Computational Systems Biology
Deep Learning in the Life Sciences

6.802 6.874 20.390 20.490 HST.506

David Gifford
Lecture 2
February 7, 2019

Optimizing Feedforward Networks

http://mit6874.github.io

On tap today!

- How can we use gradients for optimization?
- How can we use gradients to train a deep neural network?
- What performance metrics should we use?
- How can we manage gradient optimization?
- How can we “regularize” a model to control parameter selection and thus model complexity?

Gradient based optimization needs a loss function to minimize

Stochastic gradient descent updates parameters to reduce loss

$$L(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\log p(y \mid \boldsymbol{x}; \boldsymbol{\theta}).$$

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim\hat{p}_{\text{data}}} L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

$$\boldsymbol{g} = \frac{1}{m'}\nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m'} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}).$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g}$$

Learning rate

$y' = w^\top x$ find w to best approximate y

What loss function should we use to pick w?

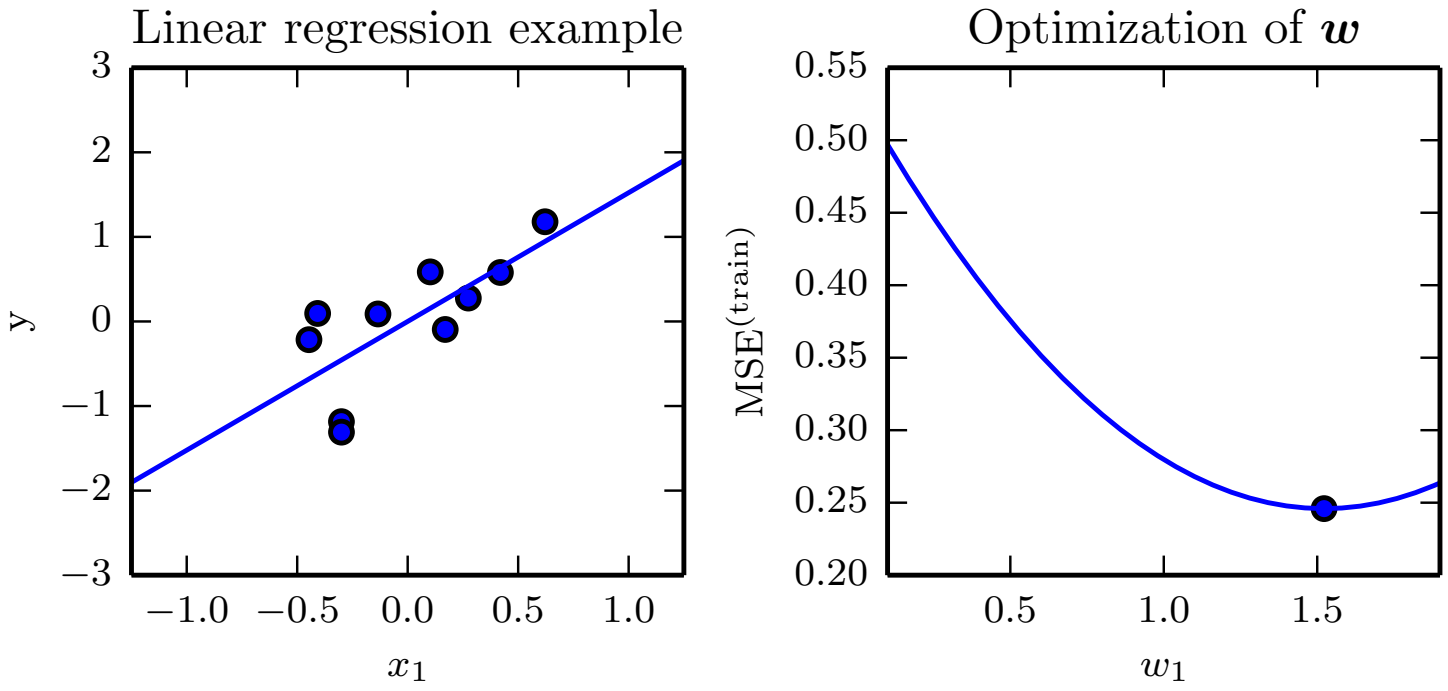


Figure 5.1

Minimizing mean-square error maximizes Gaussian likelihood

$$f(x \mid \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\log f(x \mid \theta) = C_1 - \frac{(x-\mu)^2}{C_2}$$

$$\arg\max_{\theta} f(x \mid \theta) = \arg\min_{\theta} (x-\mu)^2$$

Gradient Descent for linear regression

$$\vec{w}, b = \arg\min_{\vec{w},b} \sum_{i=1}^{n} (y^{(i)} - \vec{w}^T \vec{x}^{(i)} - b)^2$$

$$\frac{\partial E}{\partial \vec{w}} = \frac{2}{n} \sum_{i=1}^{n} (\vec{w}^T \vec{x}^{(i)} + b - y^{(i)}) \vec{x}^{(i)}$$

$$\frac{\partial E}{\partial \vec{w}} = \frac{2}{n} \sum_{i=1}^{n} (\widehat{y}^{(i)} - y^{(i)}) \vec{x}^{(i)}$$

$$\frac{\partial E}{\partial b} = \frac{2}{n} \sum_{i=1}^{n} (\widehat{y}^{(i)} - y^{(i)})$$

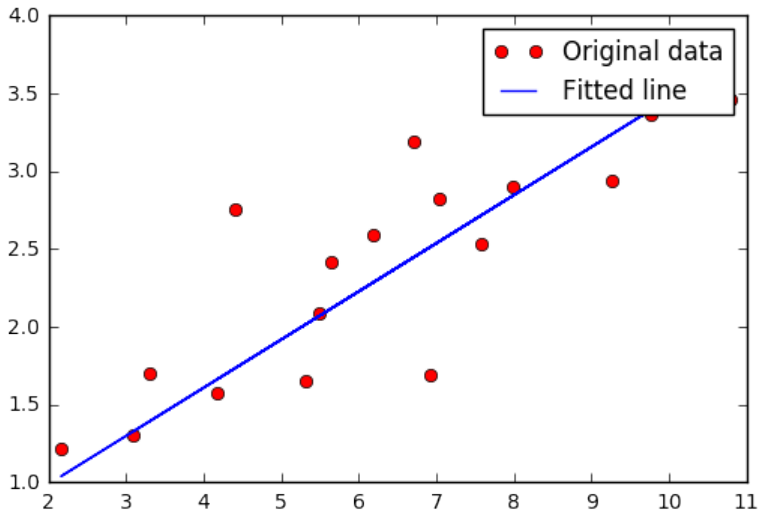
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g}$$

Linear regression can be solved by gradient descent

```
# tf Graph Input
X = tf.placeholder("float")
Y = tf.placeholder("float")

# Set model weights
W = tf.Variable(rng.randn(), name="weight")
b = tf.Variable(rng.randn(), name="bias")

# Construct a linear model
pred = tf.add(tf.mul(X, W), b)

# Mean squared error
cost = tf.reduce_sum(tf.pow(pred-Y, 2))/(2*n_samples)

# Gradient descent
optimizer =
tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
```

Cost is minimized after 1000 epochs

```
Epoch: 0050 cost= 0.195095107 W= 0.441748 b=
-0.580876
Epoch: 0100 cost= 0.181448311 W= 0.430319 b=
-0.498661
Epoch: 0150 cost= 0.169377610 W= 0.419571 b=
-0.421336
Epoch: 0200 cost= 0.158700854 W= 0.409461 b=
-0.348611
Epoch: 0250 cost= 0.149257123 W= 0.399953 b= -0.28021
Epoch: 0300 cost= 0.140904188 W= 0.391011 b=
-0.215878
Epoch: 0350 cost= 0.133515999 W= 0.3826 b= -0.155372
Epoch: 0400 cost= 0.126981199 W= 0.374689 b=
-0.0984639
Epoch: 0450 cost= 0.121201262 W= 0.367249 b=
-0.0449408
Epoch: 0500 cost= 0.116088994 W= 0.360252 b=
0.00539905
Epoch: 0550 cost= 0.111567356 W= 0.35367 b= 0.052745
Epoch: 0600 cost= 0.107568085 W= 0.34748 b= 0.0972751
Epoch: 0650 cost= 0.104030922 W= 0.341659 b= 0.139157
Epoch: 0700 cost= 0.100902475 W= 0.336183 b= 0.178547
Epoch: 0750 cost= 0.098135538 W= 0.331033 b= 0.215595
Epoch: 0800 cost= 0.095688373 W= 0.32619 b= 0.25044
Epoch: 0850 cost= 0.093524046 W= 0.321634 b= 0.283212
Epoch: 0900 cost= 0.091609895 W= 0.317349 b= 0.314035
Epoch: 0950 cost= 0.089917004 W= 0.31332 b= 0.343025
Epoch: 1000 cost= 0.088419855 W= 0.30953 b= 0.370291
Optimization Finished!
Training cost= 0.0884199 W= 0.30953 b= 0.370291
```



We may not always find the best solution for non-convex functions

Approximate Optimization

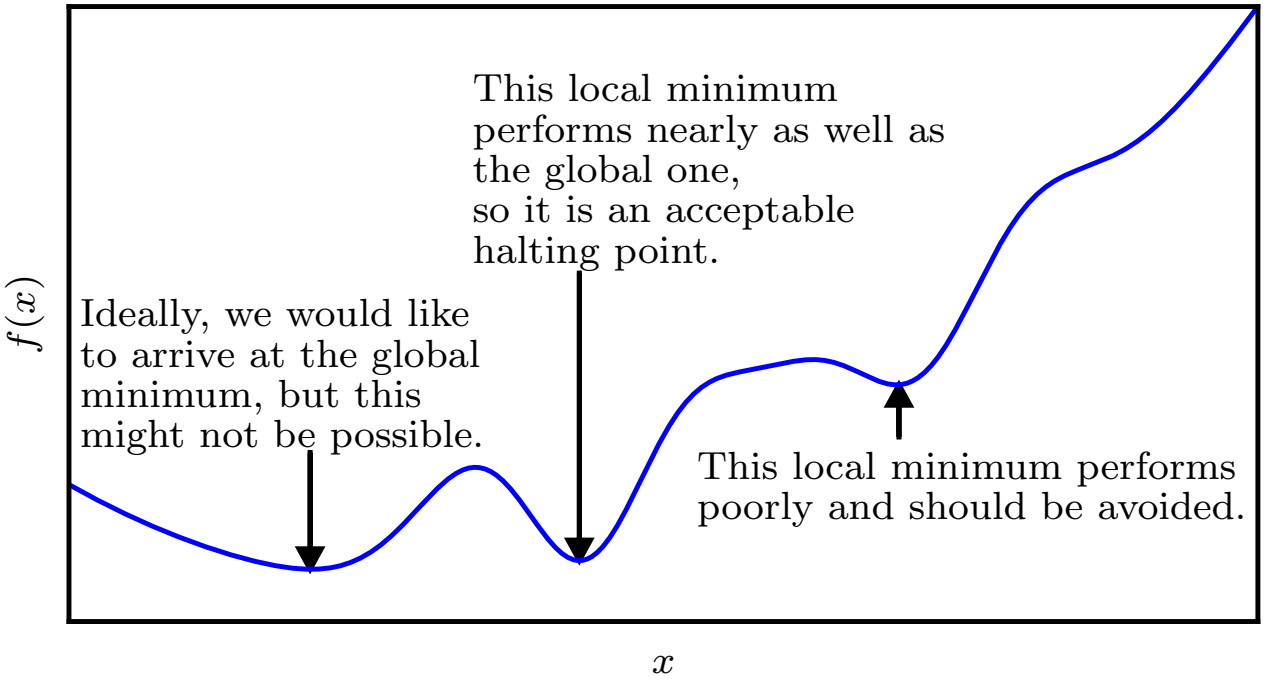


Figure 4.3

Example convex functions

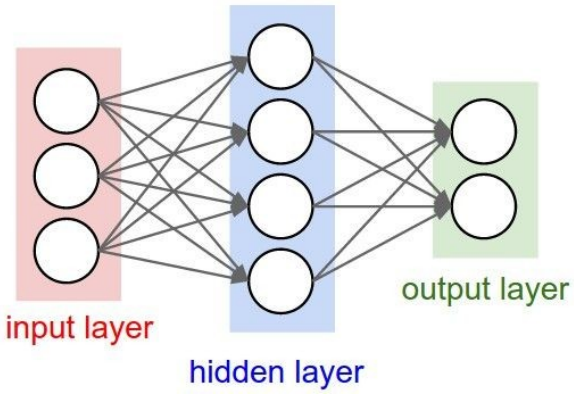
- c(x) = Mx + b
- $c(x) = e^{c1(x)}$
- c(x) = c1(x) + c2(x)
- $c(x) = x^p$
- c(x) = |x|
- c(x) = x log x
- c(x) = max(c1(x), c2(x))

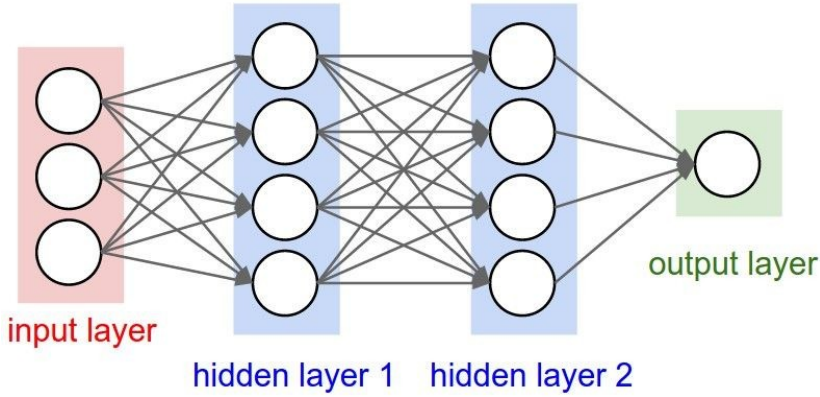
Log convex functions

- They are not convex but their log is convex
- Example - sigmoid
- They can be optimized by convex solvers

Deep neural networks are typically non-convex functions

Each hidden layer has an activation function at its output

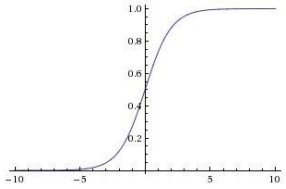




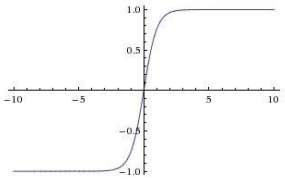
Popular activation functions at node outputs

Sigmoid

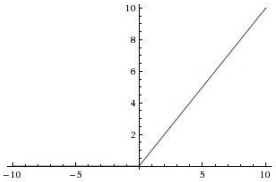
$$\sigma(x) = 1/(1 + e^{-x})$$



tanh tanh(x)

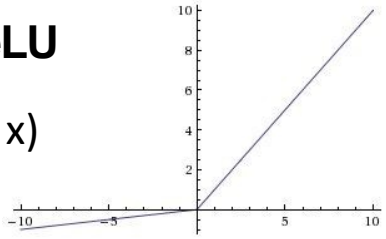


ReLU max(0,x)



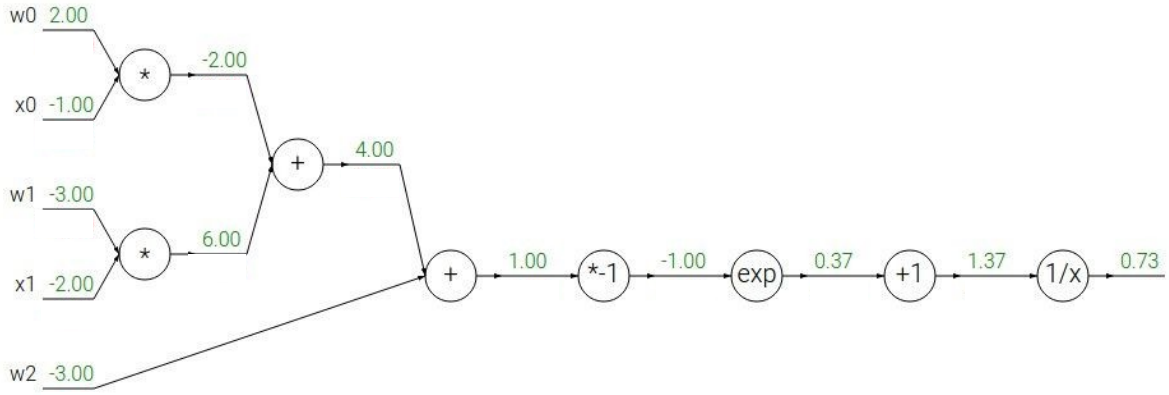
Leaky ReLU

max(0.1x, x)

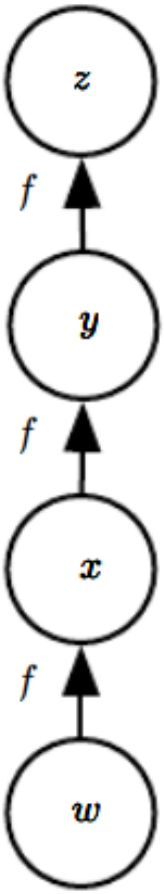


Example of backpropagation of errors

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

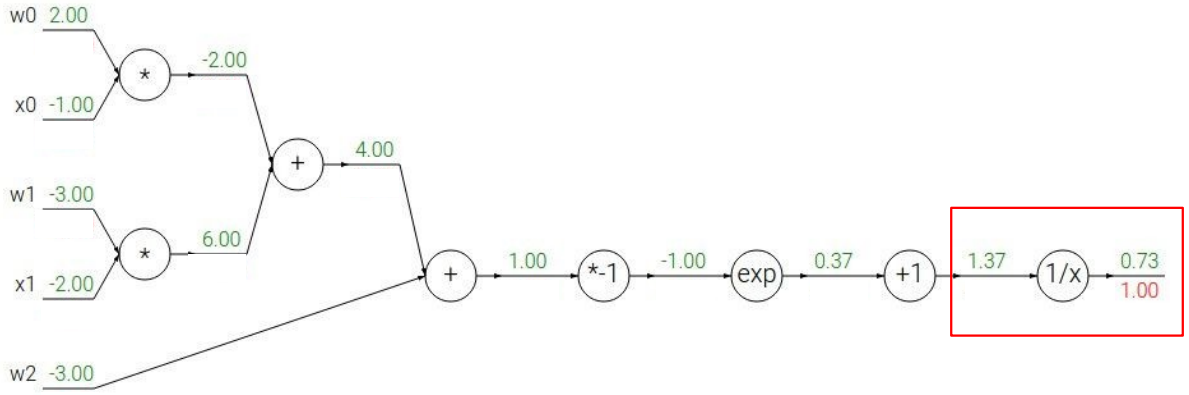


Backpropogation computes gradients via the chain rule



$$\frac{\partial z}{\partial w}$$
$$= \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w}$$
$$= f'(y)f'(x)f'(w)$$
$$= f'(f(f(w)))f'(f(w))f'(w)$$

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



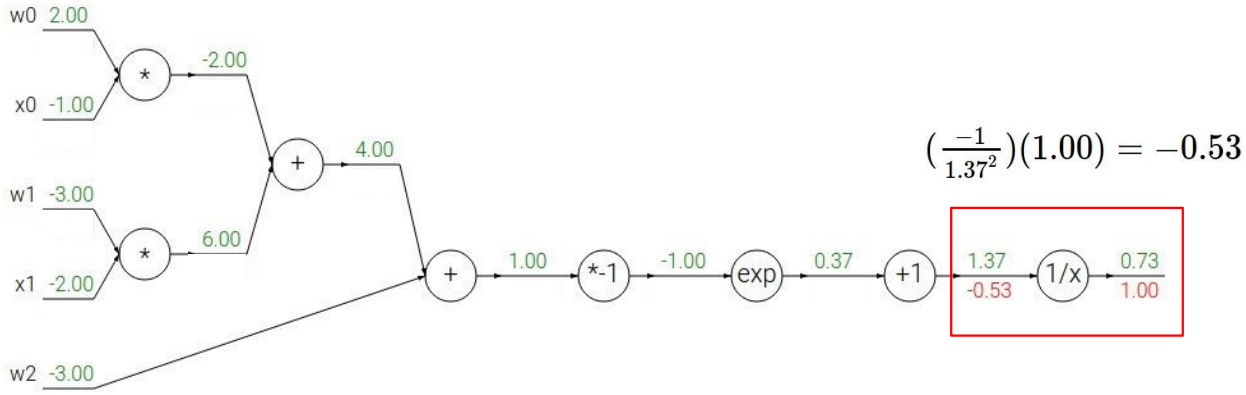
$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

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



$$(\frac{-1}{1.37^2})(1.00) = -0.53$$

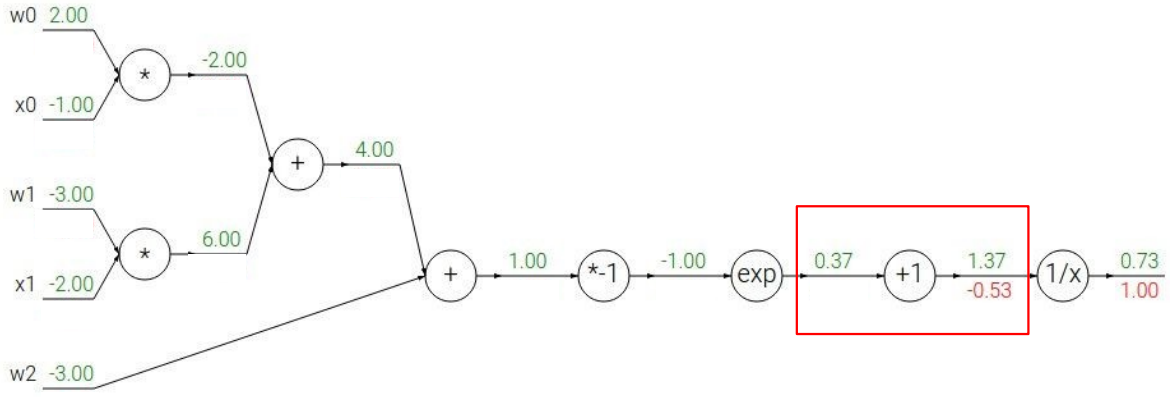
$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

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



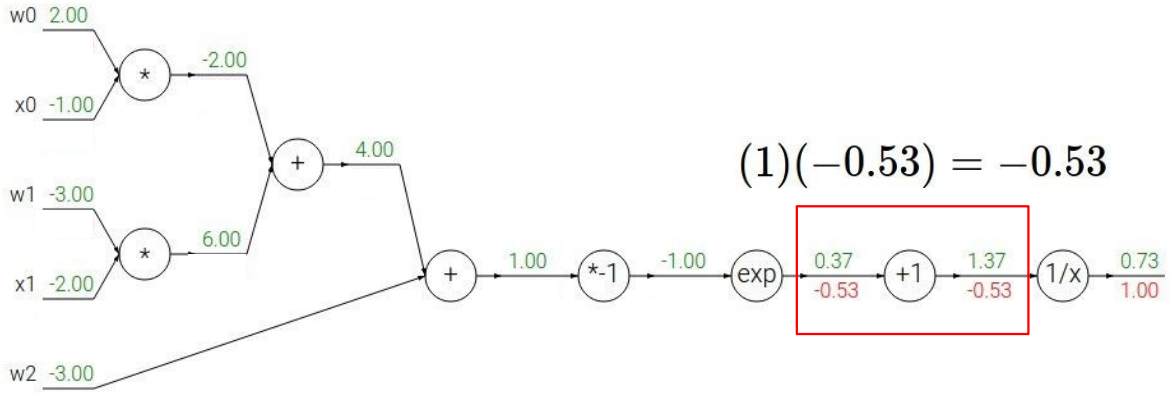
$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

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



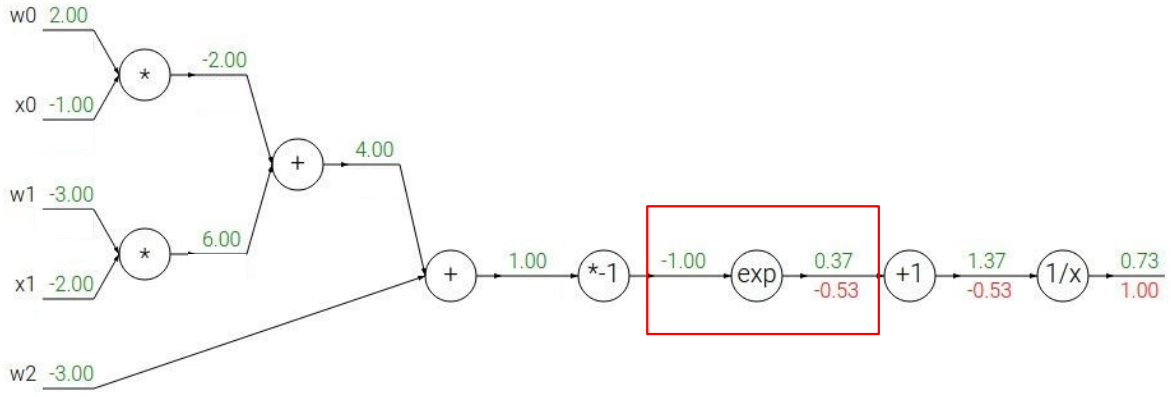
$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

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



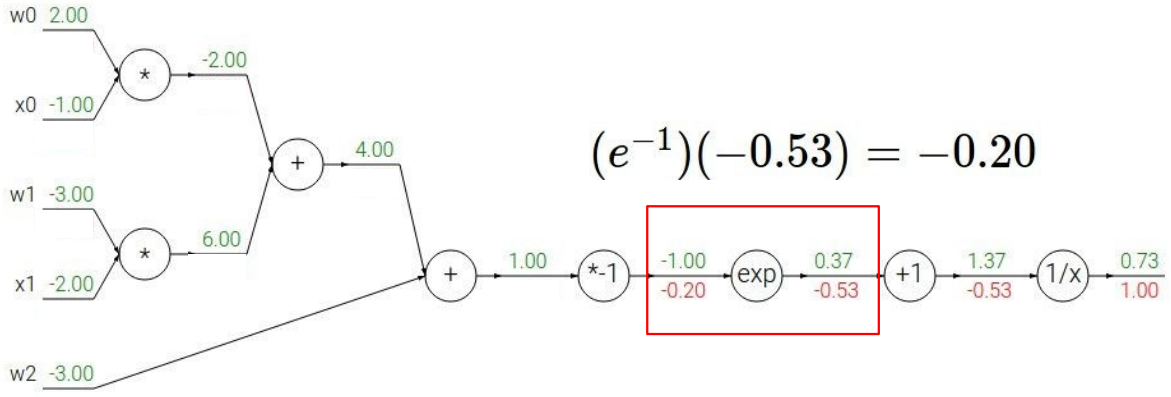
$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

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

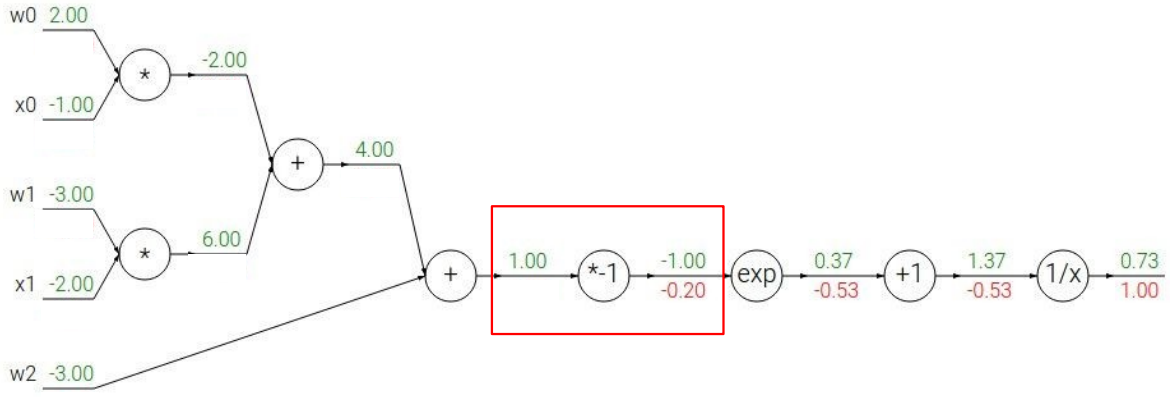


$$(e^{-1})(-0.53) = -0.20$$

$f(x) = e^x$	$\rightarrow$	$\frac{df}{dx} = e^x$
$f_a(x) = ax$	$\rightarrow$	$\frac{df}{dx} = a$

$f(x) = \frac{1}{x}$	$\rightarrow$	$\frac{df}{dx} = -1/x^2$
$f_c(x) = c + x$	$\rightarrow$	$\frac{df}{dx} = 1$

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



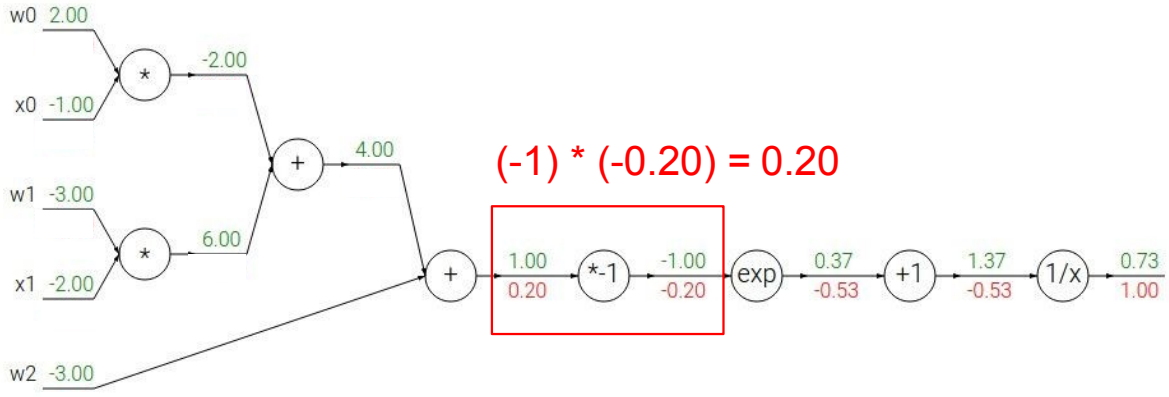
$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

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



(-1) * (-0.20) = 0.20

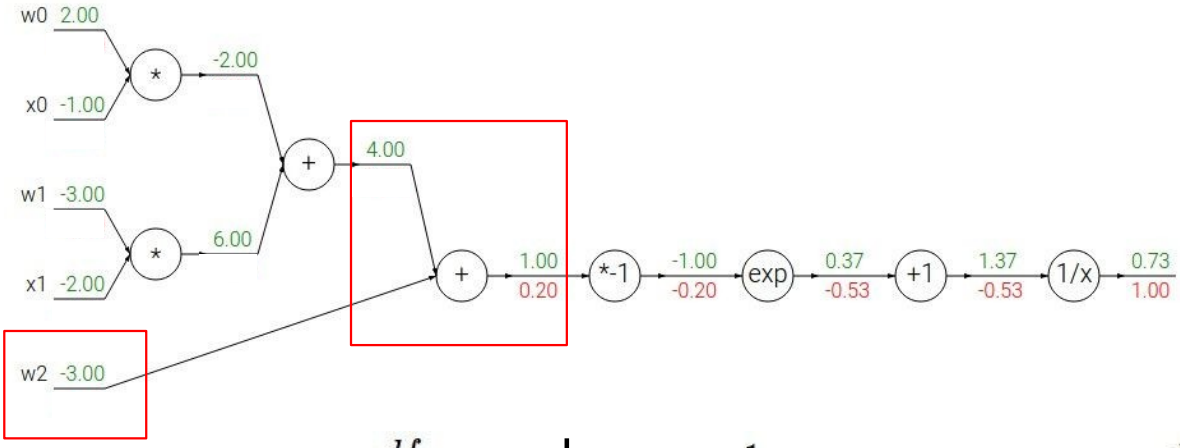
$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

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

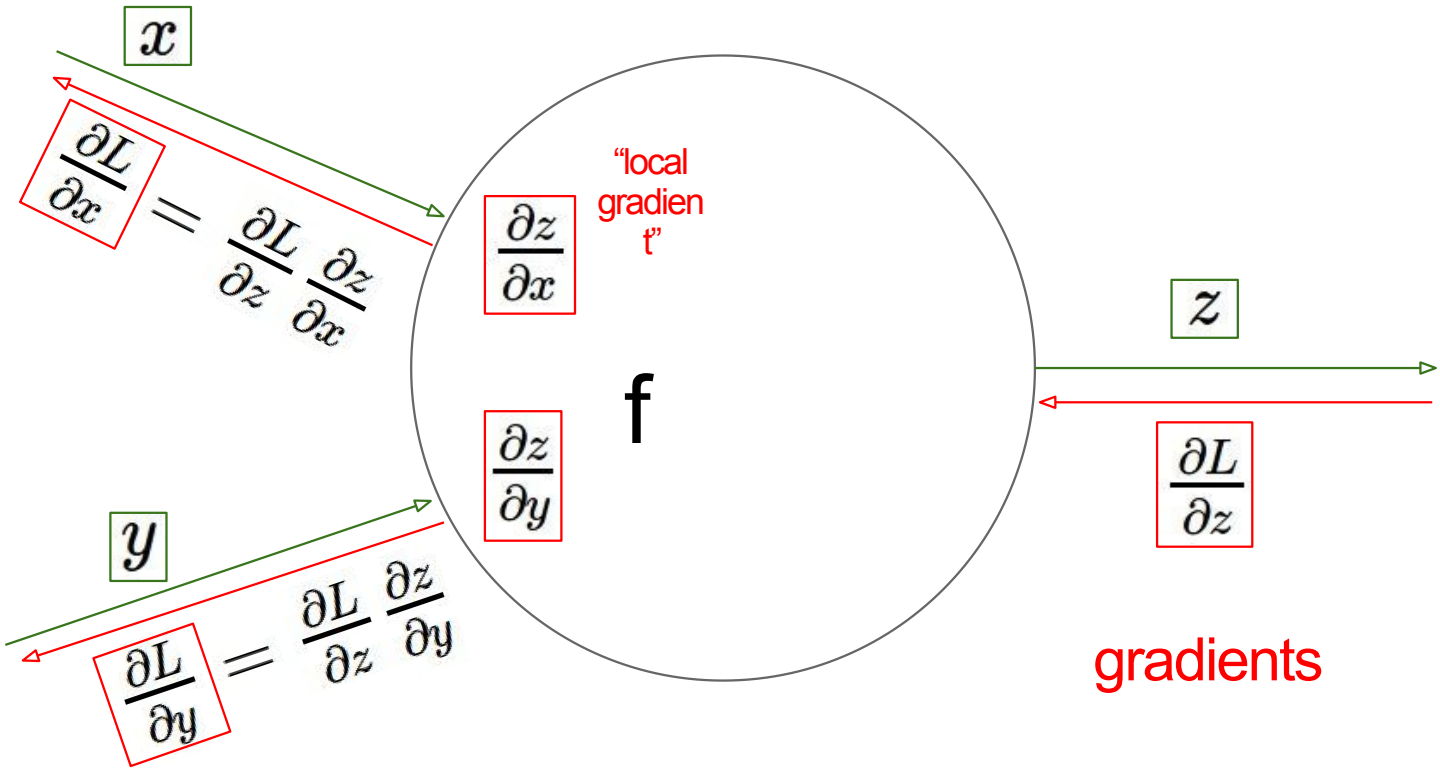


$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

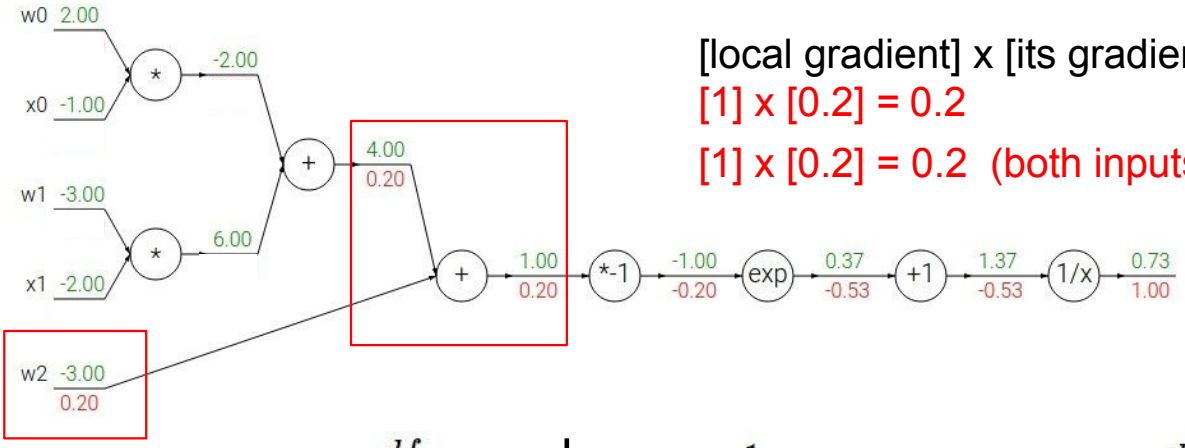
$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$


x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradien t"
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y}$
z
$\frac{\partial L}{\partial z}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
gradients

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



[local gradient] x [its gradient]
[1] x [0.2] = 0.2
[1] x [0.2] = 0.2 (both inputs!)

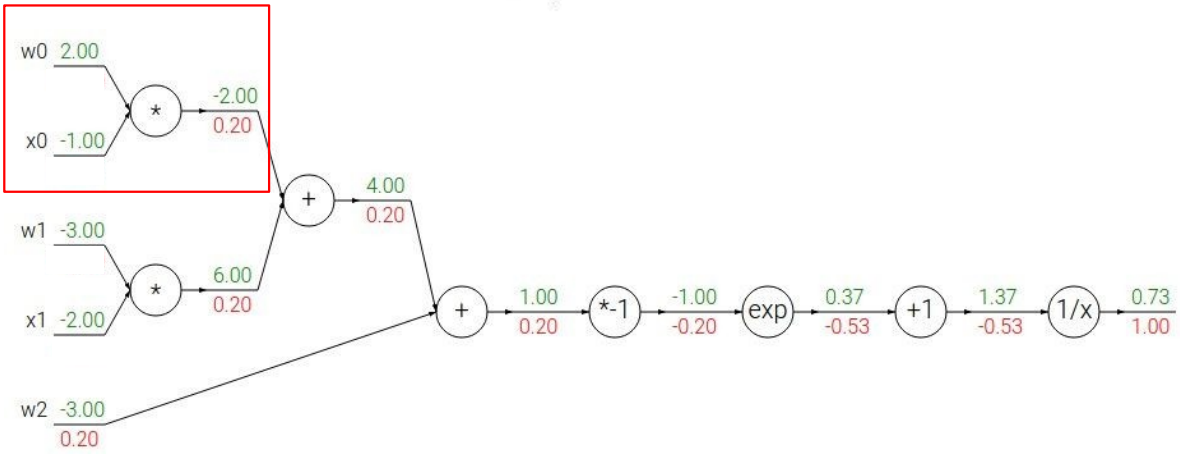
$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

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

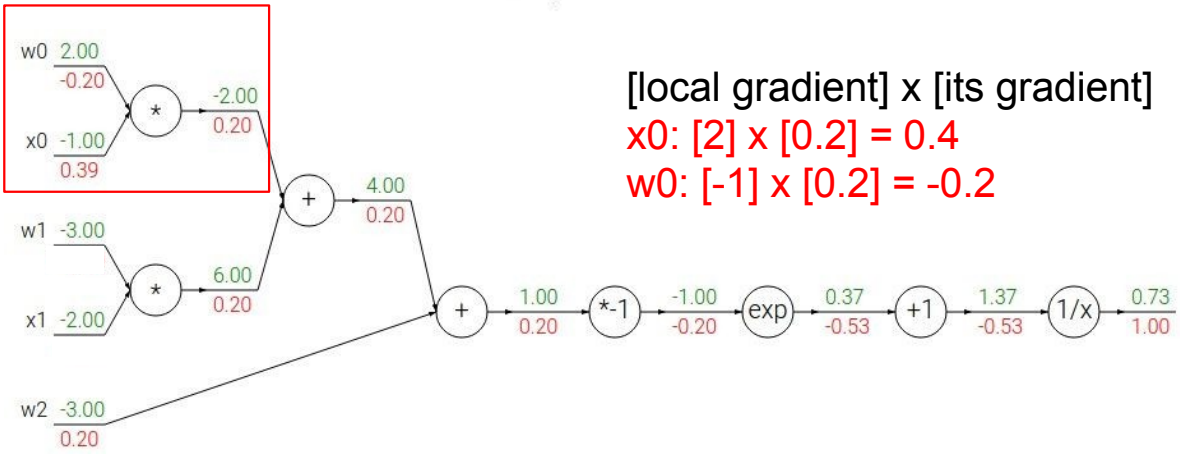


$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x \qquad f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a \qquad f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

Another example:

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



[local gradient] x [its gradient]
x0: [2] x [0.2] = 0.4
w0: [-1] x [0.2] = -0.2

$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

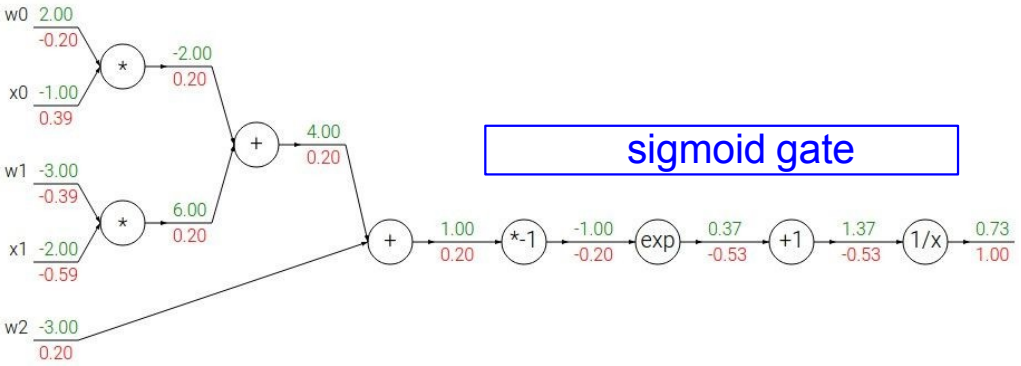
$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

Symbolic differentiation can save work and improve accuracy

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

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

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

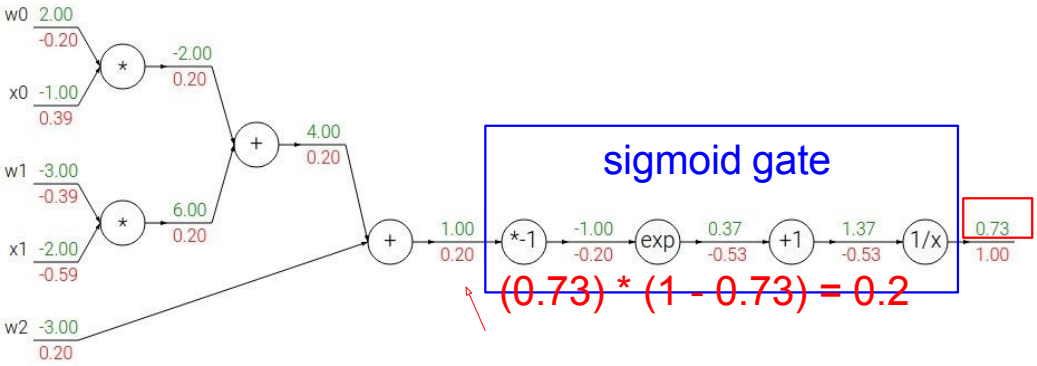


sigmoid gate

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

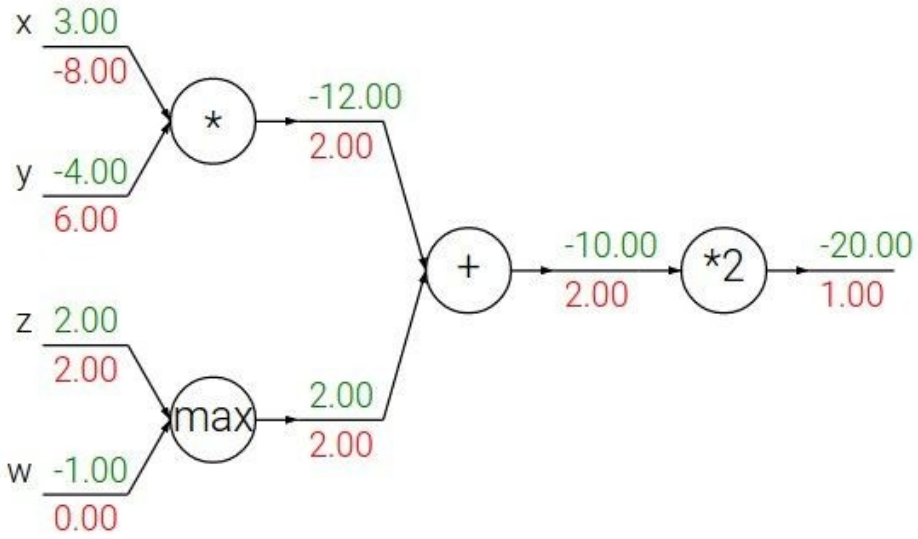
$$\sigma(x) = \frac{1}{1+e^{-x}}$$ sigmoid function

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

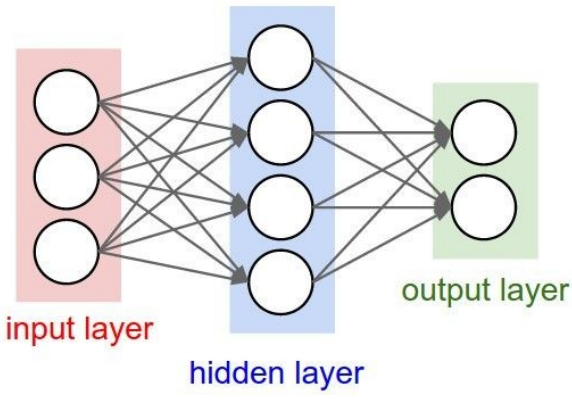


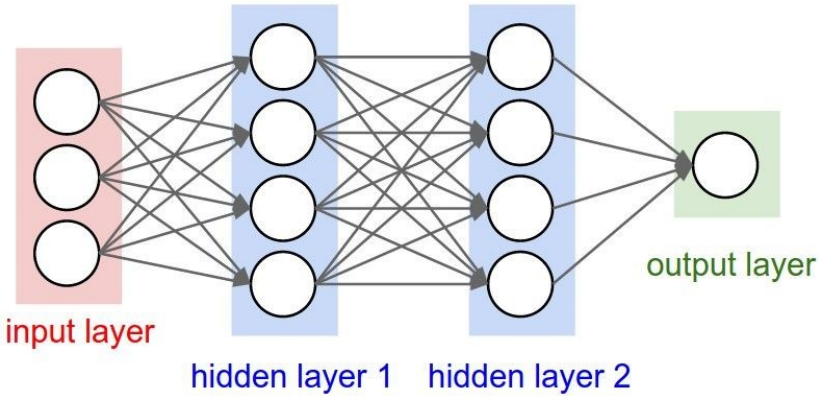
Patterns in backward flow

add gate: gradient distributor
max gate: gradient router
mul gate: gradient… "switcher"?

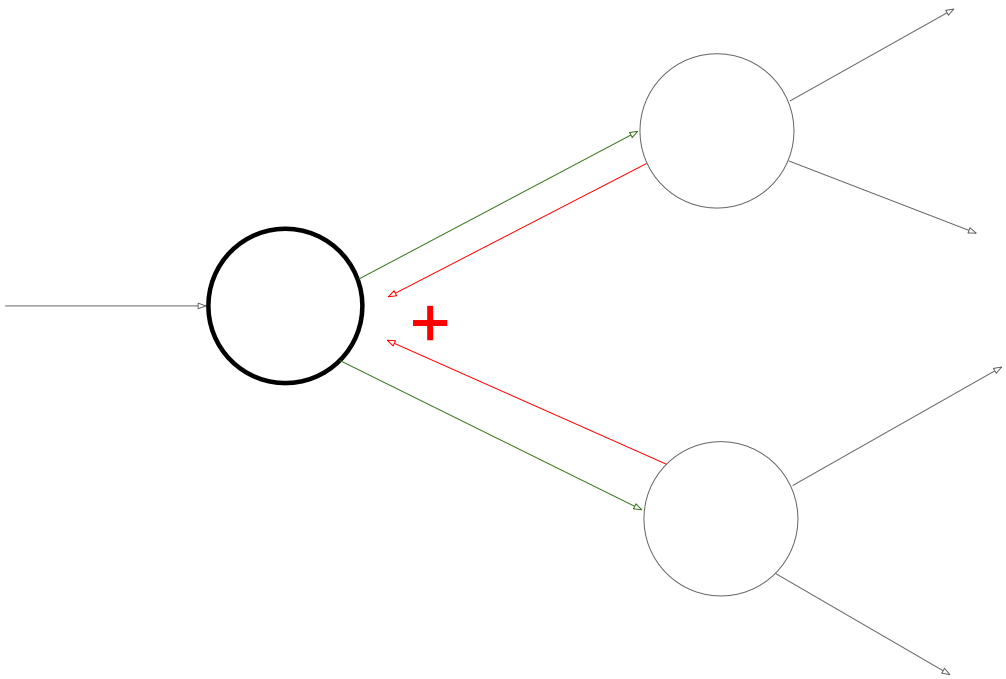


How do we handle gradients for nodes with multiple output connections?





Gradients add when there are multiple output connections



How do we backdrop through a RELU?

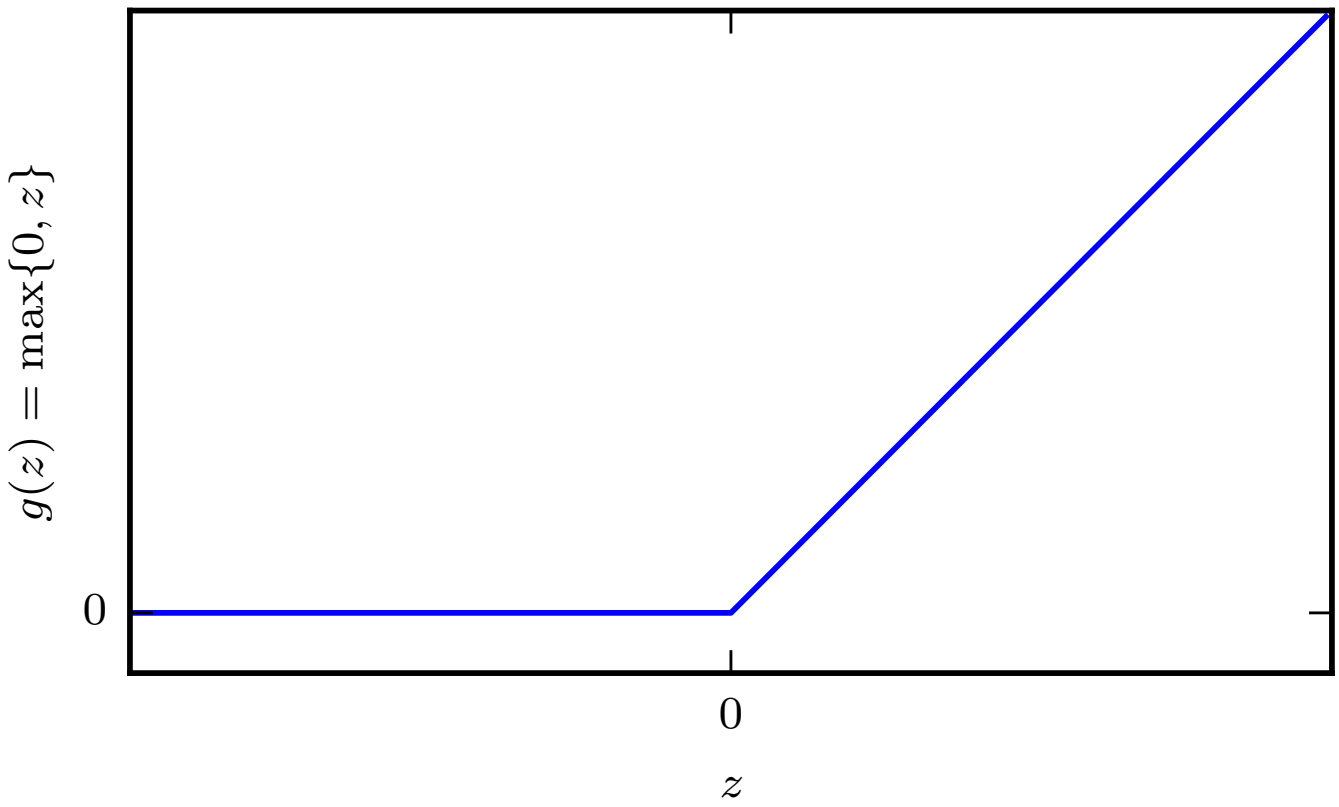


Figure 6.3

Only propagate gradient if input is non-zero

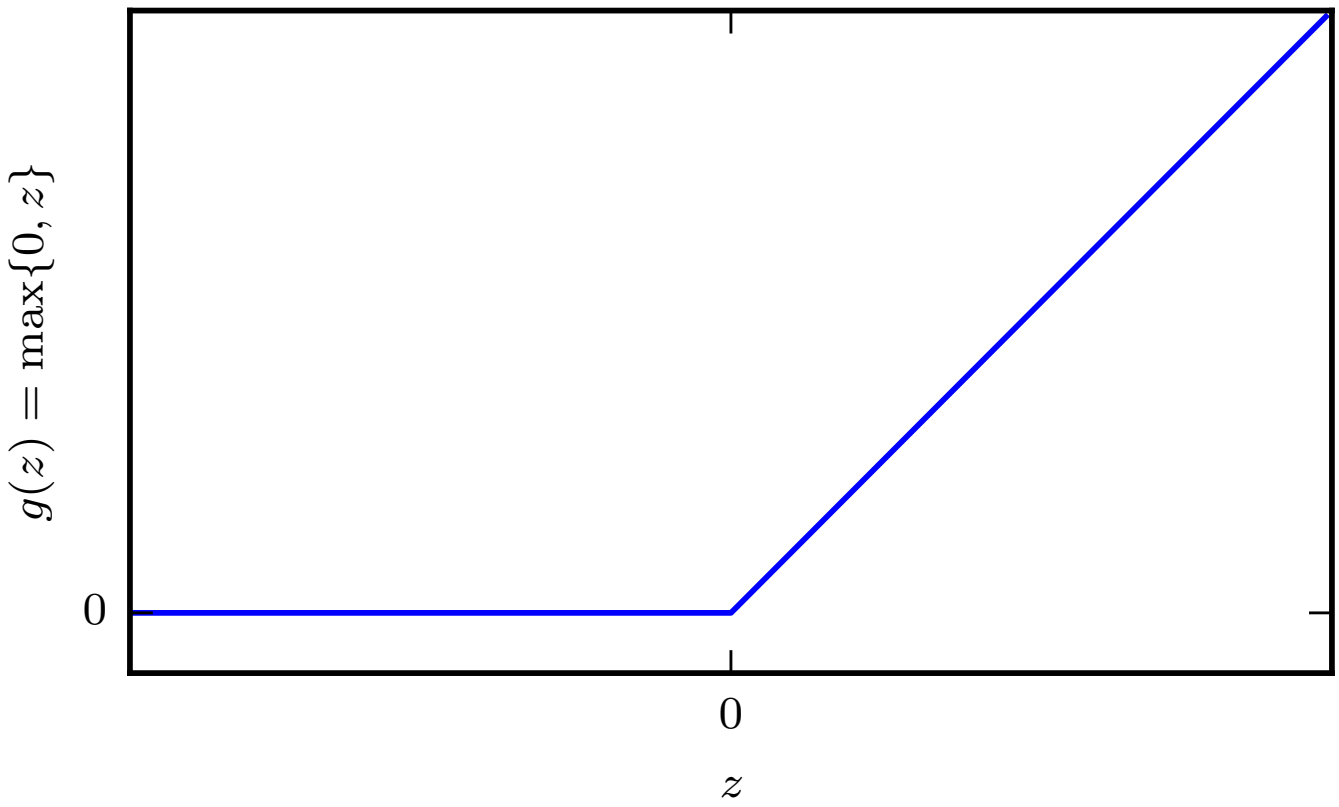


Figure 6.3

Managing gradients

The steepest gradient is not necessarily toward the optimum point

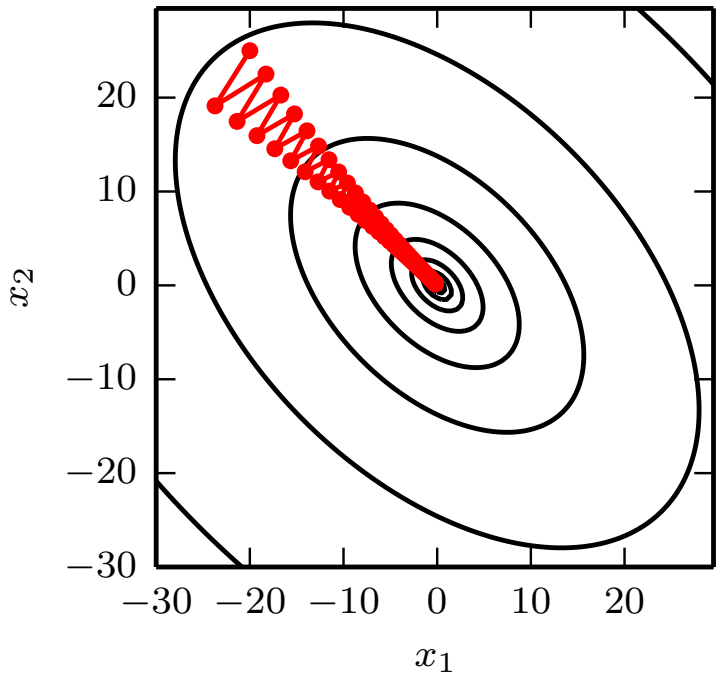
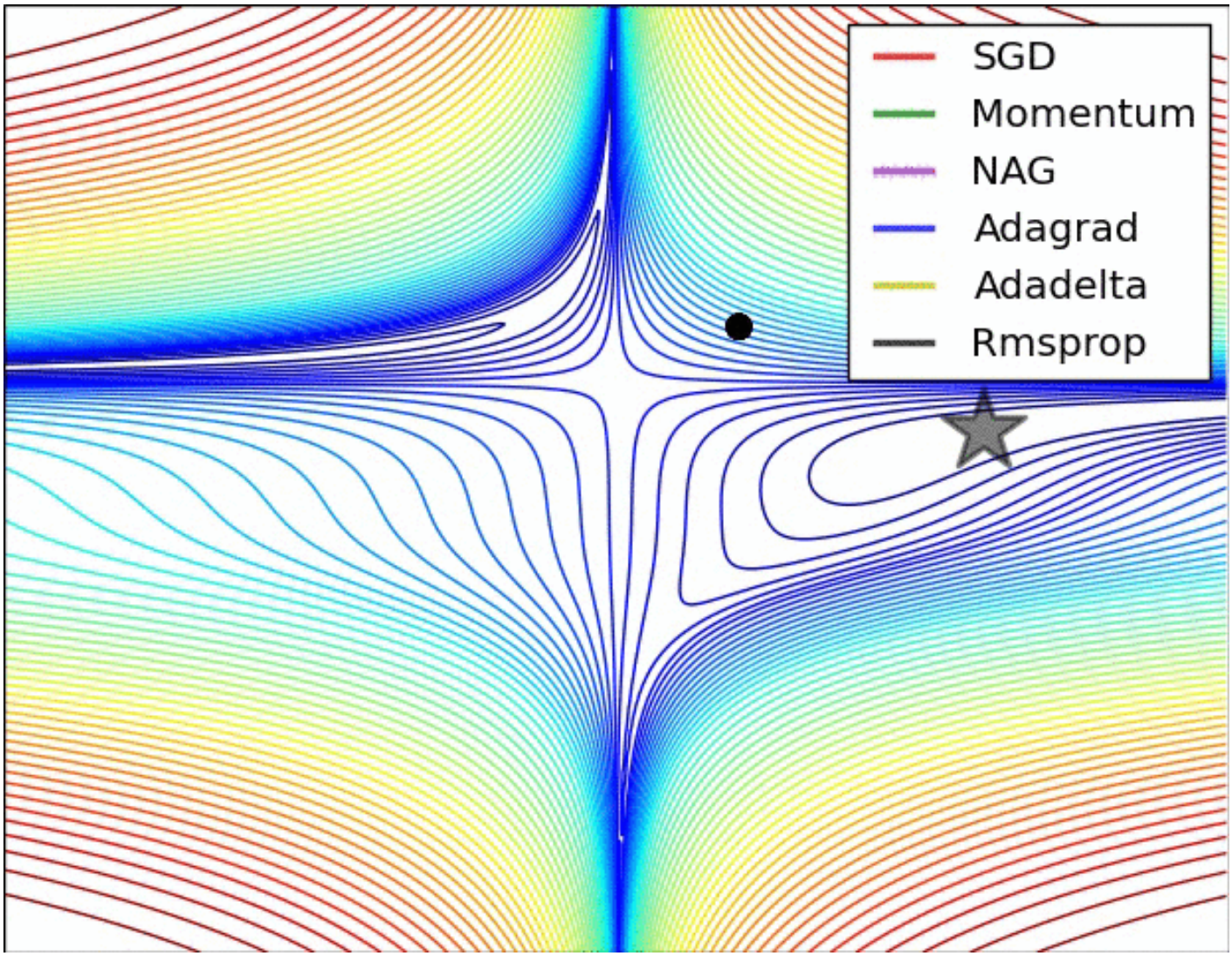
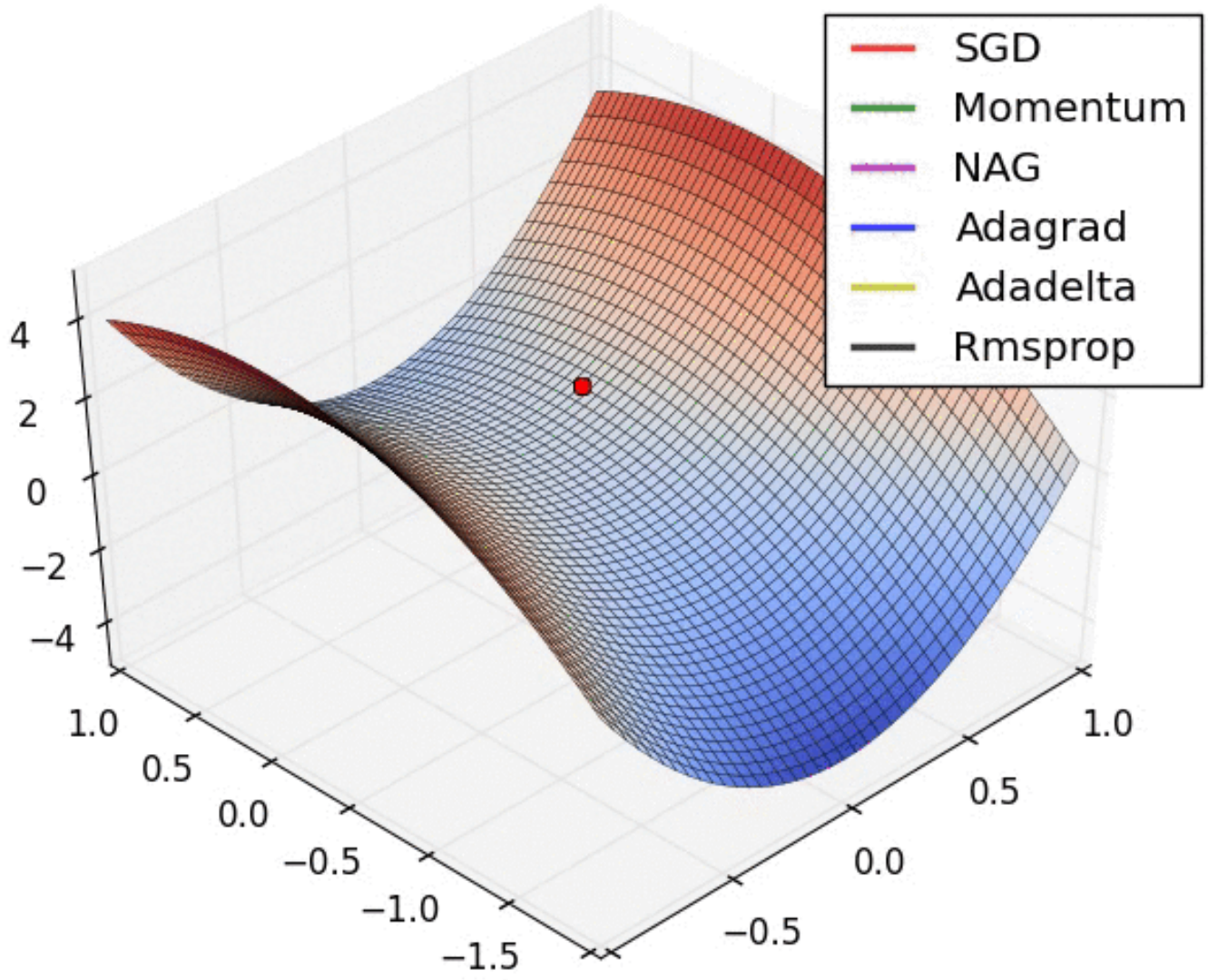


Figure 4.6

Example gradient management methods

- Stochastic gradient descent (update on each training value)
- Mini-batch gradient descent (average gradient over a mini-batch)
- Momentum methods
  - Adam - exponentially decaying average of past gradients and parameter specific learning rates
  - Adadelta - parameters specific learning rates with fixed memory window


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5

Evaluation metrics

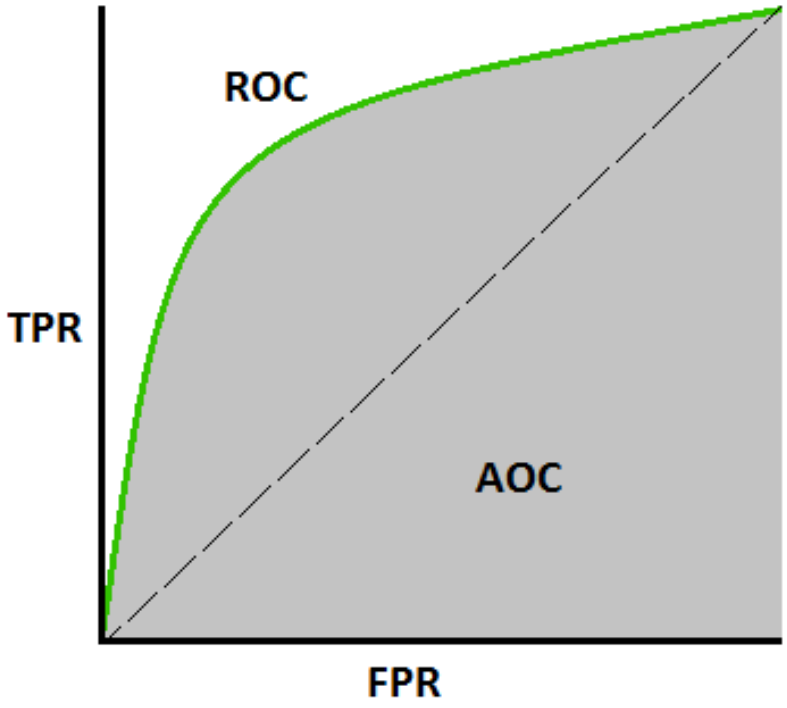
		True condition			
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$	F_1 score = $\frac{1}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}{2}$

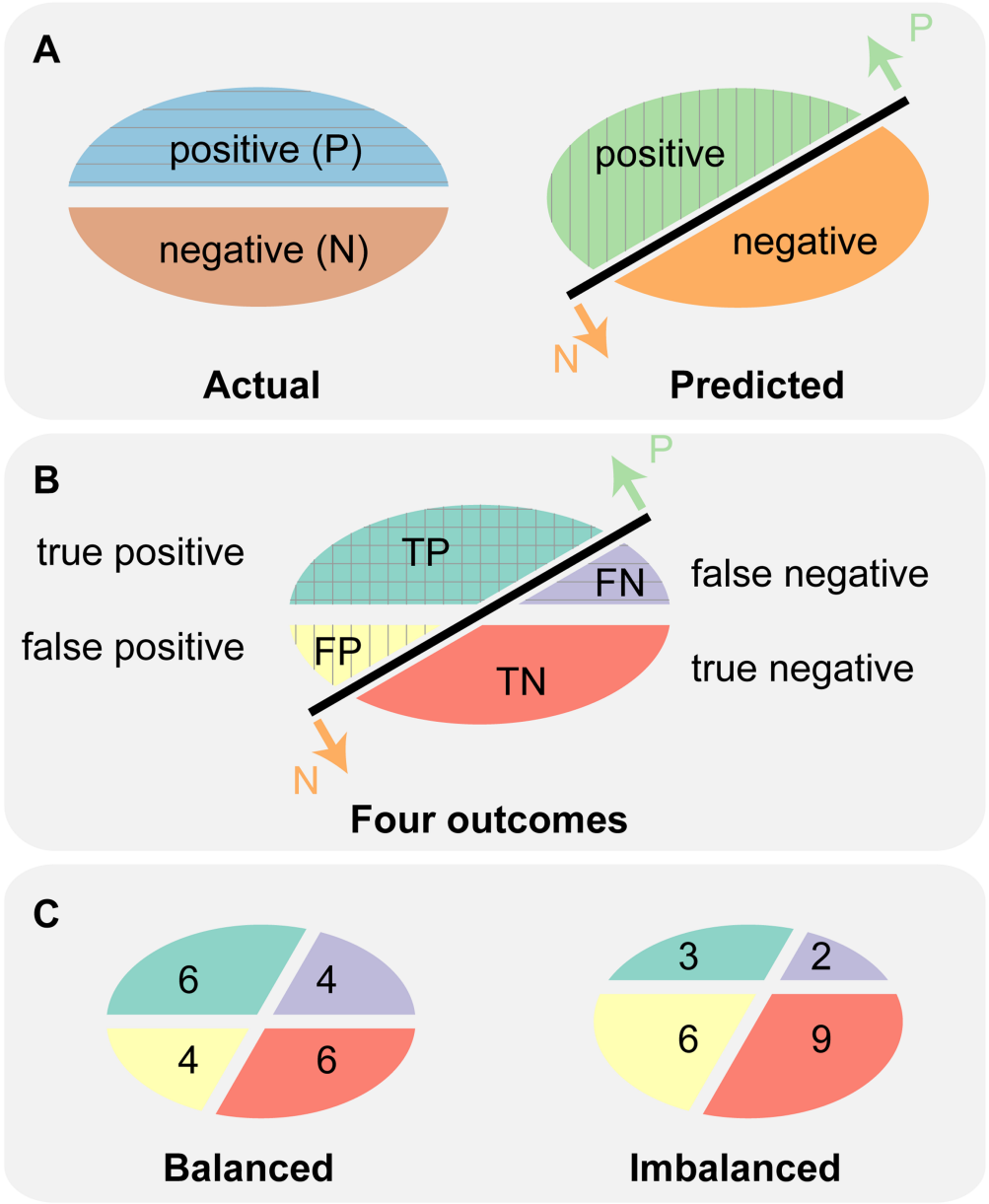
https://en.wikipedia.org/wiki/Sensitivity_and_specificity

Example confusion matrix

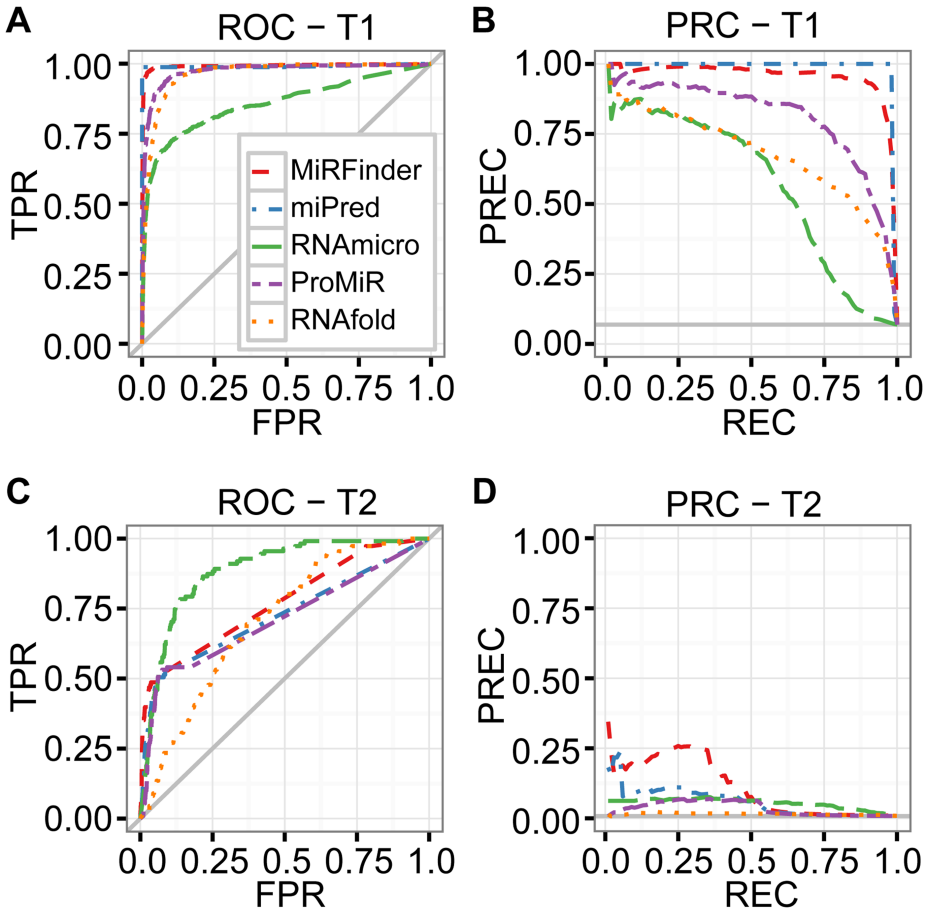
n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

A receiver operating characteristic (ROC) curve is made by changing the decision boundary




A
positive (P)
negative (N)
Actual
P
positive
negative
N
Predicted
B
true positive
false positive
TP
FN
FP
TN
P
N
false negative
true negative
Four outcomes
C
6
4
4
6
Balanced
3
2
6
9
Imbalanced

ROC and Precision-Recall curves are both useful



Training sets are best balanced

Substantially unbalanced training sets have undesirable properties

- For example, 1000 Class 1 examples, 100 Class 2 examples
- Result can be overfit models that do not generalize well to new examples
- Performance metrics can deceive as they represent underlying class distribution

How to deal with unbalanced training data

- Aquire more data
- Use different performance metrics (confusion matrix, precision-recall curves)
- Resample the data to be more balanced
- Use synthetic examples
- Try different methods (e.g. decision trees)
- Use methods specialized to anomaly or change detection

Controlling model complexity

We need to control model complexity for good generalization

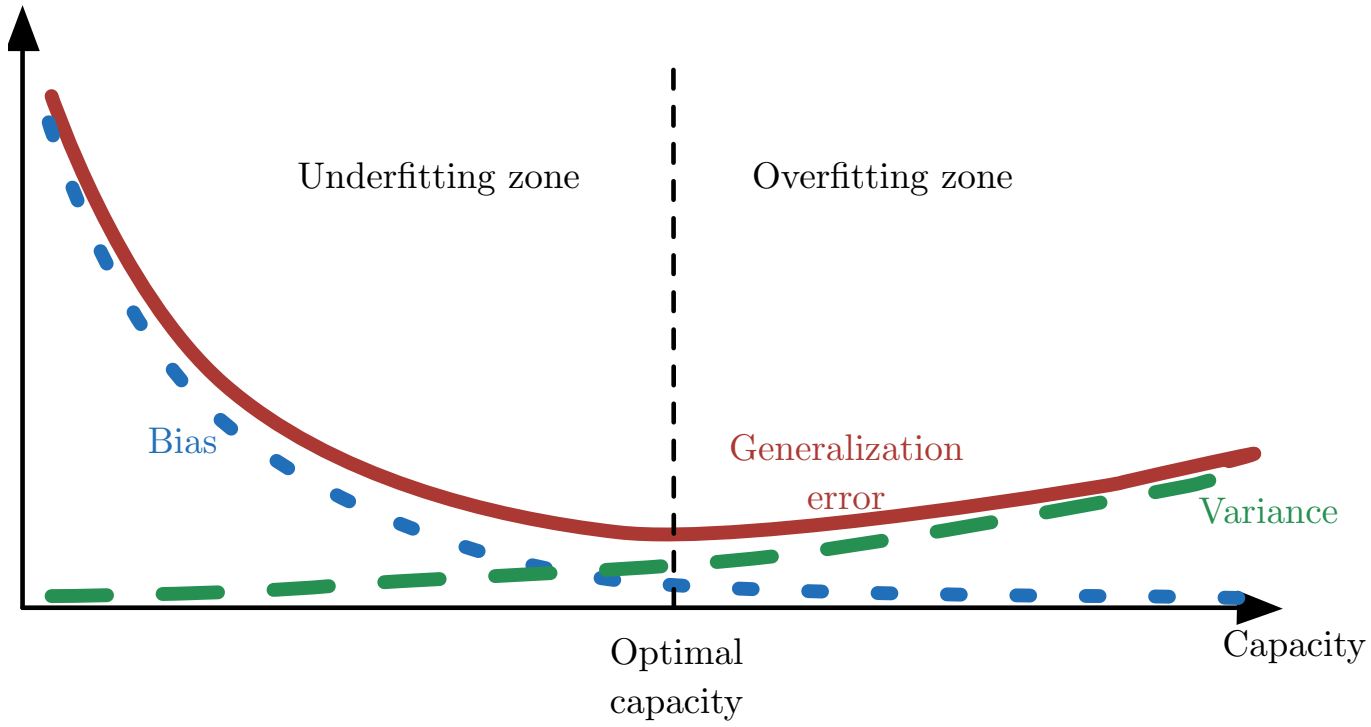
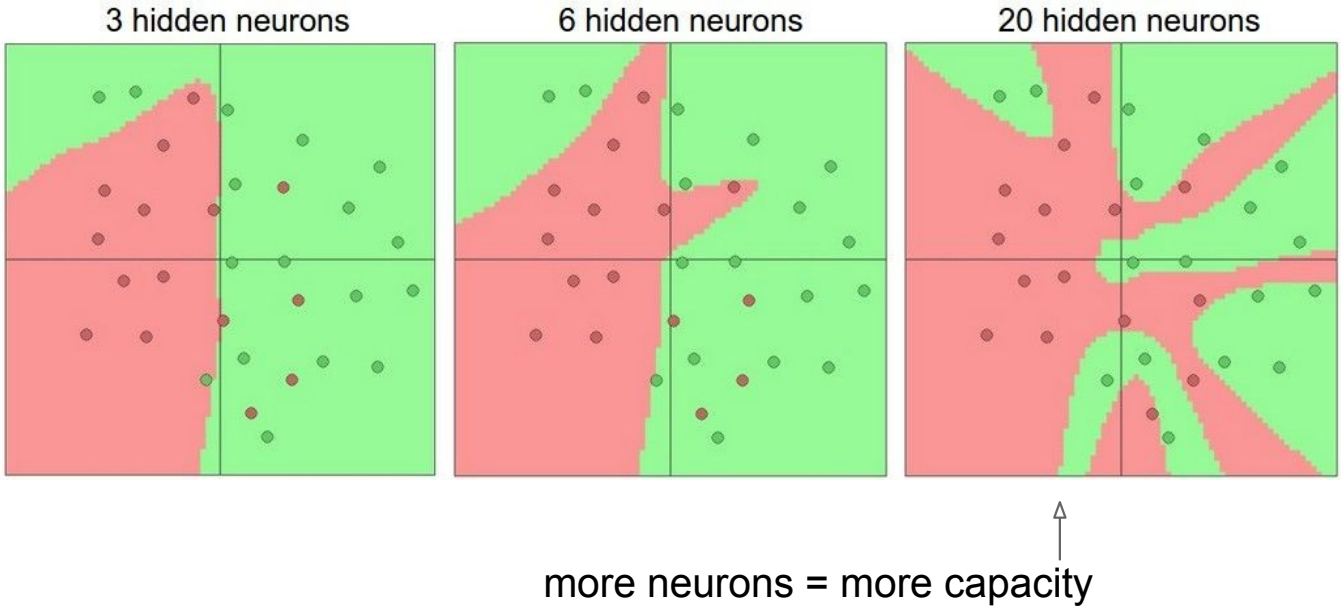


Figure 5.6

(Goodfellow 2016)

Setting the number of layers and their sizes



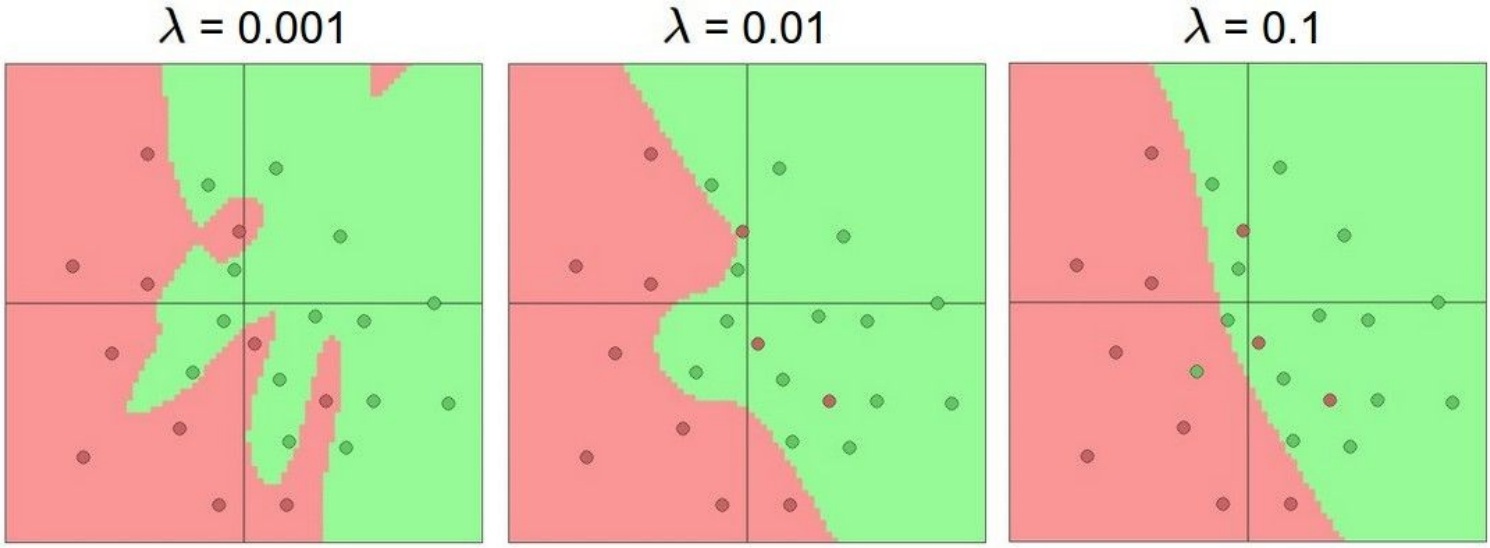
We can regularize our parameters

$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\frac{1}{2}\lambda \sum_k \sum_l W_{k,l}^2}_{\text{regularization loss}}$$

$$\frac{d}{dw}(\frac{1}{2}\lambda w^2) = \lambda w$$

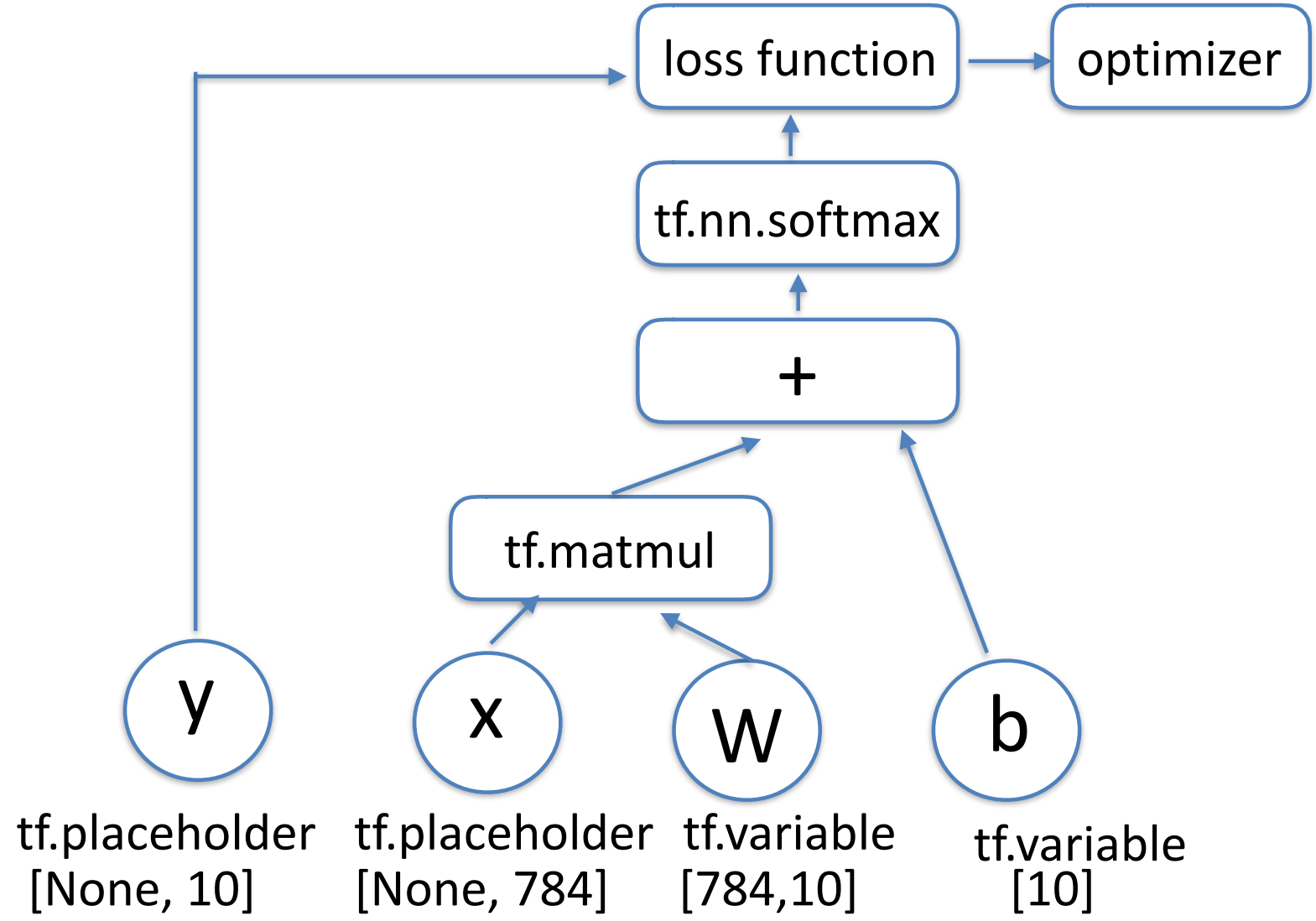
http://cs231n.github.io/neural-networks-case-study/

Do not use size of neural network as a regularizer. Use stronger regularization instead:



(you can play with this demo over at ConvNetJS: http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html)

Problem Set 1 Structure



Gradient of loss with respect to logits

$$z^i = W^T x^i + b \qquad p_k^i = \frac{e^{z_k^i}}{\sum_j e^{z_j^i}}$$

$$L_i = -\log(p_k^i) \quad \text{where} \quad k = y^i$$

$$\frac{\partial L_i}{\partial z_j} = p_j^i - \mathbf{1}(\mathbf{y_i} = \mathbf{j})$$ Update weights for j

$p^i = [0.6, 0.3, 0.1]$ then gradient $\rightarrow$ $[0.6, -0.4, 0.1]$

Correct Label

http://cs231n.github.io/neural-networks-case-study/

We can regularize our parameters

$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\frac{1}{2}\lambda \sum_k \sum_l W_{k,l}^2}_{\text{regularization loss}}$$

$$\frac{d}{dw}\left(\frac{1}{2}\lambda w^2\right) = \lambda w$$

http://cs231n.github.io/neural-networks-case-study/

FIN - Thank You