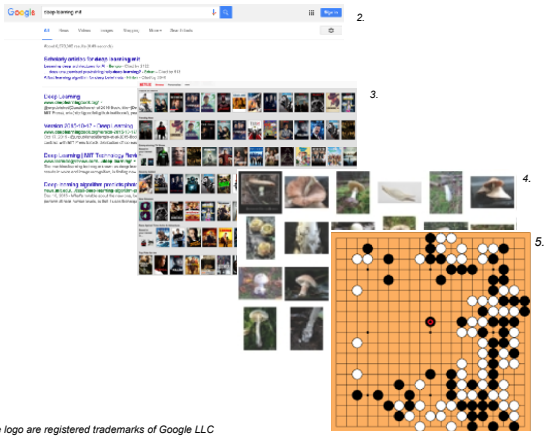
Machine Learning Lecture 1 [1]

Machine learning is everywhere

- Search, content recommendation, image/scene analysis, machine translation, dialogue systems, automated assistants, game playing, sciences (biology, chemistry, etc), …



2,4. Google and the Google logo are registered trademarks of Google LLC

3. Screenshot of Netflix (n.d.), (c) Netflix, Inc.

5. © Senseis Library

Machine learning: what is it?

- A brief definition

  Machine learning as a discipline aims to design, understand and apply computer programs that learn from experience (i.e., data) for the purpose of modeling, prediction, or control

Prediction problems

- About future events

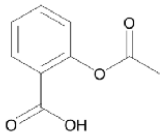
- Also collision avoidance, monitoring, medical risk, etc.

Prediction problems

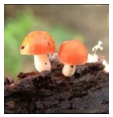
- About properties we don't yet know

6. © Black Bear Pictures

would I like this movie?



soluble in water?



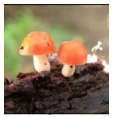
7. , © Neural Information Processing Systems Foundation, Inc.

what is the image about?

"ML is very cool" what is it in Spanish?

Example: supervised learning

- It is easier to express tasks in terms of examples of what you want (rather than how to solve them)
- E.g., image classification (1K categories)

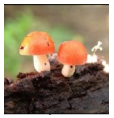
Image	**Category**
	mushroom
	cherry
...	...

. © Neural Information Processing Systems Foundation, Inc.

8. © Geoffrey Hinton, University of Toronto.

Example: supervised learning

- It is easier to express tasks in terms of examples of what you want (rather than how to solve them)
- E.g., image classification (1K categories)

Image	**Category**
	mushroom
	cherry
...	...

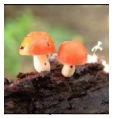
© Neural Information Processing Systems Foundation, Inc

© Geoffrey Hinton, University of Toronto.

- Rather than specify the solution directly (hard), we automate the process of finding one based on examples

Example: supervised learning

- It is easier to express tasks in terms of examples of what you want (rather than how to solve them)
- E.g., image classification (1K categories)

Image		**Category**
$h\left(\right.$  $; \theta \left. \right)$	$=$	mushroom
		cherry
...		...

© Neural Information Processing Systems Foundation, Inc

© Geoffrey Hinton, University of Toronto.

- Rather than specify the solution directly (hard), we automate the process of finding one based on examples

Example: supervised learning

- It is easier to express tasks in terms of examples of what you want (rather than how to solve them)
- No limit to what you can learn to predict…

English		**Spanish**
$h($ Is it real? ; $\theta)$	$=$	¿Es real?
Will it continue?		¿Continuará?
For how long?		¿Por cuanto tiempo?
…		…

- Already in production for some language pairs (Google)

A concrete example

- Learning to predict preferences from just a little data…

11.© 2020 Sony Pictures Digital Productions Inc.

10. © Twentieth Century Fox.

9. © 2020 The Studios At Paramount.

12.© Disney


© Black Bear Pictures

A concrete example

- Learning to predict preferences from just a little data…

11.© 2020 Sony Pictures Digital Productions Inc.

10. © Twentieth Century Fox.

9. © 2020 The Studios At Paramount.

12.© Disney


© Black Bear Pictures

-1 -1 +1 +1 ?, ?, ….

A concrete example

- Learning to predict preferences from just a little data…

11.© 2020 Sony Pictures Digital Productions Inc.

10. © Twentieth Century Fox.

9. © 2020 The Studios At Paramount.

12.© Disney


© Black Bear Pictures

-1 -1 +1 +1 ?, ?, ….

comedy? top lead?

$x^{(1)} = [1\,0\,0\,1\,1\,\ldots\,0]^T$ (feature vector)

action? romance? Spielberg?

Supervised learning

- Learning to predict preferences from just a little data…


11.© 2020 Sony Pictures Digital Productions Inc.

$x^{(1)}$

-1


10. © Twentieth Century Fox.

$x^{(2)}$

-1


9. © 2020 The Studios At Paramount.

$x^{(3)}$

+1


12.© Disney

$x^{(4)}$

+1

Training set

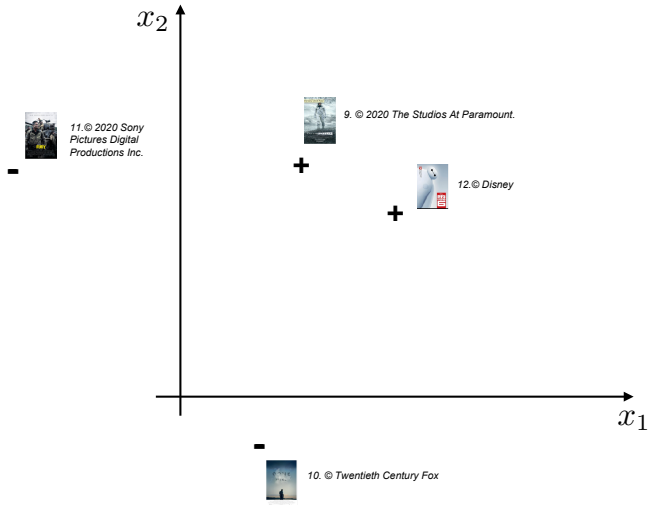

© Black Bear Pictures

$x^{(5)}, x^{(6)}, \ldots$

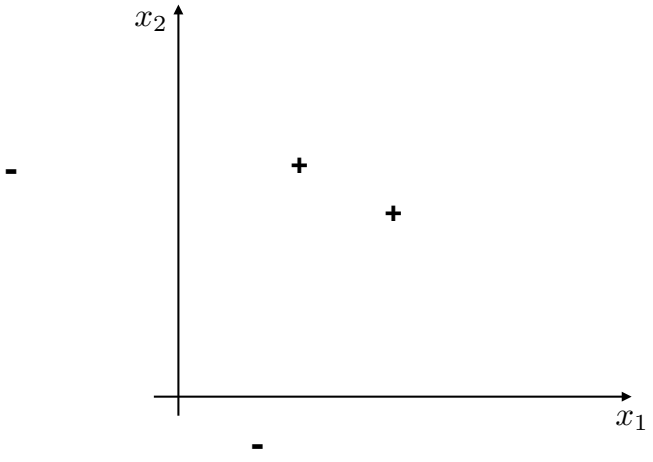
?, ?, ….

Test set

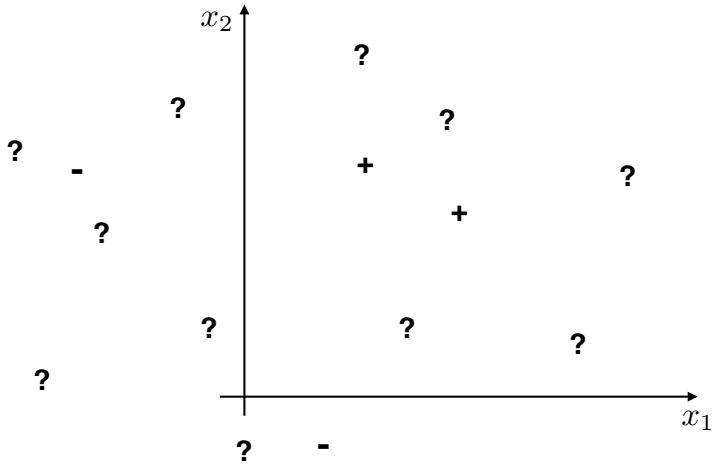
Supervised learning



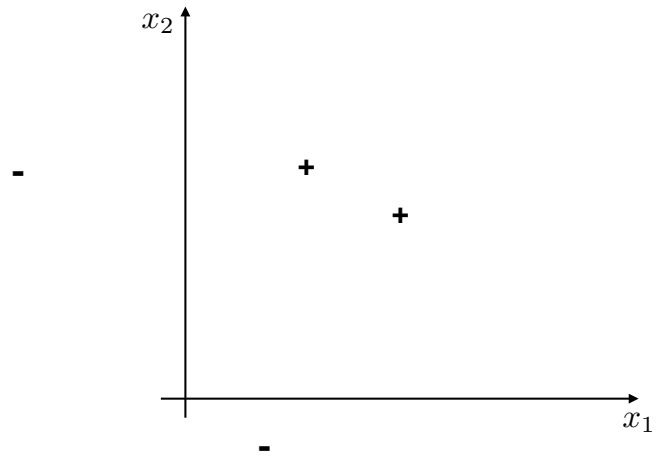
Supervised learning: training set



Supervised learning: test set

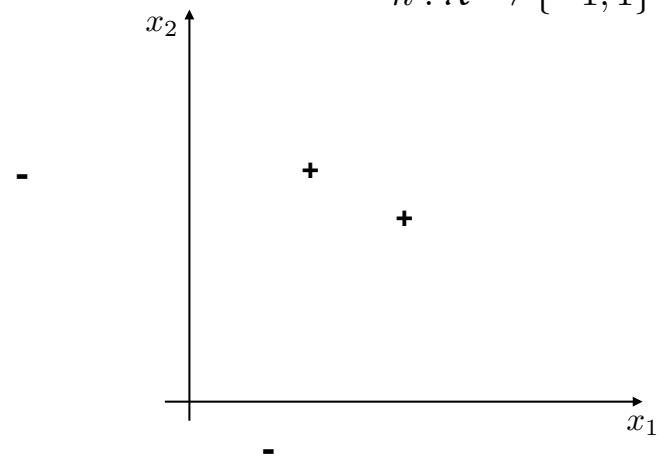


Supervised learning: training set

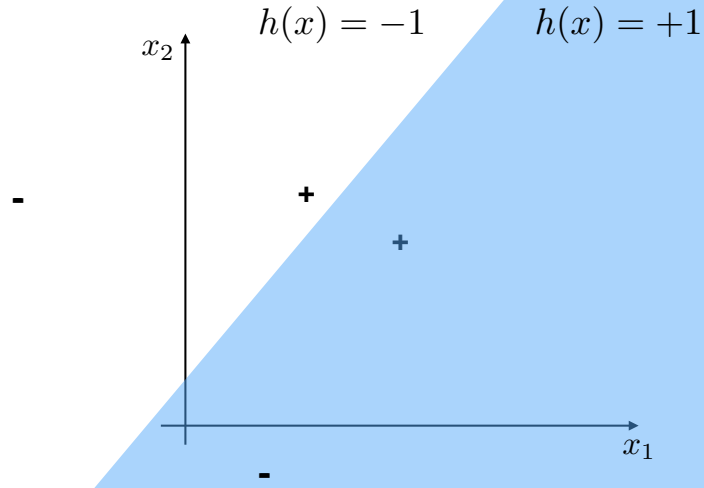


Supervised learning: classifier

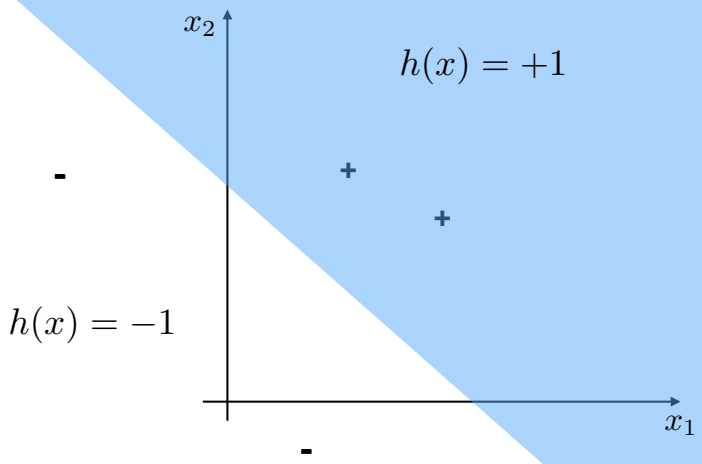
$h : \mathcal{X} \to \{-1, 1\}$



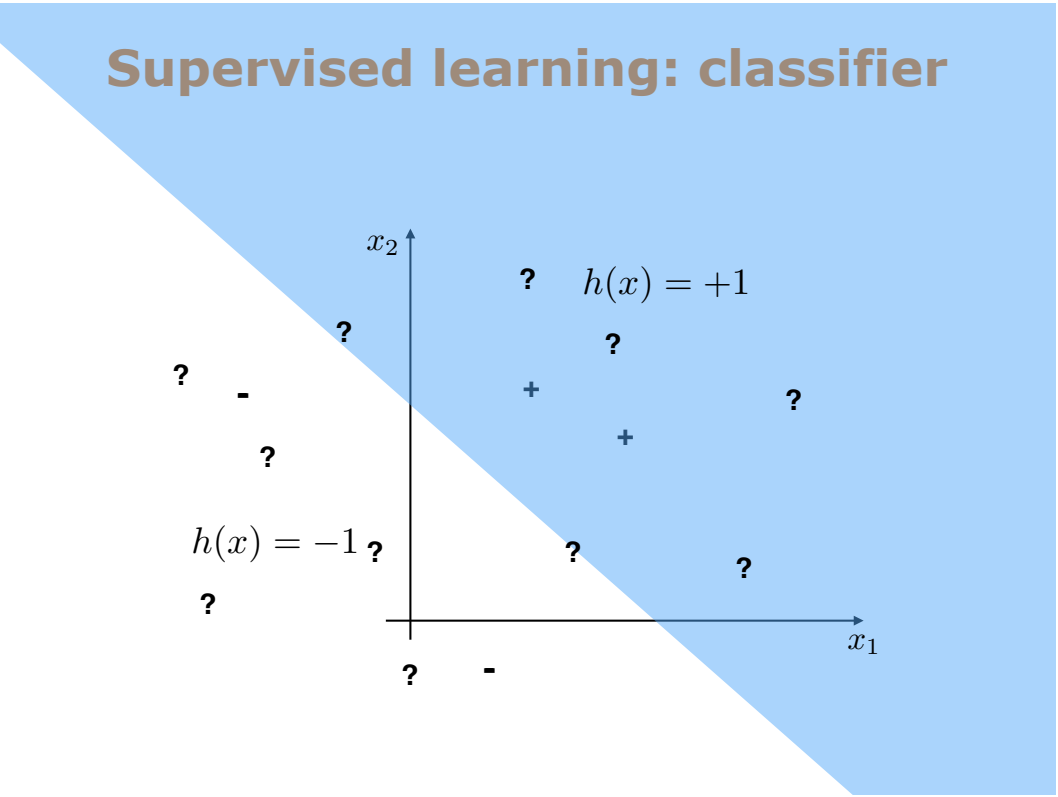
Supervised learning: classifier



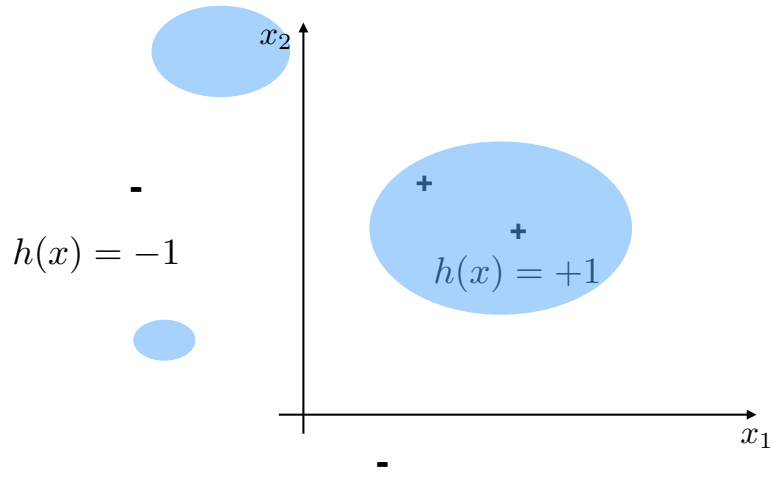
Supervised learning: classifier



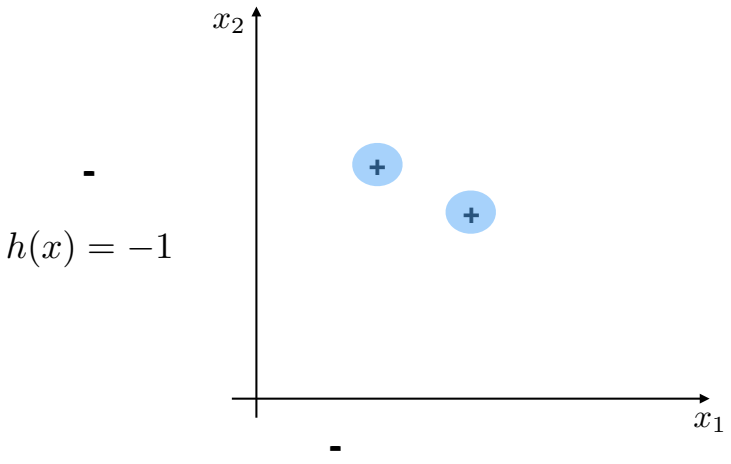
Supervised learning: classifier



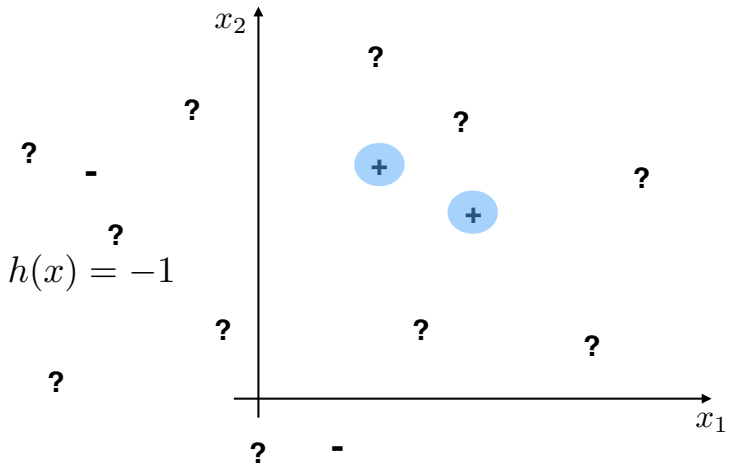
Supervised learning: classifier



Supervised learning: classifier



Supervised learning: generalization



Supervised learning +

- Multi-way classification (e.g., three-way classification)

$$h\left(\text{  }\right) = \text{politics} \qquad h : \mathcal{X} \to \{\text{politics}, \text{sports}, \text{other}\}$$

13. BBC news and the BBC logo are registered trademarks of Google LLC.

- Regression

$$h\left(\text{  }\right) = \$1{,}349{,}000 \qquad h : \mathcal{X} \to \mathbb{R}$$

14. © Robert Paul Properties, Cambridge.

- Structured prediction

$$h\left(\text{  }\right) = \begin{array}{l}\text{A group of people}\\ \text{shopping at an}\\ \text{outdoor market}\end{array} \qquad h : \mathcal{X} \to \{\text{English sentences}\}$$

15. © Computer Vision & Pattern Recognition Lab, 2017.

Types of machine learning

- Supervised learning
  - prediction based on examples of correct behavior
- Unsupervised learning
  - no explicit target, only data, goal to model/discover
- Semi-supervised learning
  - supplement limited annotations with unsupervised learning
- Active learning
  - learn to query the examples actually needed for learning
- Transfer learning
  - how to apply what you have learned from A to B
- Reinforcement learning
  - learning to act, not just predict; goal to optimize the consequences of actions
- Etc.

Key things to understand

- Posing supervised machine learning problems
- Supervised classification
- The role of training/test sets
- A classifier
- A set of classifiers
- Errors, generalization

Attribution List - Lecture 1 - 6.86x Maxchine Learning

1.
Location: Throughout course
Object Source / URL: own work
Citation/Attribution: Instructor content (c) Tommi Jaakkola. Used with permission.

2.
Unit 1 Lecture 1: Introduction to Machine Learning
Screenshot of a Google search of "deep learning"
Slide: # 2
Object Source / URL: Screenshot of a Google search of "deep learning"
Citation/Attribution: Google and the Google logo are registered trademarks of Google LLC.

3.
Unit 1 Lecture 1: Introduction to Machine Learning
Screenshot of a Google search of "Netflix"
Slides: #2
Object Source / URL: Screenshot of a Google search of "Netflix"
Citation/Attribution: (c) Netflix, Inc.

4.
Screenshot of a Google search of "mushrooms"
Slides: #2
Object Source / URL: Screenshot of a Google search of "mushrooms"
Citation/Attribution: Google and the Google logo are registered trademarks of Google LLC.

5.
Unit 1 Lecture 1: Introduction to Machine Learning
Image of "Ear-reddening game move"
Slides: #2
Object Source / URL: https://senseis.xmp.net/?EarReddeningMove
Citation/Attribution: © Senseis Library

6.
Unit 1 Lecture 1: Introduction to Machine Learning
Movie poster thumbnail for the film "Imitation Game"
Slides: 1 at #5, #6, #7, #8. Lecture 12 at #3, #4, #5
Object Source / URL: https://www.imdb.com/title/tt2084970/
Citation/Attribution: © Black Bear Pictures

7.
Unit 1 Lecture 1: Introduction to Machine Learning
Thumbnail of two mushrooms
Slides: # 5, #10, #11, #12, #13, #14
Object Source / URL: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
Citation/Attribution: Image from "ImageNet Classification with Deep Convolutional Neural Networks" by Alex Krizhevsky et all, © Neural Information Processing Systems Foundation, Inc.

Attribution List - Lecture 1 - 6.86x Maxchine Learning

8.
Unit 1 Lecture 1: Introduction to Machine Learning
Thumbnail of one Dalmatian dog
Slides: # 5, #10, #11, #12, #13, #14
Object Source / URL: https://www.slideserve.com/rufin/an-overview-of-deep-learning
Citation/Attribution: "An Overview of Deep Learning" © Geoffrey Hinton, University of Toronto.

9.
Unit 1 Lecture 1: Introduction to Machine Learning
Movie poster thumbnail for the film "Interstellar"
Slides: #10, #11, #12, #13, #14
Object Source / URL: https://www.imdb.com/title/tt0816692/
Citation/Attribution: © 2020 The Studios At Paramount

10.
Unit 1 Lecture 1: Introduction to Machine Learning
Movie poster thumbnail for the film "Gone Girl"
Slides: #10, #11, #12, #13, #14
Object Source / URL: https://www.imdb.com/title/tt2267998/?ref_=fn_al_tt_1
Citation/Attribution: © Twentieth Century Fox

11.
Unit 1 Lecture 1: Introduction to Machine Learning
Movie poster thumbnail for the film "Fury"
Slides: #10, #11, #12, #13, #14
Object Source / URL: https://www.imdb.com/title/tt2713180/?ref_=fn_al_tt_1
Citation/Attribution: © 2020 Sony Pictures Digital Productions Inc.

12.
Unit 1 Lecture 1: Introduction to Machine Learning
Movie poster thumbnail for "Big Hero 6"
Slides: #10, #11, #12, #13, #14
Object Source / URL: https://www.imdb.com/title/tt2245084/?ref_=nv_sr_srsg_0
Citation/Attribution: © Disney

13.
Unit 1 Lecture 1: Introduction to Machine Learning
Screenshot of BBC news page
Slides: #25
Object Source / URL: http://news.bbc.co.uk/2/hi/uk_news/7616996.stm
Citation/Attribution: BBC news and the BBC logo are registered trademarks of Google LLC.

14.
Unit 1 Lecture 1: Introduction to Machine Learning
Thumbnail of a living room
Slides: #25
Object Source / URL: https://www.robertpaul.com/real-estate/150-cambridge-st-410-cambridge-ma-02141/71952983/28587689
Citation/Attribution: © Robert Paul Properties, Cambridge.

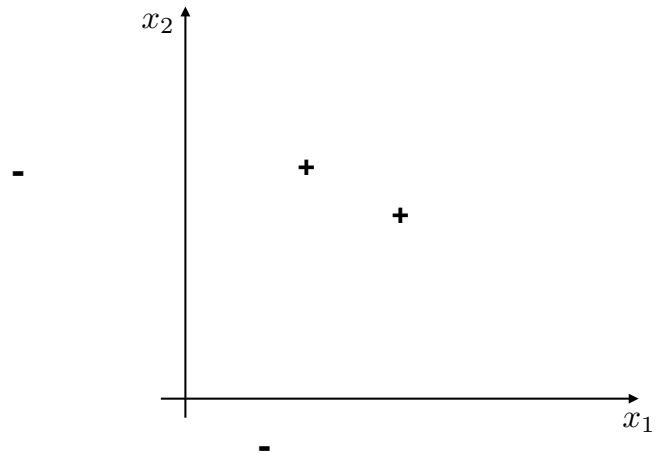
15.
Unit 1 Lecture 1: Introduction to Machine Learning
Thumbnail of a group of people shopping at an outdoor market
Slides: #25
Object Source / URL: http://166.104.231.121/ysmoon/mip2017/lecture_note/제9장.pdf -- page 2
Citation/Attribution: © Computer Vision & Pattern Recognition Lab, 2017.

Machine Learning Lecture 2

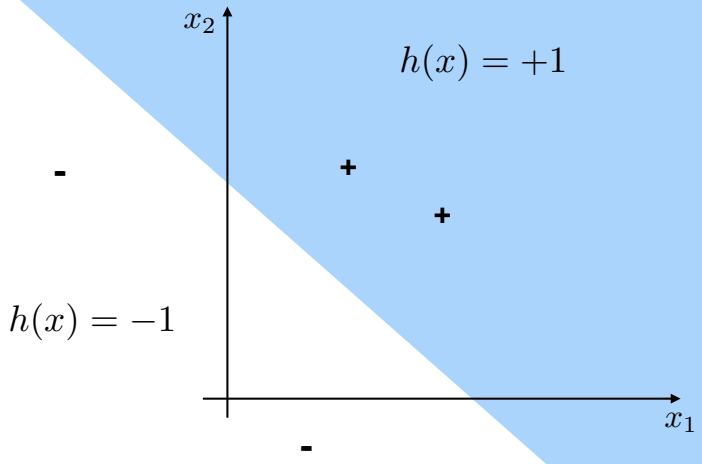
Review of basic concepts

- Feature vectors, labels
- Training set
- Classifier
- Training error
- Test error
- Set of classifiers

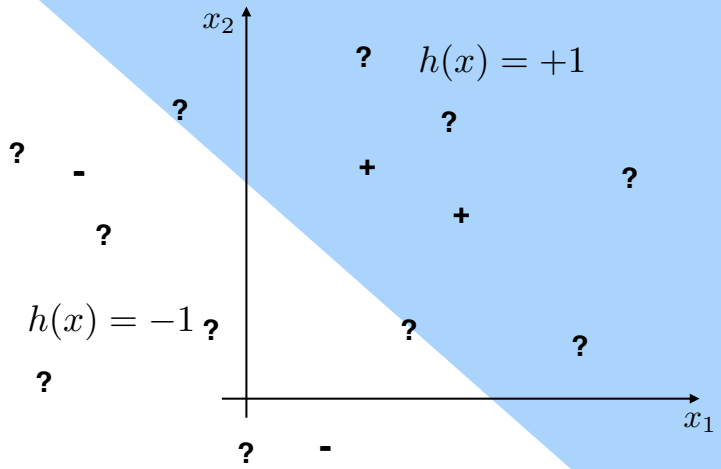
Review: training set



Review: a classifier



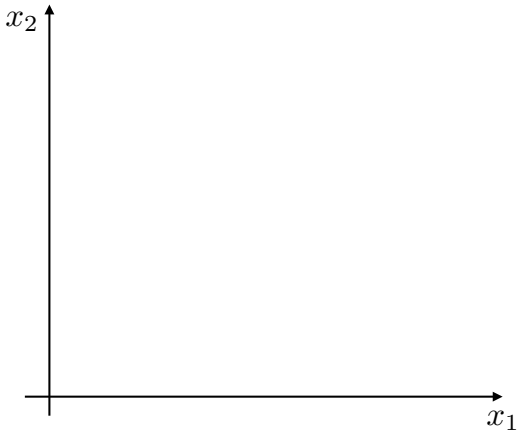
Review: test set



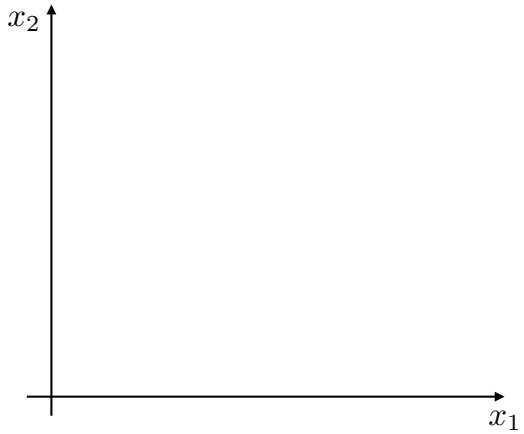
This lecture

- The set of linear classifiers
- Linear separation
- Perceptron algorithm

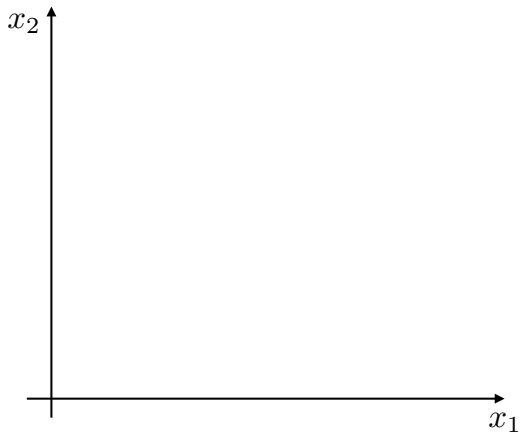
Linear classifiers



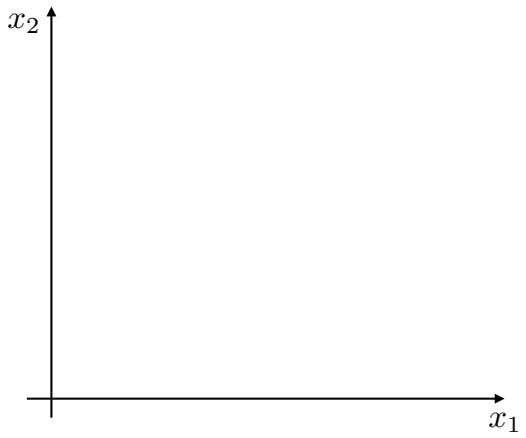
Linear classifiers through origin



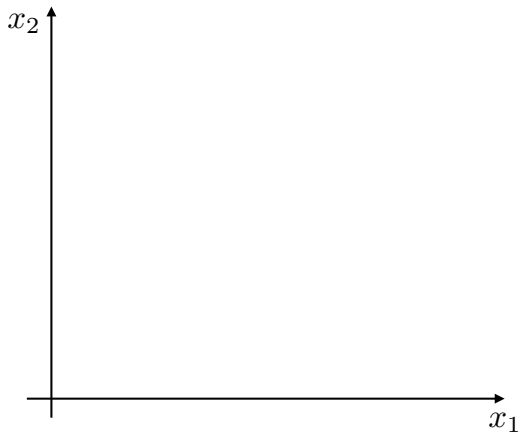
Linear classifiers



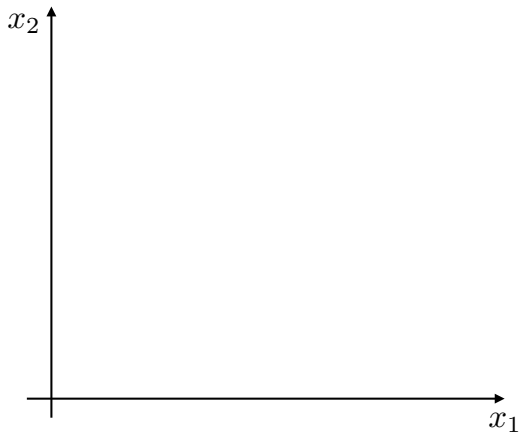
Linear separation: ex



Linear separation: ex



Linear separation: ex



Linear separation

Definition:
Training examples $S_n = \{(x^{(i)}, y^{(i)}\}), i = 1, \ldots, n\}$ are *linearly separable* if there exists a parameter vector $\hat{\theta}$ and offset parameter $\hat{\theta}_0$ such that $y^{(i)}(\hat{\theta} \cdot x^{(i)} + \hat{\theta}_0) > 0$ for all $i = 1, ..., n$.

Learning linear classifiers

- Training error for a linear classifier (through origin)

Learning linear classifiers

- Training error for a linear classifier

Learning algorithm: perceptron

$\theta = 0$ (vector)

if $y^{(i)}(\theta \cdot x^{(i)}) \leq 0$ **then**
$\theta = \theta + y^{(i)} x^{(i)}$

Learning algorithm: perceptron

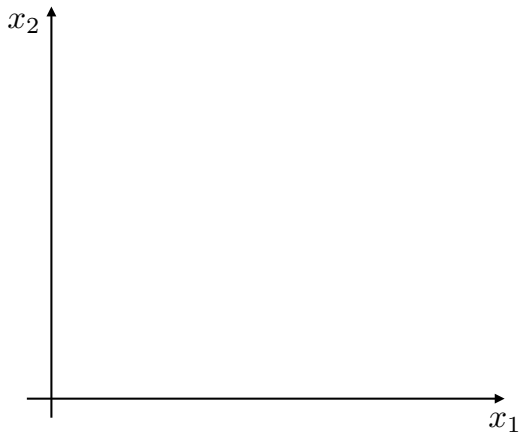
$\theta = 0$ (vector)

for $i = 1, \ldots, n$ **do**
  if $y^{(i)}(\theta \cdot x^{(i)}) \leq 0$ **then**
    $\theta = \theta + y^{(i)}x^{(i)}$

Learning algorithm: perceptron

procedure PERCEPTRON($\{(x^{(i)}, y^{(i)}), i = 1, \ldots, n\}$, T)
  $\theta = 0$ (vector)
  for $t = 1, \ldots, T$ **do**
    for $i = 1, \ldots, n$ **do**
      if $y^{(i)}(\theta \cdot x^{(i)}) \leq 0$ **then**
        $\theta = \theta + y^{(i)}x^{(i)}$
  return θ

Perceptron algorithm: ex



Perceptron (with offset)

1: **procedure** PERCEPTRON($\{(x^{(i)}, y^{(i)}), i = 1, \ldots, n\}$, T)
2: $\theta = 0$ (vector), $\theta_0 = 0$ (scalar)
3: **for** $t = 1, \ldots, T$ **do**
4: **for** $i = 1, \ldots, n$ **do**
5: **if** $y^{(i)}(\theta \cdot x^{(i)} + \theta_0) \leq 0$ **then**
6: $\theta = \theta + y^{(i)}x^{(i)}$
7: $\theta_0 = \theta_0 + y^{(i)}$
8: **return** θ, θ_0

Key things to understand

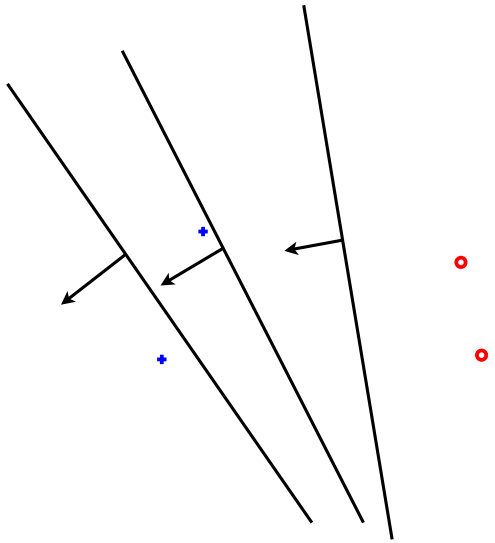
- Parametric families (sets) of classifiers
- The set of linear classifiers
- Linear separation
- Perceptron algorithm

Machine Learning
Lecture 3

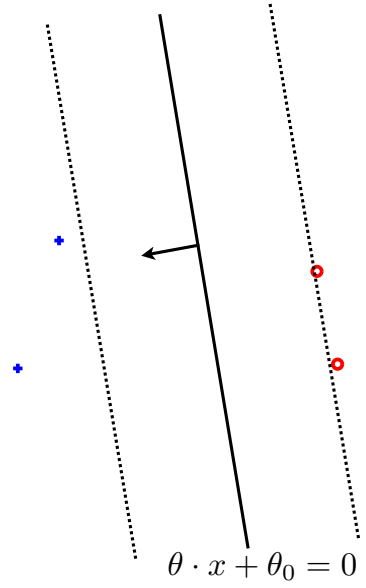
Outline

- Linear, large margin classification
  - margin, hinge loss, regularization
- Learning as an optimization problem

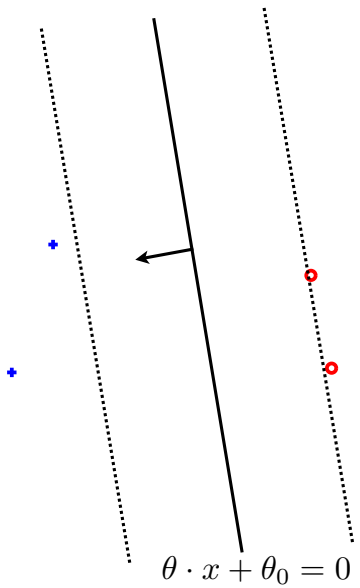
Linear classification



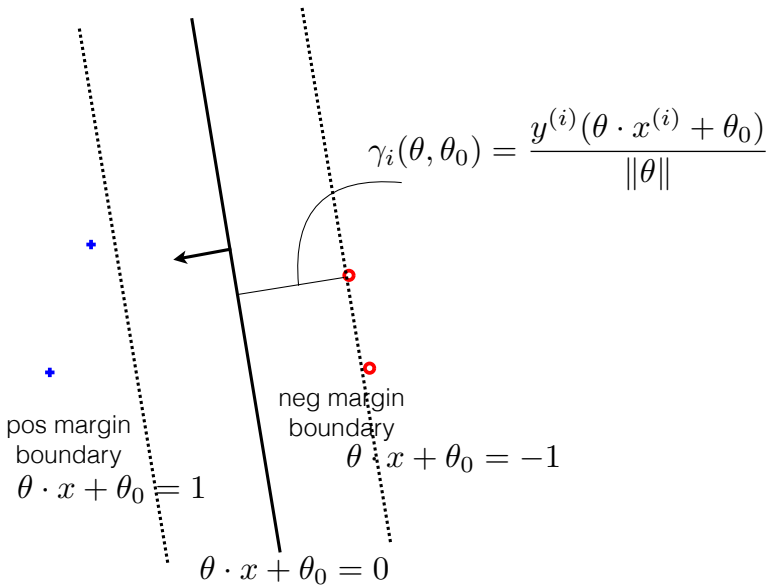
Learning as optimization



Learning as optimization



Linear classification, margin



Large margin as optimization

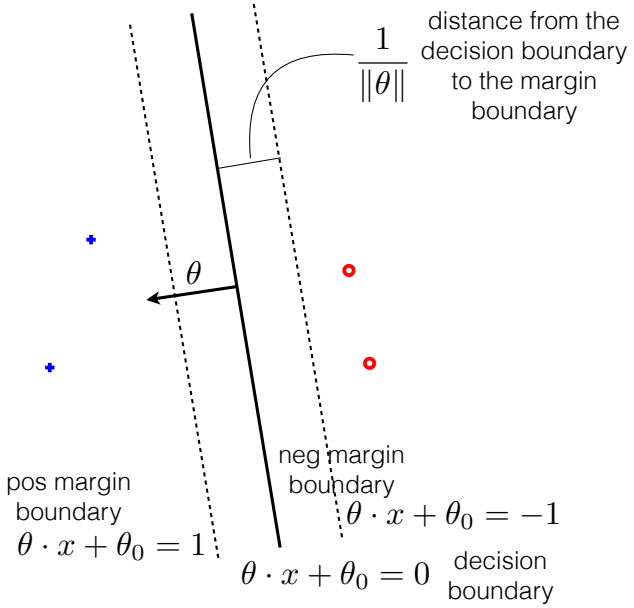
- Hinge loss

$$\text{Loss}_h\big(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)\big) =$$

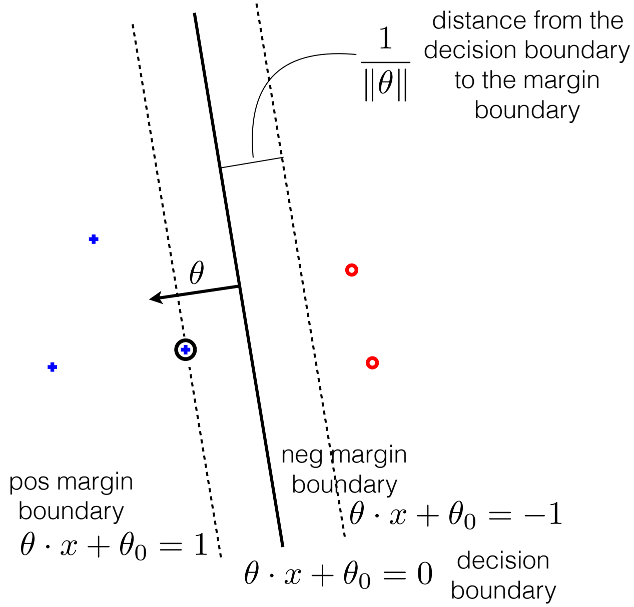
- Regularization: towards max margin

- The objective

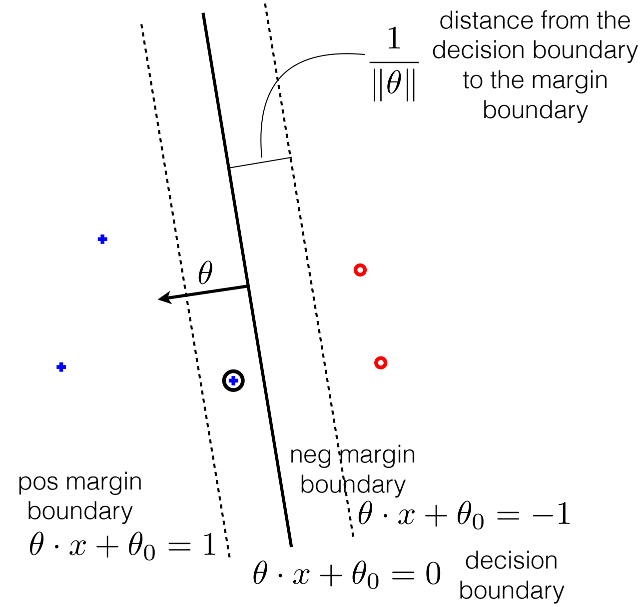
$$J(\theta, \theta_0) = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}_h\big(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)\big) + \frac{\lambda}{2}\|\theta\|^2$$



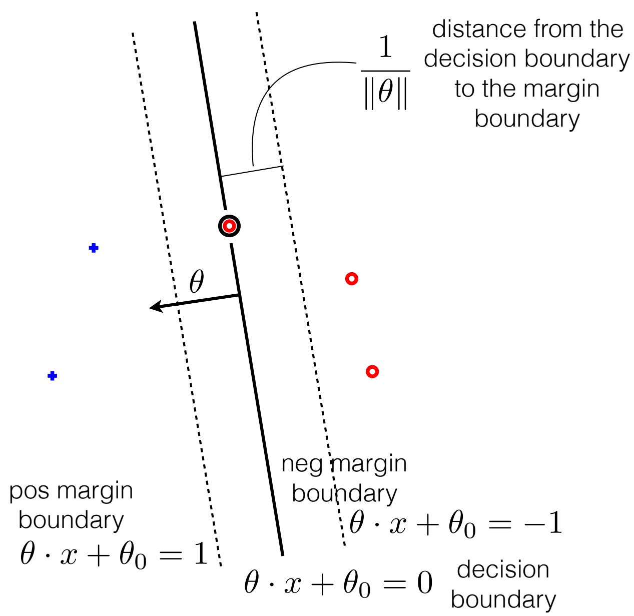
$$J(\theta, \theta_0) = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}_h\big(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)\big) + \frac{\lambda}{2}\|\theta\|^2$$



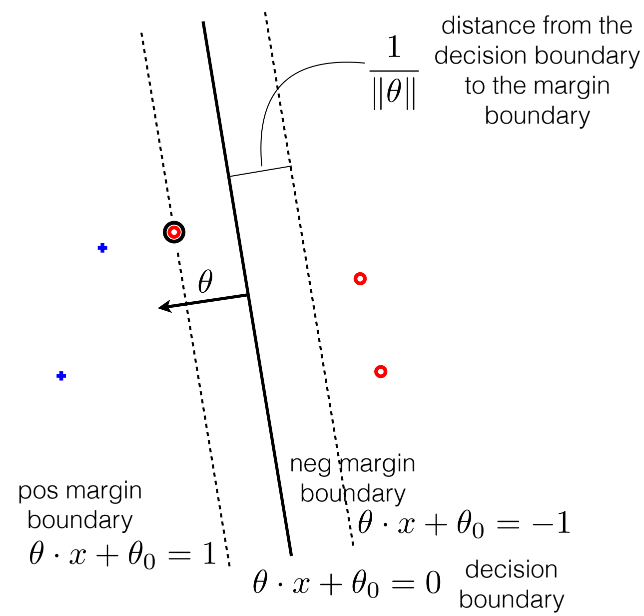
$$J(\theta, \theta_0) = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}_h\left(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)\right) + \frac{\lambda}{2}\|\theta\|^2$$



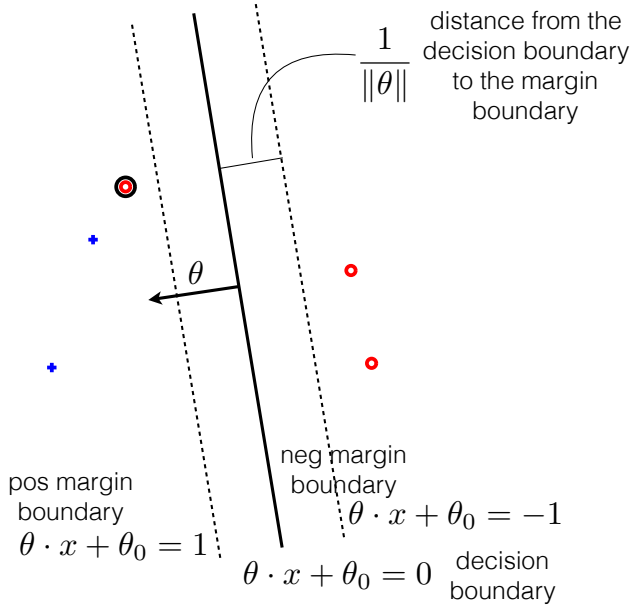
$$J(\theta, \theta_0) = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}_h\left(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)\right) + \frac{\lambda}{2}\|\theta\|^2$$



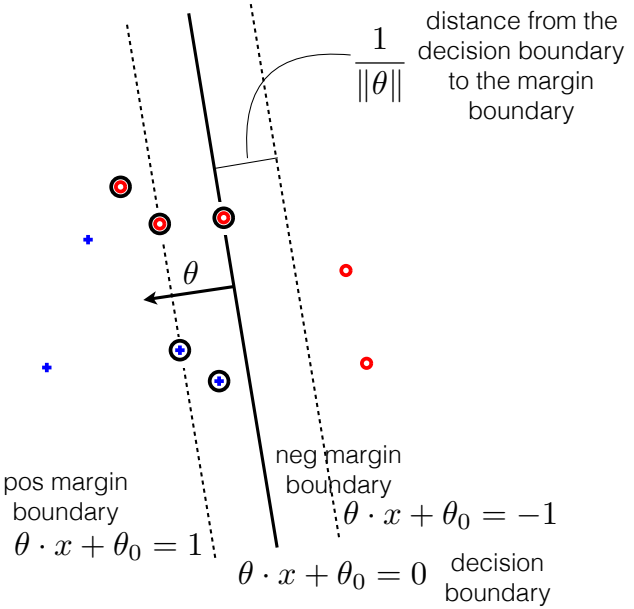
$$J(\theta, \theta_0) = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}_h\left(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)\right) + \frac{\lambda}{2}\|\theta\|^2$$



$$J(\theta, \theta_0) = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}_h\left(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)\right) + \frac{\lambda}{2}\|\theta\|^2$$



$$J(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} \text{Loss}_h\left(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)\right) + \frac{\lambda}{2}\|\theta\|^2$$



$$J(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} \text{Loss}_h\left(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)\right) + \frac{\lambda}{2}\|\theta\|^2$$

CSAIL

Things to know

- General optimization formulation of learning

  objective function = average loss + regularization

- Large margin linear classification as optimization
  - margin boundaries, hinge loss, regularization

$$J(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} \text{Loss}_h\left(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)\right) + \frac{\lambda}{2}\|\theta\|^2$$

Machine Learning Lecture 4

Outline

- Understanding optimization view of learning
  - large margin linear classification
  - regularization, generalization
- Optimization algorithms
  - preface: gradient descent optimization
  - stochastic gradient descent
  - quadratic program

Recall: learning as optimization

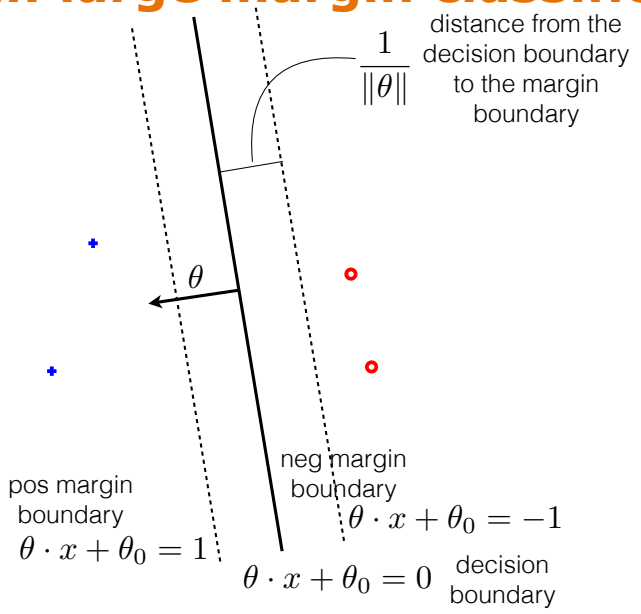
- Machine learning problems are often cast as optimization problems

  objective function = average loss + regularization

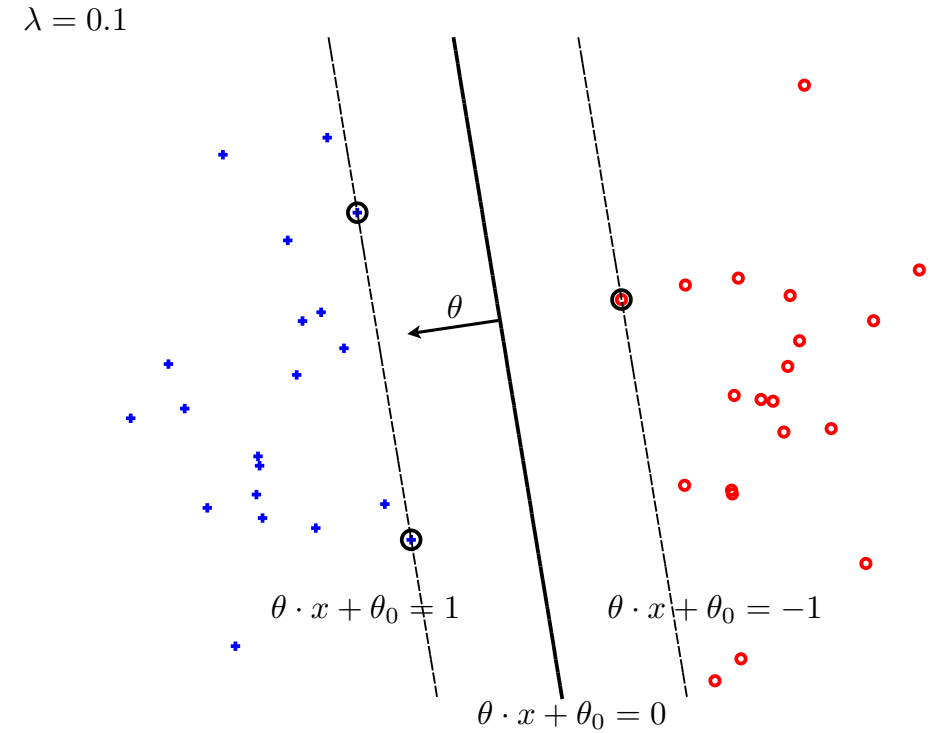
- Large margin linear classification as optimization (Support Vector Machine)

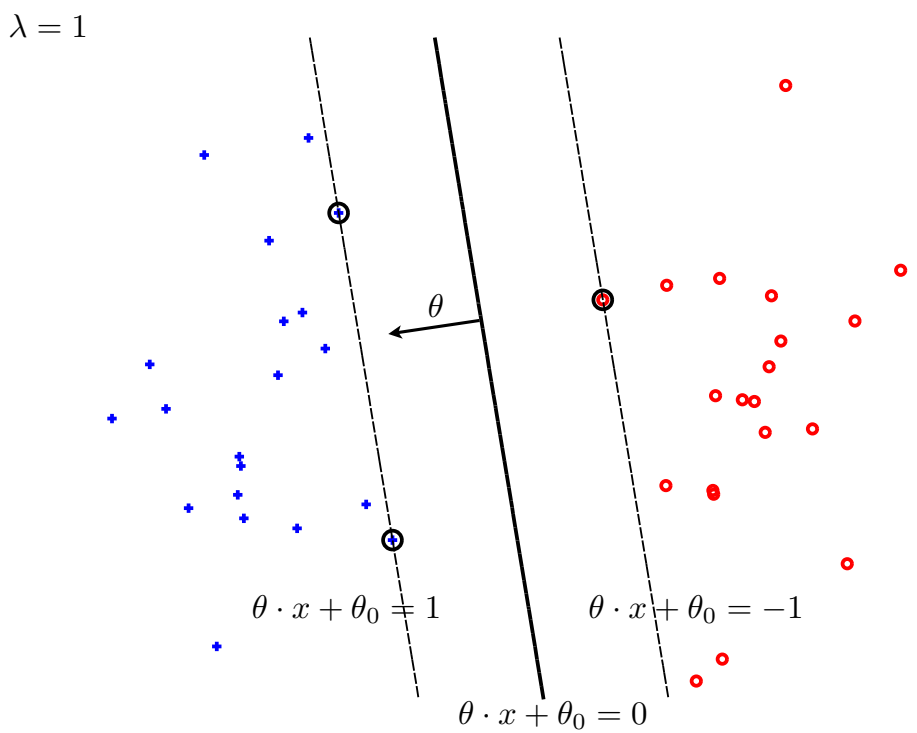
$$J(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} \text{Loss}_h\left(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)\right) + \frac{\lambda}{2}\|\theta\|^2$$

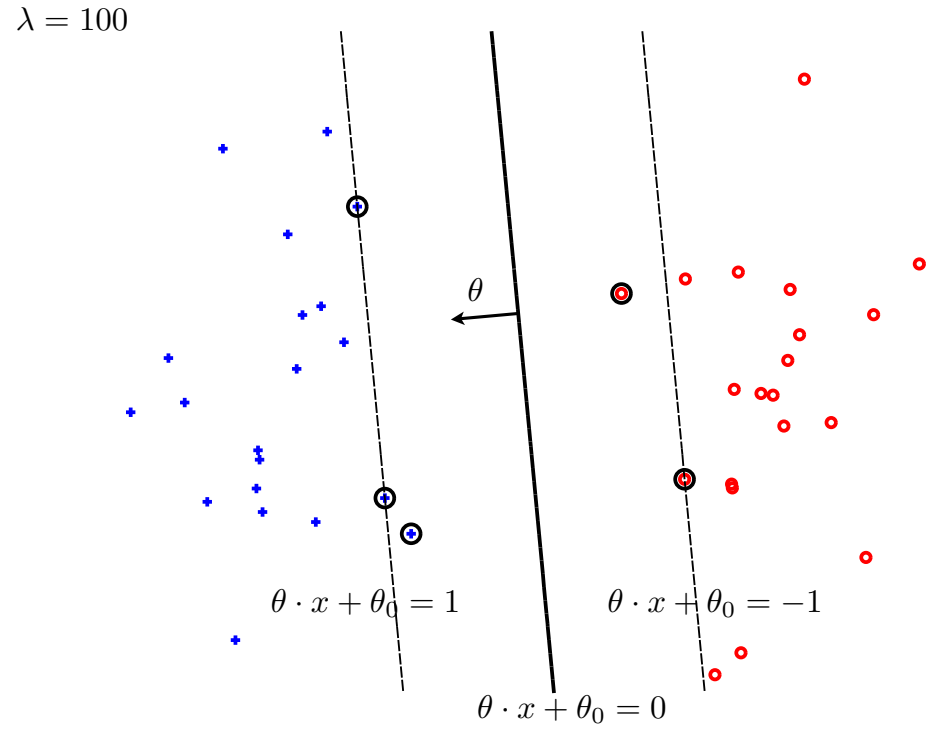
Recall: large margin classifier

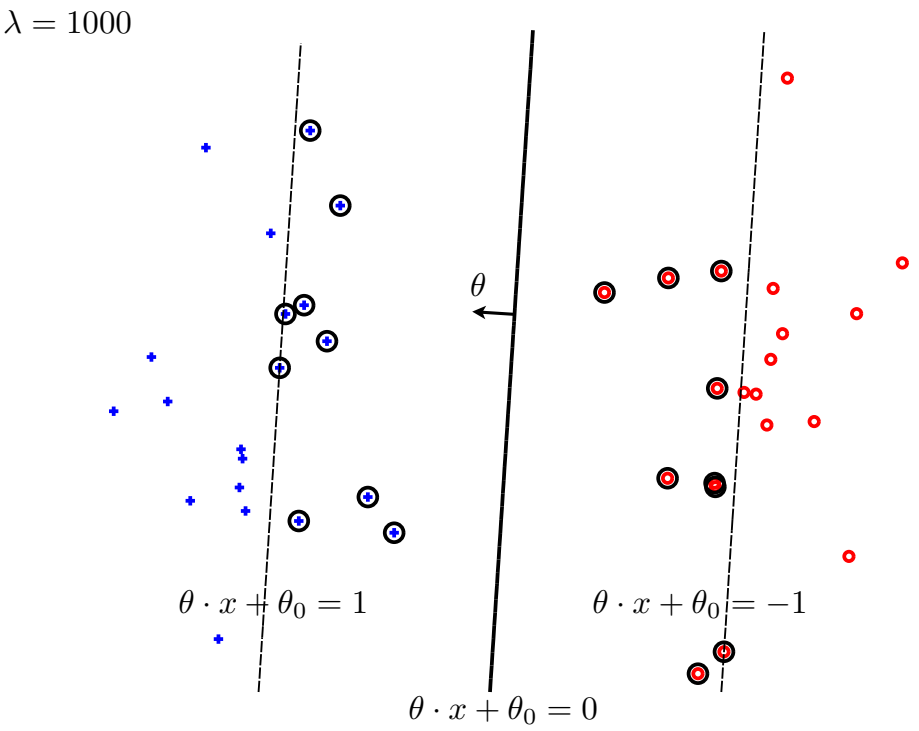


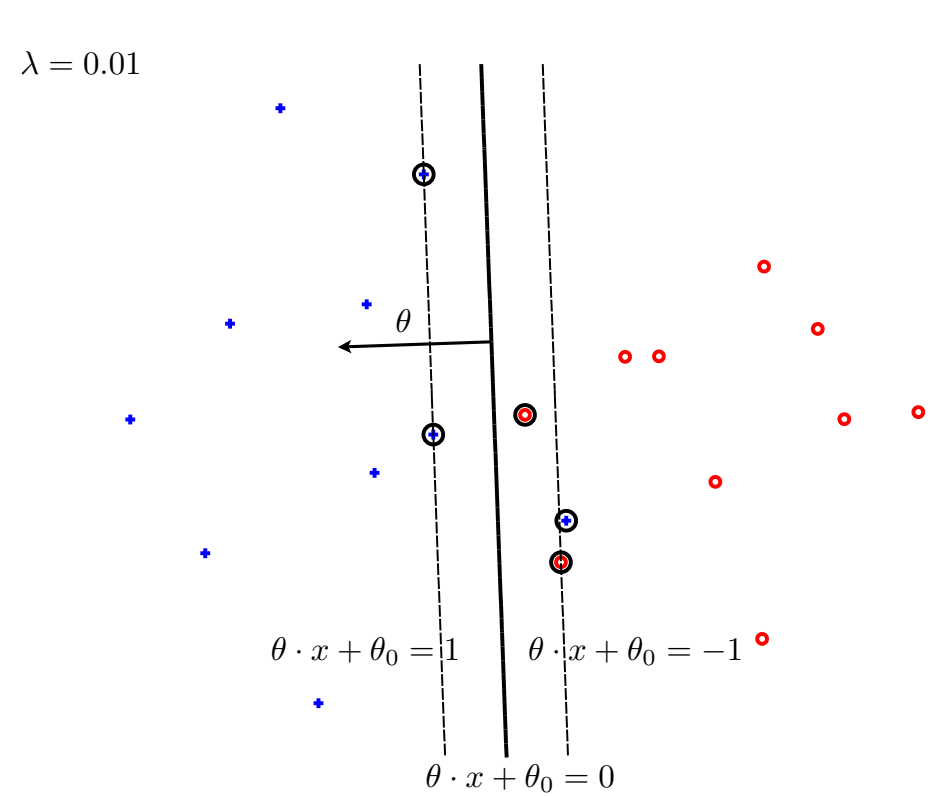
$$J(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} \text{Loss}_h\left(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)\right) + \frac{\lambda}{2}\|\theta\|^2$$

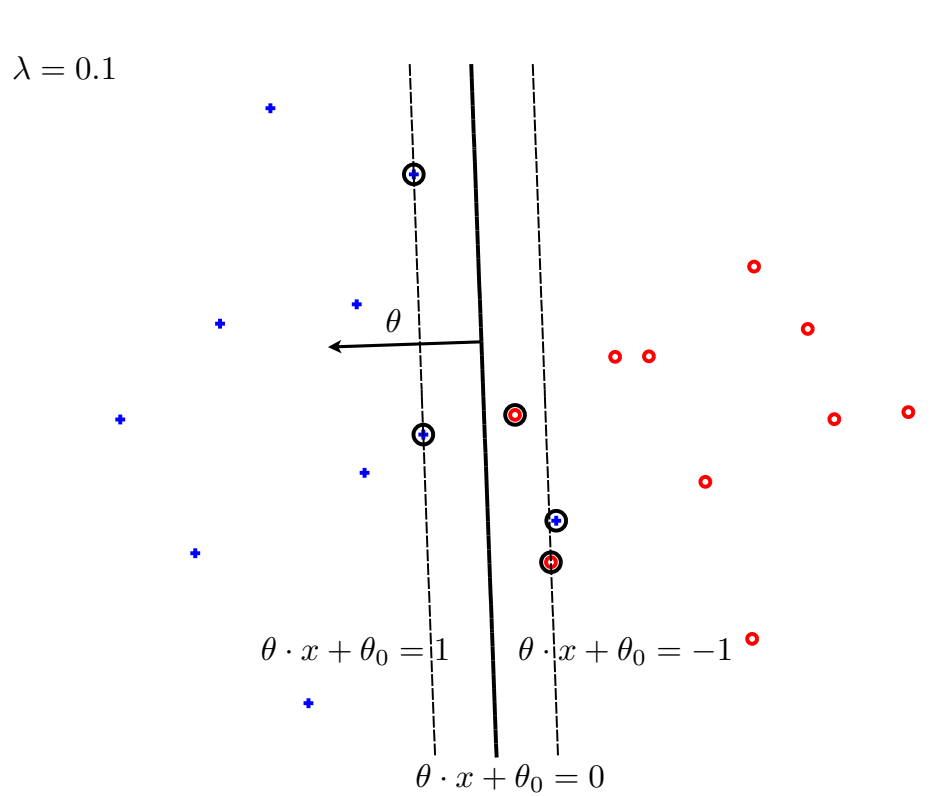

$\lambda = 0.1$
θ
$\theta \cdot x + \theta_0 = 1$
$\theta \cdot x + \theta_0 = -1$
$\theta \cdot x + \theta_0 = 0$

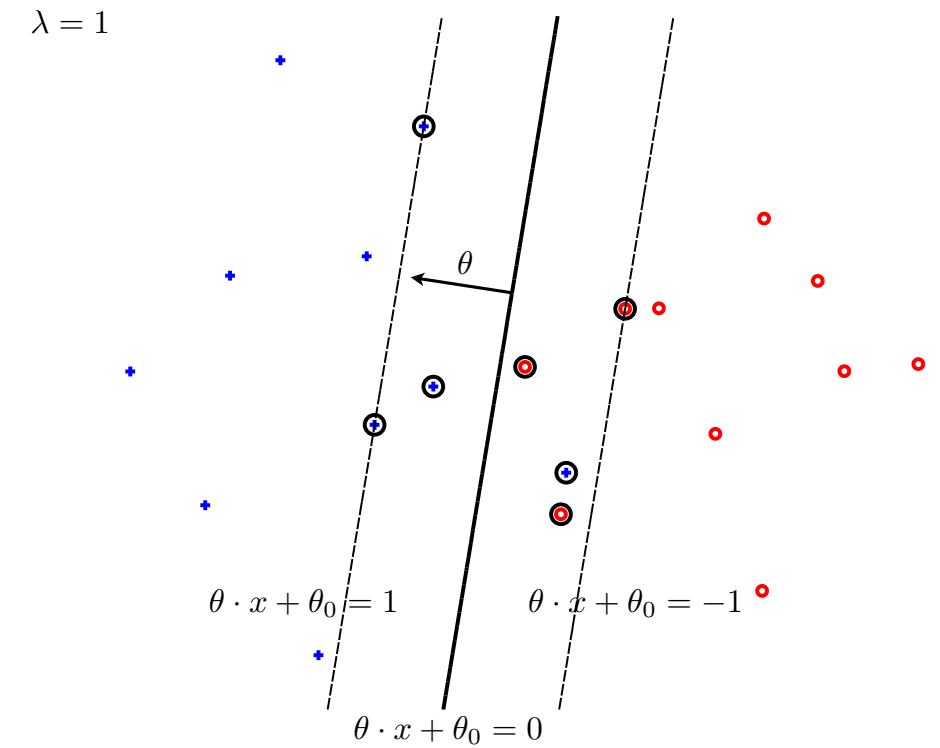

$\lambda = 1$
θ
$\theta \cdot x + \theta_0 = 1$
$\theta \cdot x + \theta_0 = -1$
$\theta \cdot x + \theta_0 = 0$

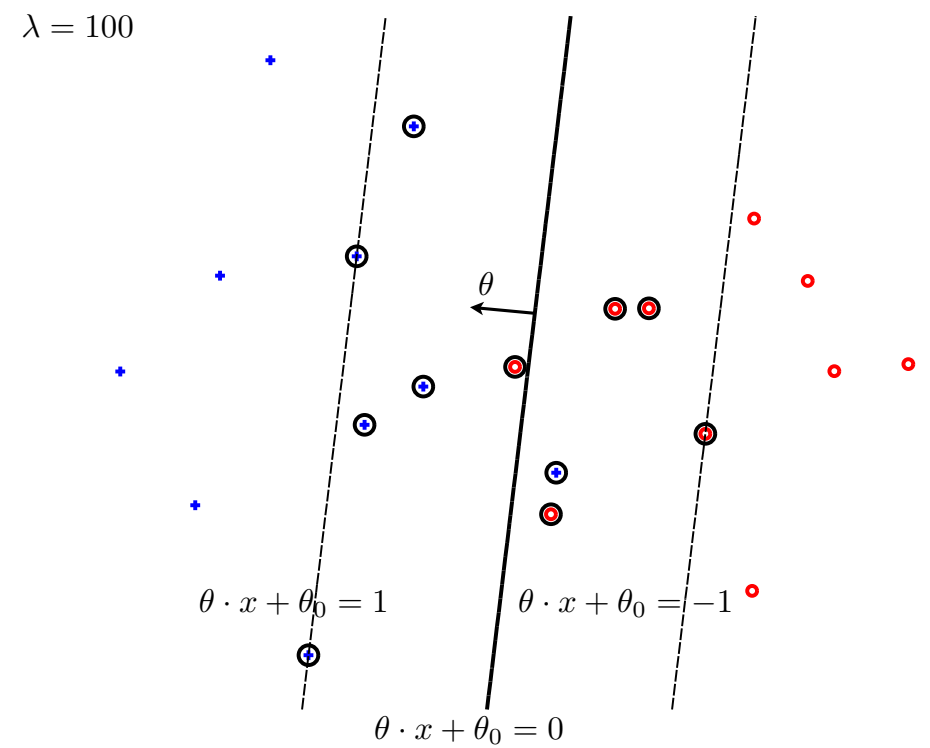

$\lambda = 100$
θ
$\theta \cdot x + \theta_0 = 1$
$\theta \cdot x + \theta_0 = -1$
$\theta \cdot x + \theta_0 = 0$


$\lambda = 1000$
θ
$\theta \cdot x + \theta_0 = 1$
$\theta \cdot x + \theta_0 = -1$
$\theta \cdot x + \theta_0 = 0$


$\lambda = 0.01$
θ
$\theta \cdot x + \theta_0 = 1$
$\theta \cdot x + \theta_0 = -1$
$\theta \cdot x + \theta_0 = 0$


$\lambda = 0.1$
θ
$\theta \cdot x + \theta_0 = 1$
$\theta \cdot x + \theta_0 = -1$
$\theta \cdot x + \theta_0 = 0$


$\lambda = 1$
θ
$\theta \cdot x + \theta_0 = 1$
$\theta \cdot x + \theta_0 = -1$
$\theta \cdot x + \theta_0 = 0$


$\lambda = 100$
θ
$\theta \cdot x + \theta_0 = 1$
$\theta \cdot x + \theta_0 = -1$
$\theta \cdot x + \theta_0 = 0$

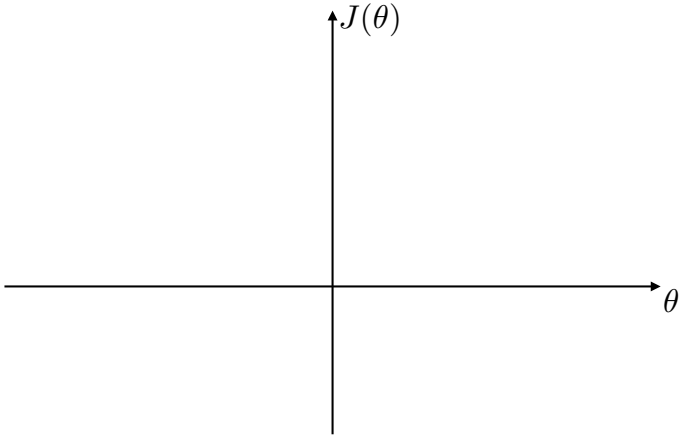
Regularization, generalization

$$J(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} \text{Loss}_h\left(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)\right) + \frac{\lambda}{2}\|\theta\|^2$$

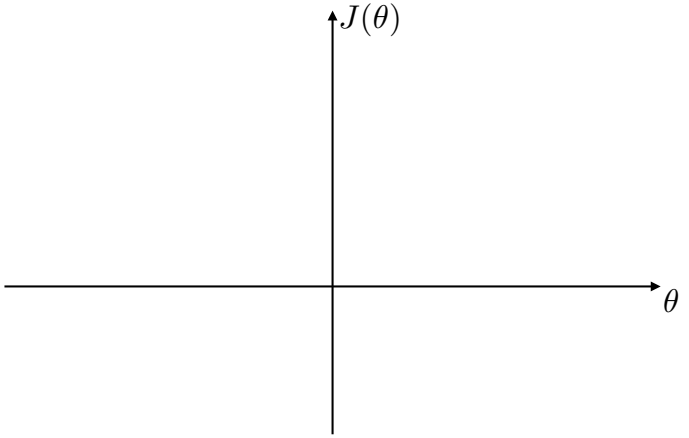
Outline

- Understanding optimization view of learning
  - large margin linear classification
  - regularization, generalization
- Optimization algorithms
  - preface: gradient descent optimization
  - stochastic gradient descent
  - quadratic program

Preface: Gradient descent



Preface: Gradient descent



Stochastic gradient descent

$$J(\theta, \theta_0) = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}_h\big(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)\big) + \frac{\lambda}{2}\|\theta\|^2$$

$$= \frac{1}{n}\sum_{i=1}^{n} \left[\text{Loss}_h\big(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)\big) + \frac{\lambda}{2}\|\theta\|^2\right]$$

Stochastic gradient descent

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \left[\text{Loss}_h(y^{(i)}\theta \cdot x^{(i)}) + \frac{\lambda}{2}\|\theta\|^2\right]$$

Stochastic gradient descent

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \left[\text{Loss}_h(y^{(i)}\theta \cdot x^{(i)}) + \frac{\lambda}{2}\|\theta\|^2\right]$$

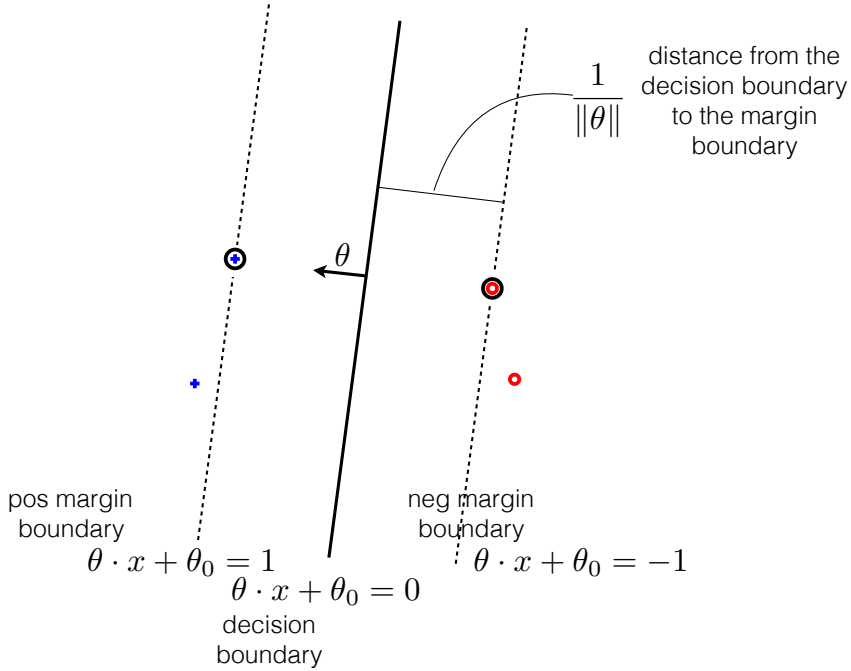
Select $i \in \{1, \ldots, n\}$ at random

$$\theta \leftarrow \theta - \eta_t \nabla_\theta \left[\text{Loss}_h(y^{(i)}\theta \cdot x^{(i)}) + \frac{\lambda}{2}\|\theta\|^2\right]$$

Support Vector Machine

- Support Vector Machine finds the maximum margin linear separator by solving the quadratic program that corresponds to $J(\theta, \theta_0)$
- In the realizable case, if we disallow any margin violations, the quadratic program we have to solve is

$$\text{Find } \theta, \theta_0 \text{ that}$$
$$\text{minimize } \frac{1}{2}\|\theta\|^2 \text{ subject to}$$
$$y^{(i)}(\theta \cdot x^{(i)} + \theta_0) \geq 1, \quad i = 1, \ldots, n$$



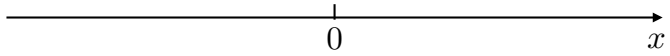
Summary

- Learning problems can be formulated as optimization problems of the form: loss + regularization
- Linear, large margin classification, along with many other learning problems, can be solved with stochastic gradient descent algorithms
- Large margin linear classifier can be also obtained via solving a quadratic program (Support Vector Machine)

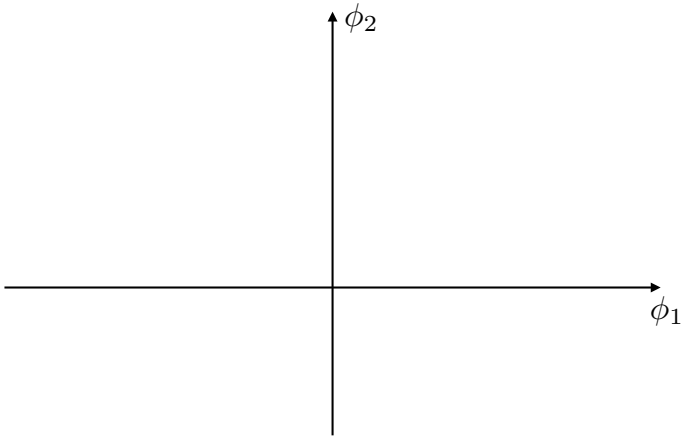
Outline

- Non-linear classification and regression
- Feature maps, their inner products
- Kernel functions induced from feature maps
- Kernel methods, kernel perceptron
- Other non-linear classifiers (e.g., Random Forest)

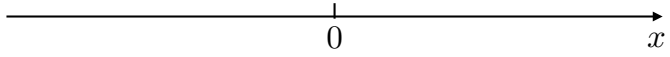
Linear classifiers on the real line



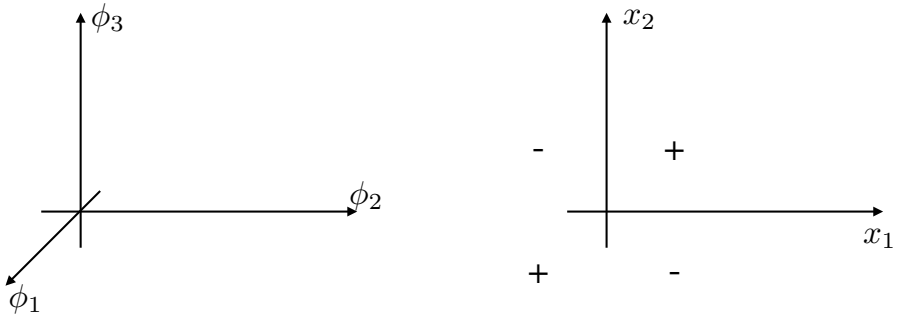
In feature space



Back to the real line



2-dim example



Polynomial features

- We can add more polynomial terms

- Means lots of features in higher dimensions

Non-lin. classification & regression

- Non-linear classification

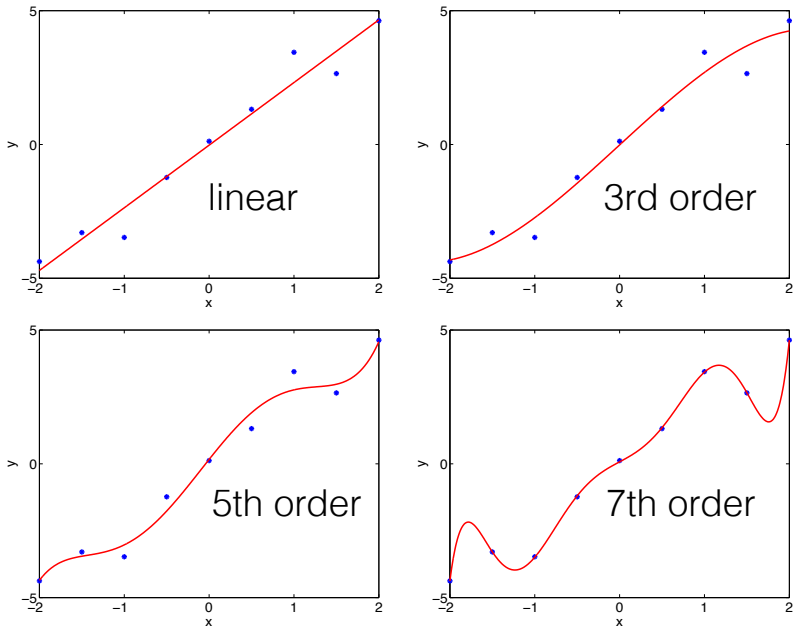
$$h(x;\theta,\theta_0) = \text{sign}\big(\theta \cdot \phi(x) + \theta_0\big)$$

- Non-linear regression

$$f(x;\theta,\theta_0) = \theta \cdot \phi(x) + \theta_0$$

e.g., $\phi(x) = [x, x^2]^T$

Non-linear regression



Why not feature vectors?

- By mapping input examples explicitly into feature vectors, and performing linear classification or regression on top of such feature vectors, we get a lot of expressive power
- But the downside is that these vectors can be quite high dimensional

Inner products, kernels

- Computing the inner product between two feature vectors **can be** cheap even if the vectors are very high dimensional

$$\phi(x) = [x_1, x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2]^T$$
$$\phi(x') = [x'_1, x'_2, x'^2_1, \sqrt{2}x'_1x'_2, x'^2_2]^T$$

Kernels vs features

- For **some** feature maps, we can evaluate the inner products very efficiently, e.g.,
- In those cases, it is advantageous to express the linear classifiers (regression methods) in terms of kernels rather than explicitly constructing feature vectors

Recall perceptron

$\theta = 0$

run through $i = 1, \ldots, n$

  if $y^{(i)}\, \theta \cdot \phi(x^{(i)}) \leq 0$

    $\theta \leftarrow \theta + y^{(i)}\phi(x^{(i)})$

Recall perceptron

$\theta = 0$

run through $i = 1, \ldots, n$

if $y^{(i)}\, \theta \cdot \phi(x^{(i)}) \leq 0$

$\theta \leftarrow \theta + y^{(i)} \phi(x^{(i)})$

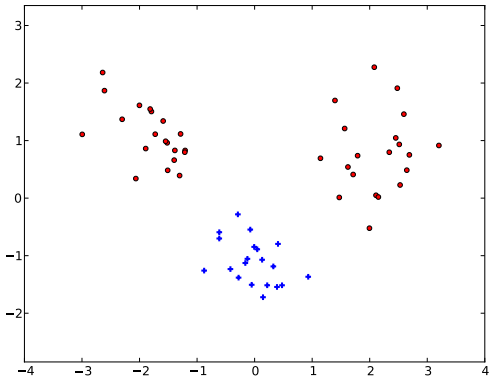
Feature engineering, kernels

- Composition rules:
  1. $K(x, x') = 1$ is a kernel function.
  2. Let $f : \mathbb{R}^d \to \mathbb{R}$ and $K(x, x')$ is a kernel. Then so is $\tilde{K}(x, x') = f(x) K(x, x') f(x')$
  3. If $K_1(x, x')$ and $K_2(x, x')$ are kernels, then $K(x, x') = K_1(x, x') + K_2(x, x')$ is a kernel
  4. If $K_1(x, x')$ and $K_2(x, x')$ are kernels, then $K(x, x') = K_1(x, x') K_2(x, x')$ is a kernel

Radial basis kernel

- The feature vectors can be infinite dimensional… this means that they have unlimited expressive power

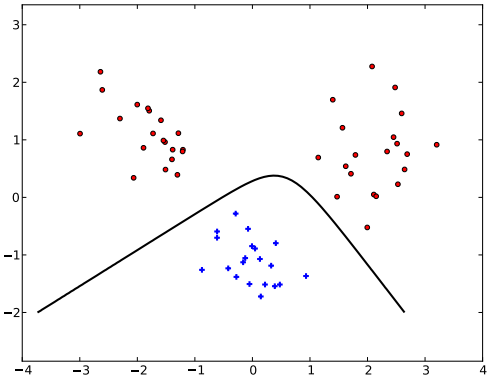
$$K(x, x') = \exp(-\frac{1}{2}\|x - x'\|^2)$$



Radial basis kernel

- The feature vectors can be infinite dimensional… this means that they have unlimited expressive power

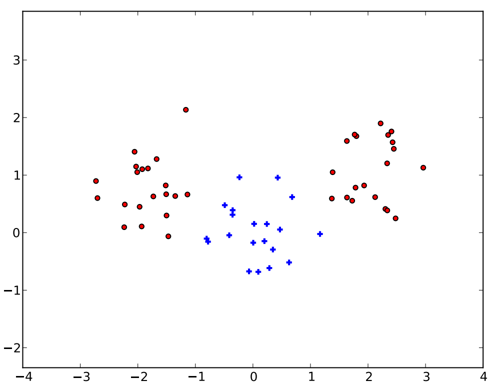
$$K(x, x') = \exp(-\frac{1}{2}\|x - x'\|^2)$$



Radial basis kernel

- The feature vectors can be infinite dimensional… this means that they have unlimited expressive power

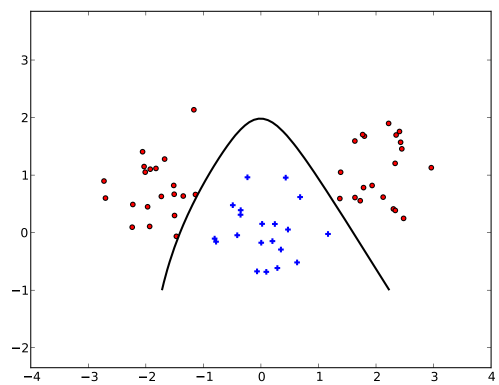
$$K(x,x') = \exp(-\frac{1}{2}\|x - x'\|^2)$$



Radial basis kernel

- The feature vectors can be infinite dimensional… this means that they have unlimited expressive power

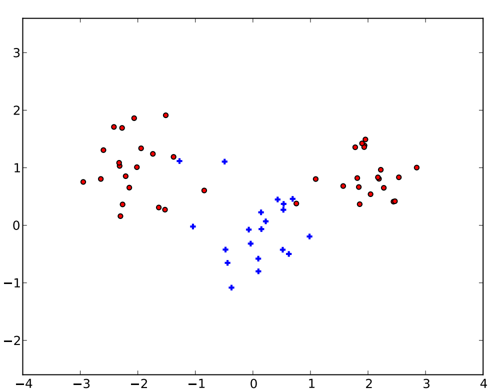
$$K(x,x') = \exp(-\frac{1}{2}\|x - x'\|^2)$$



Radial basis kernel

- The feature vectors can be infinite dimensional… this means that they have unlimited expressive power

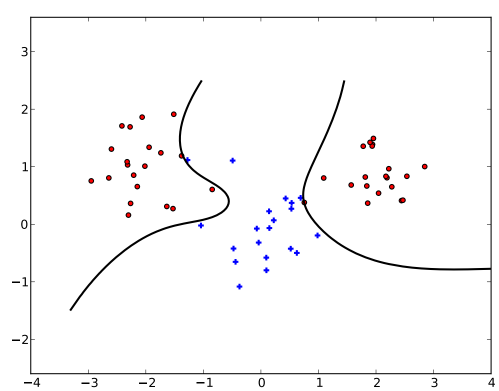
$$K(x,x') = \exp(-\frac{1}{2}\|x - x'\|^2)$$



Radial basis kernel

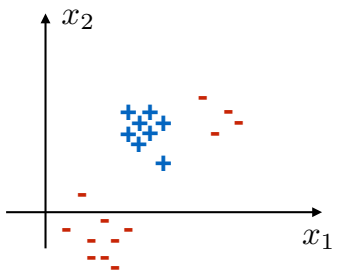
- The feature vectors can be infinite dimensional… this means that they have unlimited expressive power

$$K(x,x') = \exp(-\frac{1}{2}\|x - x'\|^2)$$



Other non-linear classifiers

- Random forest is a good default classifier for (almost) any setting



- Procedure:
  - boostrap sample
  - build a (randomized) decision tree
  - average predictions (ensemble)

Summary

- We can get non-linear classifiers or regression methods by simply mapping examples into feature vectors non-linearly, and applying a linear method on the resulting vectors
- These feature vectors can be high dimensional, however
- We can turn the linear methods into kernel methods by casting the computations in terms of inner products
- A kernel function is simply an inner product between two feature vectors
- Using kernels is advantageous when the inner products are faster to evaluate than using explicit vectors (e.g., when the vectors would be infinite dimensional!)

Feed-forward Neural Networks (Part 1)

Outline (part 1)

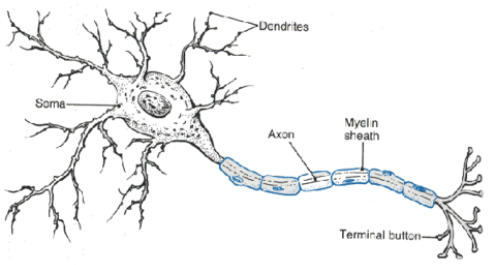
- Feed-forward neural networks
- The power of hidden layers
- Learning feed-forward networks
  - SGD and back-propagation

Motivation

- So far our classifiers rely on pre-compiled features

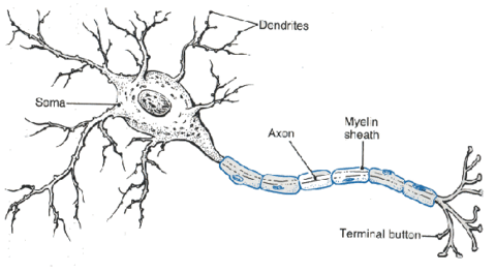
$$\hat{y} = \text{sign}\big(\theta \cdot \phi(x)\big)$$

Neural Networks

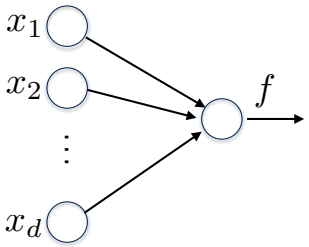


1. (c) Michael DeBellis

(Artificial) Neural Networks



(c) Michael DeBellis



(e.g., a linear classifier)

2. Image on Wikimedia by Users: Ramón Santiago y Cajal.

A unit in a neural network

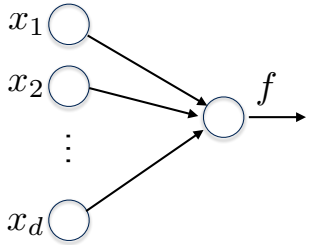


Image on Wikimedia by Users: Ramón Santiago y Cajal.

A unit in a neural network

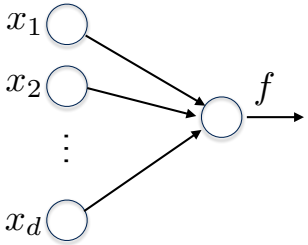
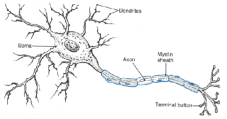
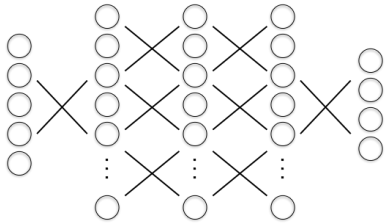


Image on Wikimedia by Users: Ramón Santiago y Cajal.

Deep Neural Networks



(c) Michael DeBellis



- Deep neural networks
  - loosely motivated by biological neurons, networks
  - adjustable processing units (~ linear classifiers)
  - highly parallel, typically organized in layers
  - deep = many transformations (layers) before output

  e.g., edges -> simple parts-> parts -> objects -> scenes

Deep Learning

- Deep learning has overtaken a number of academic disciplines in just a few years
  - computer vision (e.g., image, scene analysis)
  - natural language processing (e.g., machine translation)
  - speech recognition
  - computational biology, etc.

Deep Learning

- Deep learning has overtaken a number of academic disciplines in just a few years
  - computer vision (e.g., image, scene analysis)
  - natural language processing (e.g., machine translation)
  - speech recognition
  - computational biology, etc.
- Key role in recent successes
  - self driving vehicles
  - speech interfaces
  - conversational agents
  - superhuman game playing

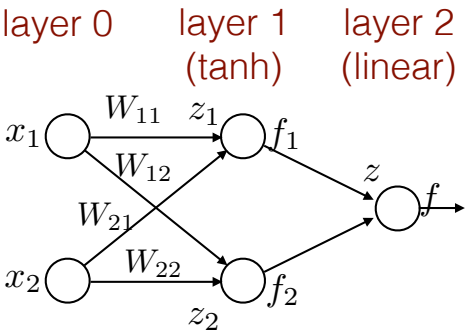
Deep Learning

- Deep learning has overtaken a number of academic disciplines in just a few years
  - computer vision (e.g., image, scene analysis)
  - natural language processing (e.g., machine translation)
  - speech recognition
  - computational biology, etc.
- Key role in recent successes
  - self driving vehicles
  - speech recognition
  - conversational agents
  - superhuman game playing
- Many more underway
  - personalized/automated medicine
  - chemistry, robotics, materials science, etc.

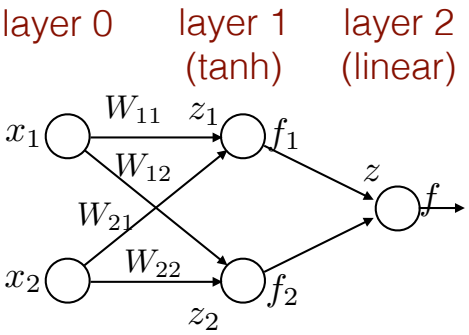
Deep learning … why now?

- **Reason #1:** lots of data
  - many significant problems can only be solved at scale
- **Reason #2:** computational resources (esp. GPUs)
  - platforms/systems that support running deep (machine) learning algorithms at scale
- **Reason #3:** large models are easier to train
  - large models can be successfully estimated with simple gradient based learning algorithms
- **Reason #4:** flexible neural "lego pieces"
  - common representations, diversity of architectural choices

One hidden layer model

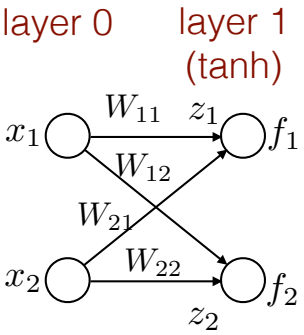


One hidden layer model

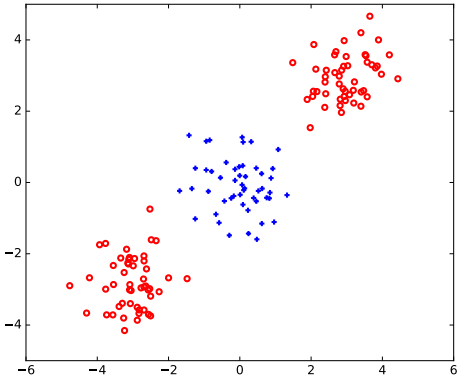


One hidden layer model

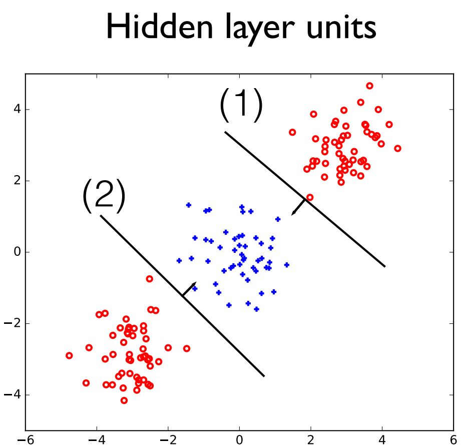
- Neural signal transformation



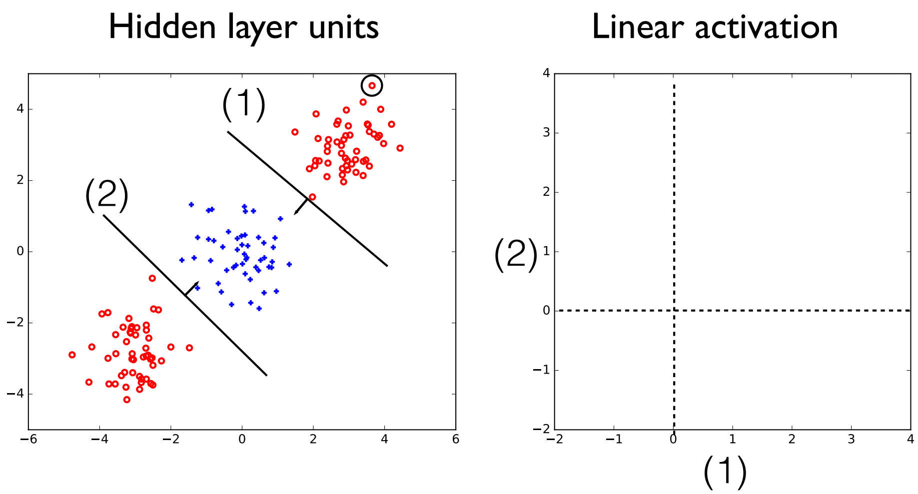
Example Problem



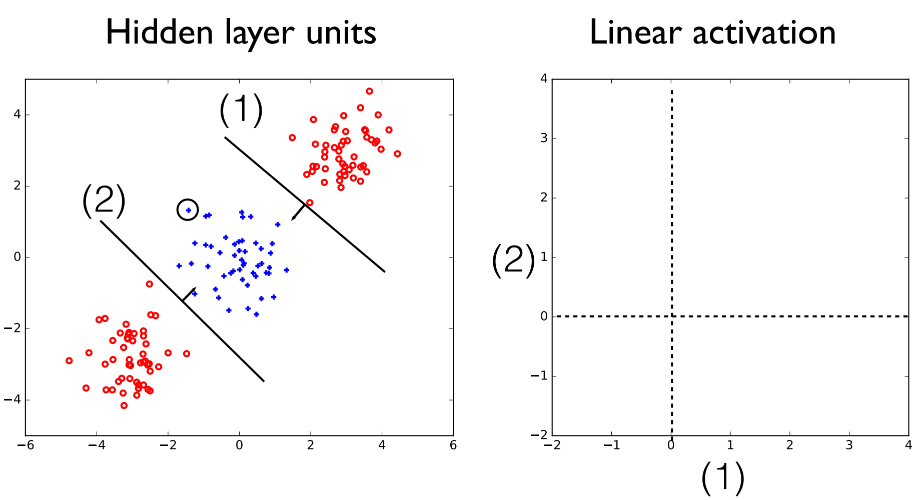
Hidden layer representation



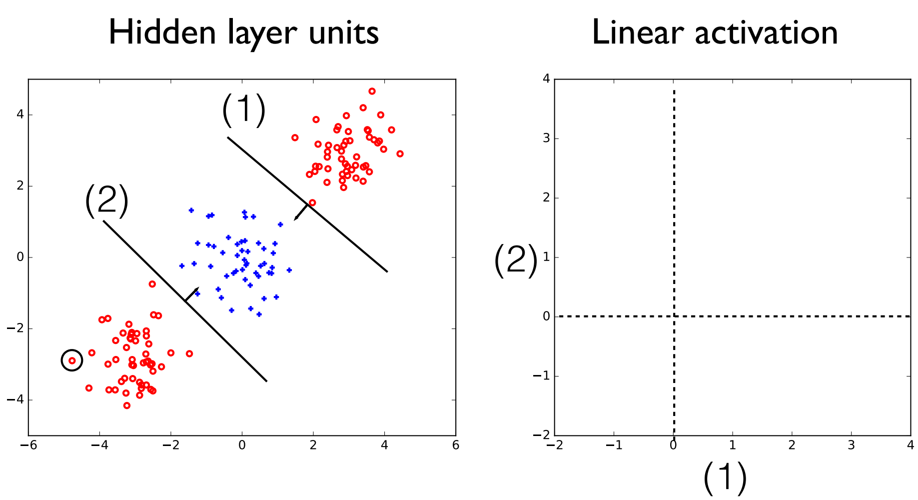
Hidden layer representation



Hidden layer representation

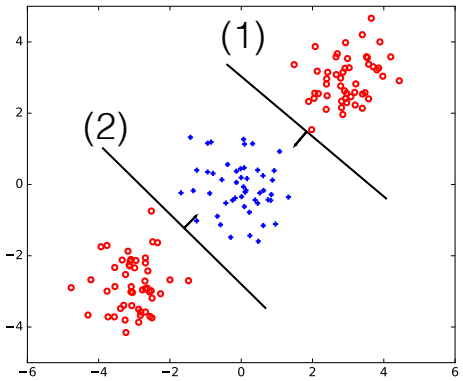


Hidden layer representation

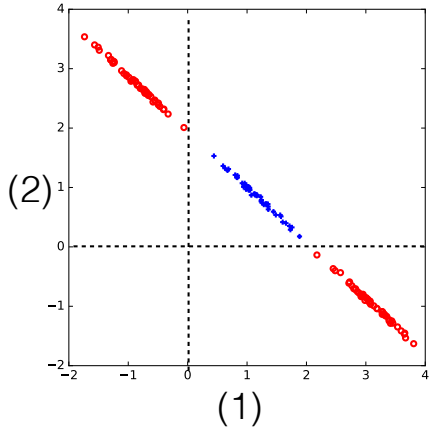


Hidden layer representation

Hidden layer units

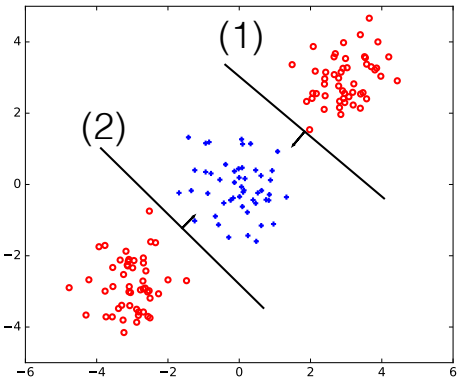


Linear activation

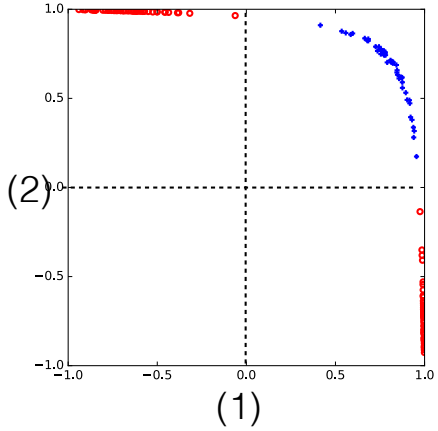


Hidden layer representation

Hidden layer units

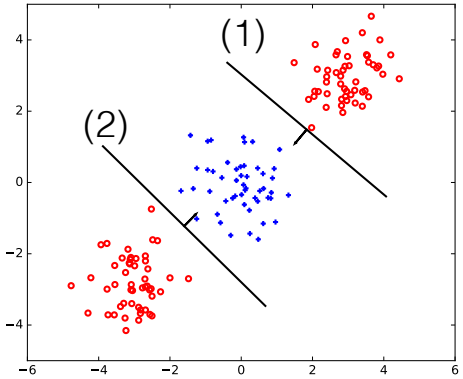


tanh activation

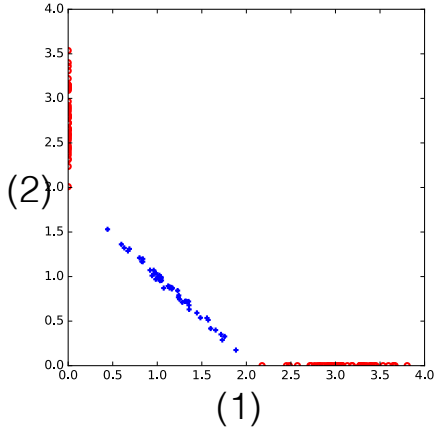


Hidden layer representation

Hidden layer units

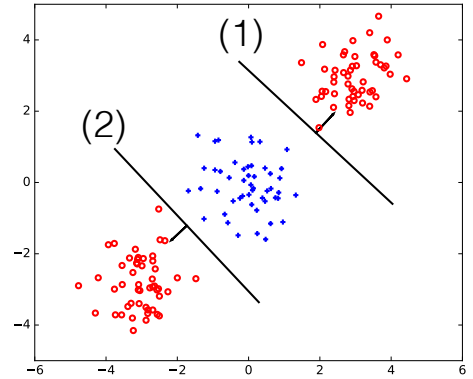


ReLU activation



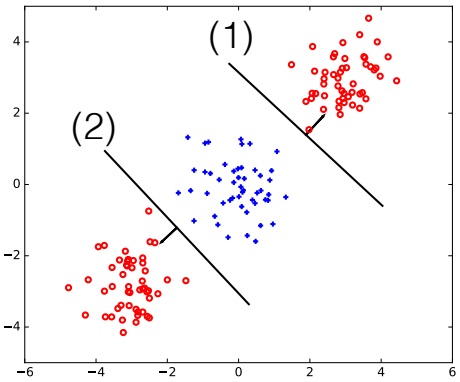
Does orientation matter?

Hidden layer units

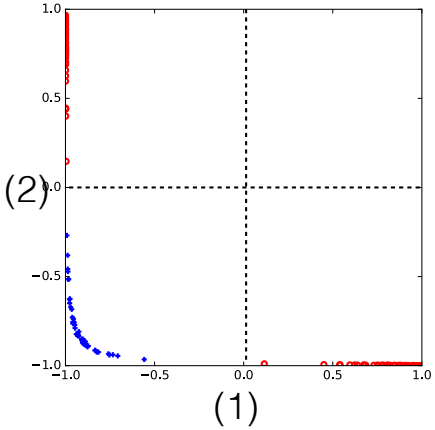


Does orientation matter?

Hidden layer units

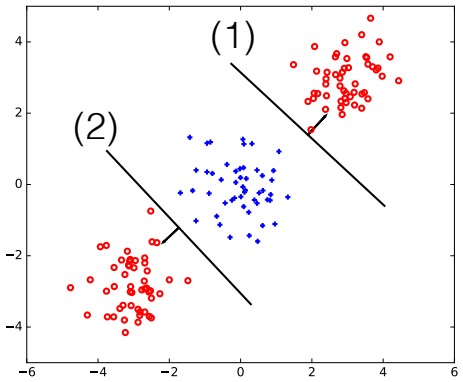


tanh activation

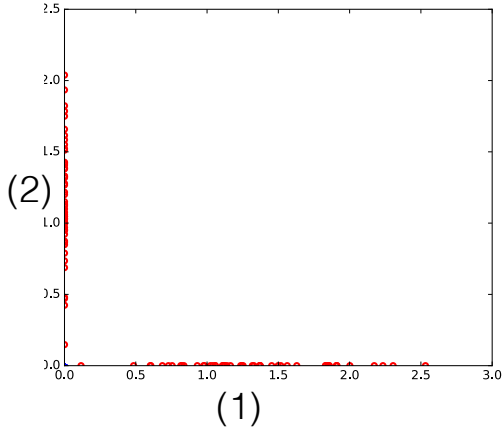


Does orientation matter?

Hidden layer units

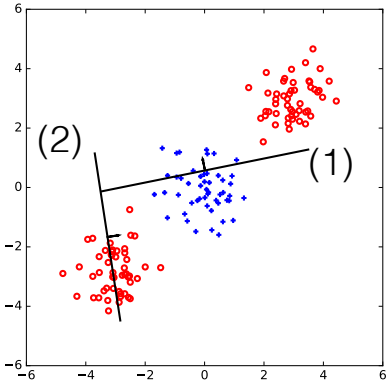


ReLU activation



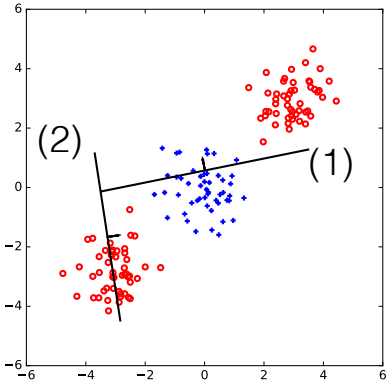
Random hidden units

Hidden layer units

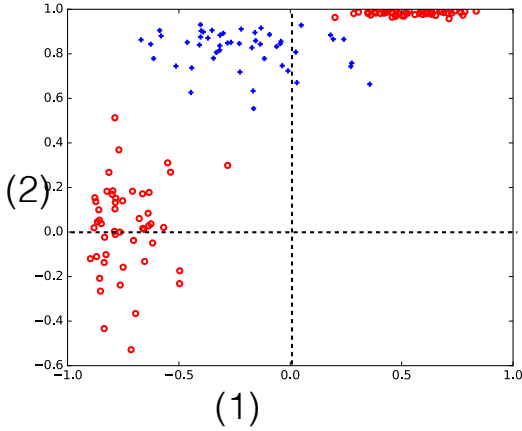


Random hidden units

Hidden layer units

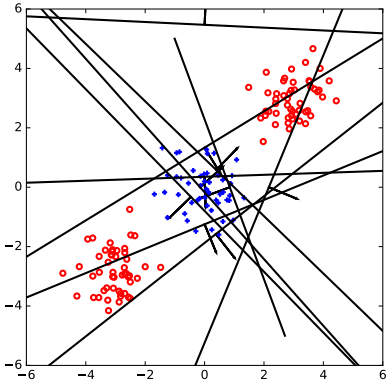


tanh activation



Random hidden units

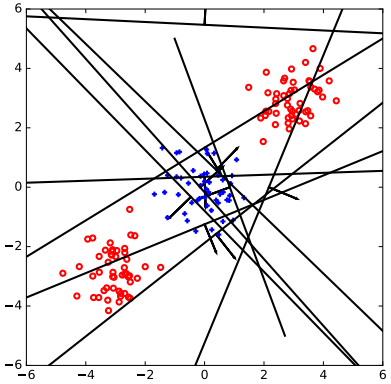
Hidden layer units



(10 randomly chosen units)

Random hidden units

Hidden layer units

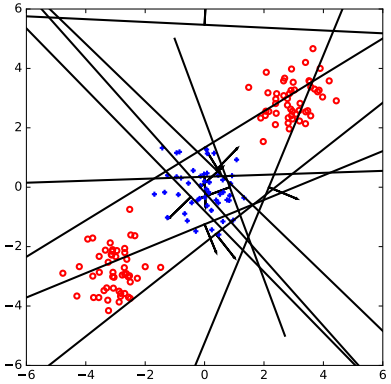


(10 randomly chosen units)

Are the points
linearly separable
in the resulting
10 dimensional space?

Random hidden units

Hidden layer units



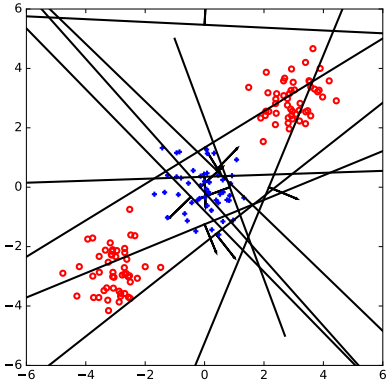
(10 randomly chosen units)

Are the points
linearly separable
in the resulting
10 dimensional space?

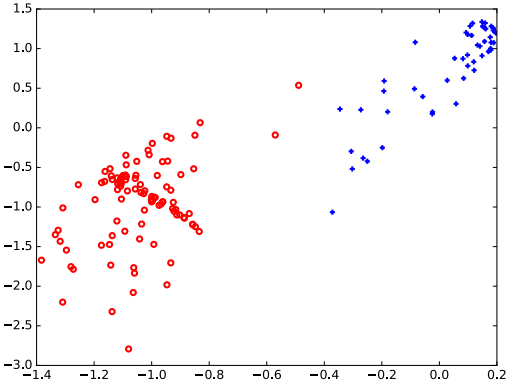
YES!

Random hidden units

Hidden layer units



(10 randomly chosen units)



what are the coordinates??

Summary

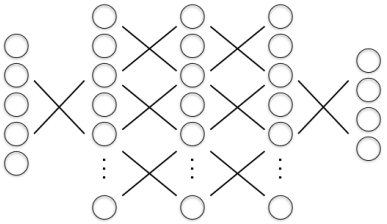
- Units in neural networks are linear classifiers, just with different output non-linearity
- The units in feed-forward neural networks are arranged in layers (input, hidden,…, output)
- By learning the parameters associated with the hidden layer units, we learn how to represent examples (as hidden layer activations)
- The representations in neural networks are learned directly to facilitate the end-to-end task
- A simple classifier (output unit) suffices to solve complex classification tasks if it operates on the hidden layer representations

Feed-forward Neural Networks (Part 2: learning)

Outline (part 2)

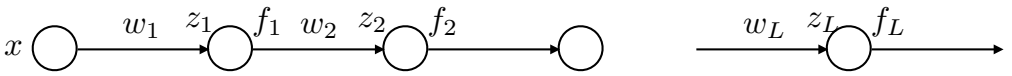
- Learning feed-forward neural networks
- SGD and back-propagation

Learning neural networks

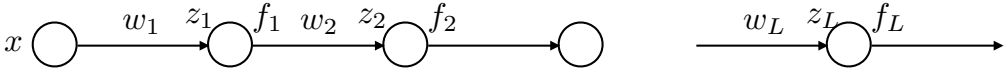


Simple example

- A long chain like neural network

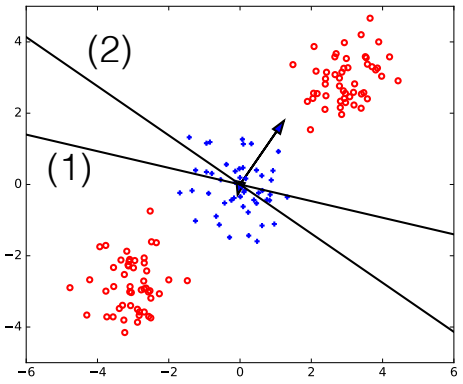


Back-propagation

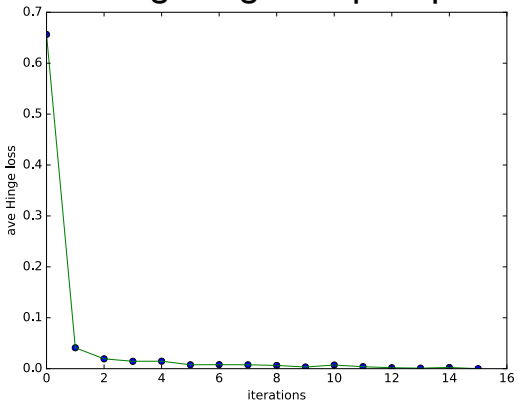


2 hidden units: training

Initial network (hidden units)

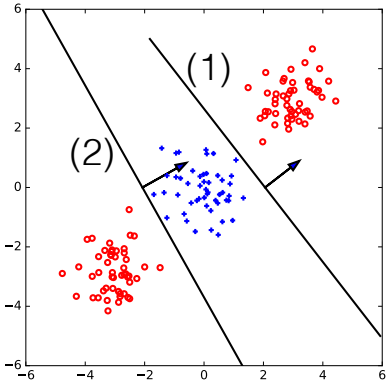


Average hinge loss per epoch

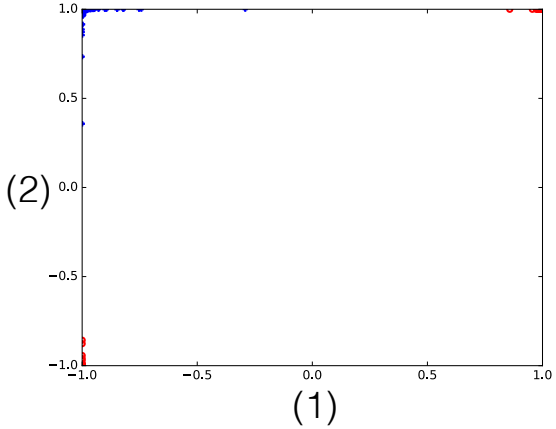


2 hidden units: training

- After ~10 passes through the data

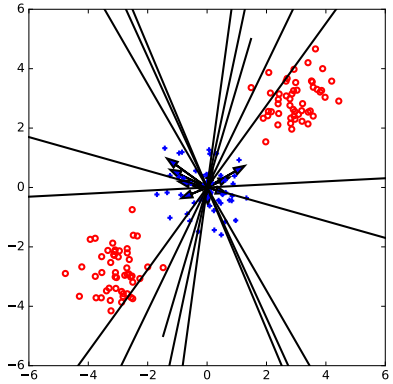


hidden unit activations



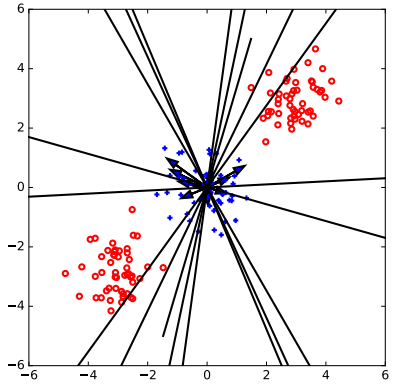
10 hidden units

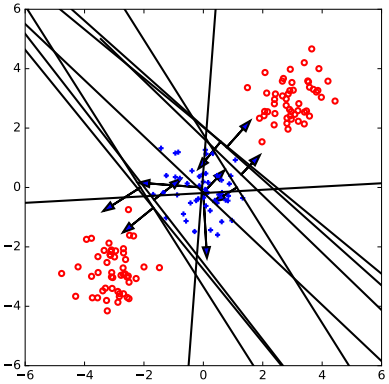
- Randomly initialized weights (zero offset) for the hidden units



10 hidden units

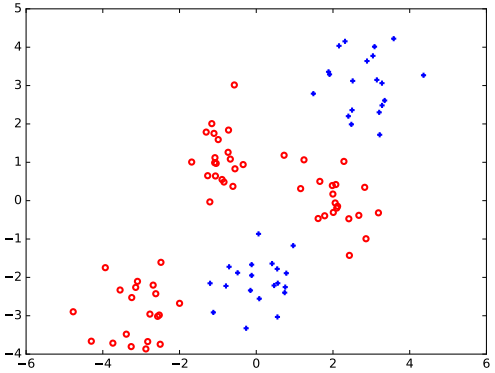
- After ~ 10 epochs the hidden units are arranged in a manner sufficient for the task (but not otherwise perfect)





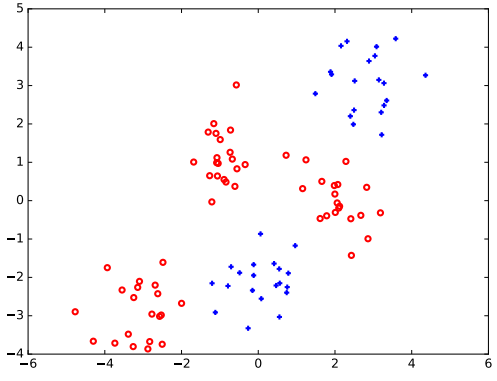
Decisions (and a harder task)

- 2 hidden units can no longer solve this task

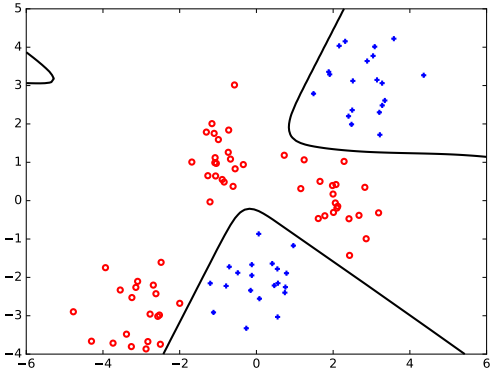


Decisions (and a harder task)

- 2 hidden units can no longer solve this task

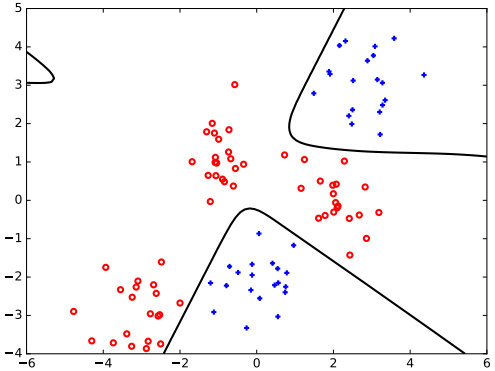


10 hidden units

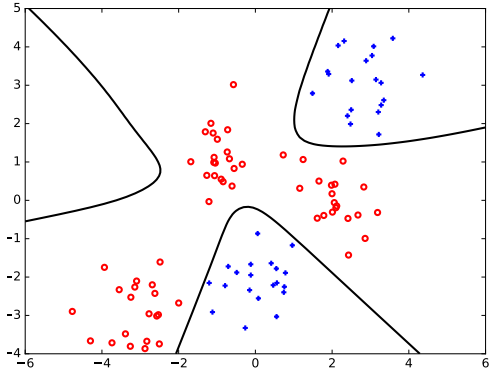


Decisions (and a harder task)

10 hidden units



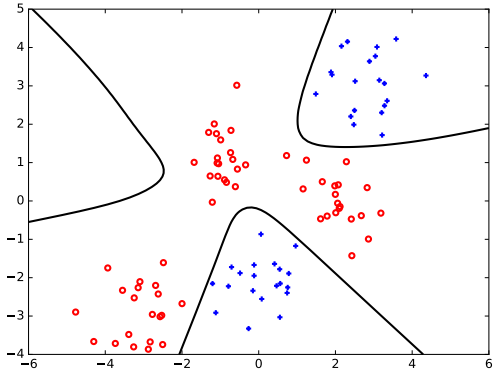
100 hidden units



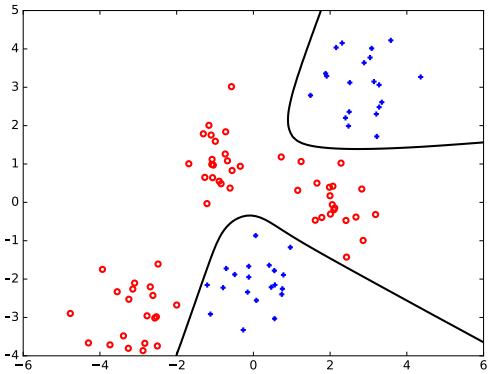
Decision boundaries

- Symmetries introduced in initialization can persist…

100 hidden units
(zero offset initialization)



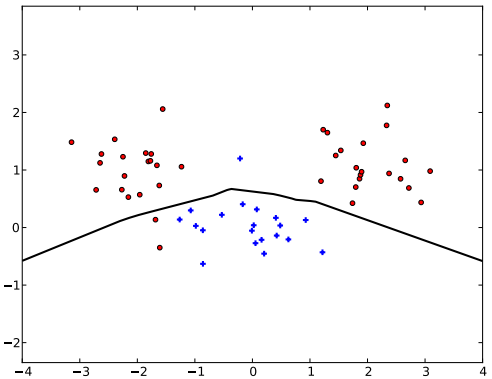
100 hidden units
(random offset initialization)



Size, optimization

- Many recent architectures use ReLU units (cheap to evaluate, sparsity)
- Easier to learn as large models…

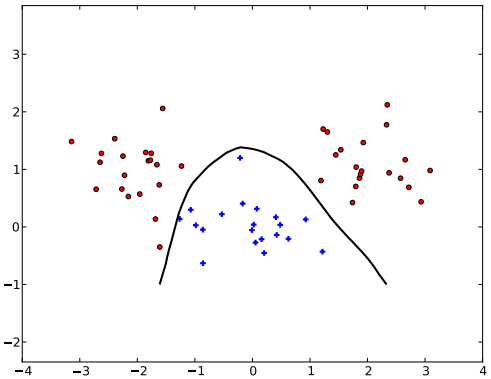
10 hidden units



Size, optimization

- Many recent architectures use ReLU units (cheap to evaluate, sparsity)
- Easier to learn as large models…

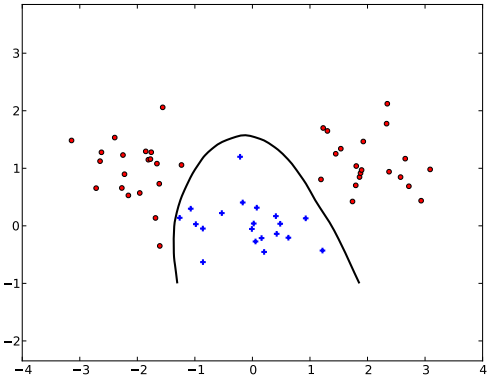
100 hidden units



Size, optimization

- Many recent architectures use ReLU units (cheap to evaluate, sparsity)
- Easier to learn as large models…

500 hidden units



Summary (part 2)

- Neural networks can be learned with SGD similarly to linear classifiers
- The derivatives necessary for SGD can be evaluated effectively via back-propagation
- Multi-layer neural network models are complicated… we are no longer guaranteed to reach global (only local) optimum with SGD
- Larger models tend to be easier to learn … units only need to be adjusted so that they are, collectively, sufficient to solve the task

Modeling with Machine Learning: RNN (part 1)

Outline (part 1)

- Modeling sequences
- The problem of encoding sequences
- Recurrent Neural Networks (RNNs)

Temporal/sequence problems

- How to cast as a supervised learning problem?

Temporal/sequence problems

- How to cast as a supervised learning problem?

- Historical data can be broken down into feature vectors and target values (sliding window)

$$\begin{bmatrix} 0.82 \\ 0.80 \\ 0.73 \\ 0.72 \end{bmatrix} \qquad 0.89$$

$$\phi(t) \qquad y^{(t)}$$

Temporal/sequence problems

- Language modeling: what comes next?

This course has been a tremendous …

Temporal/sequence problems

- Language modeling: what comes next?

This course has been a tremendous …

tremendous $\begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}$ **?**

a $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

$\phi(t)$ $y^{(t)}$

Temporal/sequence problems

- Language modeling: what comes next?

This course has been a tremendous …

a $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ **tremendous**

been $\begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$

$\phi(t)$ $y^{(t)}$

What are we missing?

- Sequence prediction problems can be recast in a form amenable to feed-forward neural networks
- But we have to engineer how "history" is mapped to a vector (representation). This vector is then fed into, e.g., a neural network
  - how many steps back should we look at?
  - how to retain important items mentioned far back?
- Instead, we would like to learn how to encode the "history" into a vector

Learning to encode/decode

- Language modeling

  This course has been a | → success (?)

- Sentiment classification

  I have seen better lectures → -1

- Machine translation

  I have seen better lectures → Olen nähnyt parempia luentoja

encoding **decoding**

Key concepts

- **Encoding** (this lecture)
  - e.g., mapping a sequence to a vector
- **Decoding** (next lecture)
  - e.g., mapping a vector to, e.g., a sequence

Encoding everything

words $\begin{bmatrix}.1\\.3\\.4\end{bmatrix}$ $\begin{bmatrix}.7\\.1\\.0\end{bmatrix}$ $\begin{bmatrix}.2\\.8\\.3\end{bmatrix}$ ...

"Efforts and courage are not enough without purpose and direction" — JFK

sentences $\begin{bmatrix}.2\\.3\\.6\end{bmatrix}$

images $\begin{bmatrix}.3\\.3\\.5\end{bmatrix}$

1. Wikimedia, public domain

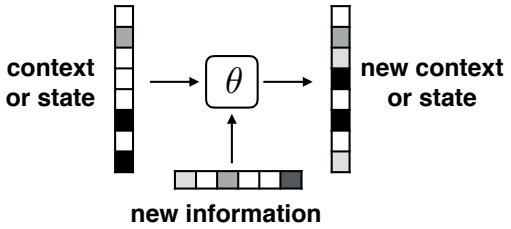
$\begin{bmatrix}.2\\.4\\.6\end{bmatrix}$

events

2. Defense.gov, Public Doman,

Example: encoding sentences

- Easy to introduce adjustable "lego pieces" and optimize them for end-to-end performance



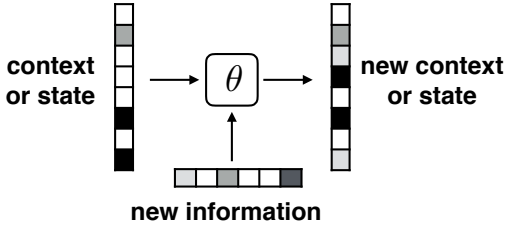
RNN

<null>

Efforts and courage are not ...

Example: encoding sentences

- Easy to introduce adjustable "lego pieces" and optimize them for end-to-end performance



RNN

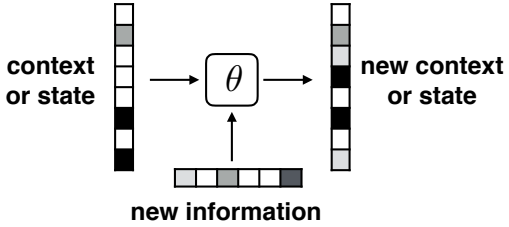
$$s_t = \tanh(W^{s,s}s_{t-1} + W^{s,x}x_t)$$

<null>

Efforts and courage are not ...

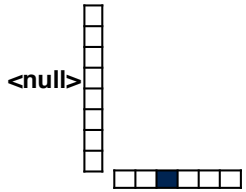
Example: encoding sentences

- Easy to introduce adjustable "lego pieces" and optimize them for end-to-end performance



RNN

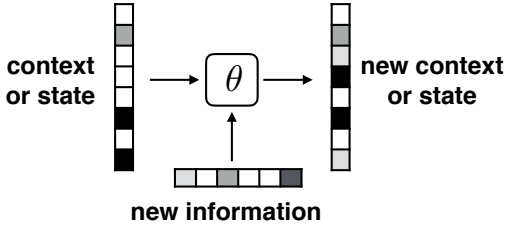
$$s_t = \tanh(W^{s,s}s_{t-1} + W^{s,x}x_t)$$



Efforts and courage are not ...

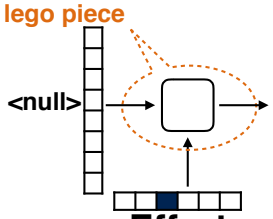
Example: encoding sentences

- Easy to introduce adjustable "lego pieces" and optimize them for end-to-end performance



RNN

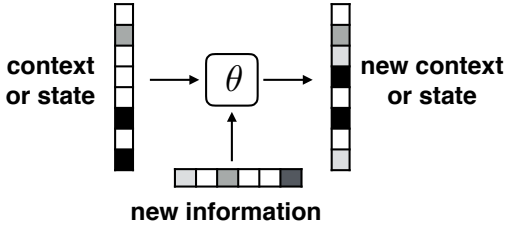
$$s_t = \tanh(W^{s,s}s_{t-1} + W^{s,x}x_t)$$



Efforts and courage are not ...

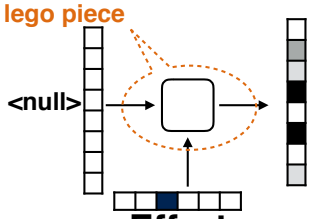
Example: encoding sentences

- Easy to introduce adjustable "lego pieces" and optimize them for end-to-end performance



RNN

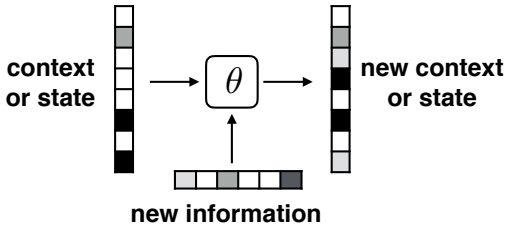
$$s_t = \tanh(W^{s,s}s_{t-1} + W^{s,x}x_t)$$



Efforts and courage are not ...

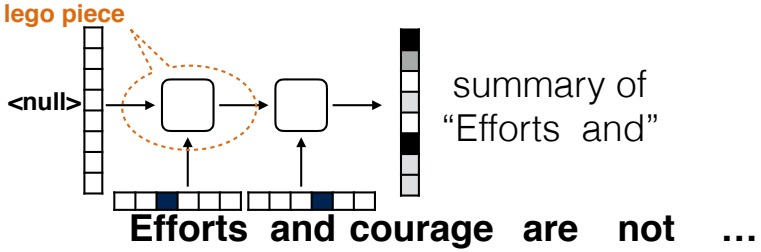
Example: encoding sentences

- Easy to introduce adjustable "lego pieces" and optimize them for end-to-end performance



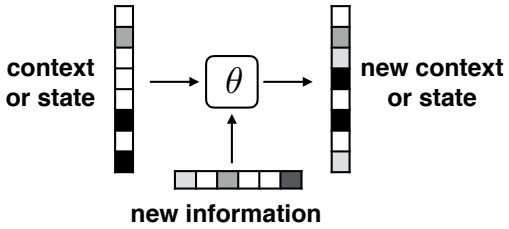
RNN

$$s_t = \tanh(W^{s,s}s_{t-1} + W^{s,x}x_t)$$



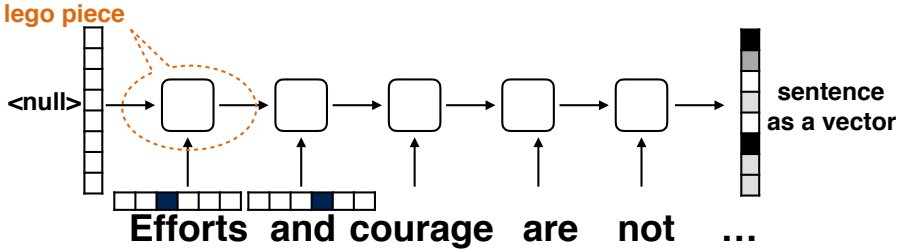
Example: encoding sentences

- Easy to introduce adjustable "lego pieces" and optimize them for end-to-end performance



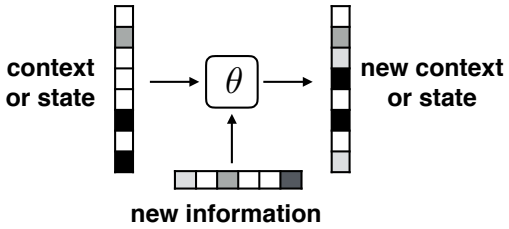
RNN

$$s_t = \tanh(W^{s,s}s_{t-1} + W^{s,x}x_t)$$



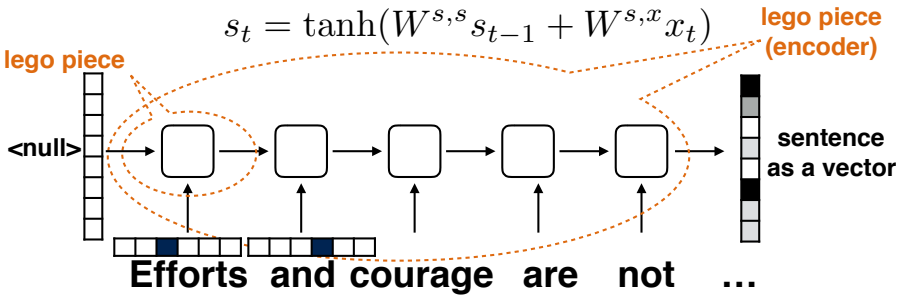
Example: encoding sentences

- Easy to introduce adjustable "lego pieces" and optimize them for end-to-end performance



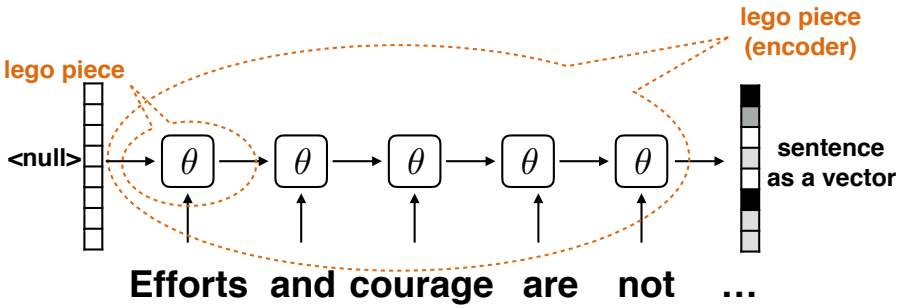
RNN

$$s_t = \tanh(W^{s,s}s_{t-1} + W^{s,x}x_t)$$



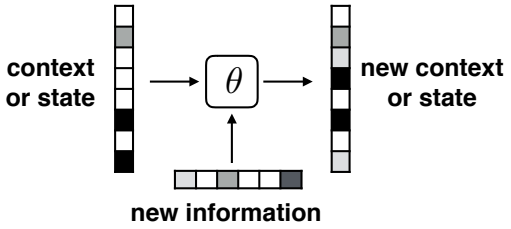
Example: encoding sentences

- There are three differences between the encoder (unfolded RNN) and a standard feed-forward architecture
  - input is received at each layer (per word), not just at the beginning as in a typical feed-forward network
  - the number of layers varies, and depends on the length of the sentence
  - parameters of each layer (representing an application of an RNN) are shared (same RNN at each step)



What's in the box?

- We can make the RNN more sophisticated…

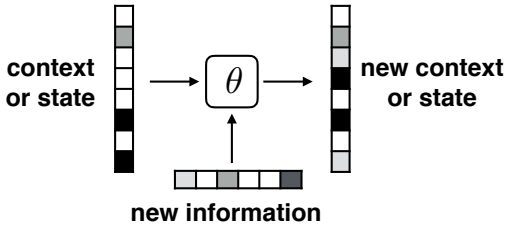


basic RNN

$$s_t = \tanh(W^{s,s} s_{t-1} + W^{s,x} x_t)$$

What's in the box?

- We can make the RNN more sophisticated…

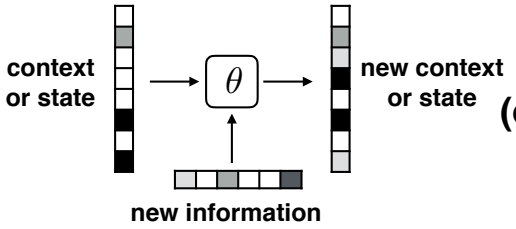


simple gated RNN

$$g_t = \text{sigmoid}(W^{g,s} s_{t-1} + W^{g,x} x_t)$$
$$s_t = (1 - g_t) \odot s_{t-1} + g_t \odot \tanh(W^{s,s} s_{t-1} + W^{s,x} x_t)$$

What's in the box?

- We can make the RNN more sophisticated…



LSTM (one of many)

$$f_t = \text{sigmoid}(W^{f,h} h_{t-1} + W^{f,x} x_t) \quad \text{forget gate}$$
$$i_t = \text{sigmoid}(W^{i,h} h_{t-1} + W^{i,x} x_t) \quad \text{input gate}$$
$$o_t = \text{sigmoid}(W^{o,h} h_{t-1} + W^{o,x} x_t) \quad \text{output gate}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W^{c,h} h_{t-1} + W^{c,x} x_t) \quad \text{memory cell}$$
$$h_t = o_t \odot \tanh(c_t) \quad \text{visible state}$$

Key things

- Neural networks for sequences: encoding
- RNNs, unfolded
  - state evolution, gates
  - relation to feed-forward neural networks
  - back-propagation (conceptually)
- Issues: vanishing/exploding gradient
- LSTM (operationally)

Attribution List - Machine learning - 6.86x

1.
Unit 1 Lecture 8: Introduction to Machine Learning
Photo portrait of John F. Kennedy
Slides: #11
Object Source / URL: https://commons.wikimedia.org/wiki/File:John_F._Kennedy,_White_House_photo_portrait,_looking_up.jpg
Citation/Attribution: This file is a work of an employee of the Executive Office of the President of the United States, taken or made as part of that person's official duties. As a work of the U.S. federal government, it is in the public domain.

2.
Unit 1 Lecture 8: Introduction to Machine Learning
John F Kennedy speech on May 25th, 1961
Slides: #11
Object Source / URL: https://www.defense.gov/Explore/Spotlight/Apollo-11/
Citation/Attribution: Image from defense.gov.This work is in the public domain

Modeling with Machine Learning: RNN (part 2)

Recall: learning to encode/decode

- Language modeling

  This course has been a |  success (?)

- Sentiment classification

  I have seen better lectures  -1

- Machine translation

  I have seen better lectures  Olen nähnyt parempia luentoja

encoding **decoding**

Outline (part 2)

- Modeling sequences: language models
  - Markov models
  - as neural networks
  - hidden state, Recurrent Neural Networks (RNNs)
- Example: decoding images into sentences

Markov Models

- Next word in a sentence depends on previous symbols already written (history = one, two, or more words)

  The lecture leaves me bumfuzzled

- Similar, next character in a word depends on previous characters already written

  bumfuzzled

- We can model such kth order dependences between symbols with Markov Models

Markov Language Models

- Let $w \in V$ denote the set of possible words/symbols that includes
  - an UNK symbol for any unknown word (out of vocabulary)
  - <beg> symbol for specifying the start of a sentence
  - <end> symbol for specifying the end of the sentence

<beg> The lecture leaves me UNK <end>

w_0 w_1 w_2 w_3 w_4 w_5 w_6

- In a first order Markov model (bigram model), the next symbol only depends on the previous one

A first order Markov model

- Each symbol (except <beg>) in the sequence is predicted using the same conditional probability table until an <end> symbol is seen

w_{i-1} \ w_i	ML	course	is	UNK	<end>
<beg>	0.7	0.1	0.1	0.1	0.0
ML	0.1	0.5	0.2	0.1	0.1
course	0.0	0.0	0.7	0.1	0.2
is	0.1	0.3	0.0	0.6	0.0
UNK	0.1	0.2	0.2	0.3	0.2

Sampling from a Markov model

w_{i-1} \ w_i	ML	course	is	UNK	<end>
<beg>	0.7	0.1	0.1	0.1	0.0
ML	0.1	0.5	0.2	0.1	0.1
course	0.0	0.0	0.7	0.1	0.2
is	0.1	0.3	0.0	0.6	0.0
UNK	0.1	0.2	0.2	0.3	0.2

Maximum likelihood estimation

- The goal is to maximize the probability that the model can generate all the observed sentences (corpus S)

$$s \in S, \quad s = \{w_1^s, w_2^s, \ldots, w_{|s|}^s\}$$

Maximum likelihood estimation

- The goal is to maximize the probability that the model can generate all the observed sentences (corpus S)

$$s \in S, \quad s = \{w_1^s, w_2^s, \ldots, w_{|s|}^s\}$$

- The ML estimate is obtained as normalized counts of successive word occurrences (matching statistics)

Feature based Markov Model

- We can also represent the Markov model as a feed-forward neural network (very extendable)

Feature based Markov Model

- We can also represent the Markov model as a feed-forward neural network (very extendable)

Temporal/sequence problems

- Language modeling: what comes next?

This course has been a tremendous ...

tremendous $\begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}$

?

a $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

$x^{(t)}$ $y^{(t)}$

Temporal/sequence problems

- A trigram language model

tremendous $\begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}$

a $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

$y^{(t)}$

$x^{(t)}$

Temporal/sequence problems

- A trigram language model

tremendous $\begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}$

a $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

$y^{(t)}$

$x^{(t)}$

RNNs for sequences

- Language modeling: what comes next?

This course has been a tremendous | **...**

tremendous $\begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}$ **?**

RNNs for sequences

- Language modeling: what comes next?

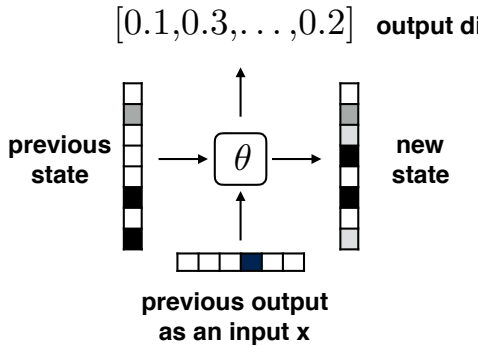
This course has been a tremendous | **...**

tremendous $\begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}$ **?**

$$s_t = \tanh(W^{s,s} s_{t-1} + W^{s,x} x_t) \quad \text{state}$$
$$p_t = \text{softmax}(W^o s_t) \quad \text{output distribution}$$

Decoding, RNNs

- Our RNN now also produces an output (e.g., a word) as well as update its state

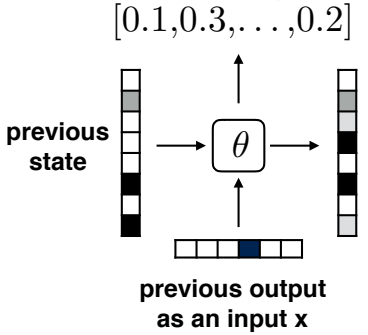


basic RNN

$$s_t = \tanh(W^{s,s}s_{t-1} + W^{s,x}x_t) \quad \text{state}$$

$$\rightarrow \quad p_t = \text{softmax}(W^o s_t) \quad \text{output distribution}$$

Decoding, LSTM



LSTM

$$f_t = \text{sigmoid}(W^{f,h}h_{t-1} + W^{f,x}x_t) \quad \text{forget gate}$$

$$i_t = \text{sigmoid}(W^{i,h}h_{t-1} + W^{i,x}x_t) \quad \text{input gate}$$

$$o_t = \text{sigmoid}(W^{o,h}h_{t-1} + W^{o,x}x_t) \quad \text{output gate}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W^{c,h}h_{t-1} + W^{c,x}x_t) \quad \text{memory cell}$$

$$h_t = o_t \odot \tanh(c_t) \quad \text{visible state}$$

$$\rightarrow \quad p_t = \text{softmax}(W^o h_t) \quad \text{output distribution}$$

Decoding (into a sentence)

- Our RNN now needs to also produce an output (e.g., a word) as well as update its state

vector encoding of a sentence "I have seen better lectures"

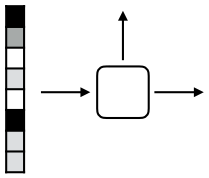
Decoding (into a sentence)

- Our RNN now needs to also produce an output (e.g., a word) as well as update its state

distribution over the possible words

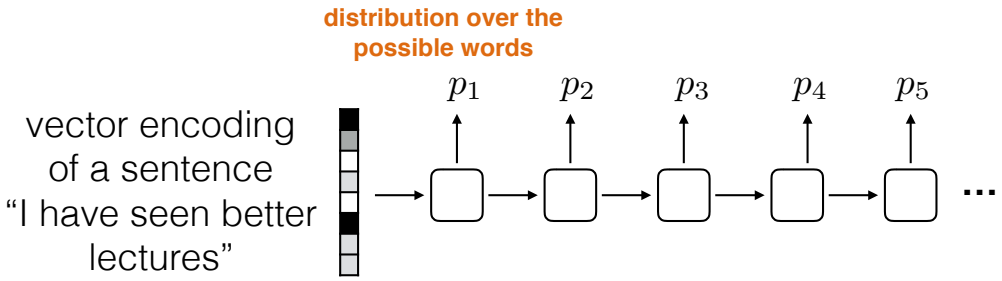
p_1

vector encoding of a sentence "I have seen better lectures"



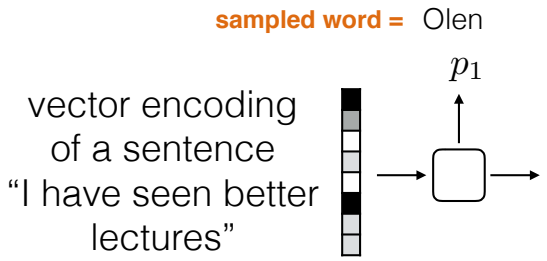
Decoding (into a sentence)

- Our RNN now needs to also produce an output (e.g., a word) as well as update its state



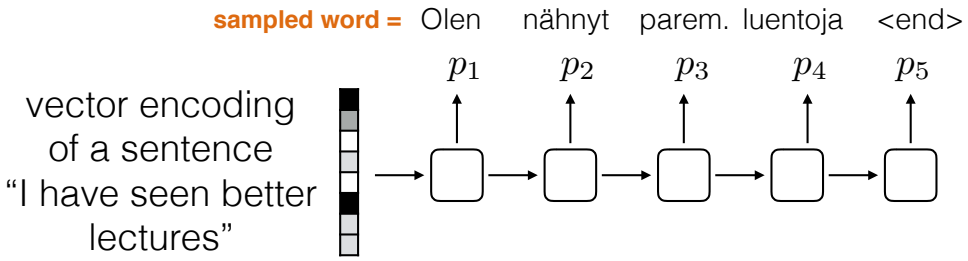
Decoding (into a sentence)

- Our RNN now needs to also produce an output (e.g., a word) as well as update its state



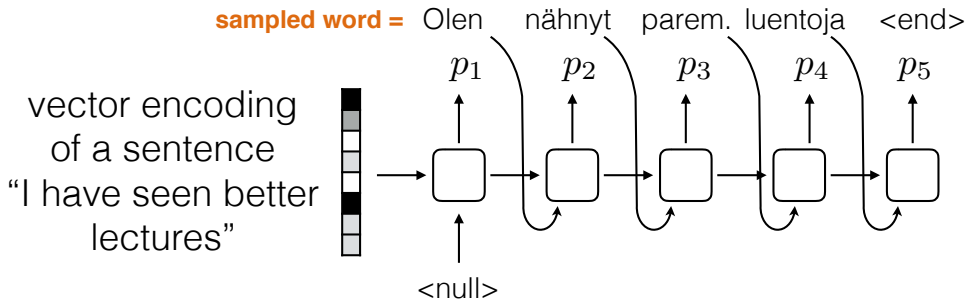
Decoding (into a sentence)

- Our RNN now needs to also produce an output (e.g., a word) as well as update its state

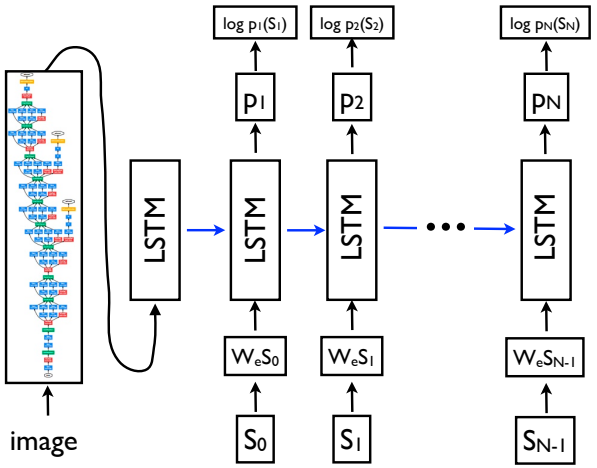


Decoding (into a sentence)

- Our RNN now needs to also produce an output (e.g., a word) as well as update its state
- The output is fed in as an input (to gauge what's left)



Mapping images to text



Examples

Key things

- Markov models for sequences
  - how to formulate, estimate, sample sequences from
- RNNs for generating (decoding) sequences
  - relation to Markov models
  - evolving hidden state
  - sampling from
- Decoding vectors into sequences

Machine Learning: CNNs

Outline

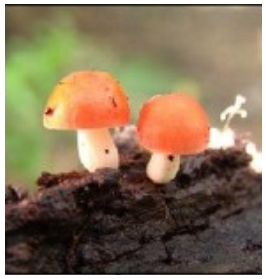
- Convolutional neural networks (CNNs)
  - why not use unstructured feed-forward models?
  - key parts: convolution, pooling
  - examples

Our problem: image classification

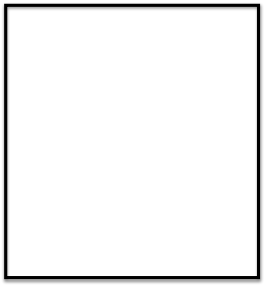
- E.g., image classification (1K categories)

Image	Category
	mushroom
	cherry
...	...

Feed-forward networks

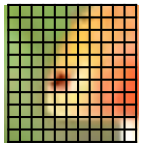


input



layer 1

Patch classifier/filter

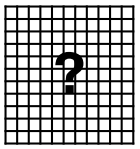


11x11
input

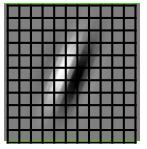
11x11
weights

Convolution

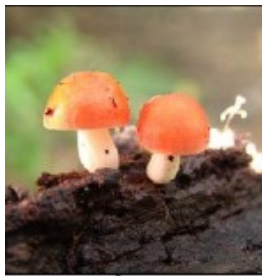
$$f = \mathrm{ReLU}\left(\; ? \; \cdot \; \right)$$



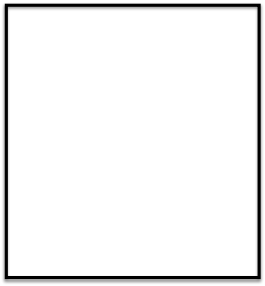
11x11
input



11x11
weights



input



feature map

Convolution, feature map

filter
patch

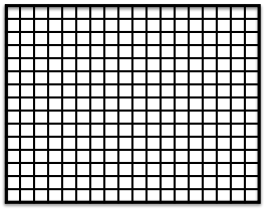
original image

resulting feature map

Pooling

- We wish to know whether a feature was there but not exactly where it was

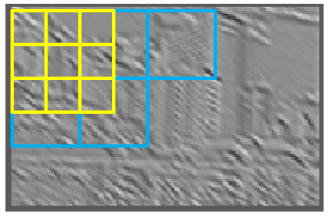
feature map



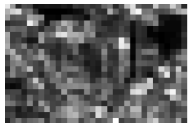
pooled map

Pooling (max)

- Pooling region and "stride" may vary
  - pooling induces translation invariance at the cost of spatial resolution
  - stride reduces the size of the resulting feature map

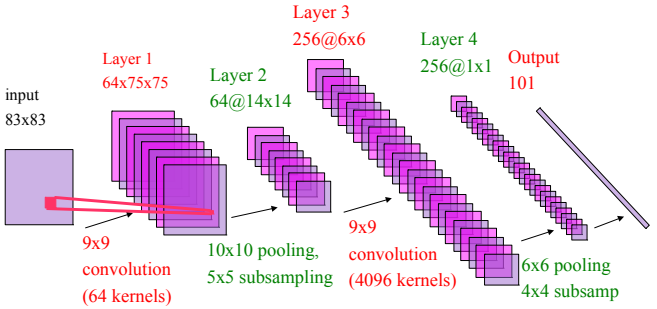


feature map



feature map
after max pooling

Convolutional Neural Network



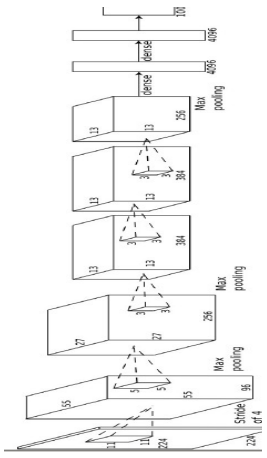
- Non-Linearity: half-wave rectification, shrinkage function, sigmoid
- Pooling: average, L1, L2, max
- Training: Supervised (1988-2006), Unsupervised+Supervised (2006-now)

(LeCun 13')

Convolutional Neural Network

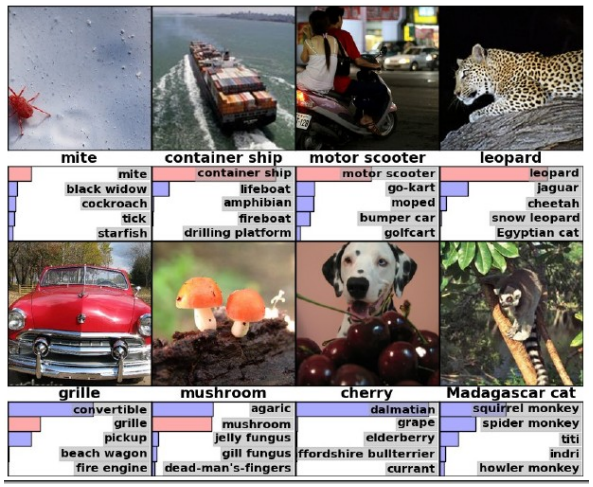
- 1000 categories, 1.5 Million labeled training samples
- Won the 2012 ImageNet LSVRC. 60 Million parameters, 832M MAC ops

4M	FULL CONNECT	4Mflop
16M	FULL 4096/ReLU	16M
37M	FULL 4096/ReLU	37M
	MAX POOLING	
442K	CONV 3x3/ReLU 256fm	74M
1.3M	CONV 3x3ReLU 384fm	224M
884K	CONV 3x3/ReLU 384fm	149M
	MAX POOLING 2x2sub	
	LOCAL CONTRAST NORM	
307K	CONV 11x11/ReLU 256fm	223M
	MAX POOL 2x2sub	
	LOCAL CONTRAST NORM	
35K	CONV 11x11/ReLU 96fm	105M



(Krizhevsky et al., 12')

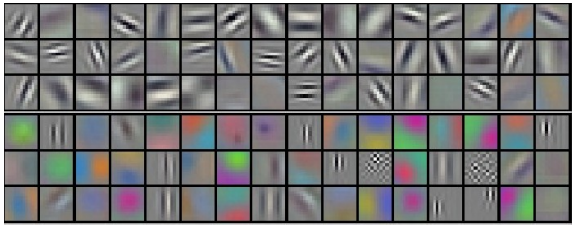
Convolutional Neural Network



(Krizhevsky et al., 12')

ConvNet features

Learned layer 1 CNN filters



96 convolutional filters on the first layer
(filters are of size 11x11x3, applied across
input images of size 224x224x3)

(Krizhevsky et al., 12')