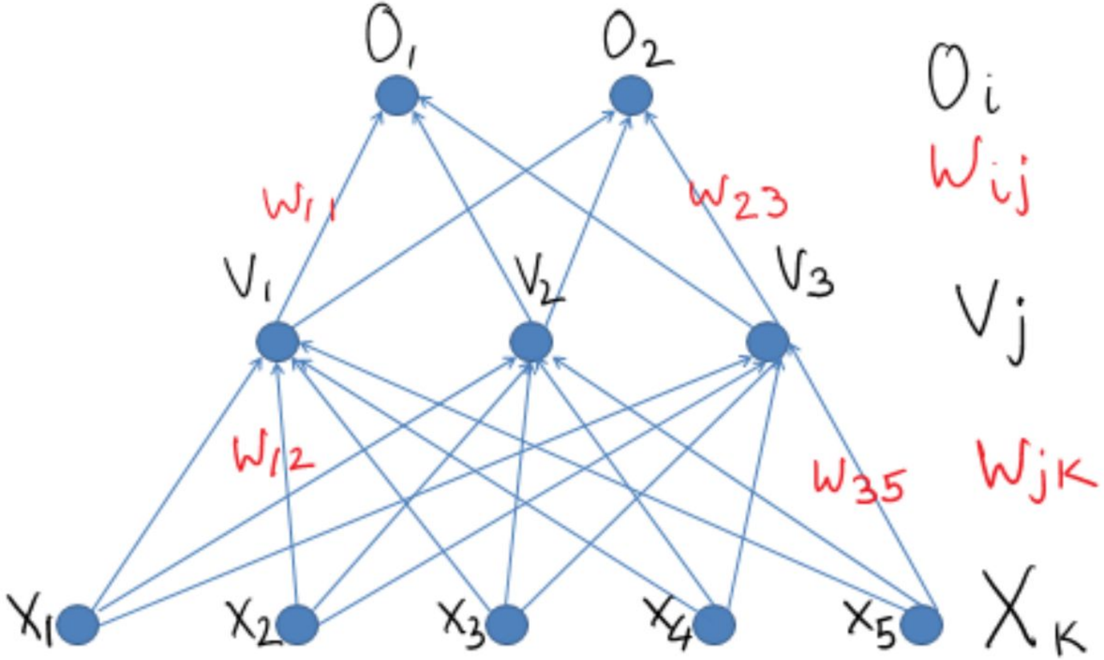
Neural Networks: Backpropagation

+ some SVMs

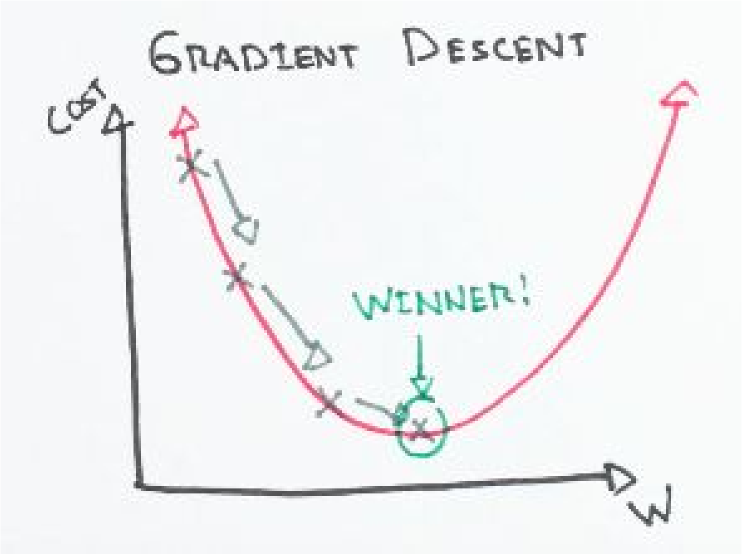
* Many figures shamelessly taken from Hinton, Karpathy, Efros

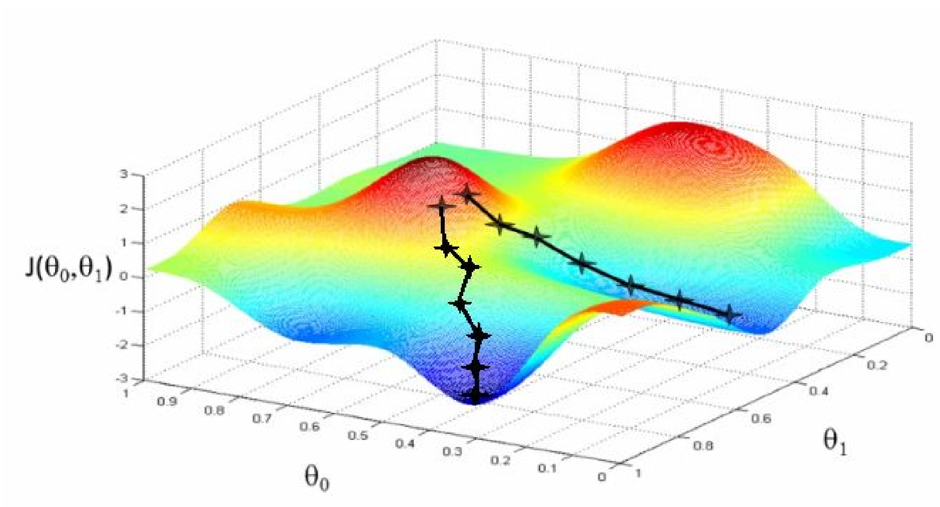
Training a neural network

$$O_i = g\left(\sum_j W_{ij}\, g\left(\sum_k W_{jk} x_k\right)\right)$$

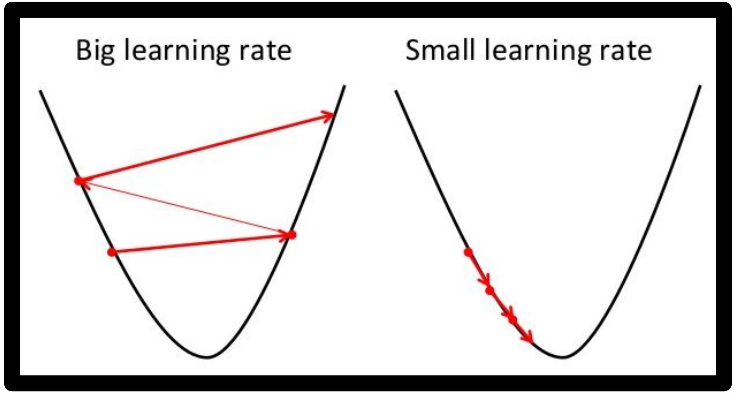


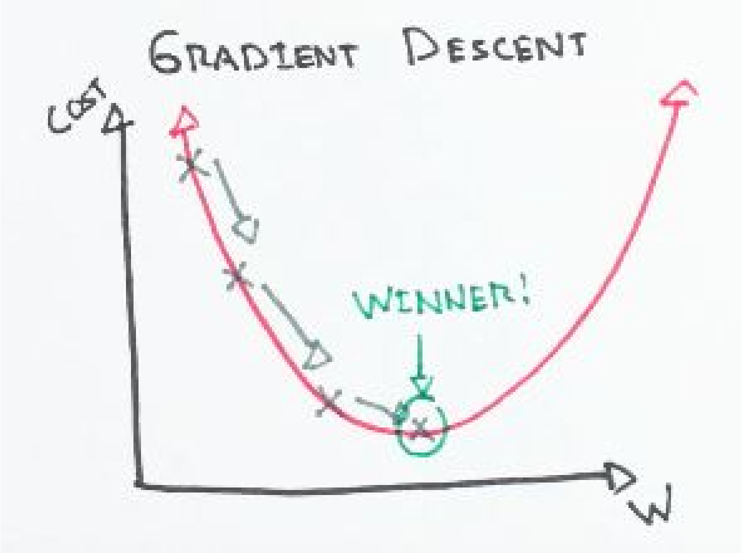
Gradient Descent

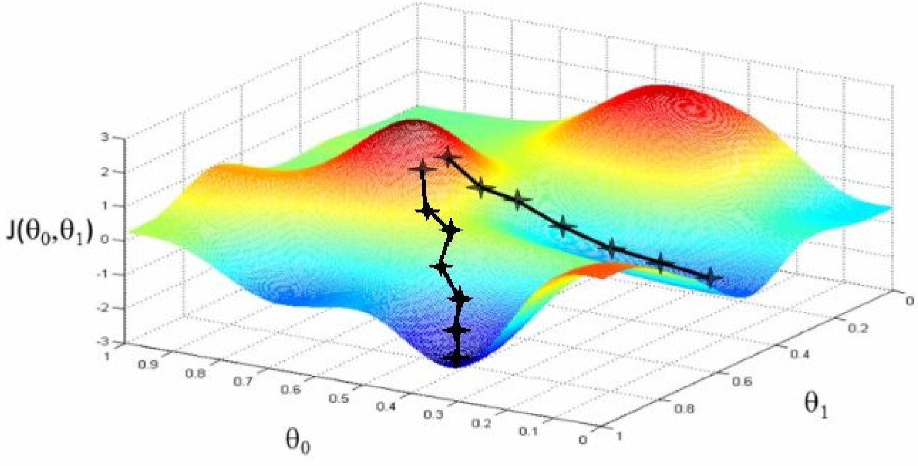




Gradient Descent







Gradient Descent

- **Numerical gradient**: easy to write ☺, slow ☹, approximate ☹
  - $O(N_w^2)$

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- **(native) analytic gradient**: exact ☺, compicated ☹, slow ☹
  - $O(N_w^2)$
- **back-propagation (cached analytic gradient)**: exact ☺, fast ☺, error-prone ☹
  - $O(N_w)$, similar to dynamic programming
  - glorified chain rule

In practice: Derive analytic gradient, check your implementation with numerical gradient

The idea behind Backprop

- We don’t know what the hidden units ought to do, but we can compute how fast the error changes as we change a hidden activity.
  - Each hidden activity can affect many output units and can therefore have many separate effects on the error. These effects must be combined.

- We can compute error derivatives for all the hidden units efficiently at the same time.
  - Once we have the error derivatives for the hidden activities, it’s easy to get the error derivatives for the weights going into a hidden unit.

$$f(x,y) = xy \qquad \rightarrow \qquad \frac{\partial f}{\partial x} = y \qquad \frac{\partial f}{\partial y} = x$$

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \qquad\qquad f(x+h) = f(x) + h\,\frac{df(x)}{dx}$$

$$f(x,y) = xy \qquad \rightarrow \qquad \frac{\partial f}{\partial x} = y \qquad \frac{\partial f}{\partial y} = x$$

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \qquad f(x+h) = f(x) + h\frac{df(x)}{dx}$$

Example: x = 4, y = -3. => f(x,y) = -12

$$\boxed{\frac{\partial f}{\partial x} = -3} \qquad \boxed{\frac{\partial f}{\partial y} = 4} \qquad \boxed{\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]}$$

partial derivatives

gradient

Compound expressions: $f(x,y,z) = (x+y)z$

$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

Compound expressions: $f(x, y, z) = (x + y)z$

$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

Chain rule:

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}}$$

Compound expressions: $f(x,y,z) = (x+y)z$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

```
# set some inputs
x = -2; y = 5; z = -4

# perform the forward pass
q = x + y # q becomes 3
f = q * z # f becomes -12

# perform the backward pass (backpropagation) in reverse order:
# first backprop through f = q * z
dfdz = q # df/fz = q, so gradient on z becomes 3
dfdq = z # df/dq = z, so gradient on q becomes -4
# now backprop through q = x + y
dfdx = 1.0 * dfdq # dq/dx = 1. And the multiplication here is the chain rule!
dfdy = 1.0 * dfdq # dq/dy = 1
```

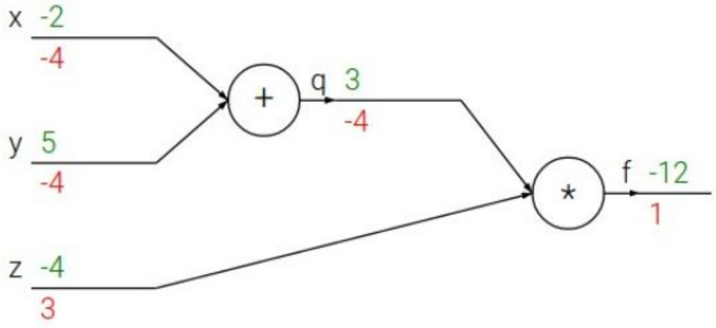
Compound expressions: $f(x,y,z) = (x+y)z$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$



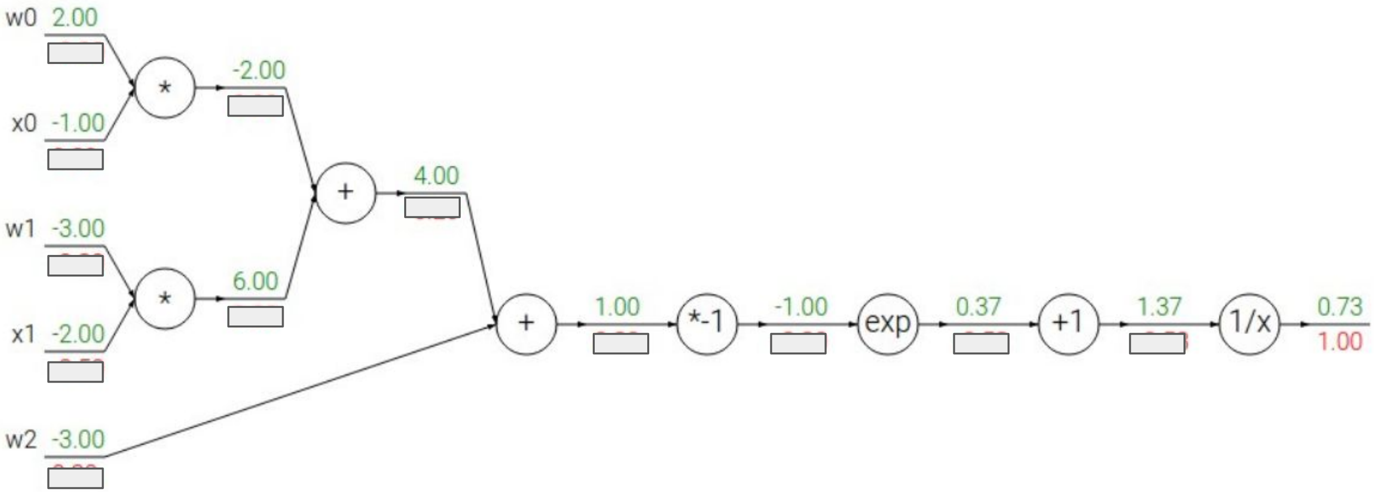

```
# set some inputs
x = -2; y = 5; z = -4

# perform the forward pass
q = x + y # q becomes 3
f = q * z # f becomes -12

# perform the backward pass (backpropagation) in reverse order:
# first backprop through f = q * z
dfdz = q # df/fz = q, so gradient on z becomes 3
dfdq = z # df/dq = z, so gradient on q becomes -4
# now backprop through q = x + y
dfdx = 1.0 * dfdq # dq/dx = 1. And the multiplication here is the chain rule!
dfdy = 1.0 * dfdq # dq/dy = 1
```

Another example

$$f(w,x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

Another example:

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



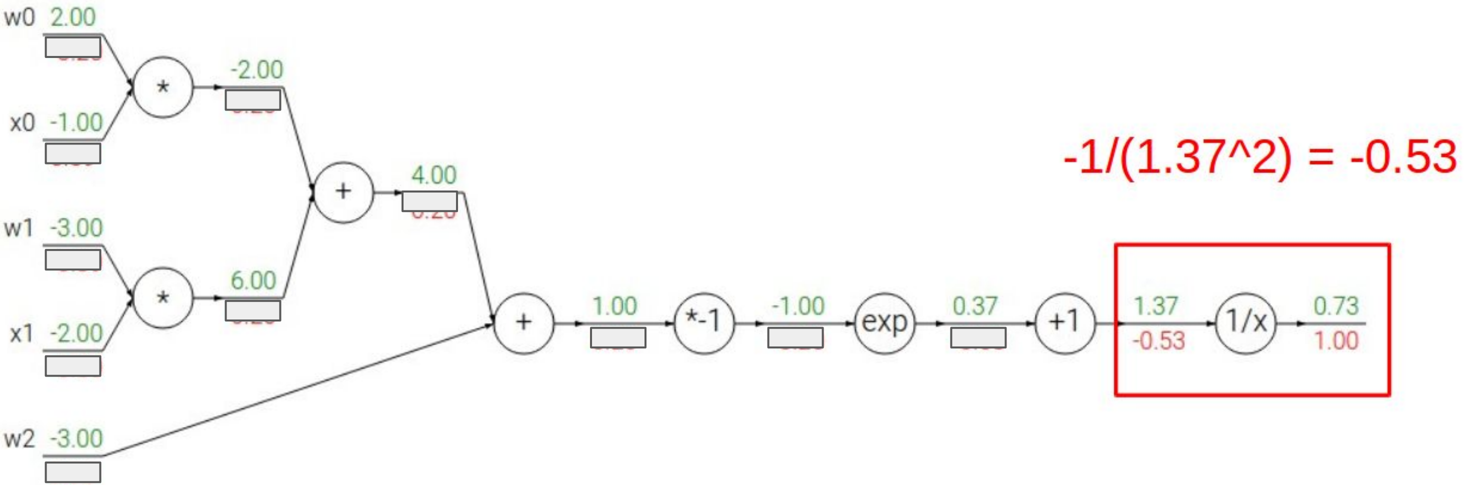
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

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

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

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

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



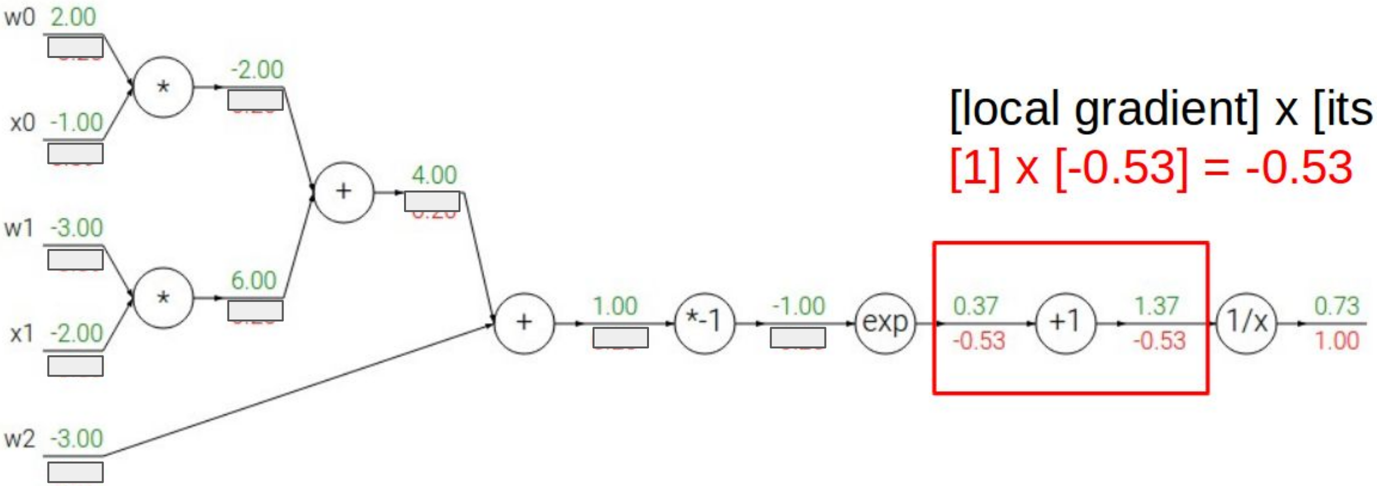
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

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

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

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

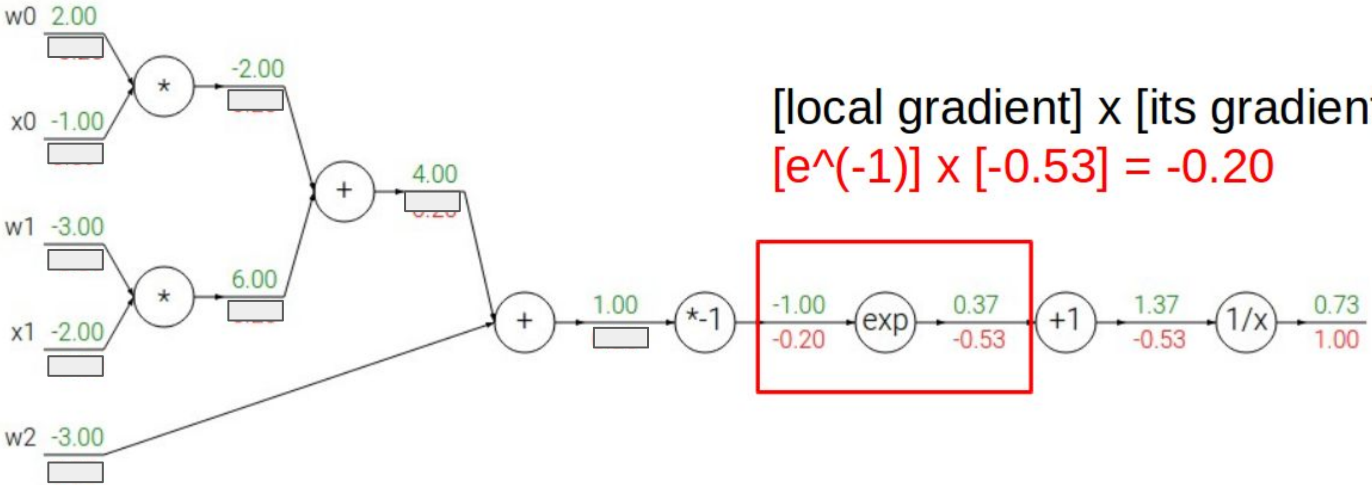
Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



[local gradient] x [its gradient]
[1] x [-0.53] = -0.53

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

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



[local gradient] x [its gradient]

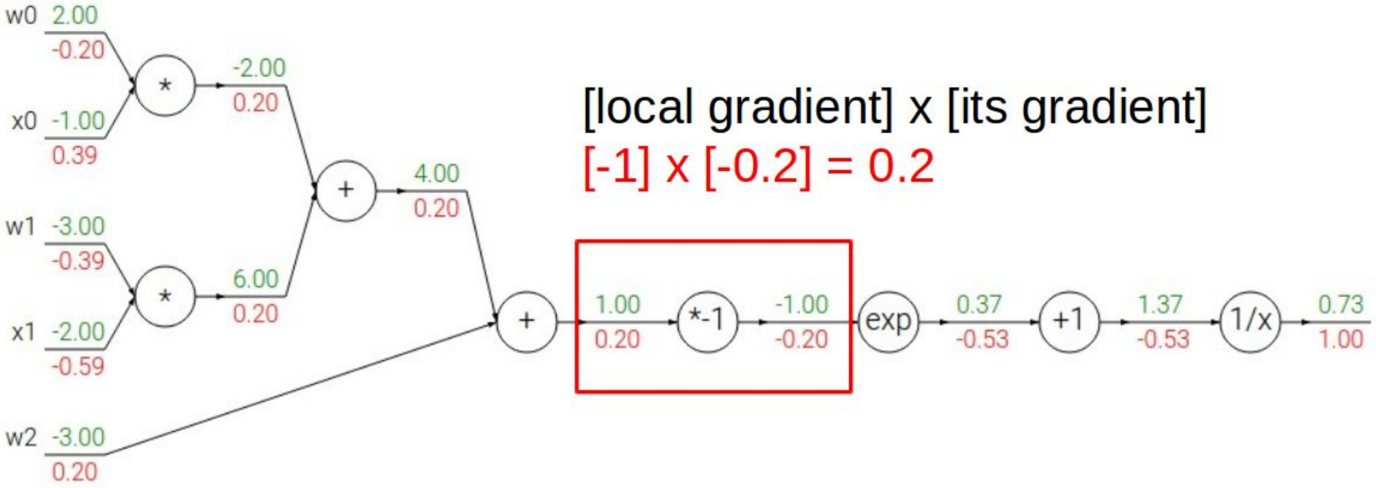
[e^(-1)] x [-0.53] = -0.20

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

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

Another example:

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



[local gradient] x [its gradient]

[-1] x [-0.2] = 0.2

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

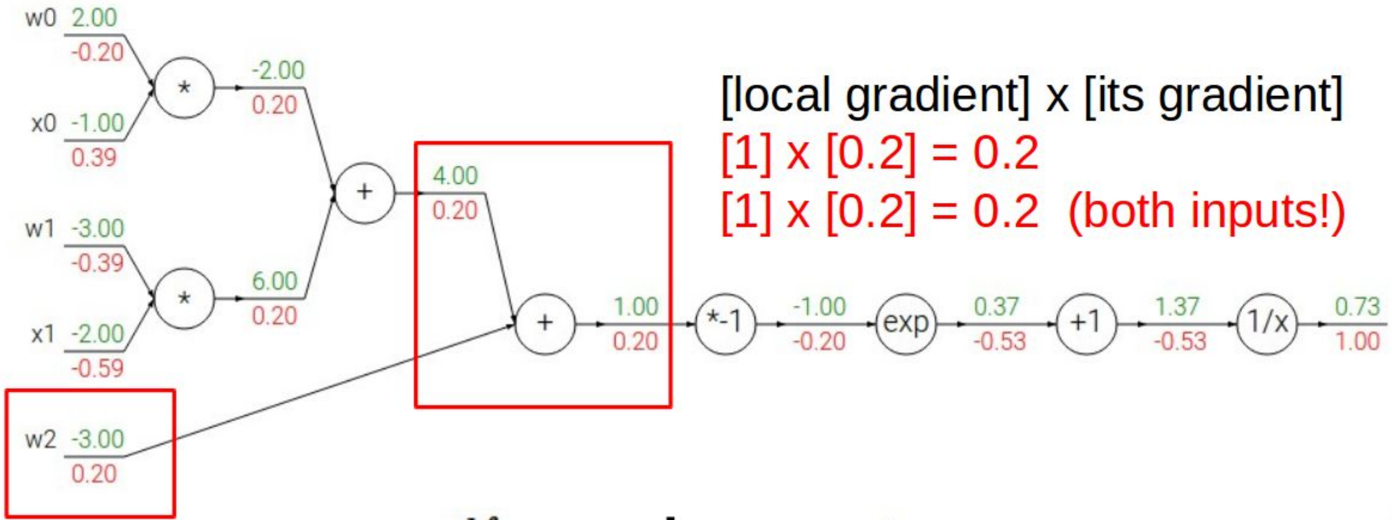
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

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

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

Another example:

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

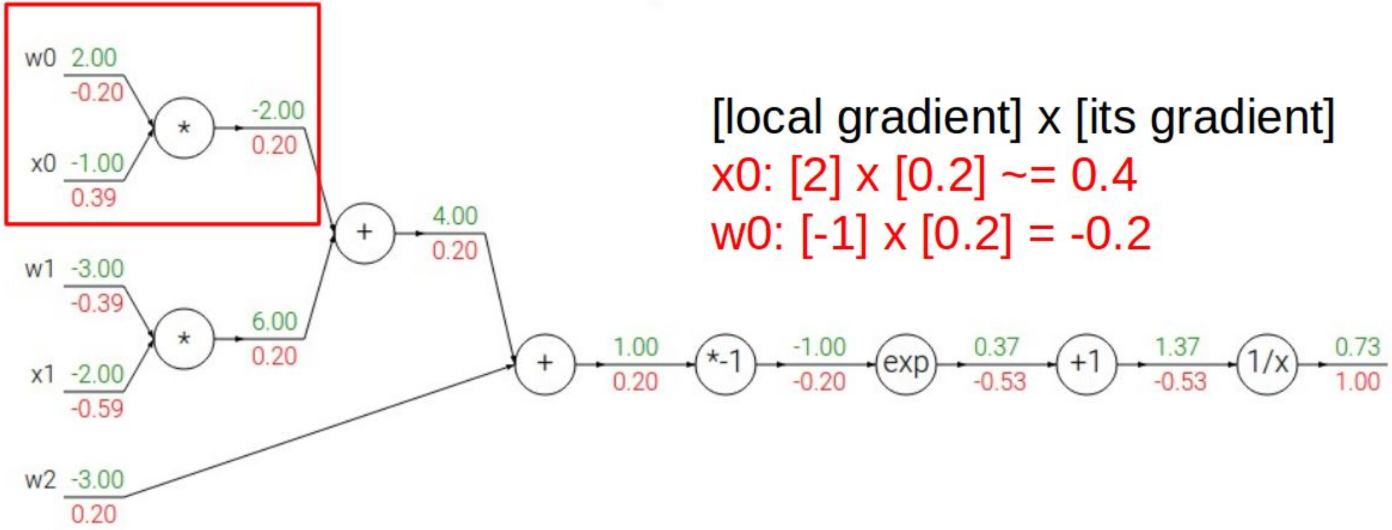


[local gradient] x [its gradient]
[1] x [0.2] = 0.2
[1] x [0.2] = 0.2 (both inputs!)

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

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

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

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

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

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

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

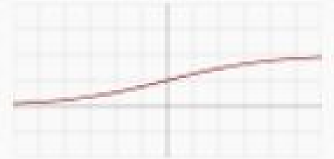
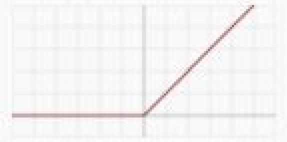
Every gate during backprop computes, for all its inputs:

[LOCAL GRADIENT] x [GATE GRADIENT]

Can be computed right away, even during forward pass

The gate receives this during backpropagation

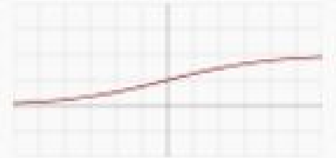
Backprop: Activation Functions

Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

Backprop: Activation Functions

<u>Derivatives</u>

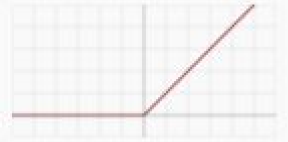
Logistic (a.k.a Soft step)



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

$$f'(x) = f(x)(1 - f(x))$$

Rectified Linear Unit (ReLU)

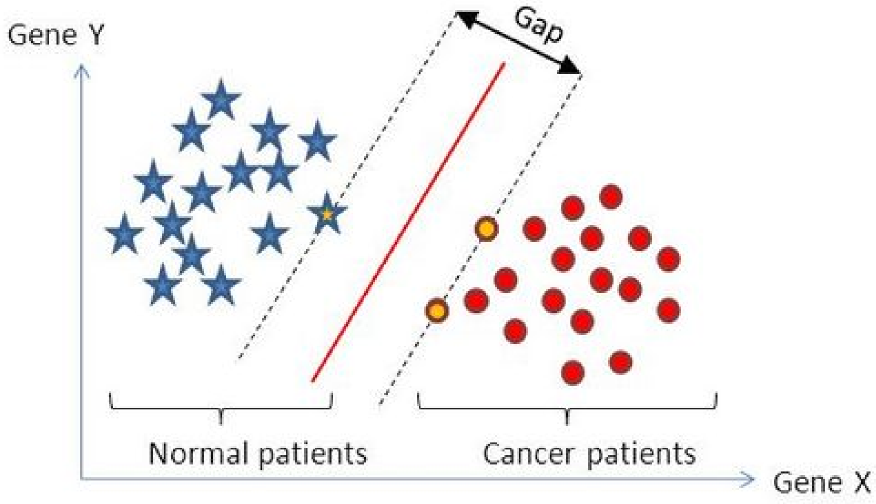


$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

$$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

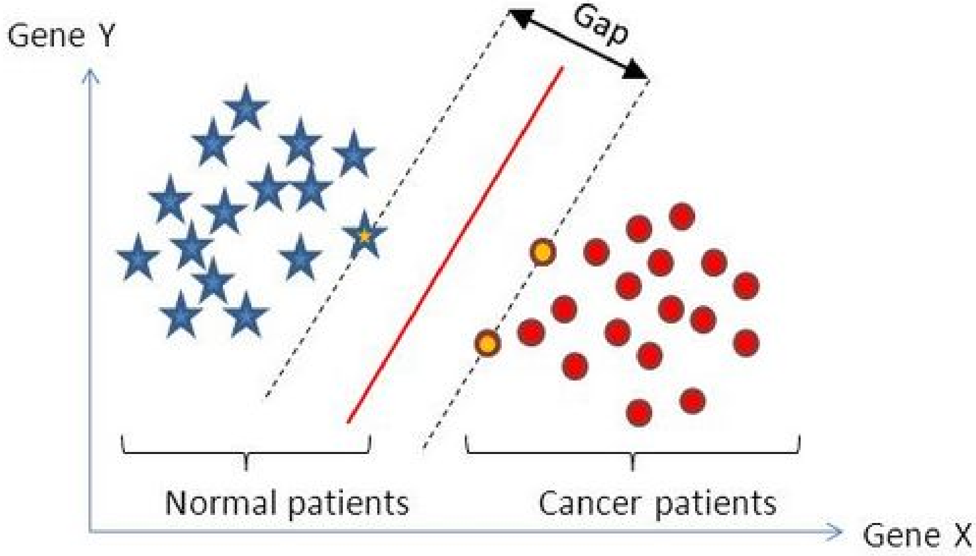
Before the glory of deep learning: SVMs

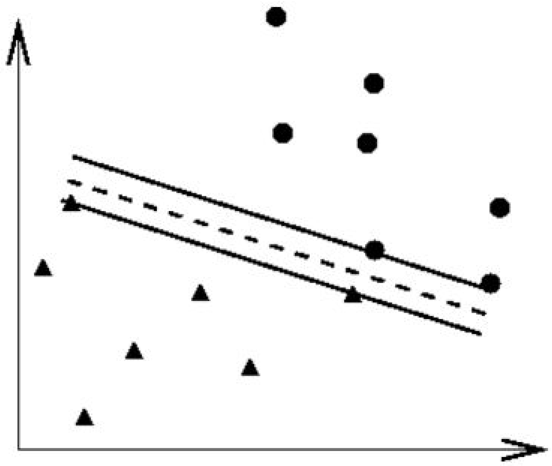
- Better theoretical understanding and guarantees
- Works well for many cases
- Simpler model to get working quickly



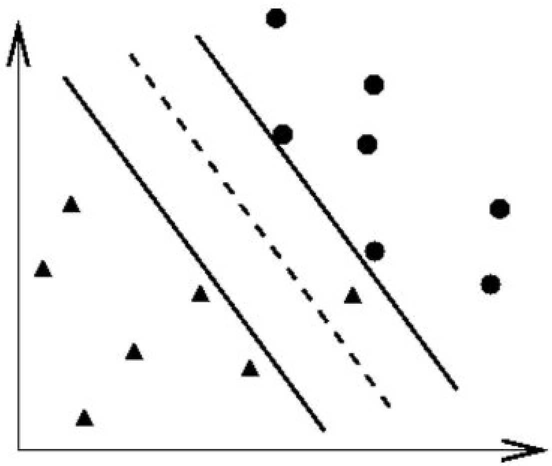
Support Vector Machines (SVMs)

Find a linear decision surface (**hyperplane**) that can separate patient classes AND has the largest distance (ie largest gap or **margin**) between border-line patients (i.e. **support vectors**)

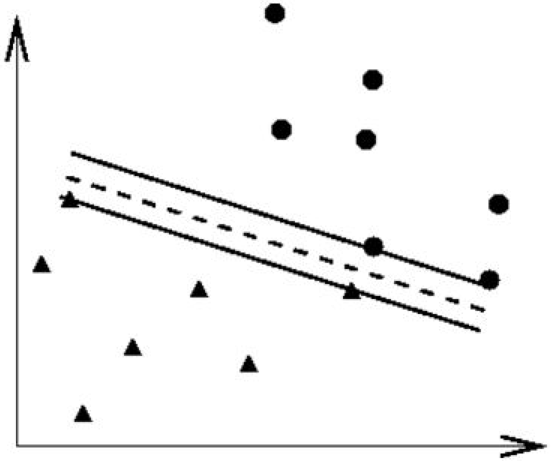




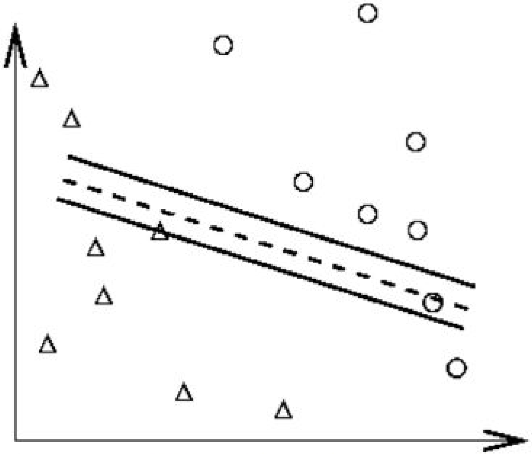
(a) Training data and an overfitting classifier



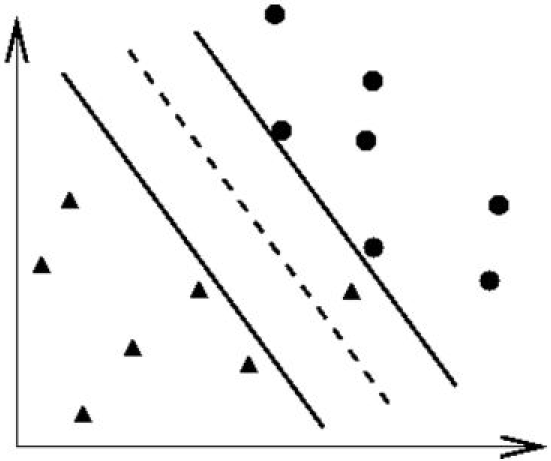
(c) Training data and a better classifier



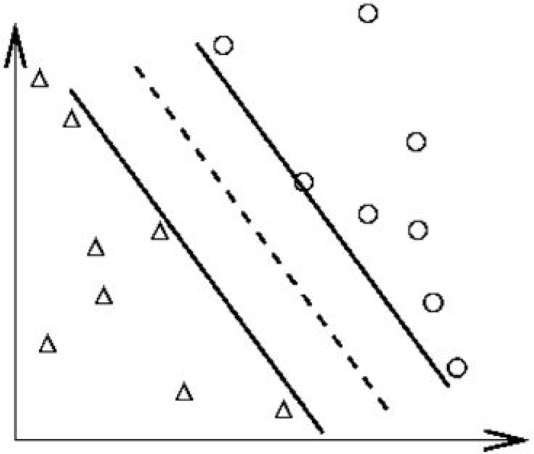
(a) Training data and an overfitting classifier



(b) Applying an overfitting classifier on testing data



(c) Training data and a better classifier



(d) Applying a better classifier on testing data

Lab 4 Tools and Hints

Libraries and Imports:

```
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC

from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.feature_extraction import DictVectorizer

from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV, ParameterGrid

import numpy as np
```

Lab 4 Tools and Hints:

```
clf = SVC(kernel='linear', random_state=234)

clf.fit(X_train, y_train)
accuracy = clf.score(X_train, y_train)
print('accuracy: {:.3f}%'.format(accuracy*100))
```

```
clf = MLPClassifier(hidden_layer_sizes=(10),
                    activation='logistic',
                    solver='lbfgs',
                    random_state=2563)

clf.fit(X_train, y_train)
accuracy = clf.score(X_train, y_train)
print('accuracy: {:.3f}%'.format(accuracy*100))
```

Lab 4 Tools and Hints:

```
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print('accuracy: {:.3f}%'.format(accuracy*100))
```