Neural Netwo

Story so far

- Neural networks are universal approximators
  - Can model any odd thing
  - Provided they have the right architecture
- We must train them to approximate any function
  - Specify the architecture
- Networks are trained to minimize total "error" on a training set
  - total "err

Recap: Gradient Descent Algorithm

- In order to minimize any function

Training Neural Nets by Gradient Descent

-

Training Neural Nets by Gradient Descent

-

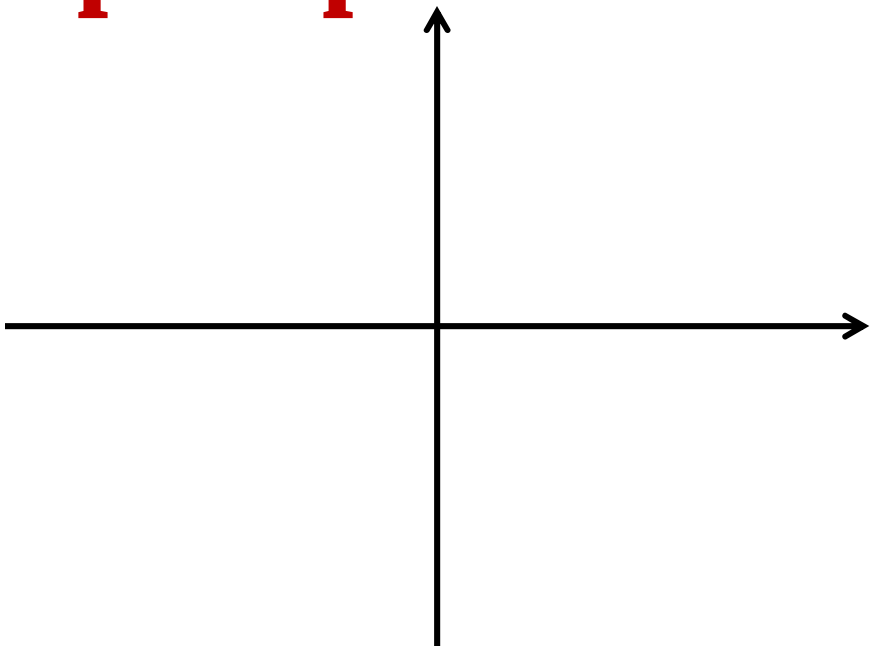
Computing

Computing

Recap: Backpropagation for training

- **Initialize all weights and biases**

Backprop fails to separate where perceptron succeeds



- Brady, Raghavan, Slawny, ’89
- Simple problem, 3 training instances, single neuron
- Perceptron training rule trivially find a perfect solution

Backprop vs. Perceptron

- **Back propagation using logistic function and**

Backprop vs. Perceptron

- **Now add a fourth point**
- t

Backprop

- **Consider backprop:**
- **Contribution of Ɛ urth point to derivative of** L

Backprop

- **The fourth point at**

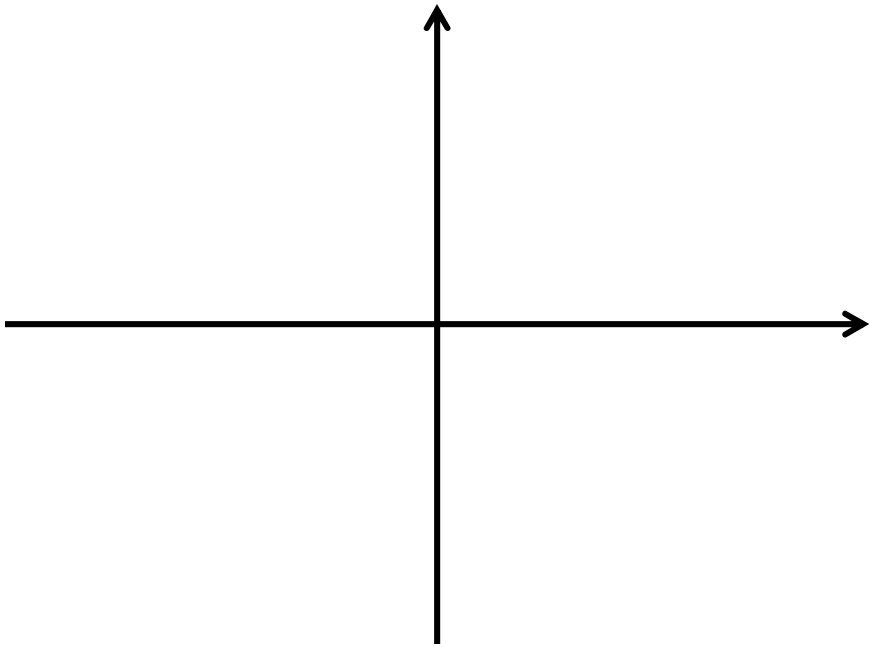
Backprop

- Solution found by backprop
- Does not separa] B] 2 e poin d’ ev en s я hоug я h s я he points are linea

Backprop fails to separate where perceptron succeeds

- **Brady, Raghavan, Slawny, ’89**
- **Several**

A more complex problem



Backprop fails to separate even when possible

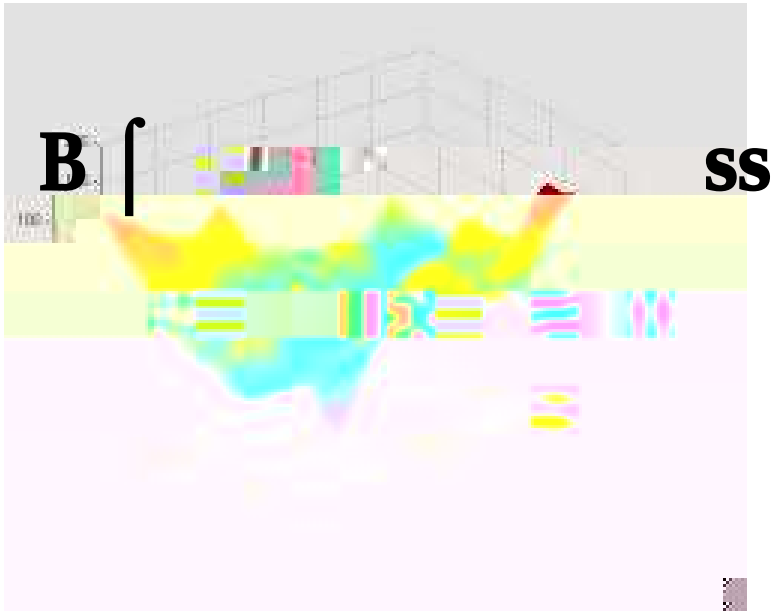
- This is not 9 o ŏ T oh to inio 3 o

Backprop fails to separate even when possible

-

The Error Surface

- The example (and statements) earlier assumed the loss objective had a ƒ siea B ∫ ss



Convergence

-

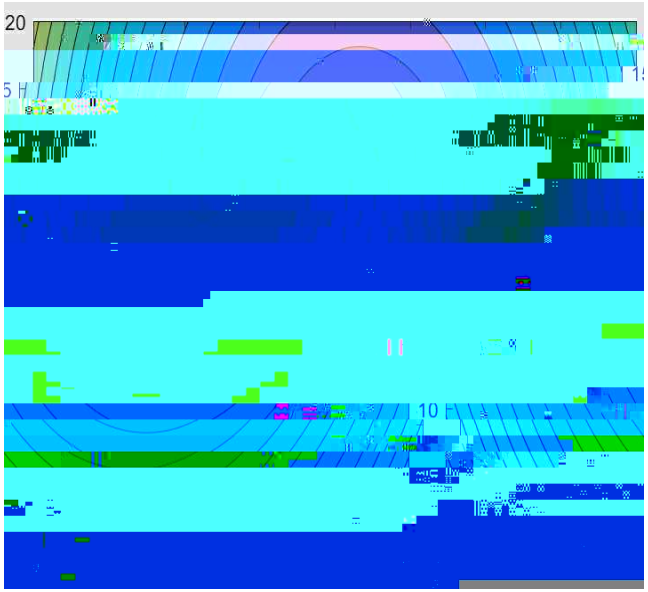
A quick tour of (convex) optimization

Multivariate Quadratic with Diagonal

“Descents” are uncoupled

- **The optimum of each coordinate is not affected by the other coordinates**
  - **I.e. we could optimize each coordinate independently**
-

Dependence on learning rate


20
10

Scaling the axes

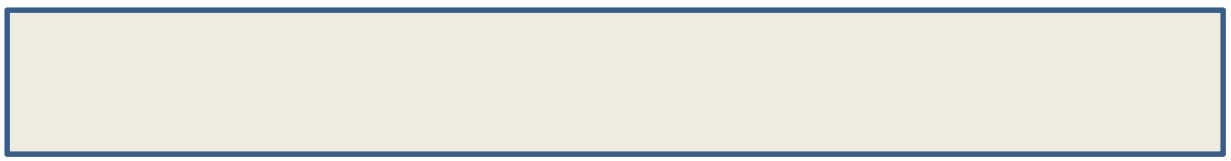
-

Returning to our problem

-

Returning to our problem

-



Minimization by Newton's method

- **Iterated localized optimization with quadratic approximations**

- **Iterated localized optimization with quadratic approximations**

Minimization by Newton's method

- **Iterated localized optimization with quadratic approximations**

Minimization by Newton's method

- **Iterated localized optimization with quadratic approximations**

Issues: 2. The learning rate

Issues: 2. The learning rate

- **For complex models such as neural networks the loss function is often not convex**

Story so far : cond-ordtr mtthods

-

Story so far : Learning rate

- **Divergence-causing learning rates may not be a bad thing**

Derivative-inspiree

Rprop

- **Compute the derivative in the new location**
  - **I the derivative has not changed sign from the previous location, increase the step size and take a longer step**
    - Δ

Rprop

-

Rprop

- **Compute the derivative in the new location**

Rprop

- **Compute the derivative in the new location**

QuickProp

QuickProp: Modification 1

- **It treats each dimension independently**
- **For**

QuickProp: Modification 2

- It approximates the second derivative through finite differences

QuickProp

- **Updates are independent for every parameter**
- **For every layer**

QuickProp

- **Updates are independent for every parameter**
- **For every layer**

Quickprop

A closer look at the convergence problem

-

The momentum methods

- **Maintain a aunning ave**
  past ste
  - **In dire**
    co

ace **aMis smo**

Momentum Update

- **The momentum method maintains a running average of all gradients until the current step**

Δ

Training with momentum

- **Initialize all weights W**

Momentum Update

- **The momentum method**

Momentum Update

- **The momentum method**

$$\Delta W$$

Nestorov's Accelerated Gradient

- Change the order of operations
- At any iteration, to compute the current step:
  - First extend the previous step
  - Then compute the gradient step at the resultant position

Nestorov's Accelerated Gradient

- Change the order of operations
- At any iteration, to compute the current step:
  - First extend the previous step
  - Then compute the gradient step at the resultant position
  - Add the two to obtain the final step

Training with Nestorov

- **Initialize all weights**