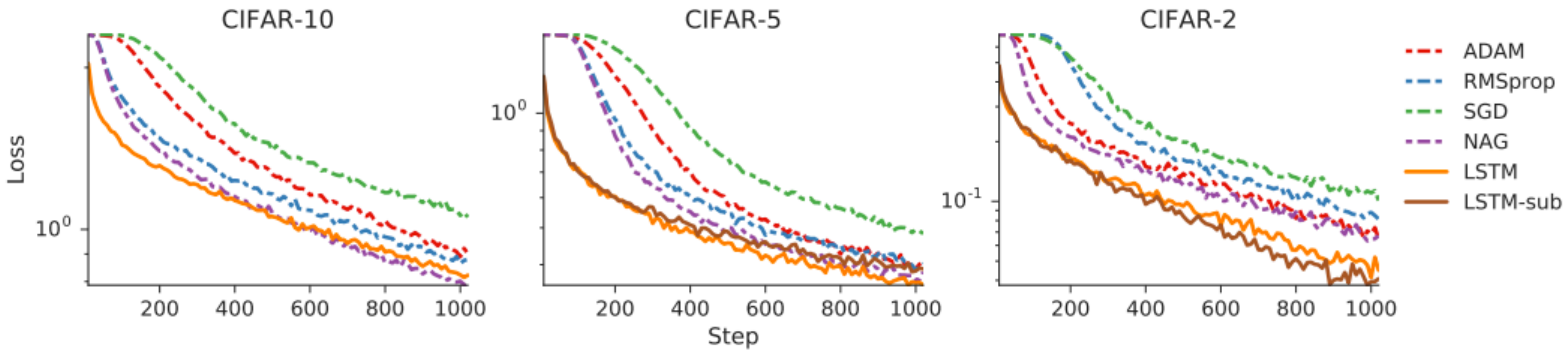
如何选择?



一个好的学习曲线，应该是一开始可以让Loss快速下降，同时在最低点收敛。

Learning to Learn





http://arxiv.org/abs/1606.04474

Optimizer in TensorFlow

Optimizers

The Optimizer base class provides methods to compute gradients for a loss and apply gradients to variables. A collection of subclasses implement classic optimization algorithms such as GradientDescent and Adagrad.

You never instantiate the Optimizer class itself, but instead instantiate one of the subclasses.

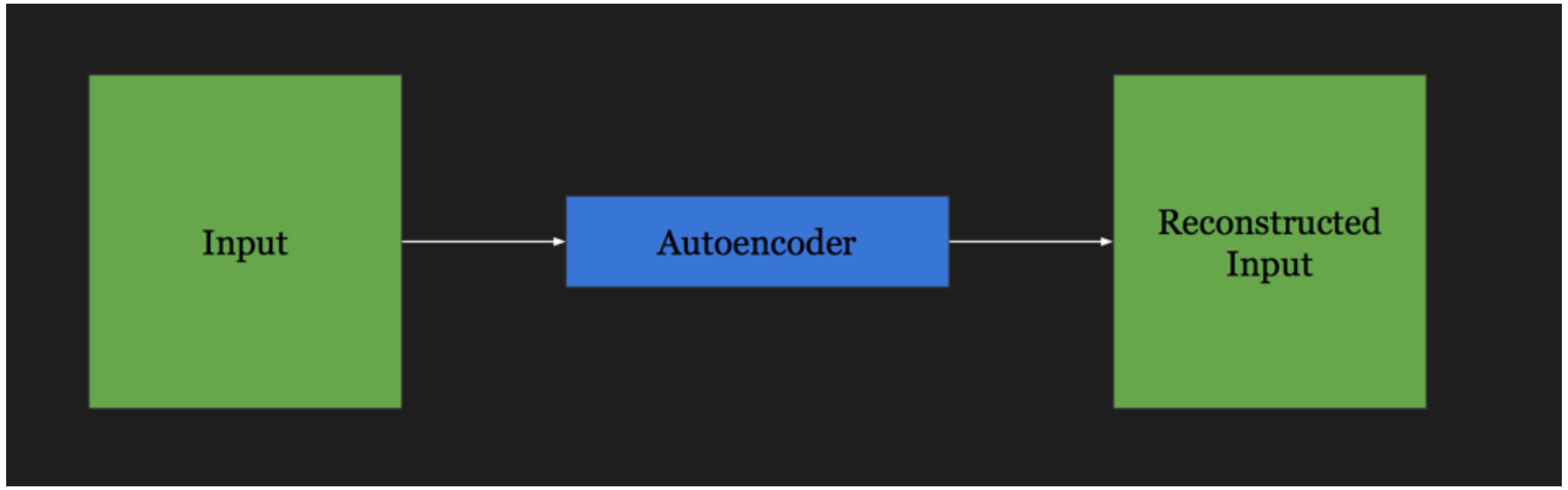
- `tf.train.Optimizer`
- `tf.train.GradientDescentOptimizer`
- `tf.train.AdadeltaOptimizer`
- `tf.train.AdagradOptimizer`
- `tf.train.AdagradDAOptimizer`
- `tf.train.MomentumOptimizer`
- `tf.train.AdamOptimizer`
- `tf.train.FtrlOptimizer`
- `tf.train.ProximalGradientDescentOptimizer`
- `tf.train.ProximalAdagradOptimizer`
- `tf.train.RMSPropOptimizer`

```
optimizer = tf.train.XXOptimizer(learning_rate=0.1).minimize(loss)
```

https://www.tensorflow.org/api_guides/python/train

自编码器（Autoencoder）

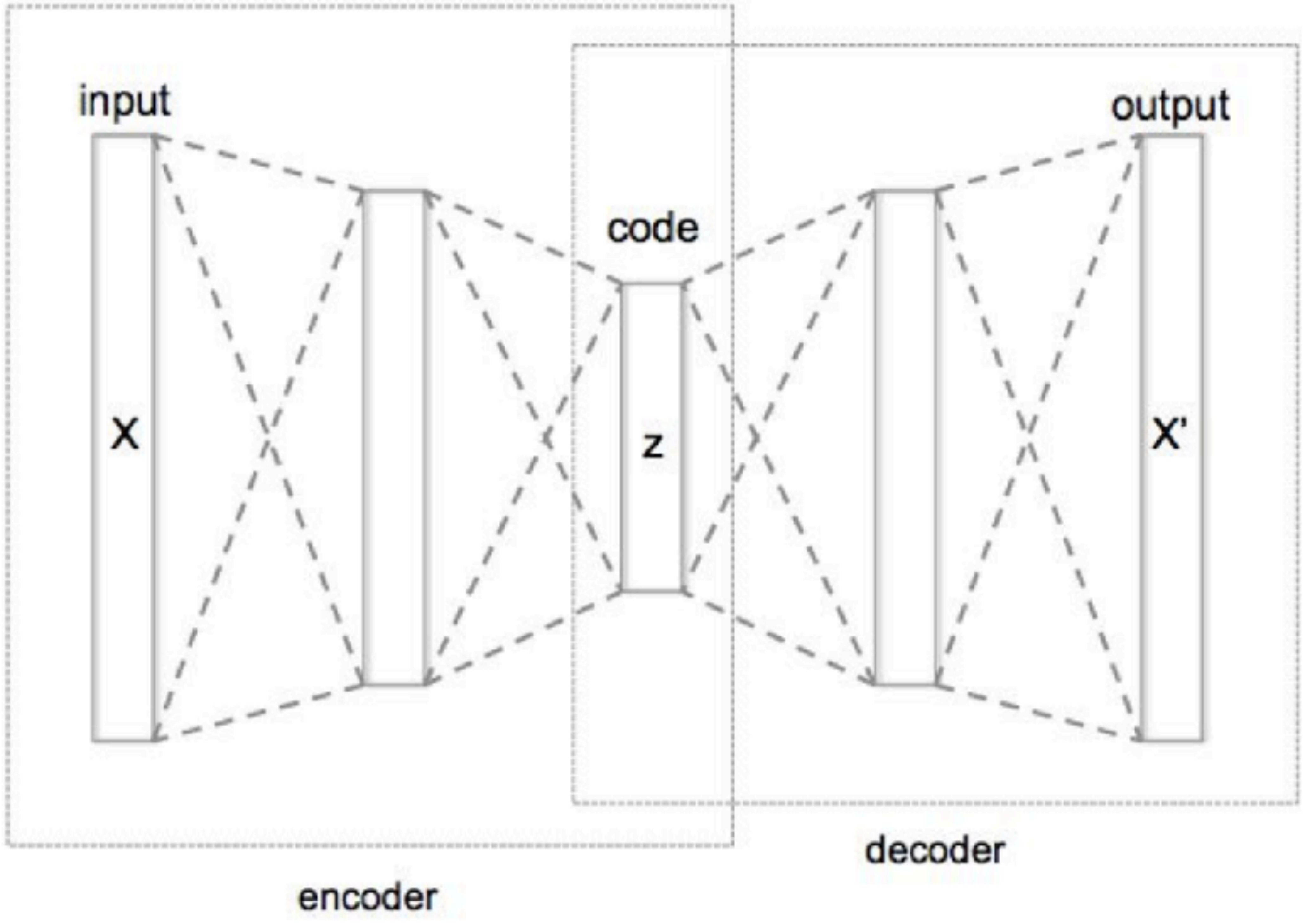
什么是AUTOENCODER



AUTOENCODER

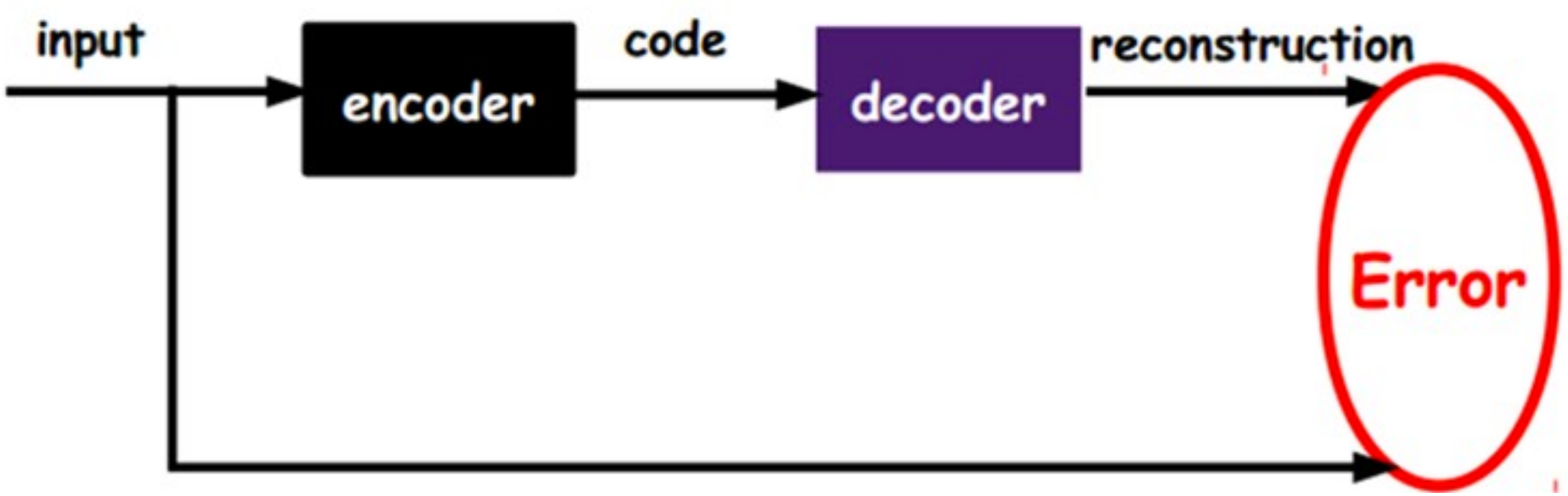
autoencoder就是一种尽可能复现输入信号的神经网络，
autoencoder是无监督学习

AUTOENCODER



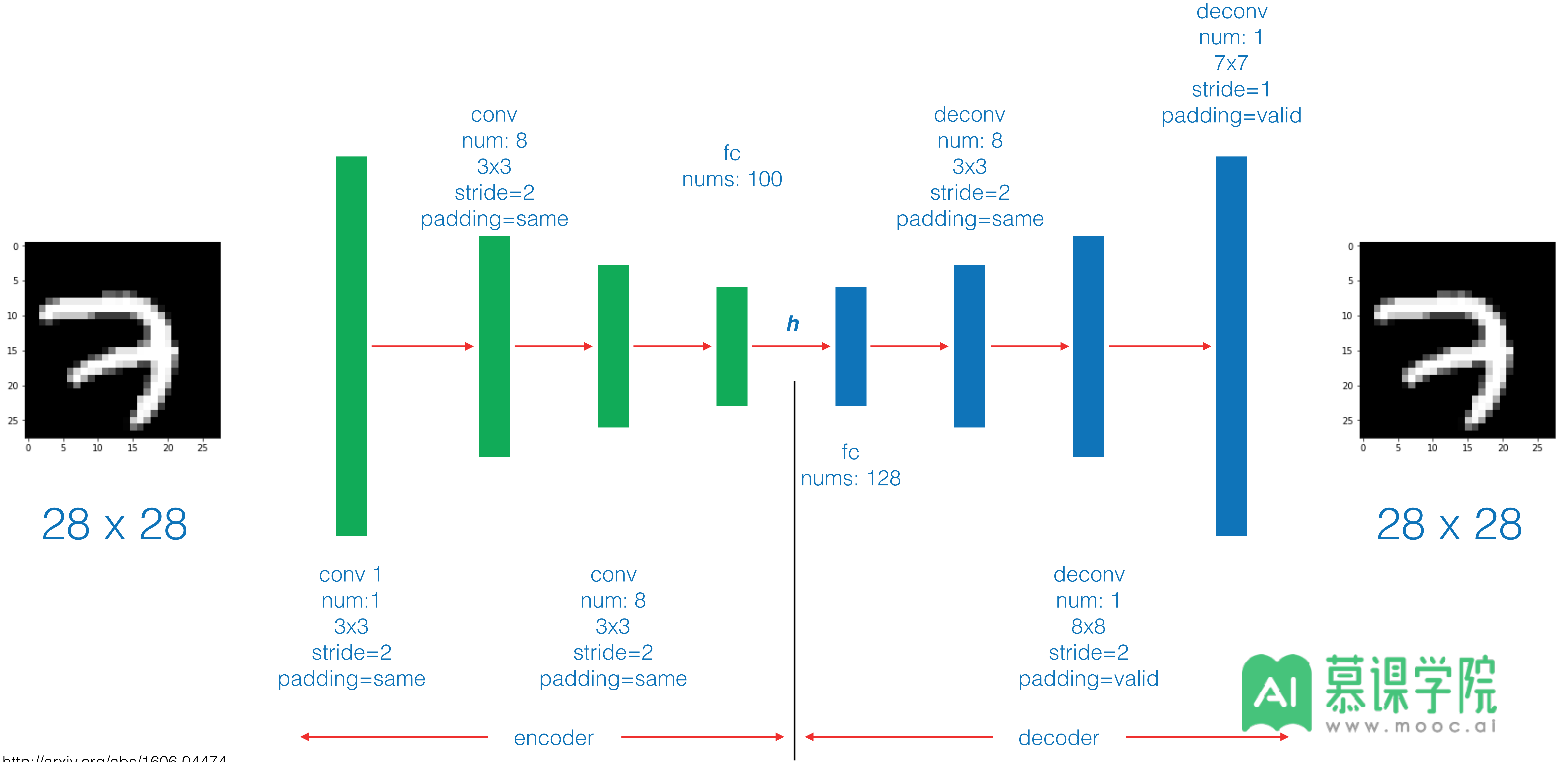
输入和输出的维度应该相同
输入和输出的范围应该相同

AUTOENCODER怎么求误差



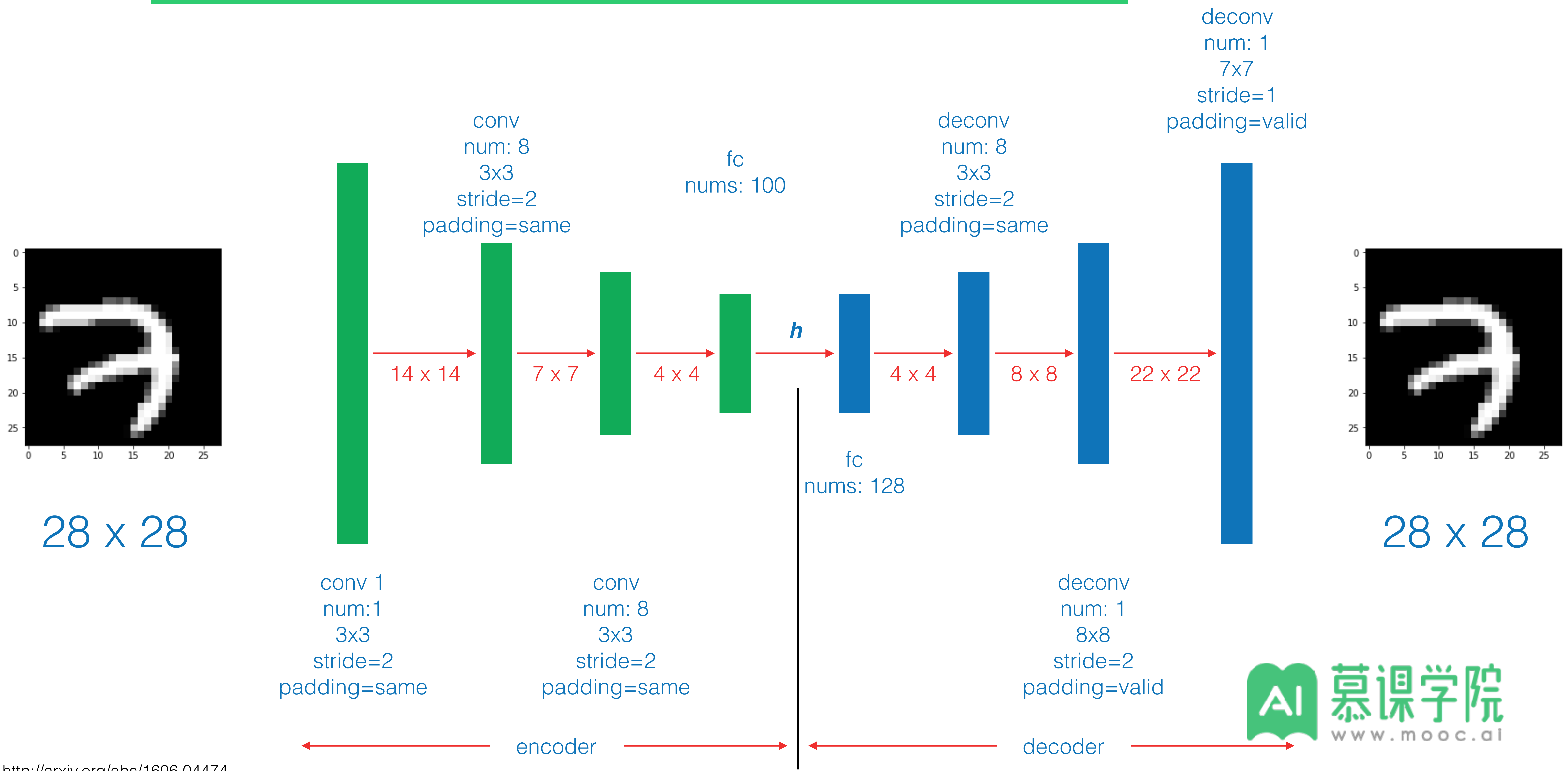
将input输入一个encoder编码器，就会得到一个code，code表示输入，加一个decoder解码器，decoder得到输出，通过通过调整encoder和decoder的参数，使得输入和输出的。

Autoencoder



http://arxiv.org/abs/1606.04474

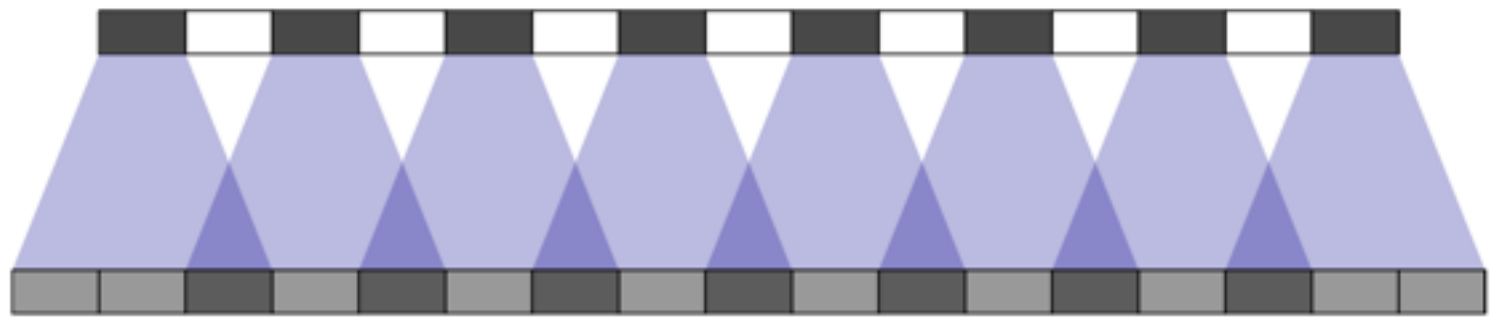
Autoencoder

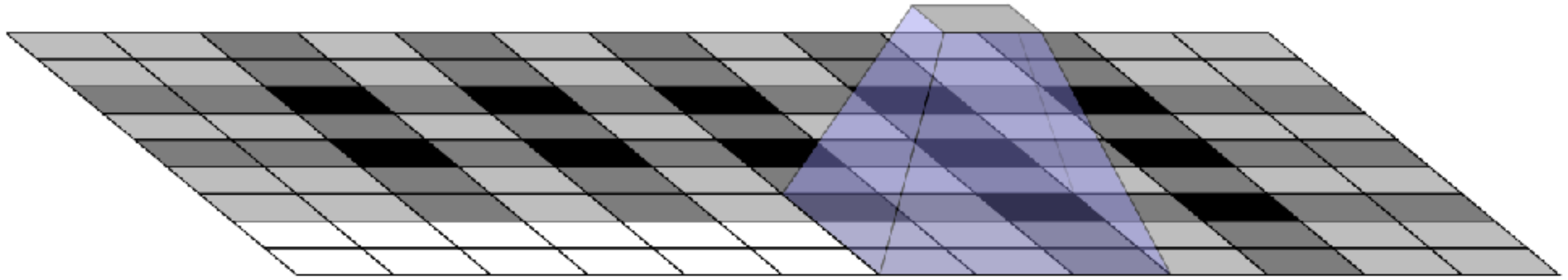


http://arxiv.org/abs/1606.04474

反卷积（Deconvolution）

反卷积





http://distill.pub/2016/deconv-checkerboard/

反卷积

`tf.nn.conv2d_transpose(value, filter, output_shape, strides, padding='SAME', data_format='NHWC', name=None)`

The transpose of `conv2d`.

This operation is sometimes called "deconvolution" after Deconvolutional Networks, but is actually the transpose (gradient) of `conv2d` rather than an actual deconvolution.

Args:

- `value`: A 4-D `Tensor` of type `float` and shape `[batch, height, width, in_channels]` for `NHWC` data format or `[batch, in_channels, height, width]` for `NCHW` data format.
- `filter`: A 4-D `Tensor` with the same type as `value` and shape `[height, width, output_channels, in_channels]`. `filter`'s `in_channels` dimension must match that of `value`.
- `output_shape`: A 1-D `Tensor` representing the output shape of the deconvolution op.
- `strides`: A list of ints. The stride of the sliding window for each dimension of the input tensor.
- `padding`: A string, either `'VALID'` or `'SAME'`. The padding algorithm. See the comment here
- `data_format`: A string. 'NHWC' and 'NCHW' are supported.
- `name`: Optional name for the returned tensor.

```
def get_deconv2d_output_dims(input_dims, filter_dims, stride_dims, padding):
    # Returns the height and width of the output of a deconvolution layer.
    batch_size, input_h, input_w, num_channels_in = input_dims
    filter_h, filter_w, num_channels_out = filter_dims
    stride_h, stride_w = stride_dims

    # Compute the height in the output, based on the padding.
    if padding == 'SAME':
        out_h = input_h * stride_h
    elif padding == 'VALID':
        out_h = (input_h - 1) * stride_h + filter_h

    # Compute the width in the output, based on the padding.
    if padding == 'SAME':
        out_w = input_w * stride_w
    elif padding == 'VALID':
        out_w = (input_w - 1) * stride_w + filter_w

    return [batch_size, out_h, out_w, num_channels_out]
```


http://distill.pub/2016/deconv-checkerboard/

Live Coding

Q&A

https://github.com/tongda/leifeng_tf_course

谢谢

如果您有任何问题和建议

请联系
dtong@thoughtworks.com

ThoughtWorks®