یادگیری ماشین

بخش دوم

مروری بر روش های یادگیری ماشین
آغاز کار با Tensorflow

محمد حسن بشری موحد
زمستان ۹۷


بسم
الله
الرحمن
الرحيم

کلمات کلیدی این بخش

- یادگیری بدون نظارت (Unsupervised Learning)
- یادگیری بانظارت (Supervised Learning)
- یادگیری تقویتی (Reinforcement Learning)
- تابع هزینه (Cost function)
- گرادیان نزولی (Gradient Descent)
- گراف محاسباتی (Calculation Graph)
- رگرسیون خطی (Linear Regression)

فهرست مطالب

چرا (چه وقت) از یادگیری ماشین استفاده کنیم/نکنیم؟

- هزینه‌ی تنظیم دستی قوانین(به کار گیری روش های فرمال) بیشتر از جمع‌آوری داده و توسعه‌ی الگوریتم یادگیری ماشین باشد.
  - اگر داده‌ی کافی قابل جمع‌آوری نیست از یادگیری ماشین صرف نظر کنید!
  - به‌طور کلی جمع‌آوری،پاک‌سازی و پیش پردازش داده **۶۰ الی ۷۰ درصد** فرآیند کاری را در بر می‌گیرد.
  - مابقی به بهینه سازی مدل اختصاص دارد
  - آموزش مدل هم **صفر** درصد.
  - کافی بودن داده تابع الگوریتم یادگیری است.
- دلیل اصلی متن باز بودن کتابخانه‌های یادگیری ماشین
- باید دنبال روشی باشیم که با بضاعت خودمان بتوانیم برای آن داده تهیه کنیم!
- متخصص یادگیری ماشین همان قدر که متخصص الگوریتم است متخصص داده نیز هست

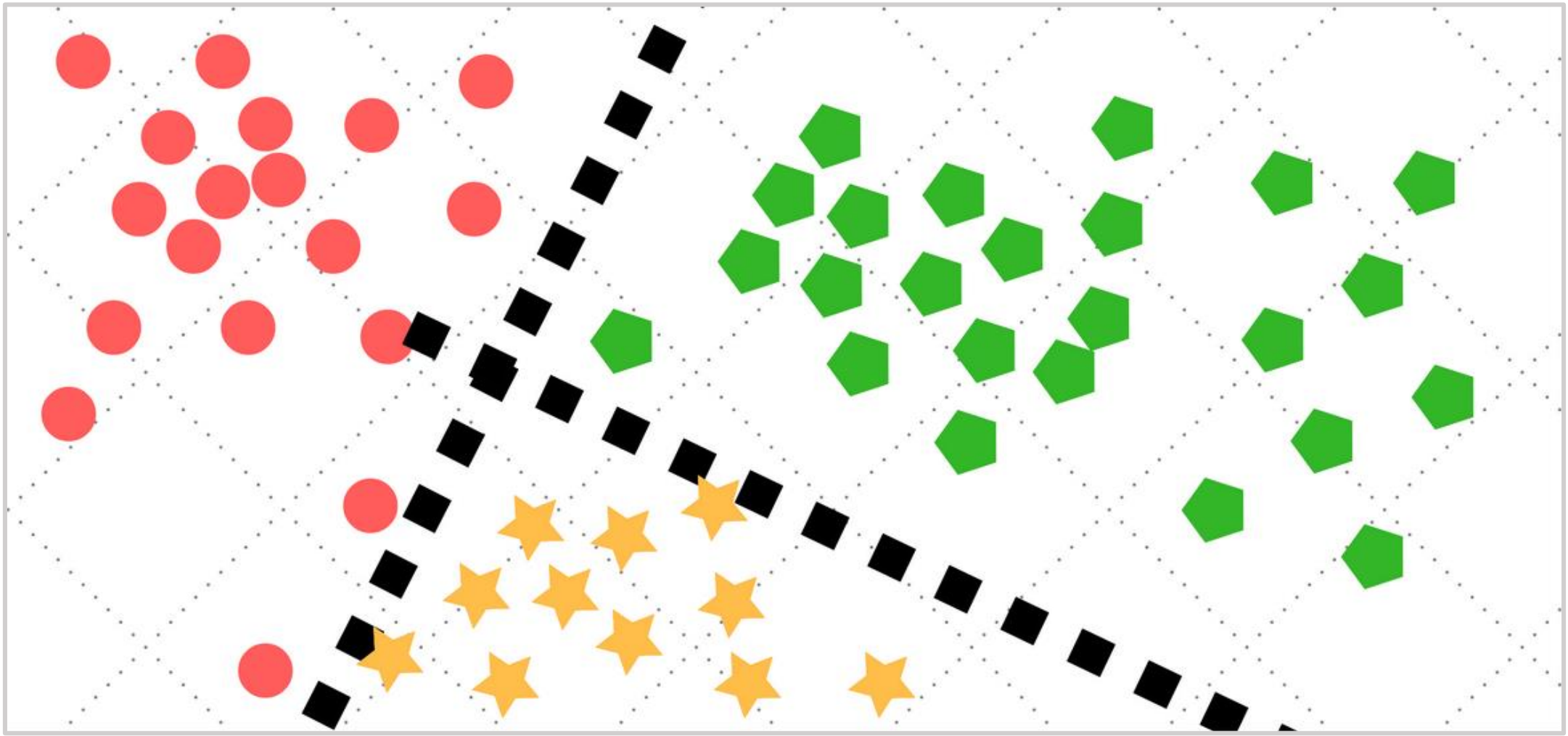
انواع روش‌های یادگیری ماشین

1. یادگیری بانظارت
2. یادگیری بدون نظارت
3. یادگیری تقویتی
4. تقسیم بندی‌های فرعی
    I. یادگیری نیمه نظارت شده
    II. یادگیری فعال
    III. یادگیری برخط
    IV. یادگیری خود نظارتی

یادگیری بانظارت

1. ورودی داده است
2. خروجی داده است.
3. **هدف:** پیدا کردن تابعی (مدلی) است که داده‌ی ورودی را بگیرد و داده‌ی خروجی متناسب با آن را تولید کند.
4. **شامل دو بخش اصلی:**
    1. طبقه بندی | رده بندی
    2. رگرسیون

طبقه‌بندی | رده‌بندی (Classification)



یادگیری بدون نظارت

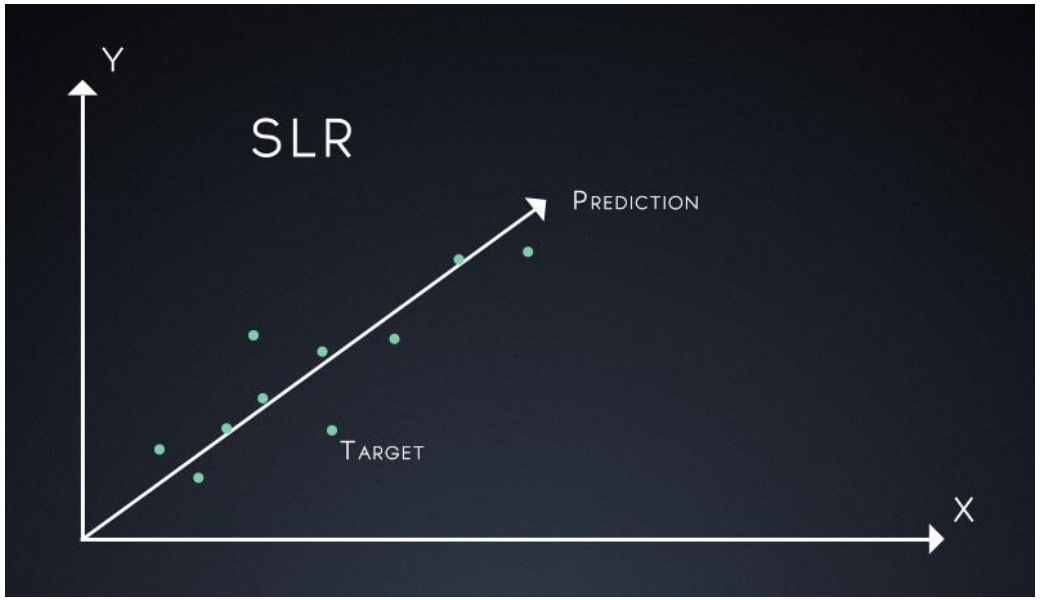
1. ورودی داده است
2. خروجی داده نیست. خروجی یک بازنمایی از داده‌ها یا برقراری یک ساختار بین آن ها است.
3. **هدف**: بیان از جدید از داده
4. **شامل**:
   - خوشه بندی
   - یادگیری بازنمایی
   - کاهش ابعاد

یادگیری تقویتی

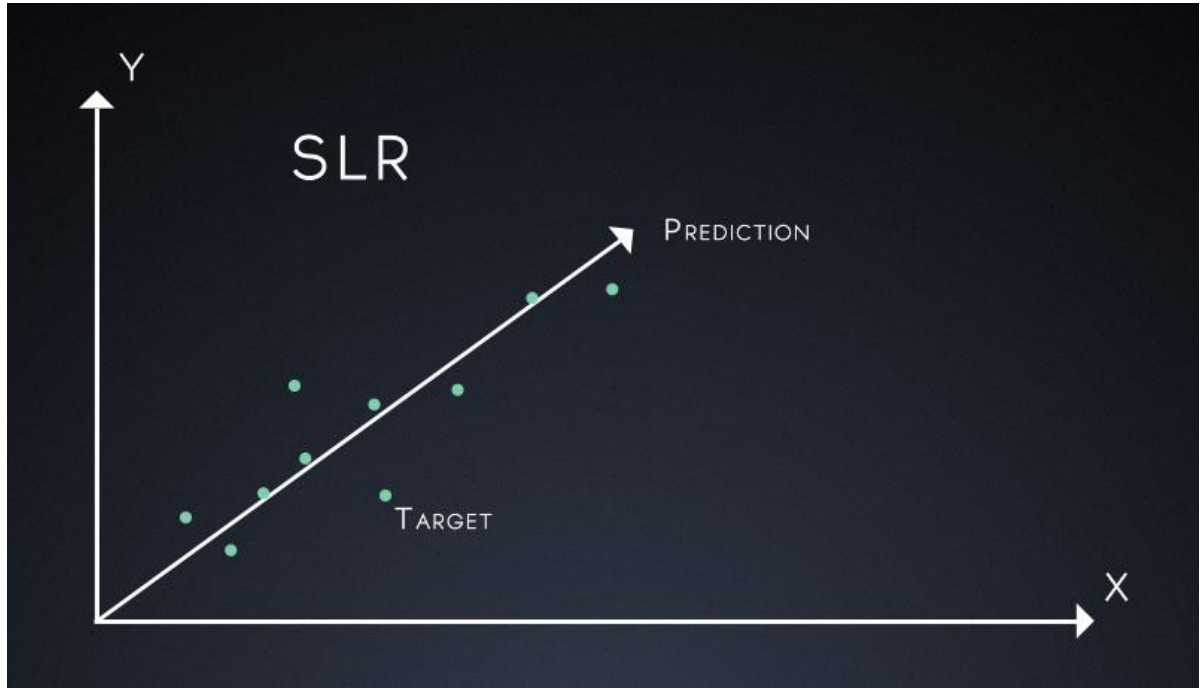
1. بر سه اساس بنا شده است:
    1. وجود موجودیتی به نام **محیط**
    2. تاثیر عمل عامل بر روی محیط
    3. عدم تبیین ویژگی های عمل موفق به صورت دقیق و بیان آن به صورت امتیازی
2. **هدف**:
    1. توسعه‌ی عامل هایی که از تعامل با محیط خودشان و امتیازی که می‌گیرند دنباله‌ی تصمیمات خودشان را اصلاح می‌کنند.

رگرسیون خطی

- یک روش با ناظر به حساب می‌آید
- تخمین متغیر Y از روی X با فرض خطی بودن رابطه‌ی بین این دو
- $y = \beta_1 X + \beta_0$



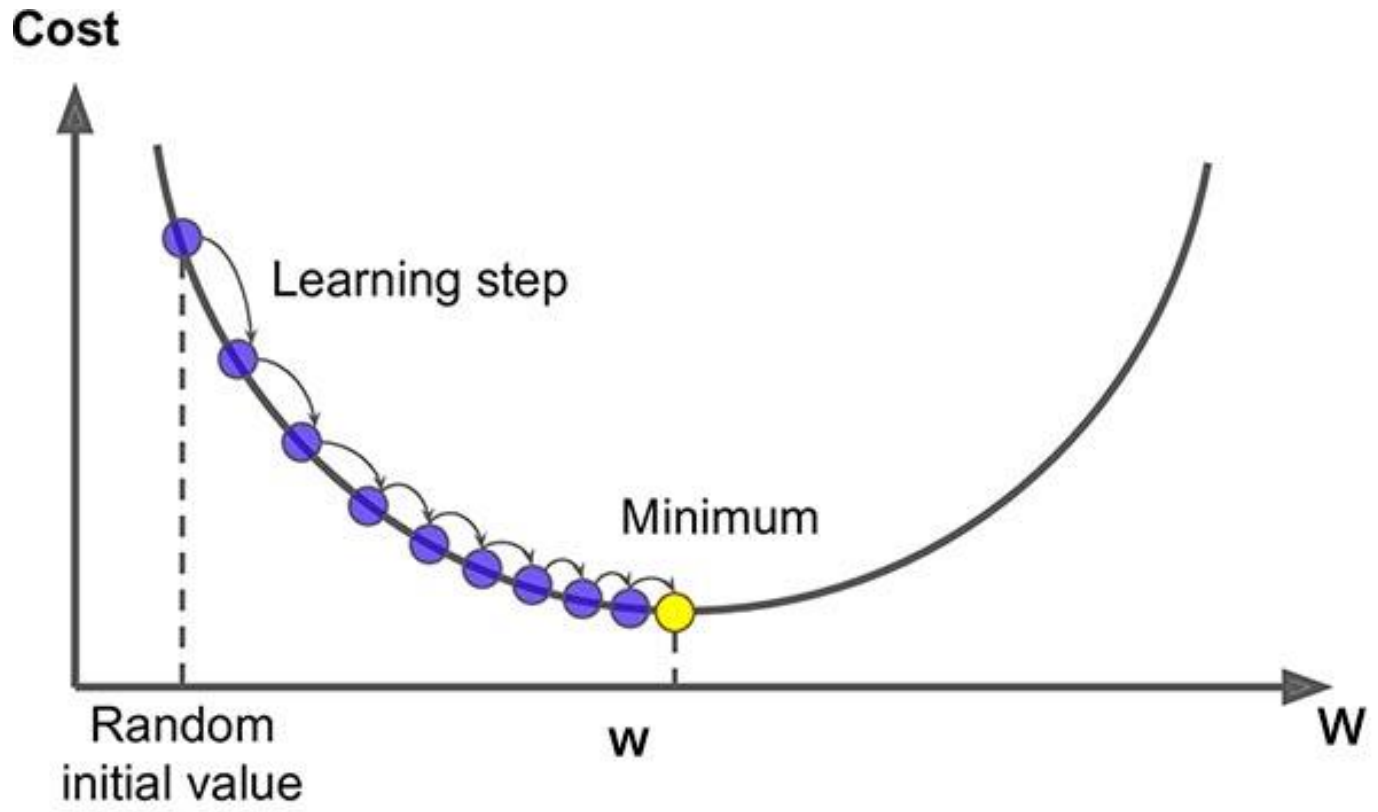
رگرسیون خطی به روش کلاسیک (آماری)



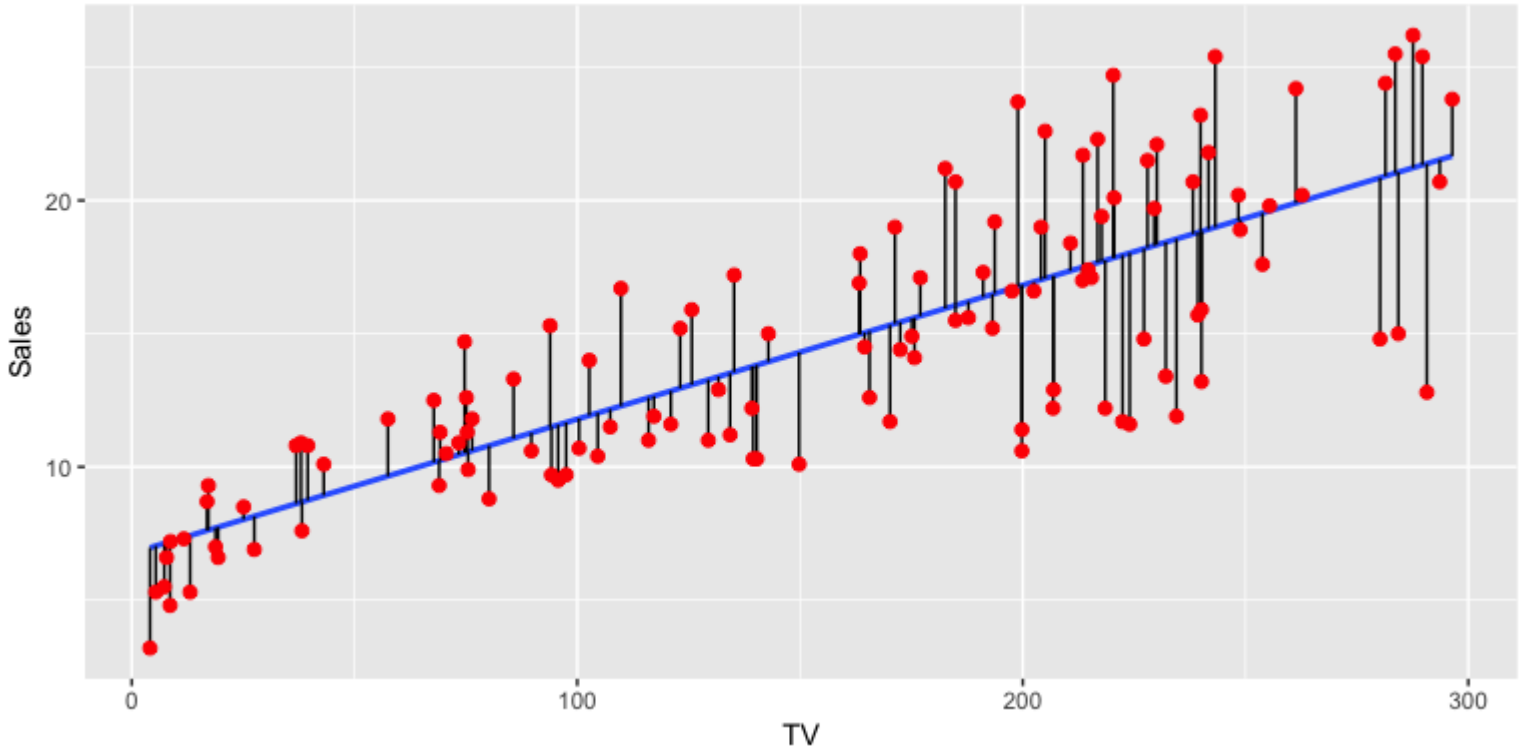
$$\widehat{\beta_1} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \widehat{\beta_1}\bar{x}$$

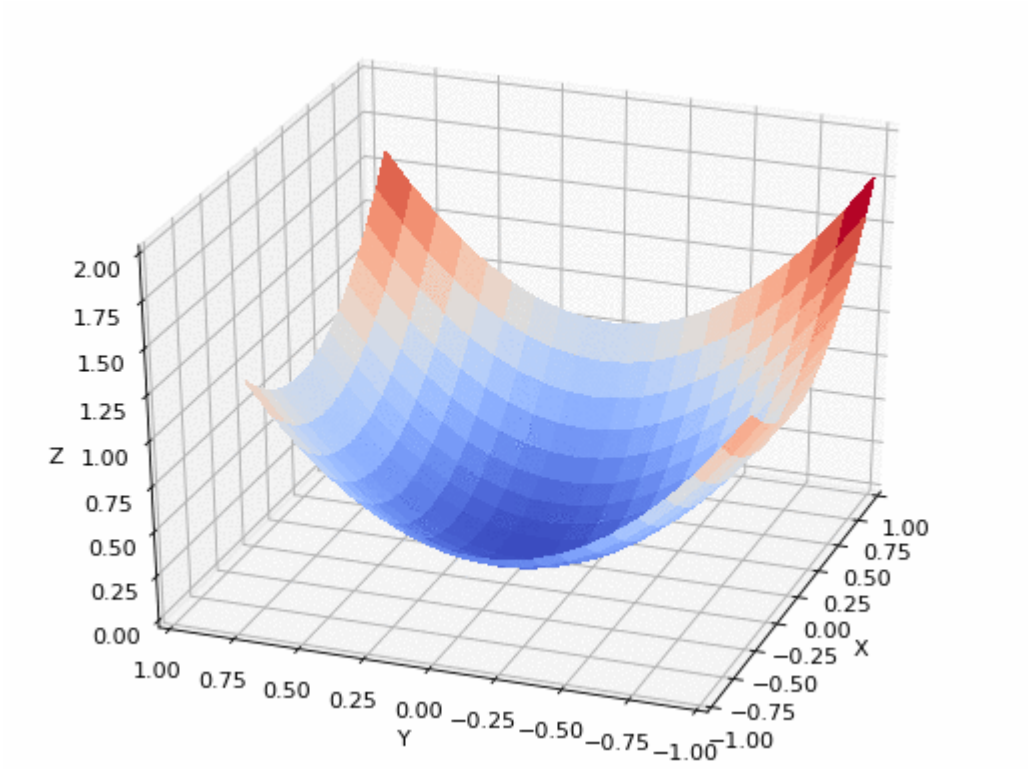
گرادیان نزولی برای بیشینه کردن تابع



تابع هزینه

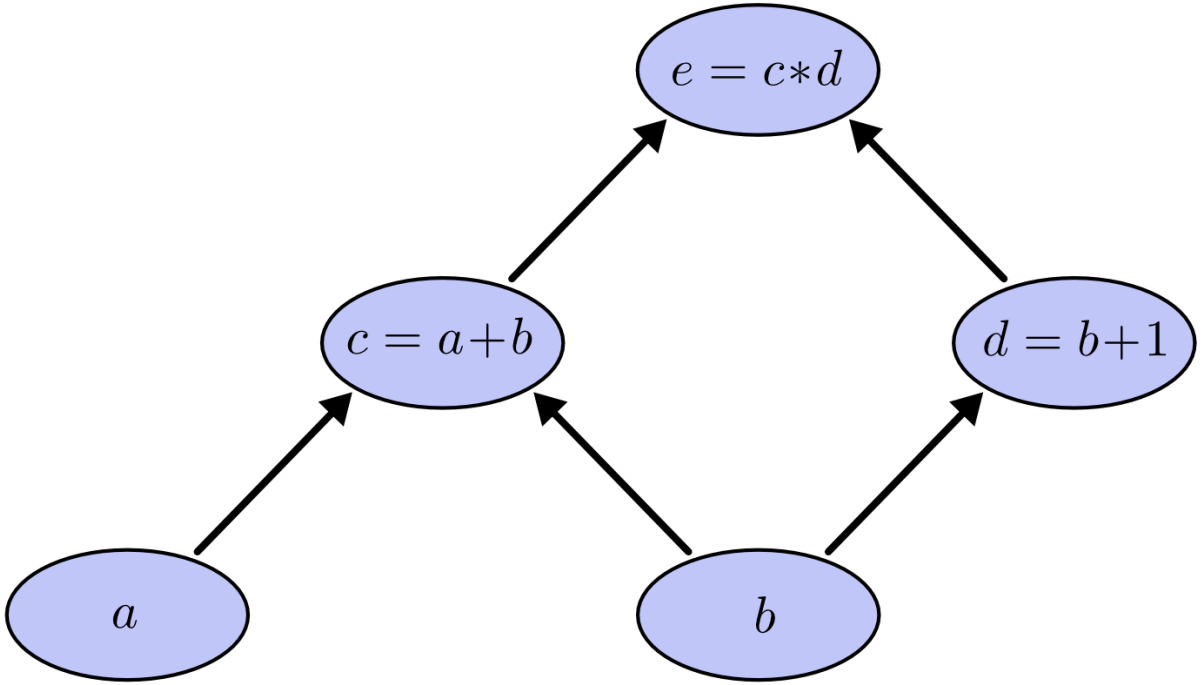


تابع هزینه-ادامه



$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta).$$

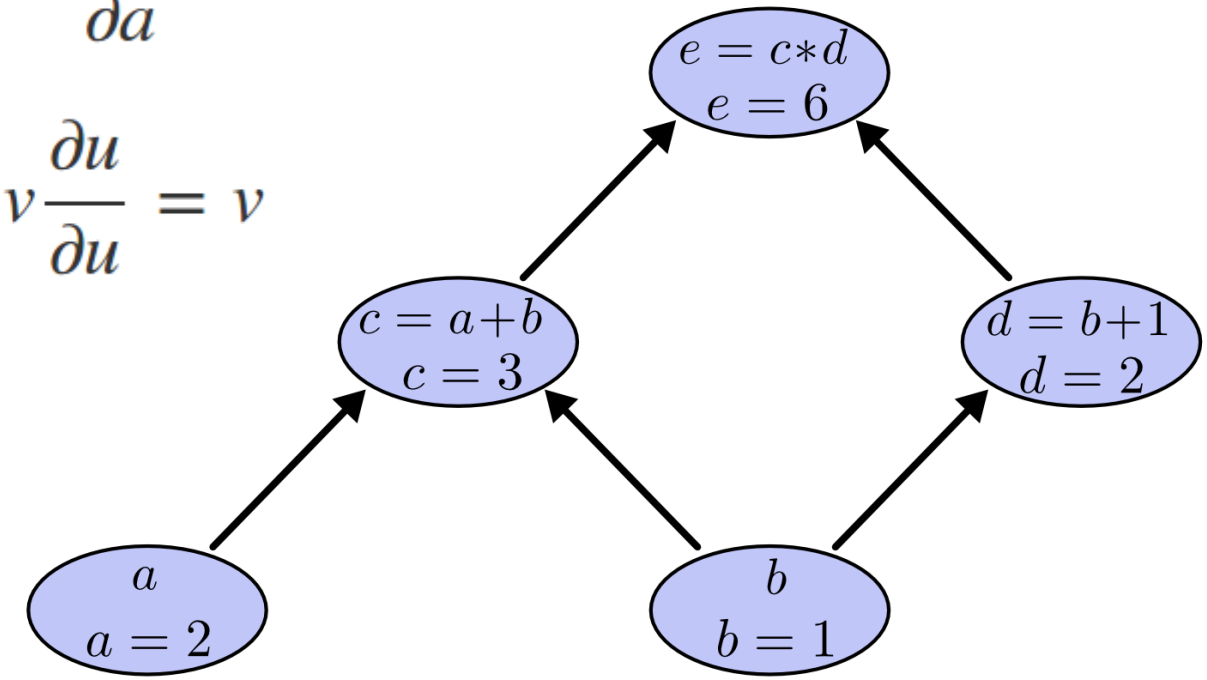
گراف محاسباتی



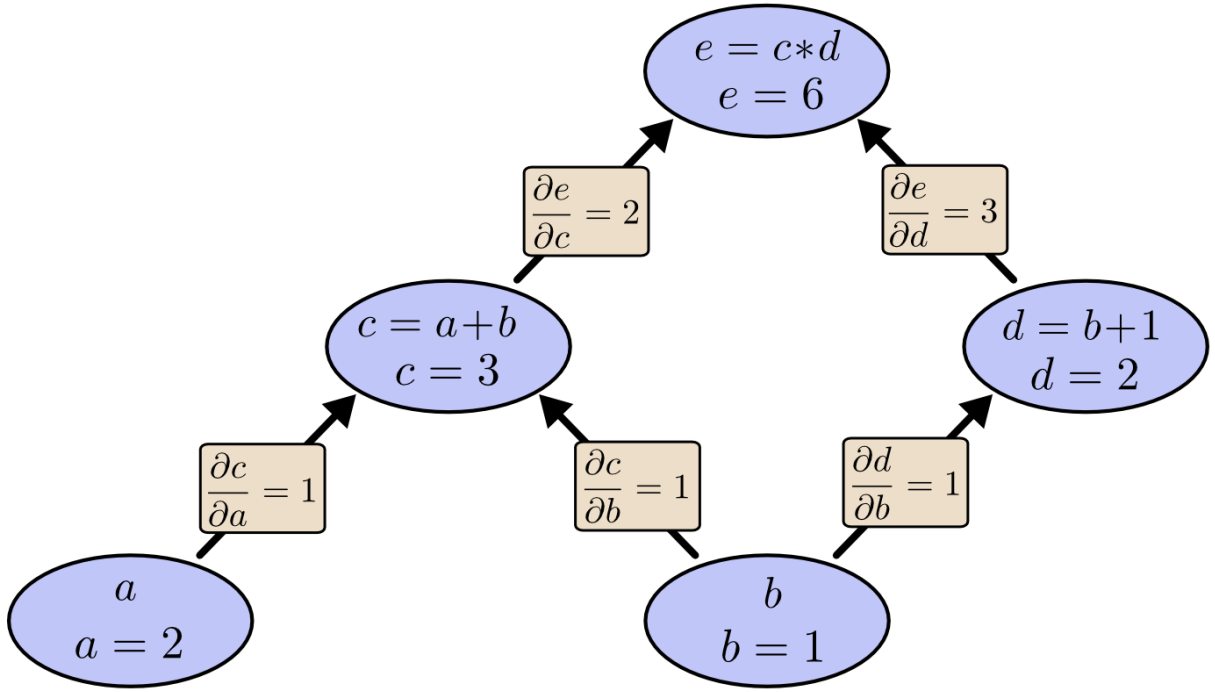
گراف محاسباتی-ادامه

$$\frac{\partial}{\partial a}(a+b) = \frac{\partial a}{\partial a} + \frac{\partial b}{\partial a} = 1$$

$$\frac{\partial}{\partial u}uv = u\frac{\partial v}{\partial u} + v\frac{\partial u}{\partial u} = v$$

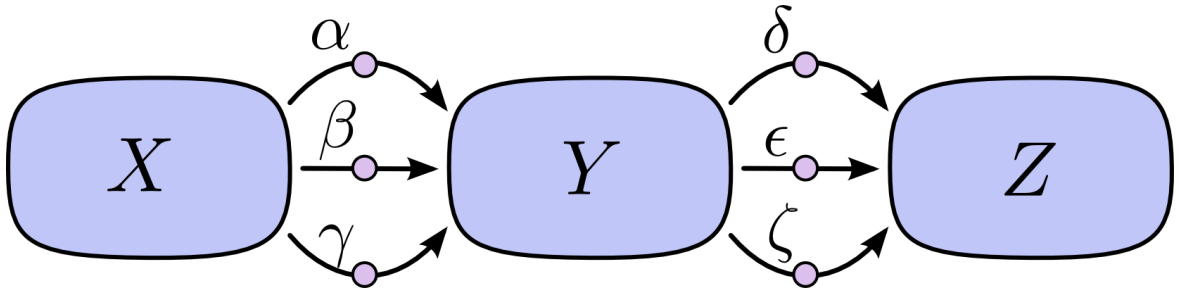


گراف محاسباتی-ادامه



گراف محاسباتی-ادامه

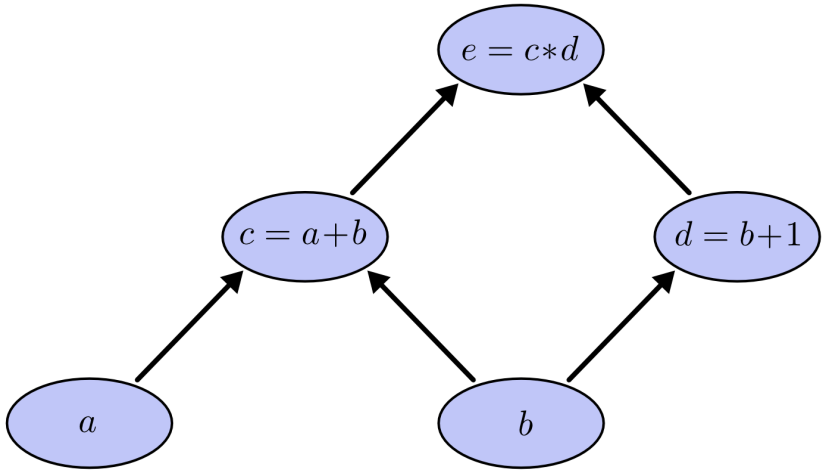
- به طور کلی دو قانون ساده برای مشتق گیری بر روی گراف محاسباتی وجود دارد.
  - مشتق های موازی با هم جمع میشوند
  - مشتق های سری در هم ضرب می شوند.



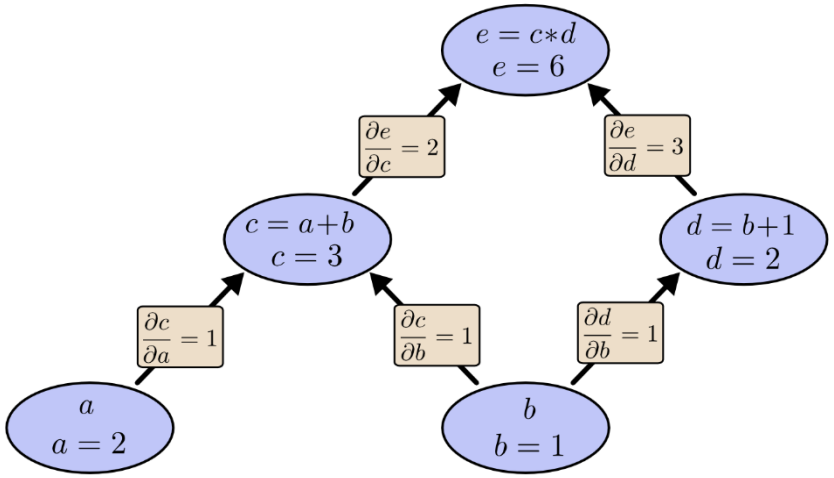
$$\frac{\partial Z}{\partial X} = \alpha\delta + \alpha\epsilon + \alpha\zeta + \beta\delta + \beta\epsilon + \beta\zeta + \gamma\delta + \gamma\epsilon + \gamma\zeta$$

گراف محاسباتی-ادامه

- آن چه ما اعلان می کنیم:



- آن چه تنسورفلو میسازد:



تنسورفلو (tf)

با اعلان پیچیده ترین گراف محاسباتی، عملیات مشتق گیری را به راحتی انجام می‌دهد.

محاسبه مشتق با تنسورفلو

In [1]:
```
import tensorflow as tf
```

In [2]:
```
def get_gradient(fx):
    opt = tf.train.GradientDescentOptimizer(0.1)
    grads = opt.compute_gradients(fx)
    sess = tf.Session()
    sess.run(tf.global_variables_initializer())
    grad_vals = sess.run([grad[0] for grad in grads])
    return grad_vals
```

In [3]:
```
point = 3.0
x = tf.Variable(float(point))
g_x = x * x
fog_x = g_x * g_x
print(get_gradient(fog_x))  # [108.0]
```

```
[108.0]
```

بخش‌های اصلی یک برنامه‌ی مبتنی بر tf

```
def fit(self):
    x = tf.placeholder("float")
    y = tf.placeholder("float")

    a = tf.Variable(1.0, name="weight")
    b = tf.Variable(1.0, name="bias")

    pred = tf.multiply(x, a) + b

    cost = tf.reduce_mean(tf.abs(pred - y))

    optimizer = tf.train.GradientDescentOptimizer(self.learning_rate).minimize(cost)

    init = tf.initialize_all_variables()

    with tf.Session() as sess:
        sess.run(init)
        for epoch in range(self.training_epochs):
            for i, out in zip(self.train_X, self.train_Y):
                sess.run(optimizer, feed_dict={x: i, y: out})
                print("Epoch:", '%04d' % (epoch + 1), "cost=", "W=", sess.run(a), "b=", sess.run(b))
        print("Optimization Finished!")
        training_cost = sess.run(cost, feed_dict={x: self.train_X, y: self.train_Y})
        print("Training cost=", training_cost, "a=", sess.run(a), "b=", sess.run(b), '\n')
        return sess.run(a), sess.run(b)
```

Placeholders

Variables

Model Structure

Optimization Section

Training Section