یادگیری عمیق

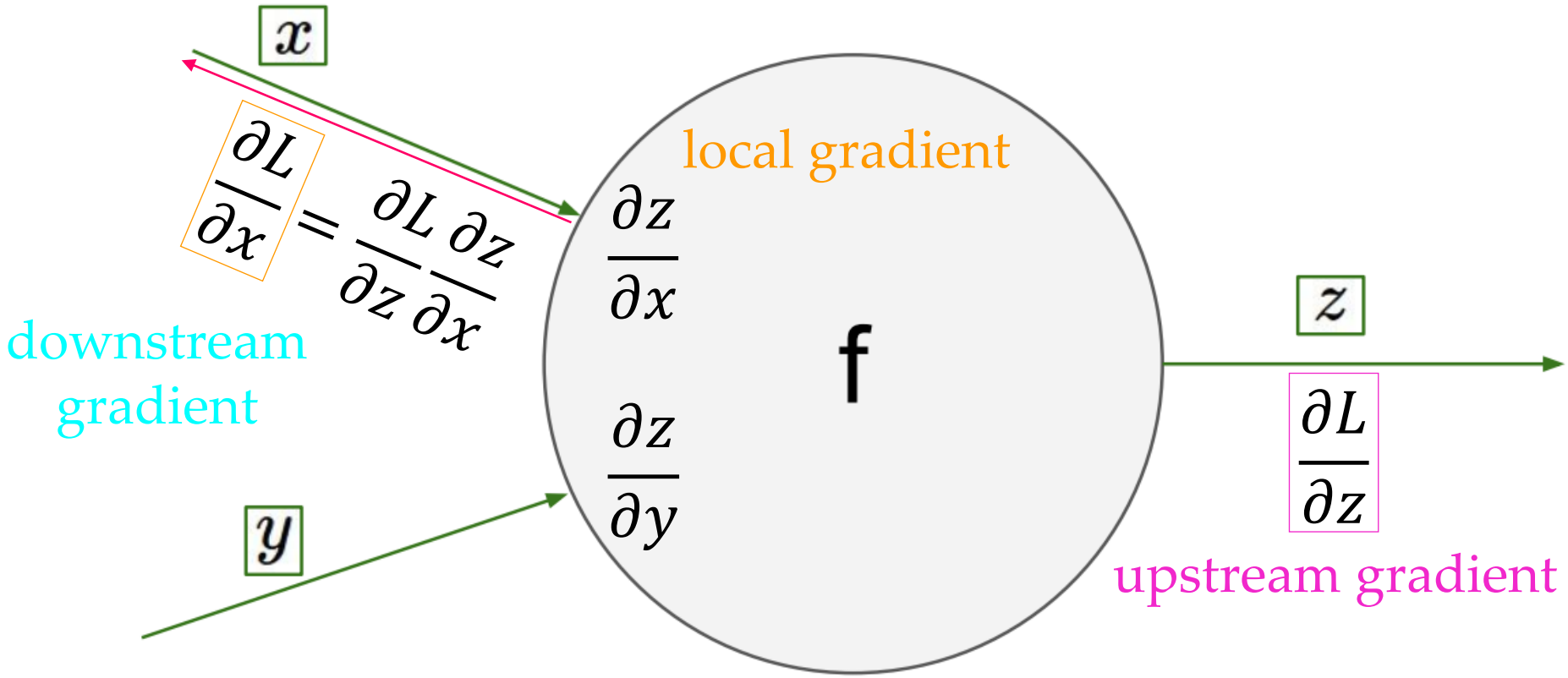
مدرس: محمدرضا محمدی

زمستان ۱۴۰۱

الگوریتم‌های بهینه‌سازی

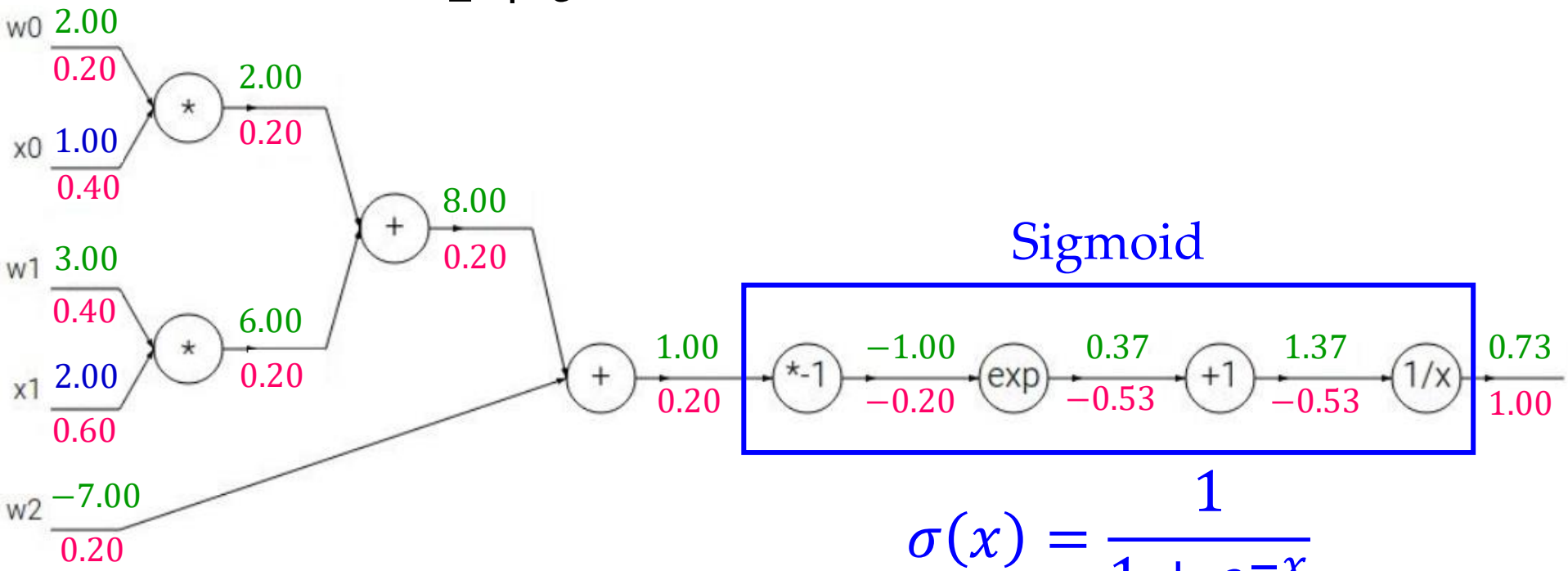
Optimization Algorithms

پس‌انتشار (Backpropagation)



مثال: Linear + Sigmoid

$$f(\boldsymbol{w},\boldsymbol{x}) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

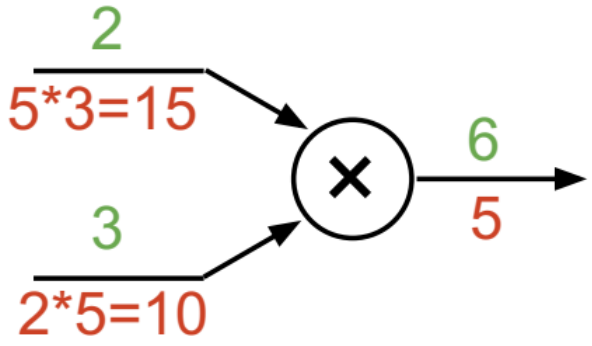


$$\sigma(x) = \frac{1}{1+e^{-x}}$$

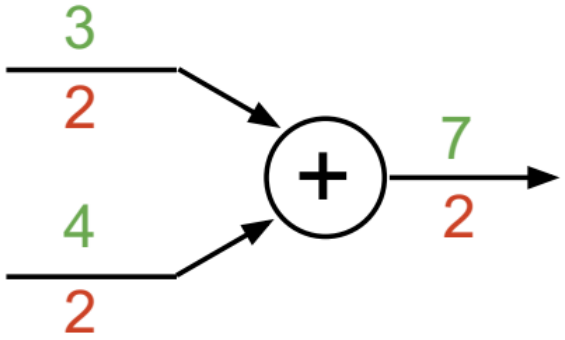
$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1-1+e^{-x}}{1+e^{-x}}\frac{1}{1+e^{-x}} = \big(1-\sigma(x)\big)\sigma(x)$$

نمونه‌هایی از جریان گرادیان

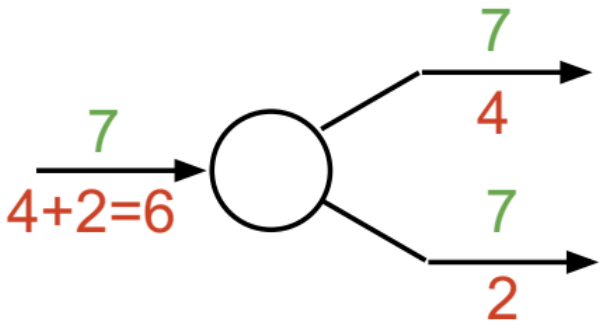
گیت **ضرب**: مبادله‌گر گرادیان



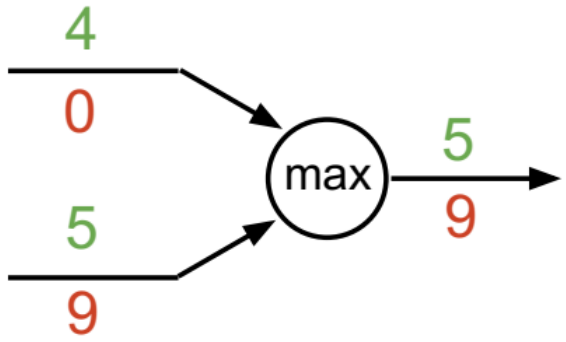
گیت **جمع**: توزیع‌کننده گرادیان



گیت **کپی**: جمع‌کننده گرادیان

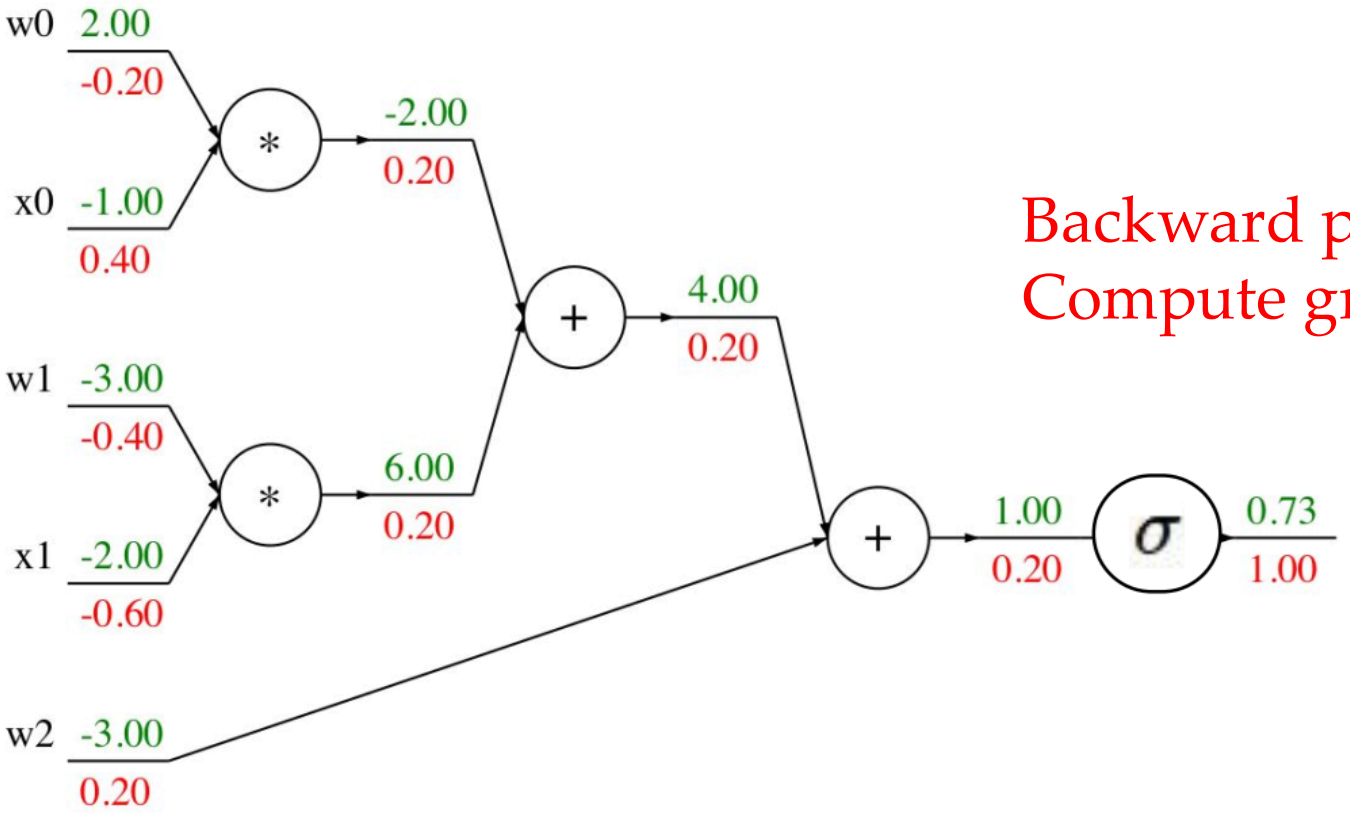


گیت **بیشینه**: روتر گرادیان



Forward pass:
Compute output

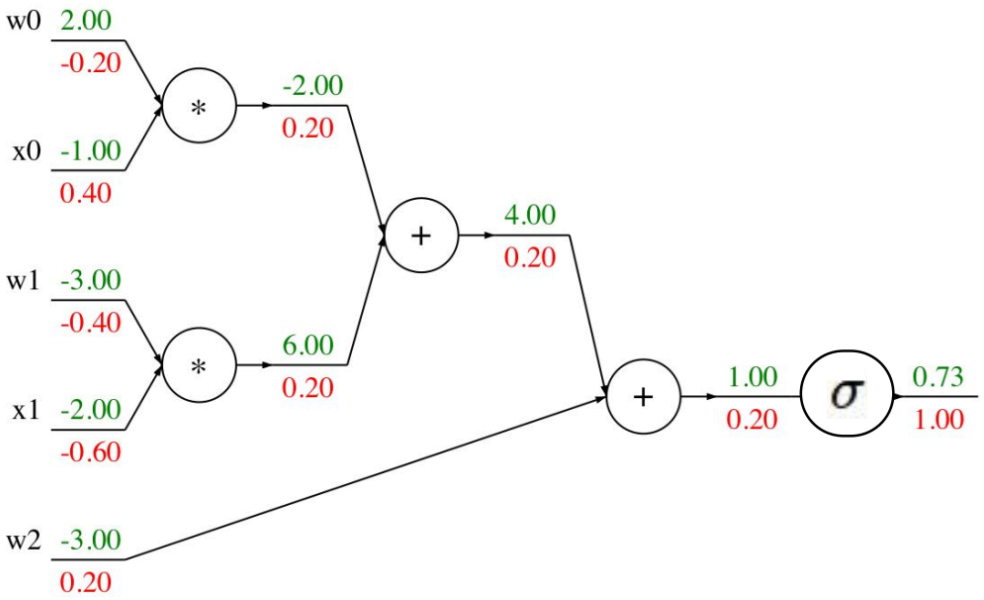
```
s0 = w0 * x0
s1 = w1 * x1
s2 = s0 + s1
s3 = s2 + w2
L = sigmoid(s3)
```



Backward pass:
Compute grads

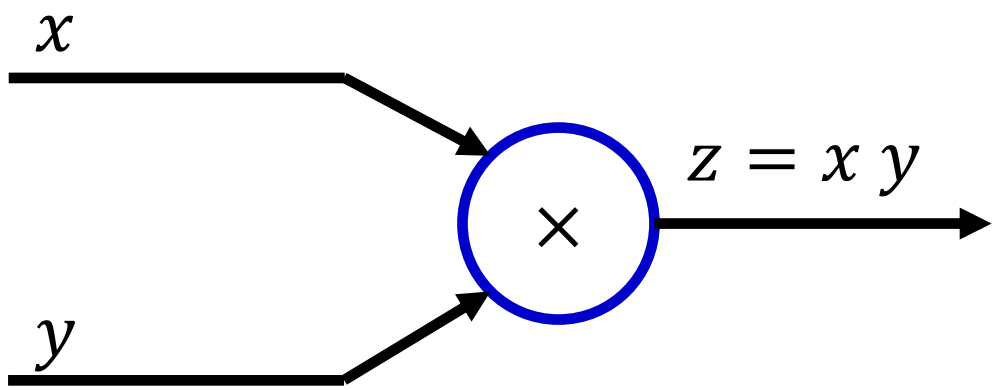
```
grad_L = 1.0
grad_s3 = grad_L * (1 - L) * L
grad_w2 = grad_s3
grad_s2 = grad_s3
grad_w0 = grad_s2
grad_s1 = grad_s2
grad_w1 = grad_s1 * x1
grad_x1 = grad_s1 * w1
grad_w0 = grad_s0 * x0
grad_x0 = grad_s0 * w0
```

پیاده‌سازی ماژولار




```python
class ComputationalGraph(object):
    #...
    def forward(inputs):
        # 1. [pass inputs to input gates...]
        # 2. forward the computational graph:
        for gate in self.graph.nodes_topologically_sorted():
            gate.forward()
        return loss # the final gate in the graph outputs the loss
    def backward():
        for gate in reversed(self.graph.nodes_topologically_sorted()):
            gate.backward() # little piece of backprop (chain rule applied)
        return inputs_gradients
```

گیت ضرب




```
class MultuplyGate(object):
  """
  x,y are scalars
  """
  def forward(x,y):
    z = x*y
    self.x = x  # Cache
    self.y = y  # Cache
    # We cache x and y because we know that the derivatives contains them.
    return z
  def backward(dz):
    dx = self.y * dz          #self.y is dx
    dy = self.x * dz
    return [dx, dy]
```

لایه‌ها در Caffe

BVLC / caffe Public

Notifications | Star 32k | Fork 18.9k

Code | Issues 882 | Pull requests 291 | Actions | Projects | Wiki | Security | Insights

master | caffe / src / caffe / layers / | Go to file

Noiredd Merge branch 'master' into patch_1 — 828dd10 on Aug 21, 2018 — History

..		
absval_layer.cpp	dismantle layer headers	6 years ago
absval_layer.cu	dismantle layer headers	6 years ago
accuracy_layer.cpp	Added count==0 safeguard to CPU accuracy calculation	4 years ago
accuracy_layer.cu	explain use of scratch diffs in comments	4 years ago
argmax_layer.cpp	dismantle layer headers	6 years ago
base_conv_layer.cpp	Shape mismatch CHECK logging improvements	5 years ago
base_data_layer.cpp	Using default from proto for prefetch	5 years ago
base_data_layer.cu	Switched multi-GPU to NCCL	5 years ago
batch_norm_layer.cpp	CPU BatchNormLayer: replace powx with sqr and sqrt	5 years ago
batch_norm_layer.cu	GPU BatchNormLayer: replace powx with mul and sqrt	5 years ago

لایه Sigmoid

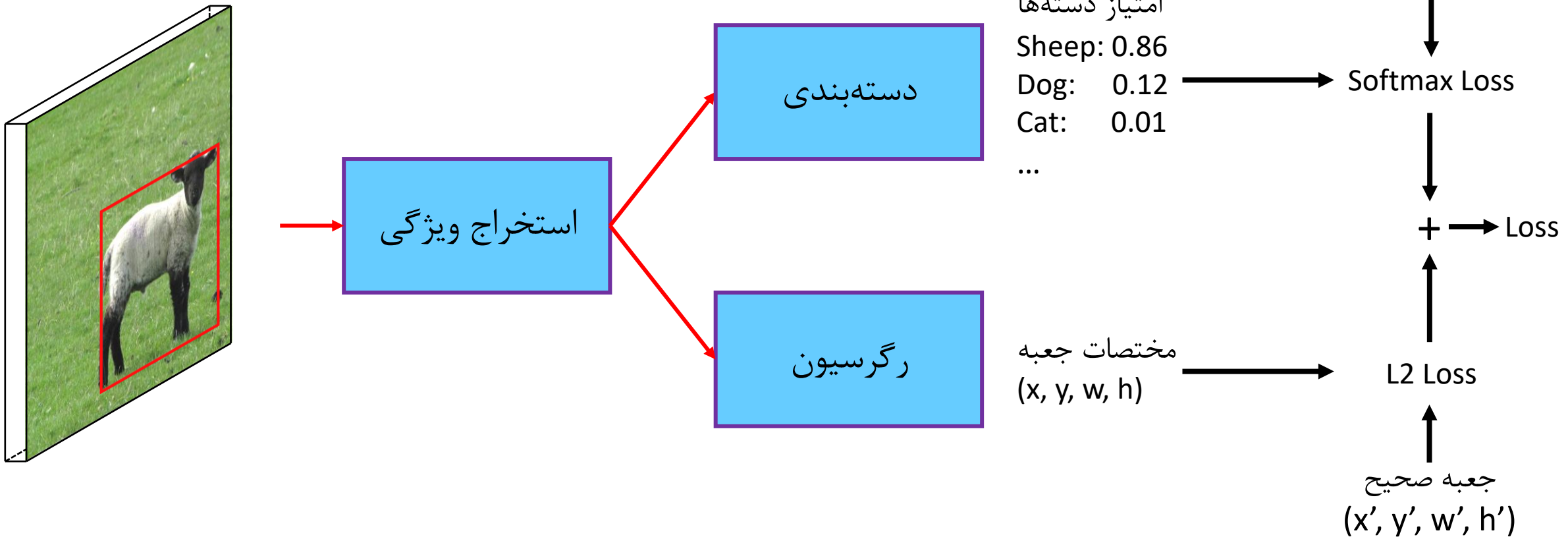
```
template <typename Dtype>
void SigmoidLayer<Dtype>::Forward_cpu(const vector<Blob<Dtype>*>& bottom,
      const vector<Blob<Dtype>*>& top) {
    const Dtype* bottom_data = bottom[0]->cpu_data();
    Dtype* top_data = top[0]->mutable_cpu_data();
    const int count = bottom[0]->count();
    for (int i = 0; i < count; ++i) {
      top_data[i] = sigmoid(bottom_data[i]);
    }
}

template <typename Dtype>
void SigmoidLayer<Dtype>::Backward_cpu(const vector<Blob<Dtype>*>& top,
      const vector<bool>& propagate_down,
      const vector<Blob<Dtype>*>& bottom) {
    if (propagate_down[0]) {
      const Dtype* top_data = top[0]->cpu_data();
      const Dtype* top_diff = top[0]->cpu_diff();
      Dtype* bottom_diff = bottom[0]->mutable_cpu_diff();
      const int count = bottom[0]->count();
      for (int i = 0; i < count; ++i) {
        const Dtype sigmoid_x = top_data[i];
        bottom_diff[i] = top_diff[i] * sigmoid_x * (1. - sigmoid_x);
      }
    }
}
```

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

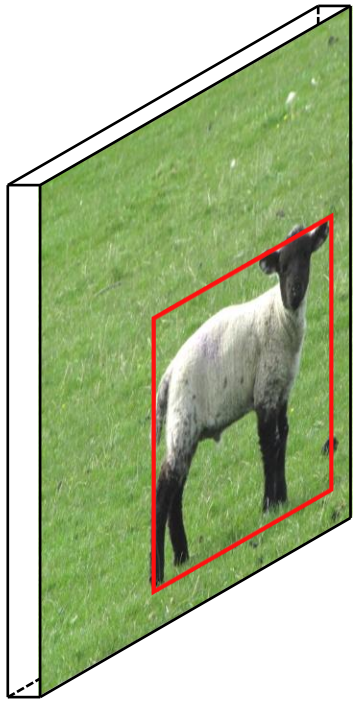
$$\frac{d\sigma(x)}{dx} = \sigma(x)\big(1 - \sigma(x)\big)$$

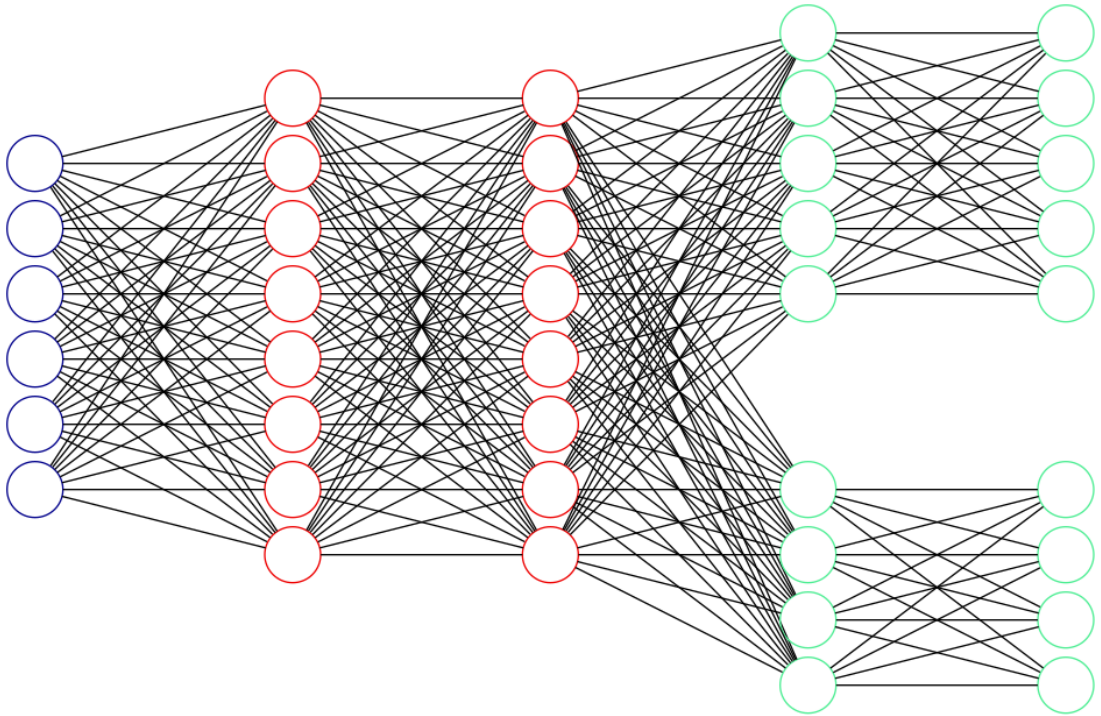
مثال: دسته‌بندی و مکان‌یابی

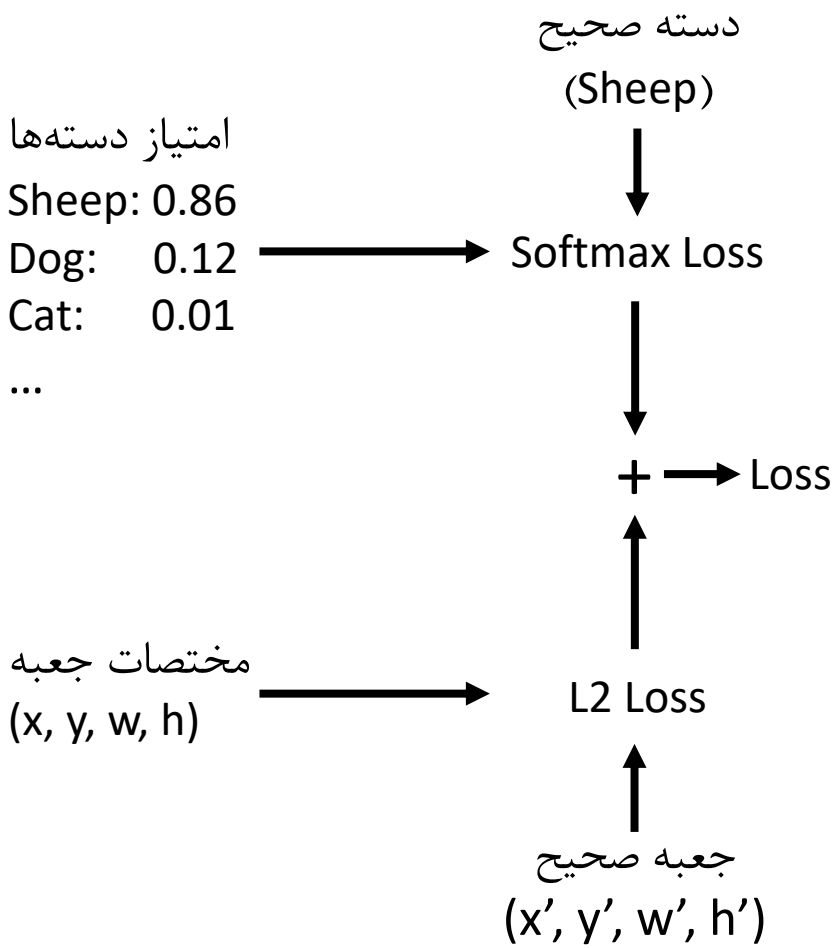


مثال: دسته‌بندی و مکان‌یابی

$$L(\mathbf{w}, b) = L_{\mathrm{C}}(\mathbf{w}, b) + L_{\mathrm{R}}(\mathbf{w}, b) + \lambda\, \Omega(\mathbf{w})$$





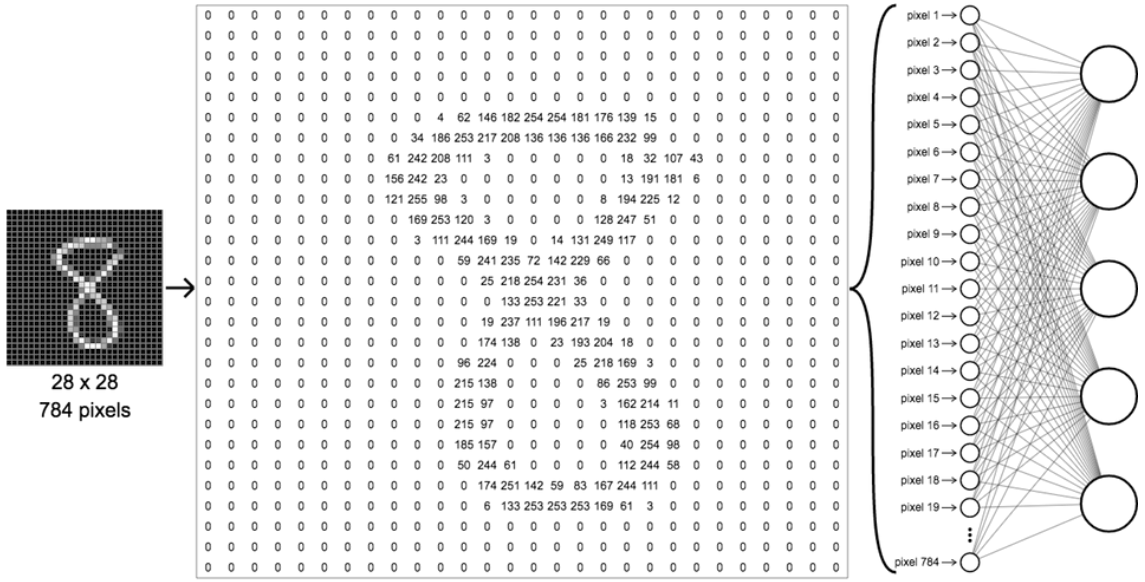


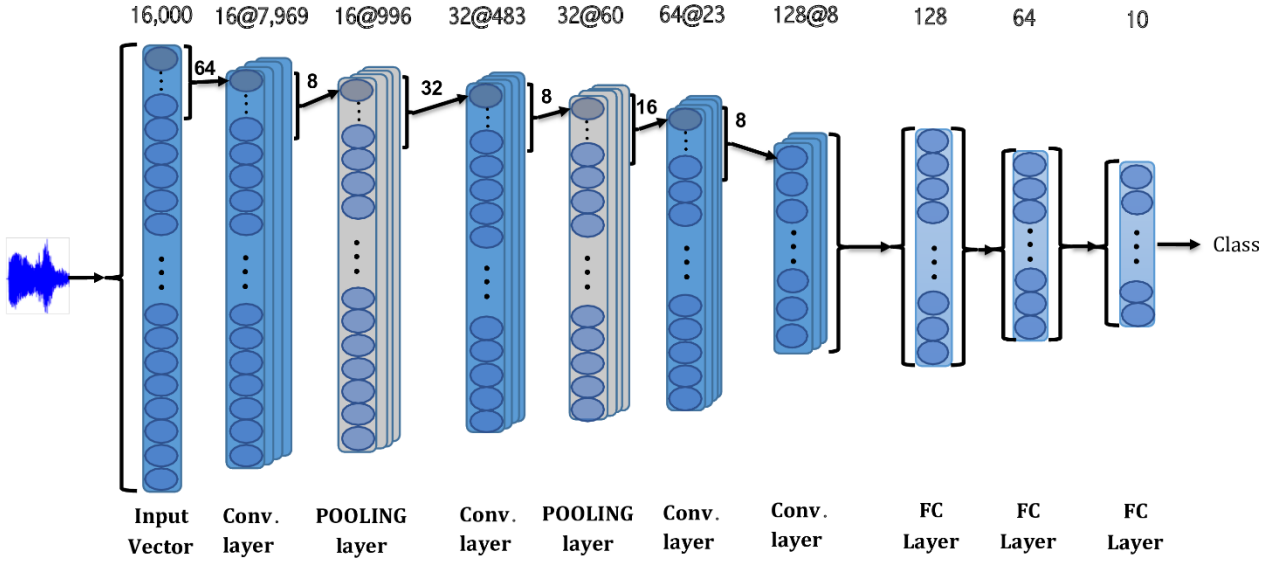
شبکه‌های عصبی کانولوشنی

Convolutional Neural Networks

شبکه‌های عصبی کانولوشنی

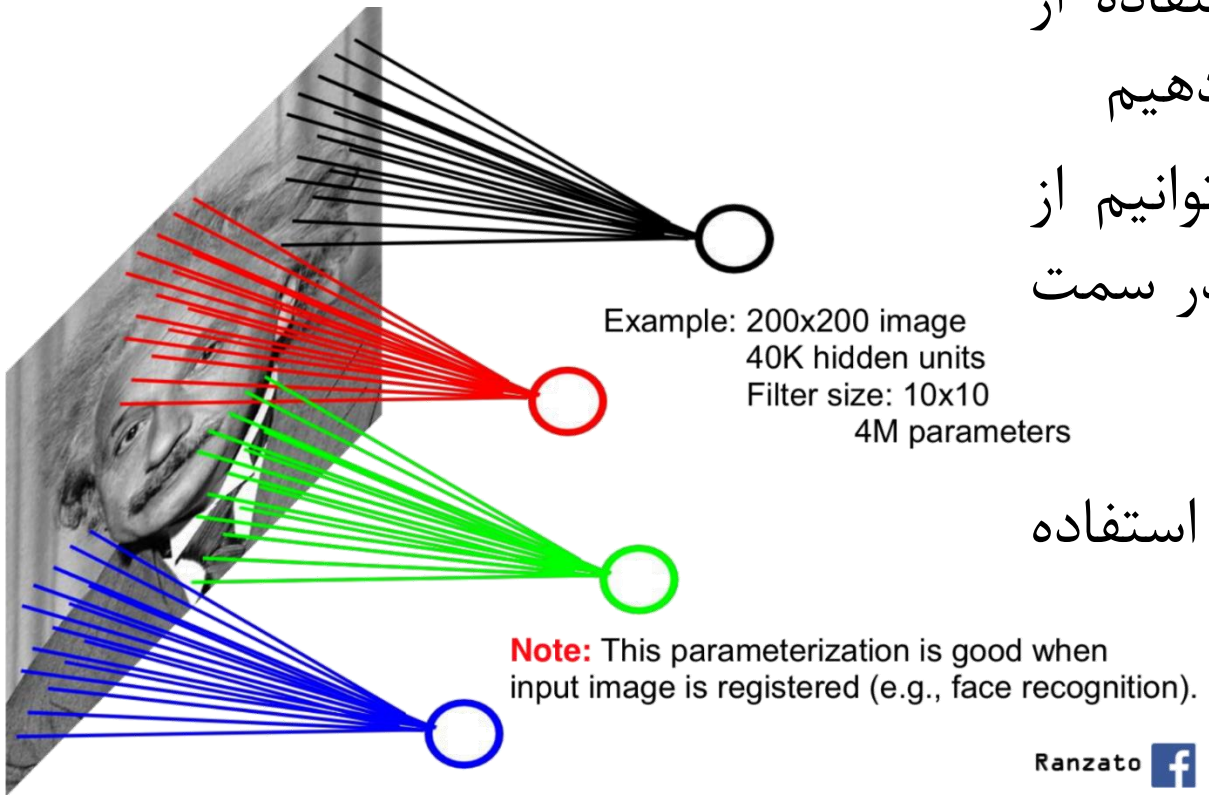
- شبکه‌های عصبی کانولوشنی (CNNs) نوع خاصی از NNها هستند که برای پردازش داده‌هایی که دارای توپولوژی شبکه‌ای شناخته‌شده‌ای هستند مناسب‌اند
- کانولوشن یک عمل خطی است





لایه‌های متصل محلّی

- بسیاری از ویژگی‌هایی که چشم انسان به راحتی می‌تواند تشخیص دهد، ویژگی‌های محلی هستند
- ما می‌توانیم لبه‌ها، بافت‌ها و حتی شکل‌ها را با استفاده از شدت پیکسل‌ها در ناحیه کوچکی از تصویر تشخیص دهیم
- اگر می‌خواهیم یک ویژگی را تشخیص بدهیم، می‌توانیم از همان آشکارساز در گوشه پایین سمت چپ تصویر و در سمت راست بالای تصویر استفاده کنیم
- ما می‌توانیم از وزن‌های یکسان در هر مکان از تصویر استفاده کنیم
  - اشتراک وزن‌ها (weight sharing)



کانولوشن و همبستگی

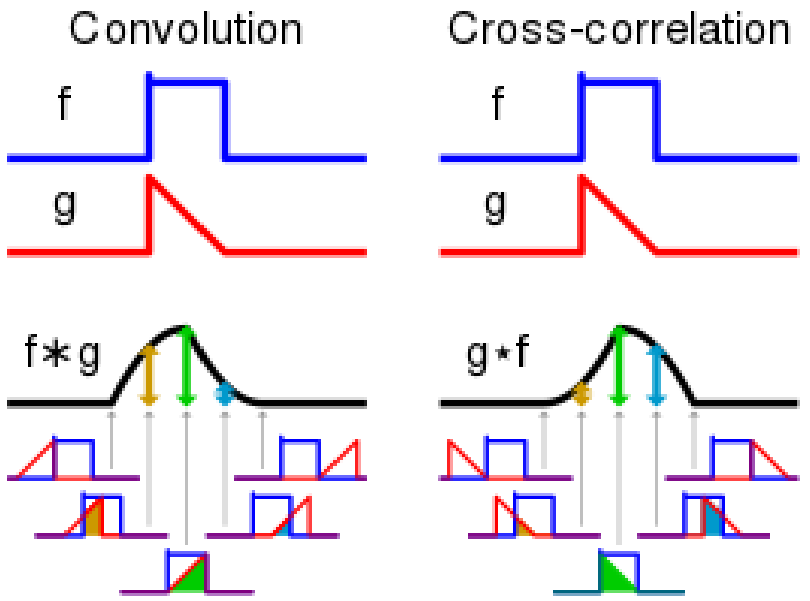
- بسیاری از کتابخانه‌های ML همبستگی متقابل را پیاده‌سازی می‌کنند اما آن را کانولوشن می‌نامند!
- الگوریتم یادگیری مقادیر مناسب هسته را در مکان مناسب یاد می‌گیرد

$$S(i) = (I * K)(i) = \sum_m I(i-m)K(m)$$

$$S(i) = (I \star K)(i) = \sum_m I(i+m)K(m)$$

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

$$S(i,j) = (I \star K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$



https://en.wikipedia.org/wiki/Cross-correlation