Jelinek Summer School

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Machine Learning

Matt Gormley
June 19, 2017

This section is based on Chapter 1 of (Mitchell, 1997)

DEFINING LEARNING PROBLEMS

Intro Outline

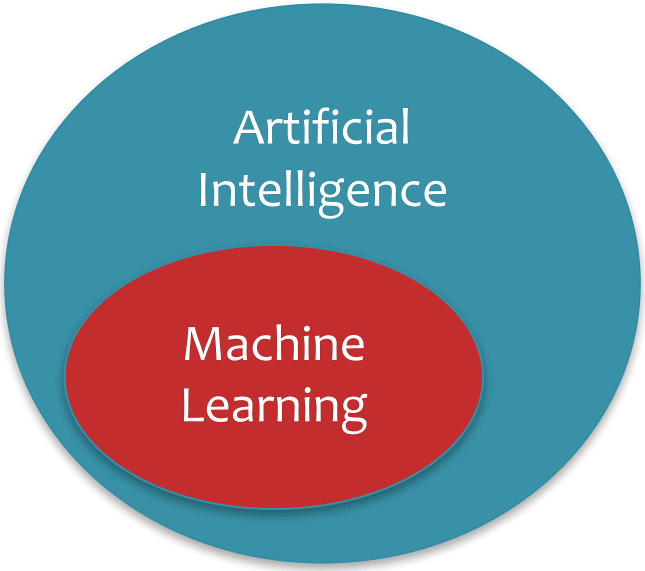
- **Defining Learning Problems**
  - Artificial Intelligence (AI)
  - Mitchell's definition of learning
  - Example learning problems
  - Data annotation
  - The Machine Learning framework

Artificial Intelligence

The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:

- Perception
- Reasoning
- Control / Motion / Manipulation
- Planning
- Communication
- Creativity
- Learning



Artificial Intelligence (AI): Example Tasks:

- Identify objects in an image
- Translate from one human language to another
- Recognize speech
- Assess risk (e.g. in loan application)
- Make decisions (e.g. in loan application)
- Assess potential (e.g. in admission decisions)
- Categorize a complex situation (e.g. medical diagnosis)
- Predict outcome (e.g. medical prognosis, stock prices, inflation, temperature)
- Predict events (default on loans, quitting school, war)
- Plan ahead under perfect knowledge (chess)
- Plan ahead under partial knowledge (Poker, Bridge)

Well-Posed Learning Problems

Three components:

1. Task, *T*
2. Performance measure, *P*
3. Experience, *E*

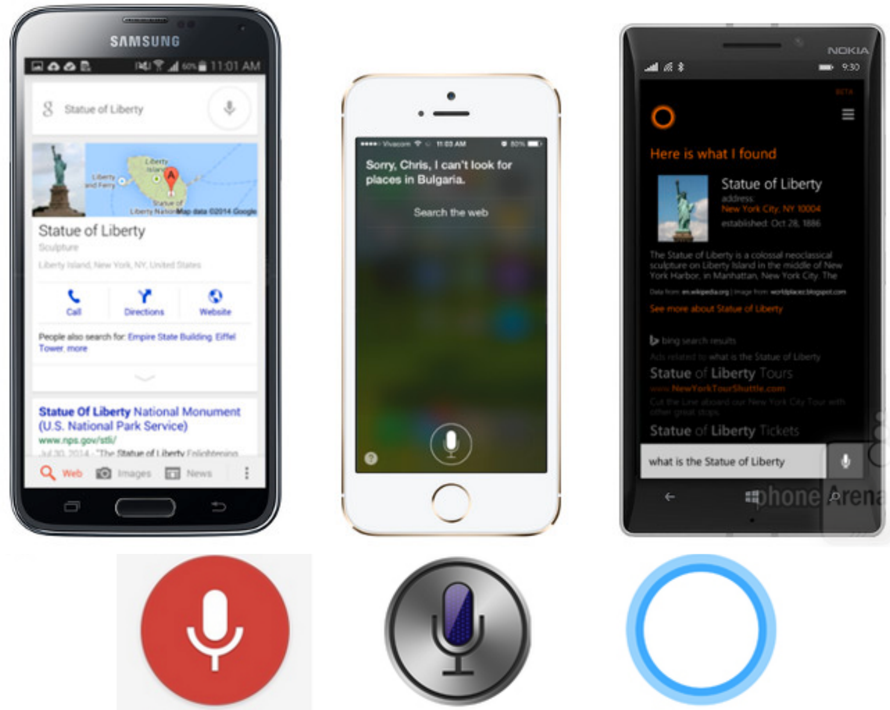
Mitchell's definition of learning:

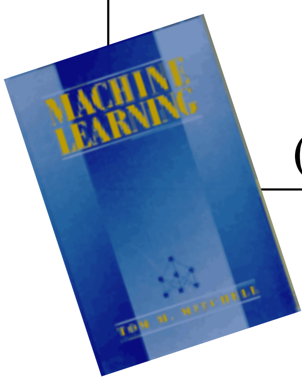
A computer program **learns** if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.

Definition from (Mitchell, 1997)

Example Learning Problems (historical perspective)

1. Learning to recognize spoken words

THEN	NOW
"…the SPHINX system (e.g. Lee 1989) learns speaker-specific strategies for recognizing the primitive sounds (phonemes) and words from the observed speech signal…neural network methods…hidden Markov models…" (Mitchell, 1997)	 Google Now Siri Cortana

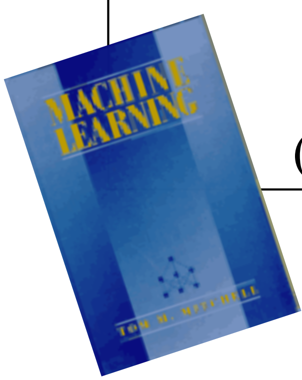


Source: https://www.stonetemple.com/great-knowledge-box-showdown/#VoiceStudyResults

Example Learning Problems (historical perspective)

2. Learning to drive an autonomous vehicle

THEN	NOW
"…the ALVINN system (Pomerleau 1989) has used its learned strategies to drive unassisted at 70 miles per hour for 90 miles on public highways among other cars…" (Mitchell, 1997)	 waymo.com



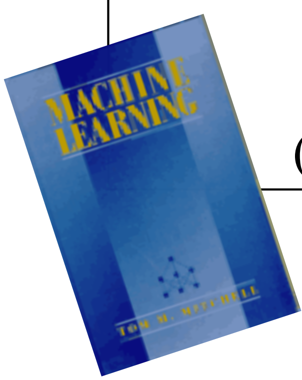
Example Learning Problems (historical perspective)

2. Learning to drive an autonomous vehicle

THEN

"…the ALVINN system (Pomerleau 1989) has used its learned strategies to drive unassisted at 70 miles per hour for 90 miles on public highways among other cars…"

(Mitchell, 1997)



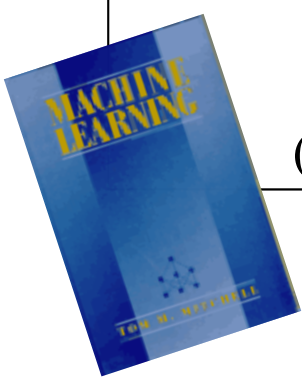
NOW

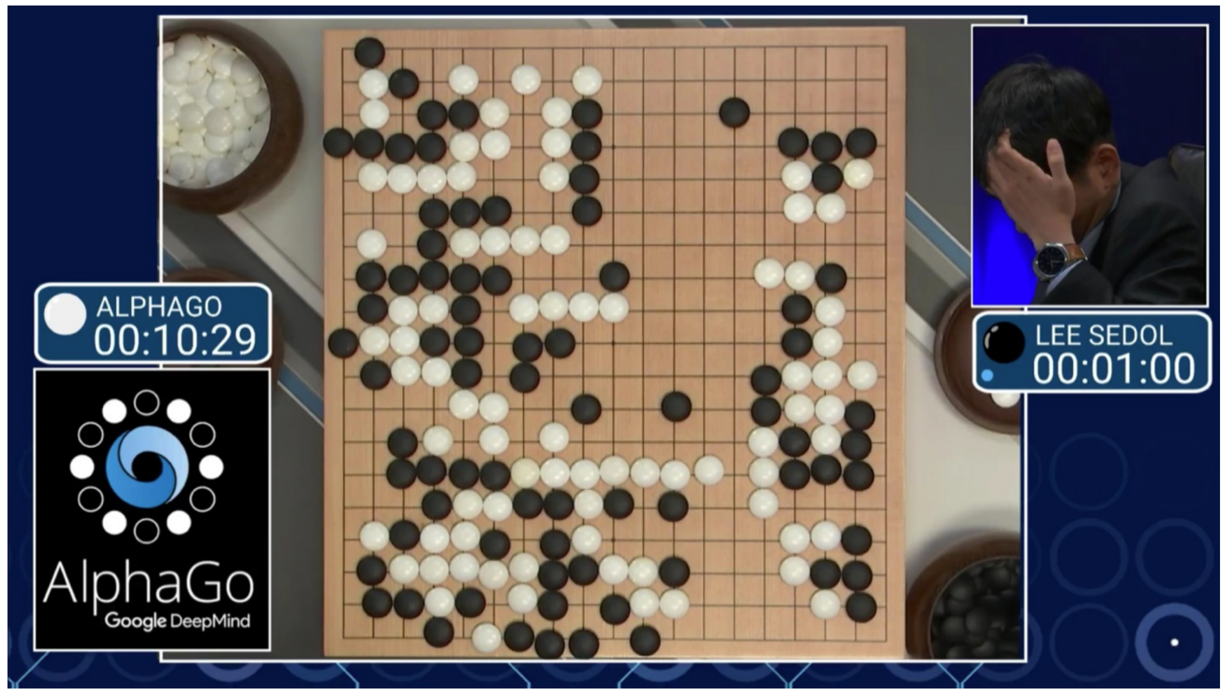
https://www.geek.com/wp-content/uploads/2016/03/uber.jpg

Example Learning Problems (historical perspective)

3. Learning to beat the masters at board games

THEN	NOW
"…the world's top computer program for backgammon, TD-GAMMON (Tesauro, 1992, 1995), learned its strategy by playing over one million practice games against itself…" (Mitchell, 1997)	





Example Learning Problems

3. Learning to beat the masters at **chess**
    1. Task, *T*:
    2. Performance measure, *P*:
    3. Experience, *E*:

Example Learning Problems

4. Learning to **respond to voice commands (Siri)**
   1. Task, *T*:

   2. Performance measure, *P*:

   3. Experience, *E*:

Capturing the Knowledge of Experts

Solution #1: Expert Systems

- Over 20 years ago, we had rule based systems
- Ask the expert to
  1. Obtain a PhD in Linguistics
  2. Introspect about the structure of their native language
  3. Write down the rules they devise

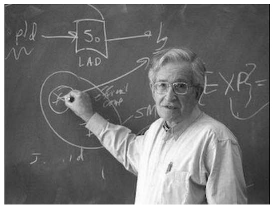
Give me directions to Starbucks
If: "give me directions to X" Then: directions(here, nearest(X))

How do I get to Starbucks?
If: "how do i get to X" Then: directions(here, nearest(X))

Where is the nearest Starbucks?
If: "where is the nearest X" Then: directions(here, nearest(X))

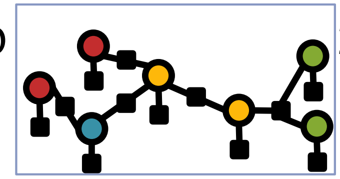
Capturing the Knowledge of Experts

1980



1990

2000



2010

Solution #1: Expert Systems

- Over 20 years ago, we had rule based systems
- Ask the expert to
  1. Obtain a PhD in Linguistics
  2. Introspect about the structure of their native language
  3. Write down the rules they devise

I need directions to Starbucks

```
If: "I need directions to X"
Then: directions(here, nearest(X))
```

Starbucks directions

```
If: "X directions"
Then: directions(here, nearest(X))
```

Is there a Starbucks nearby?

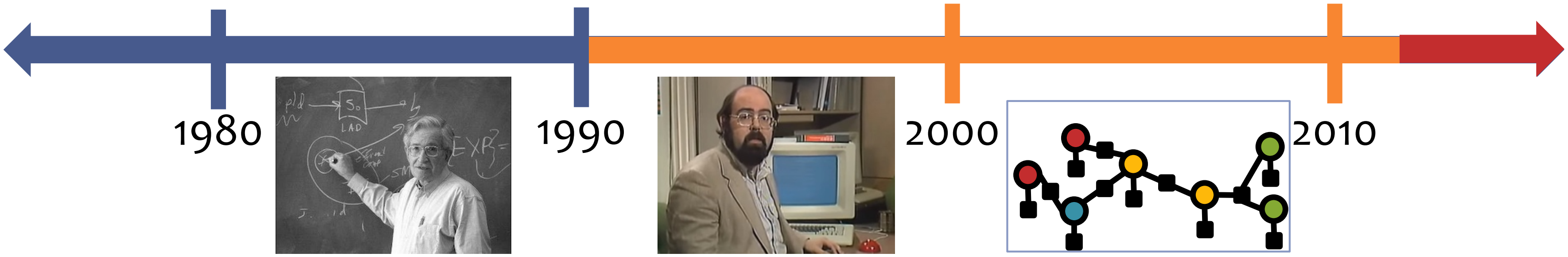
```
If: "Is there an X nearby"
Then: directions(here, nearest(X))
```

Capturing the Knowledge of Experts

Solution #2: Annotate Data and Learn

- Experts:
  - **Very good at** answering questions about specific cases
  - **Not very good at** telling **HOW** they do it
- 1990s: So why not just have them tell you what they do on **SPECIFIC CASES** and then let **MACHINE LEARNING** tell you how to come to the same decisions that they did

Capturing the Knowledge of Experts



Solution #2: Annotate Data and Learn

1. Collect raw sentences $\{x_1, \ldots, x_n\}$
2. Experts annotate their meaning $\{y_1, \ldots, y_n\}$

x_1: How do I get to Starbucks?

y_1: `directions(here, nearest(Starbucks))`

x_2: Show me the closest Starbucks

y_2: `map(nearest(Starbucks))`

x_3: Send a text to John that I'll be late

y_3: `txtmsg(John, I'll be late)`

x_4: Set an alarm for seven in the morning

y_4: `setalarm(7:00AM)`

Example Learning Problems

4. Learning to **respond to voice commands (Siri)**
   1. Task, *T*:
      predicting action from speech
   2. Performance measure, *P*:
      percent of correct actions taken in user pilot study
   3. Experience, *E*:
      examples of (speech, action) pairs

The Machine Learning Framework

- Formulate a task as a mapping from input to output
  - Task examples will usually be pairs: (input, correct_output)
- Formulate performance as an error measure
  - or more generally, as an objective function (aka Loss function)
- Examples:
  - Medical Diagnosis
    - mapping input to one of several classes/categories ➔ Classification
  - Predict tomorrow's Temperature
    - mapping input to a number ➔ Regression
  - Chance of Survival: From patient data to p(survive >= 5 years)
    - mapping input to probability ➔ Density estimation
  - Driving recommendation
    - mapping input into a plan ➔ Planning

Choices in ML Formulation

Often, the same task can be formulated in more than one way:

- Ex. 1: Loan applications
  - creditworthiness/score (regression)
  - probability of default (density estimation)
  - loan decision (classification)
- Ex. 2: Chess
  - Nature of available training examples/experience:
    - expert advice (painful to experts)
    - games against experts (less painful but limited, and not much control)
    - experts' games (almost unlimited, but only ”found data” – no control)
    - games against self (unlimited, flexible, but can you learn this way?)
  - Choice of target function: board→move vs. board→score

How to Approach a Machine Learning Problem

1. Consider your goal → definition of task **T**
   - E.g. make good loan decisions, win chess competitions, …
2. Consider the nature of available (or potential) experience **E**
   - How much data can you get? What would it cost (in money, time or effort)?
3. Choose type of output **O** to learn
   - (Numerical? Category? Probability? Plan?)
4. Choose the Performance measure **P** (error/loss function)

5. Choose a representation for the input X
6. Choose a set of possible solutions **H** (hypothesis space)
   - set of functions h: X → O
   - (often, by choosing a representation for them)
7. Choose or design a learning algorithm
   - for using examples (**E**) to converge on a member of **H** that optimizes **P**

Part I Outline

1. What is Machine Learning?
2. Optimization for ML
3. Function Approximation
4. Linear Regression
5. Logistic Regression
6. Feature Engineering
7. Regularization

OPTIMIZATION

Optimization Outline

- **Optimization for ML**
  - Differences
  - Types of optimization problems
  - Unconstrained optimization
  - Convex, concave, nonconvex
- **Optimization: Closed form solutions**
  - Example: 1-D function
  - Example: higher dimensions
  - Gradient and Hessian
- **Gradient Descent**
  - Example: 2D gradients
  - Algorithm
  - Details: starting point, stopping criterion, line search
- **Stochastic Gradient Descent (SGD)**
  - Expectations of gradients
  - Algorithm
  - Mini-batches
  - Details: mini-batches, step size, stopping criterion
  - Problematic cases for SGD
- **Convergence**
  - Comparison of Newton's method, Gradient Descent, SGD
  - Asymptotic convergence
  - Convergence in practice

Optimization for ML

Not quite the same setting as other fields…

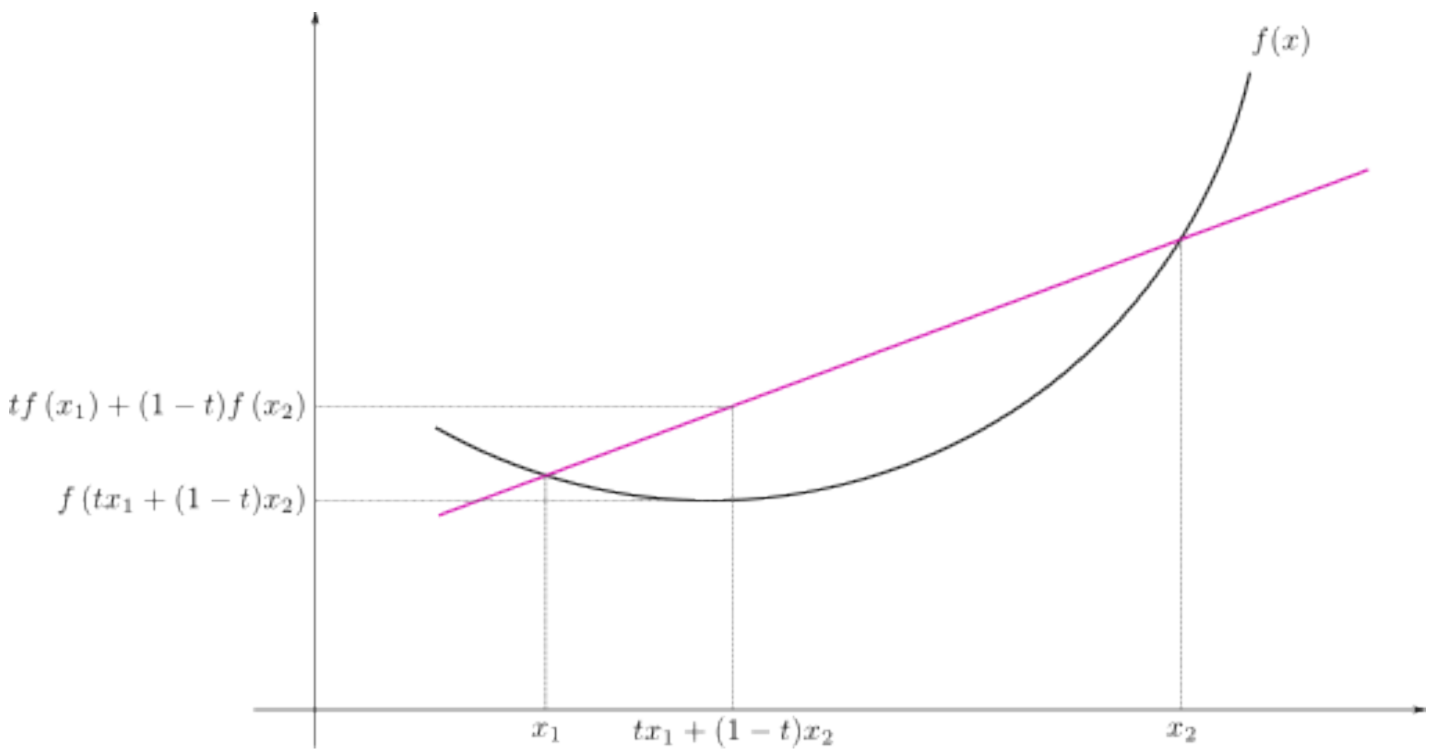
- Function we are optimizing might not be the true goal
  (e.g. likelihood vs generalization error)
- Precision might not matter
  (e.g. data is noisy, so optimal up to 1e-16 might not help)
- Stopping early can help generalization error
  (i.e. "early stopping" is a technique for regularization – discussed more next time)

Convexity

Function $f : \mathbb{R}^M \to \mathbb{R}$ is **convex**
if $\forall\ \mathbf{x}_1 \in \mathbb{R}^M, \mathbf{x}_2 \in \mathbb{R}^M, 0 \leq t \leq 1$:

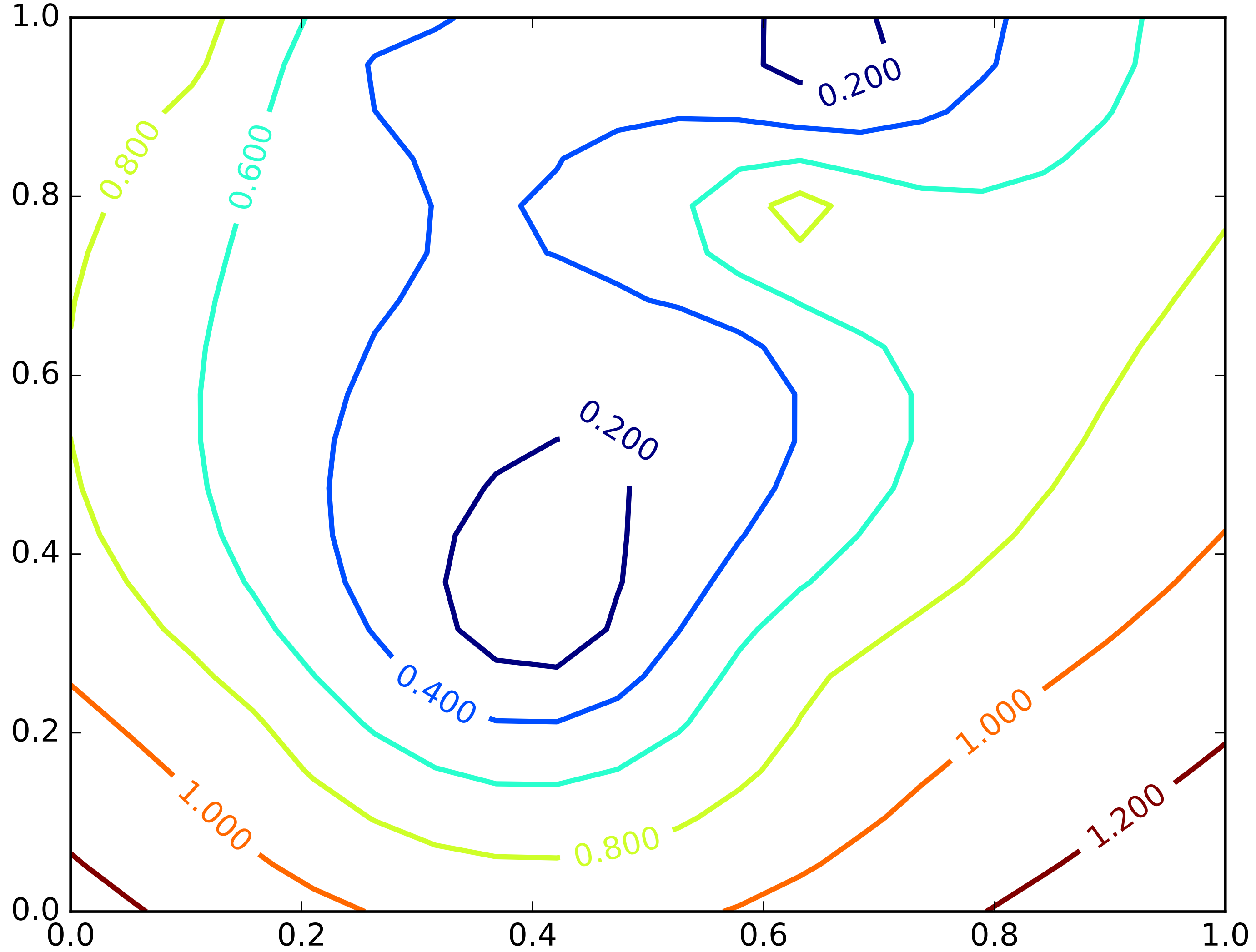
$$f(t\mathbf{x}_1 + (1-t)\mathbf{x}_2) \leq tf(\mathbf{x}_1) + (1-t)f(\mathbf{x}_2)$$

There is only one local optimum if the function is *convex*



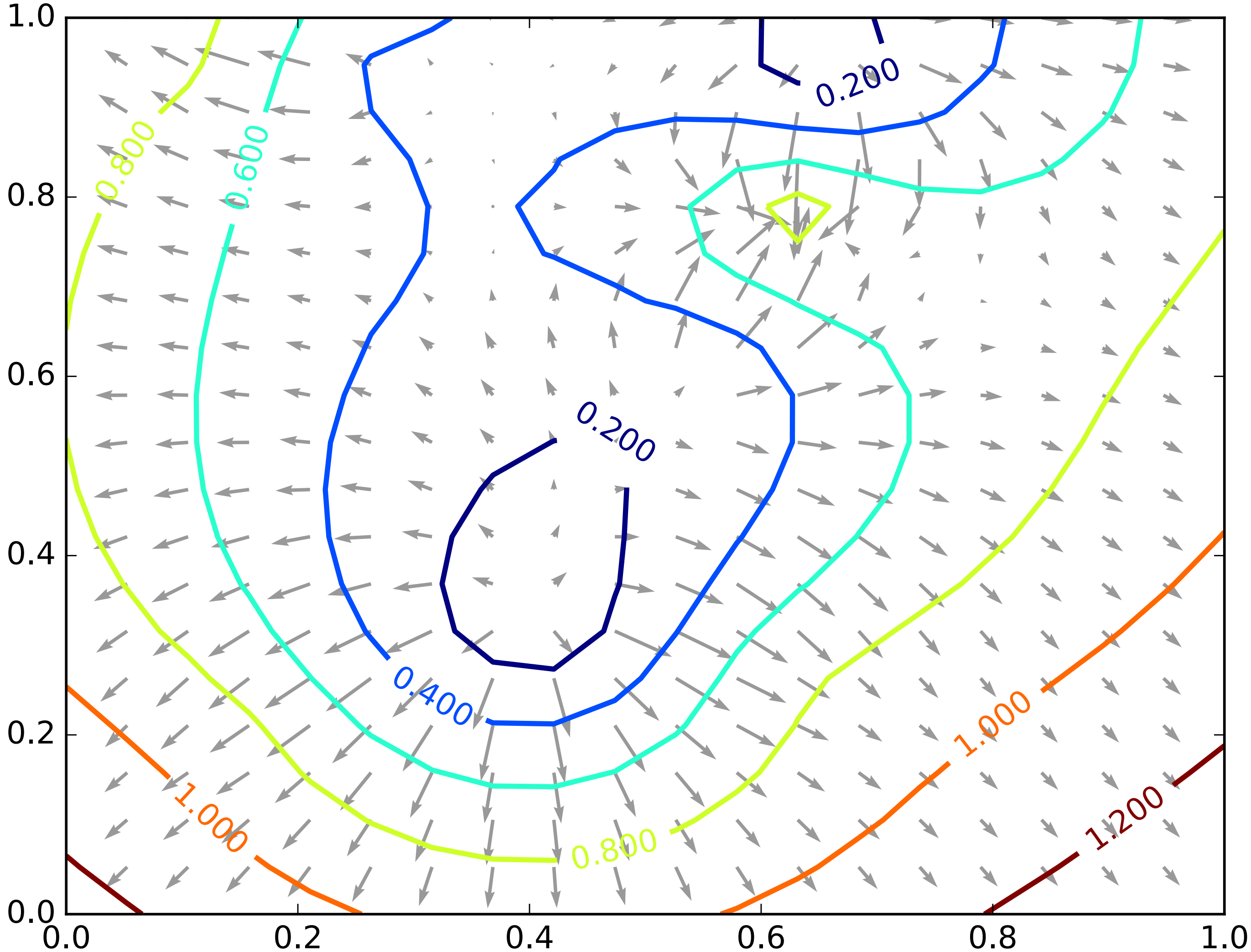
Slide adapted from William Cohen

Gradients



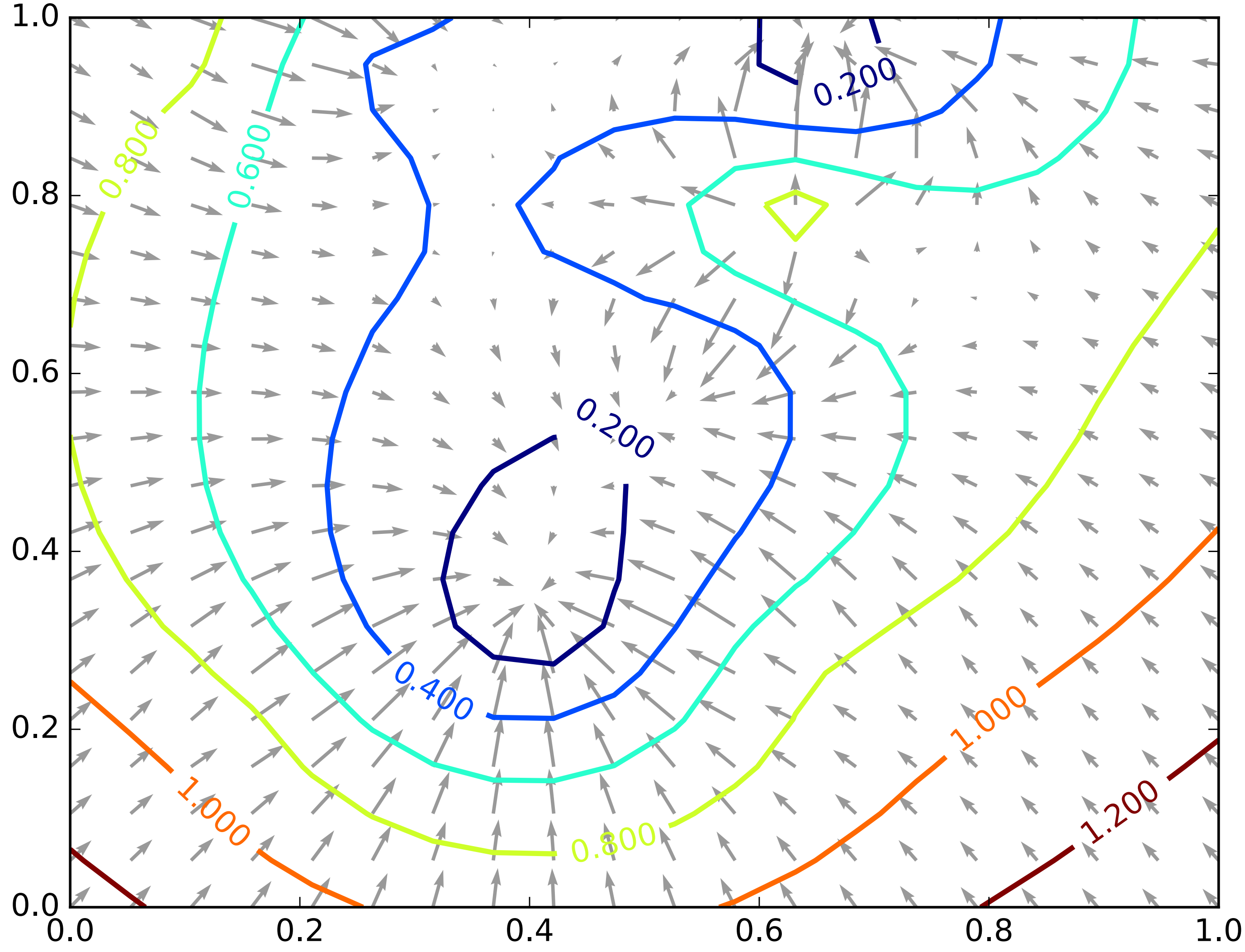
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
0.200
0.200
0.400
0.600
0.800
0.800
1.000
1.000
1.200

Gradients



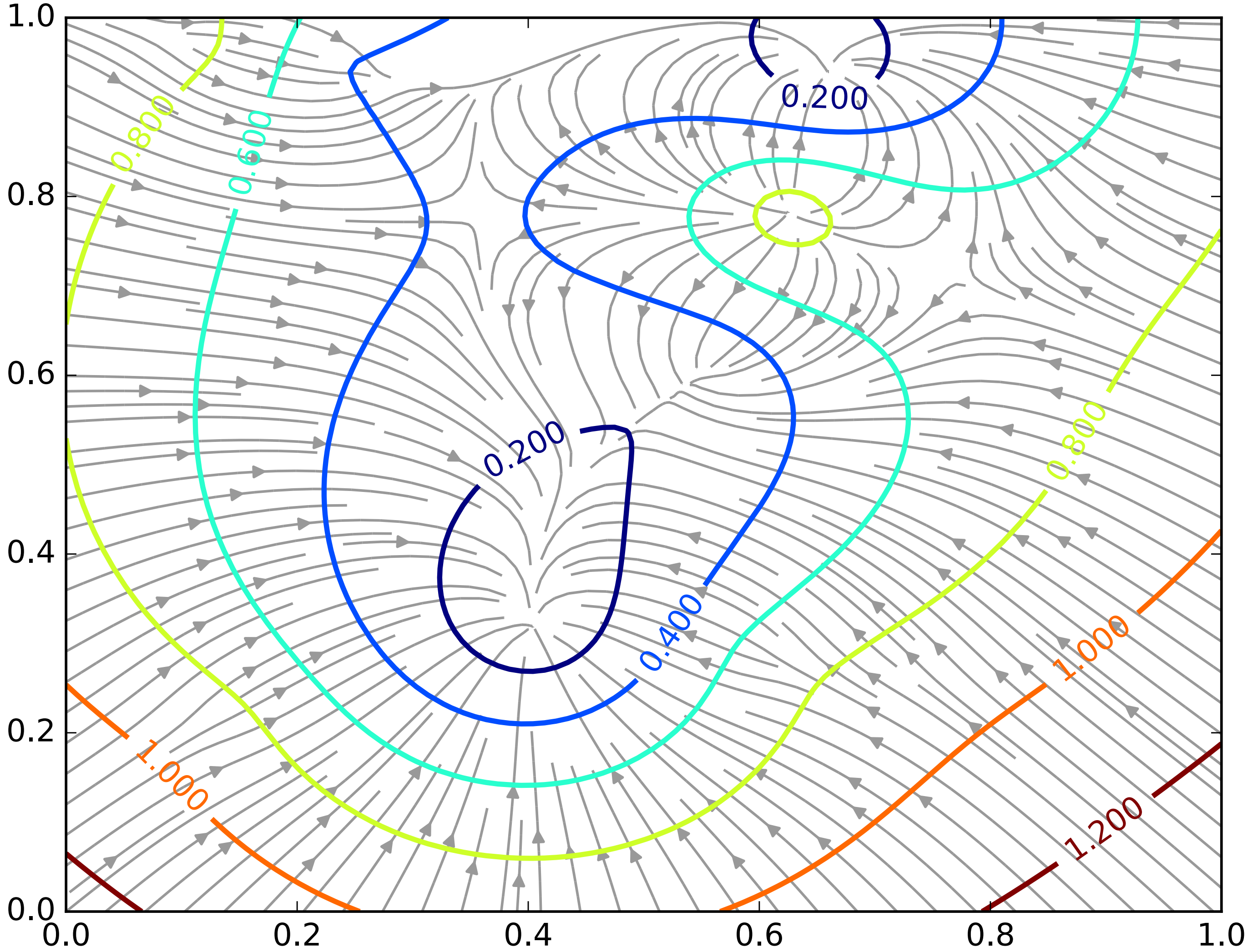
These are the **gradients** that
Gradient **Ascent** would follow.

Negative Gradients



These are the **negative** gradients that Gradient **Descent** would follow.

Negative Gradient Paths

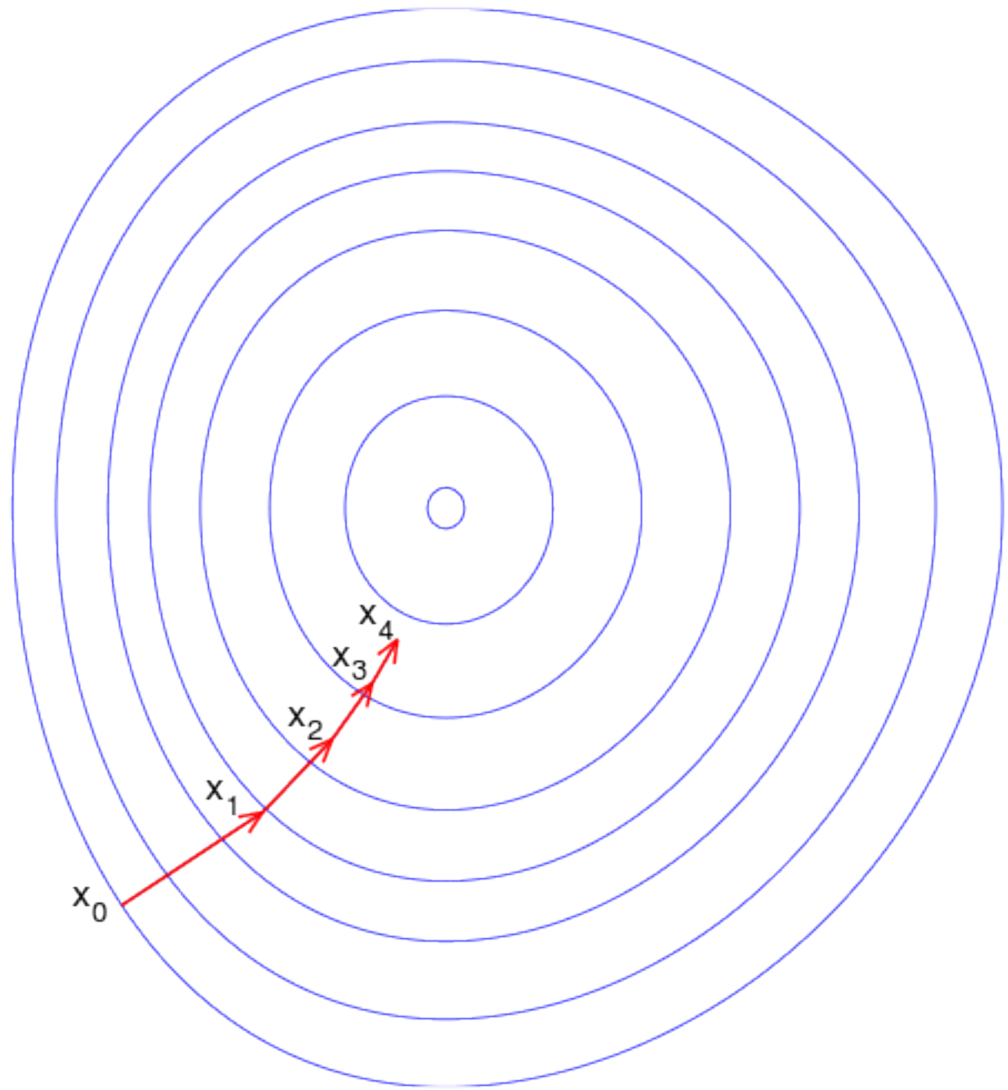


Shown are the paths that Gradient Descent would follow if it were making infinitesimally small steps.

Gradient ascent

To find $\text{argmin}_{\mathbf{x}} f(\mathbf{x})$:

- Start with $\mathbf{x}_0$
- For t=1….
  - $\mathbf{x}_{t+1} = \mathbf{x}_t + \lambda f'(\mathbf{x}_t)$ where λ is small

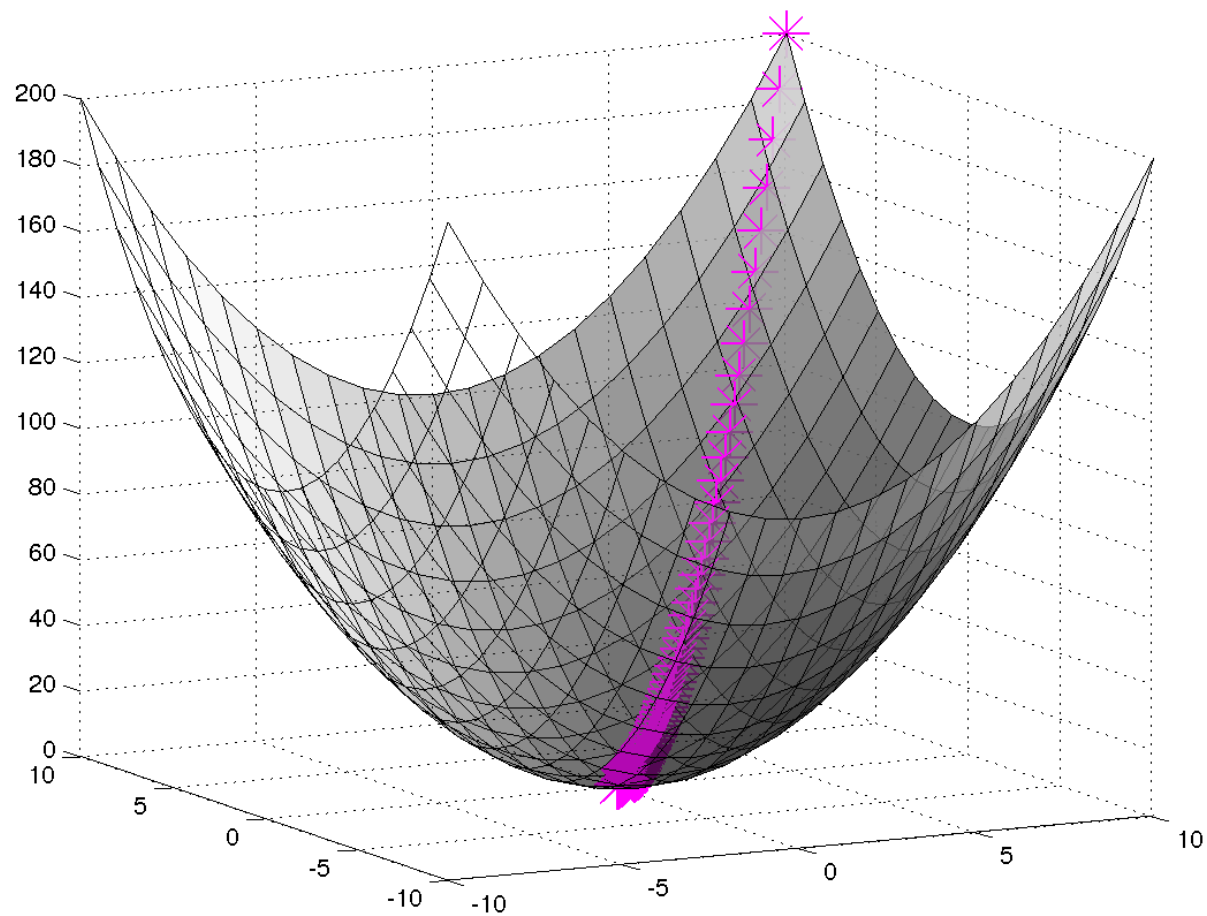


Slide courtesy of William Cohen

Gradient descent

Likelihood: ascent

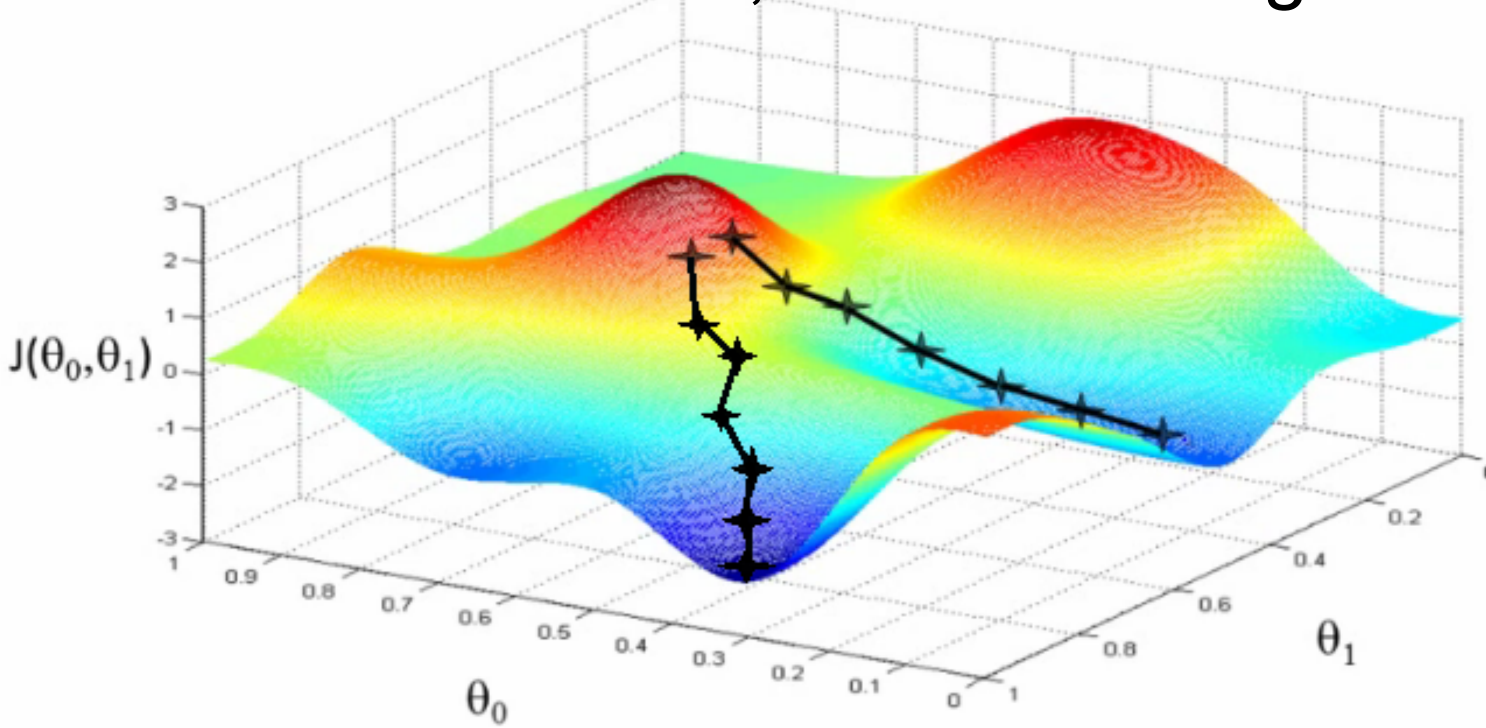
Loss: descent



Slide courtesy of William Cohen

Pros and cons of gradient descent

- Simple and often quite effective on ML tasks
- Often very scalable
- Only applies to smooth functions (differentiable)
- Might find a local minimum, rather than a global one



Slide courtesy of William Cohen

Gradient Descent

Algorithm 1 Gradient Descent

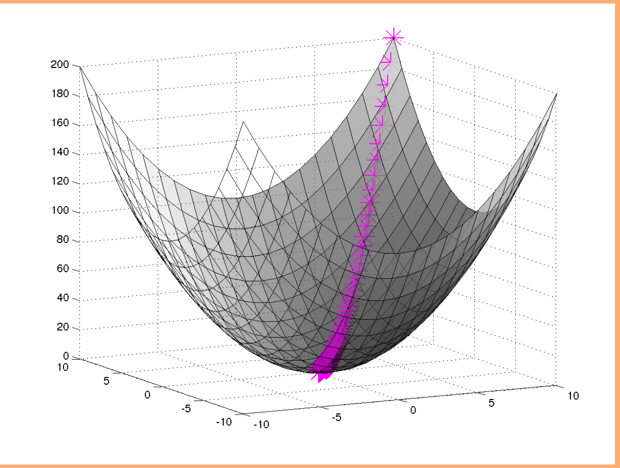
1: **procedure** GD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)

2: $\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$

3: $\quad$**while** not converged **do**

4: $\quad\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

5: $\quad$**return** $\boldsymbol{\theta}$



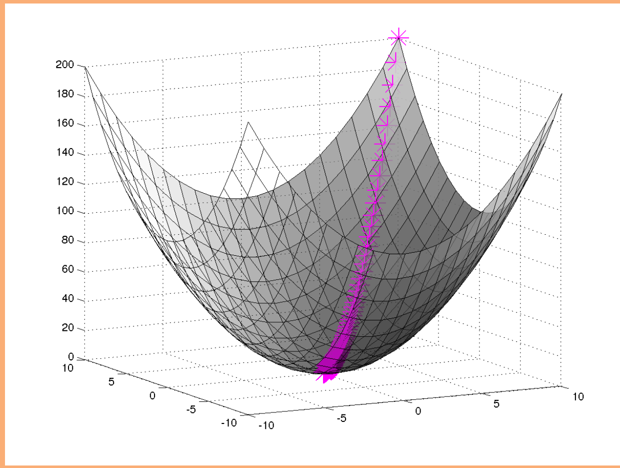
In order to apply GD to Linear Regression all we need is the **gradient** of the objective function (i.e. vector of partial derivatives).

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_N} J(\boldsymbol{\theta}) \end{bmatrix}$$

Gradient Descent

Algorithm 1 Gradient Descent

procedure GD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
 $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
 while not converged **do**
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
 return $\boldsymbol{\theta}$



There are many possible ways to detect **convergence.** For example, we could check whether the L2 norm of the gradient is below some small tolerance.

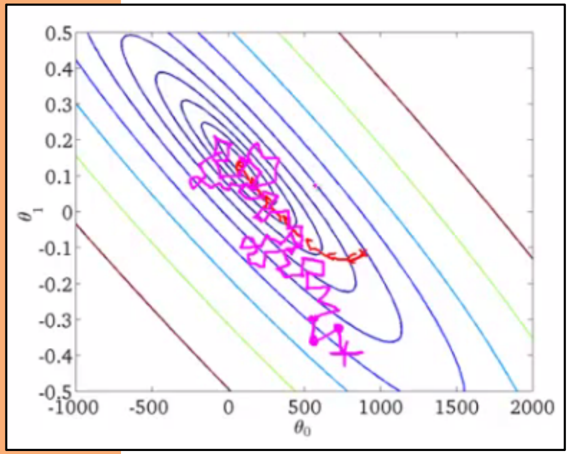
$$||\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})||_2 \leq \epsilon$$

Alternatively we could check that the reduction in the objective function from one iteration to the next is small.

Stochastic Gradient Descent (SGD)

Algorithm 2 Stochastic Gradient Descent (SGD)

1: **procedure** SGD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
2: $\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
3: $\quad$**while** not converged **do**
4: $\quad\quad$**for** $i \in \text{shuffle}(\{1, 2, \ldots, N\})$ **do**
5: $\quad\quad\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$
6: $\quad$**return** $\boldsymbol{\theta}$



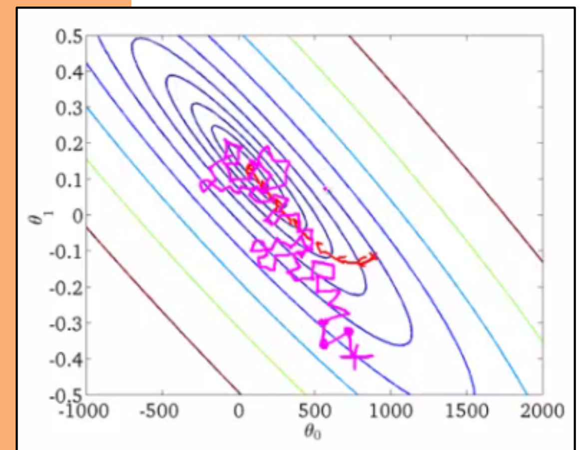
We need a per-example objective:

$$\text{Let } J(\boldsymbol{\theta}) = \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$

Stochastic Gradient Descent (SGD)

Algorithm 2 Stochastic Gradient Descent (SGD)

procedure $\text{SGD}(\mathcal{D}, \boldsymbol{\theta}^{(0)})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
    while not converged **do**
        for $i \in \text{shuffle}(\{1, 2, \ldots, N\})$ **do**
            for $k \in \{1, 2, \ldots, K\}$ **do**
                $\theta_k \leftarrow \theta_k - \lambda \frac{d}{d\theta_k} J^{(i)}(\boldsymbol{\theta})$
    return $\boldsymbol{\theta}$



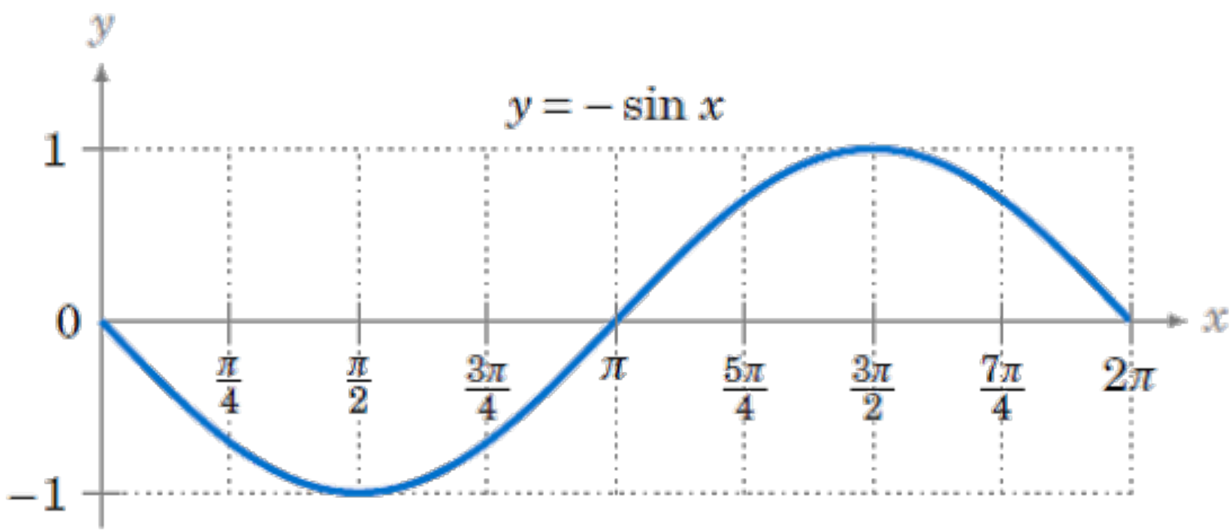
We need a per-example objective:

$$\text{Let } J(\boldsymbol{\theta}) = \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$

FUNCTION APPROXIMATION

Function Approximation

Quiz: Implement a simple function which returns sin(x).



A few constraints are imposed:

1. You can't call any other trigonometric functions
2. You *can* call an existing implementation of sin(x) a few times (e.g. 100) to test your solution
3. You only need to evaluate it for x in [0, 2*pi]

LINEAR REGRESSION

Linear Regression Outline

- **Regression Problems**
  - Definition
  - Linear functions
  - Residuals
  - Notation trick: fold in the intercept
- **Linear Regression as Function Approximation**
  - Objective function: Mean squared error
  - Hypothesis space: Linear Functions
- **Optimization for Linear Regression**
  - Normal Equations (Closed-form solution)
    - Computational complexity
    - Stability
  - SGD for Linear Regression
    - Partial derivatives
    - Update rule
  - Gradient Descent for Linear Regression
- **Probabilistic Interpretation of Linear Regression**
  - Generative vs. Discriminative
  - Conditional Likelihood
  - Background: Gaussian Distribution
  - Case #1: 1D Linear Regression
  - Case #2: Multiple Linear Regression

Regression Problems

Whiteboard

- Definition
- Linear functions
- Residuals
- Notation trick: fold in the intercept

Linear Regression as Function Approximation

Whiteboard

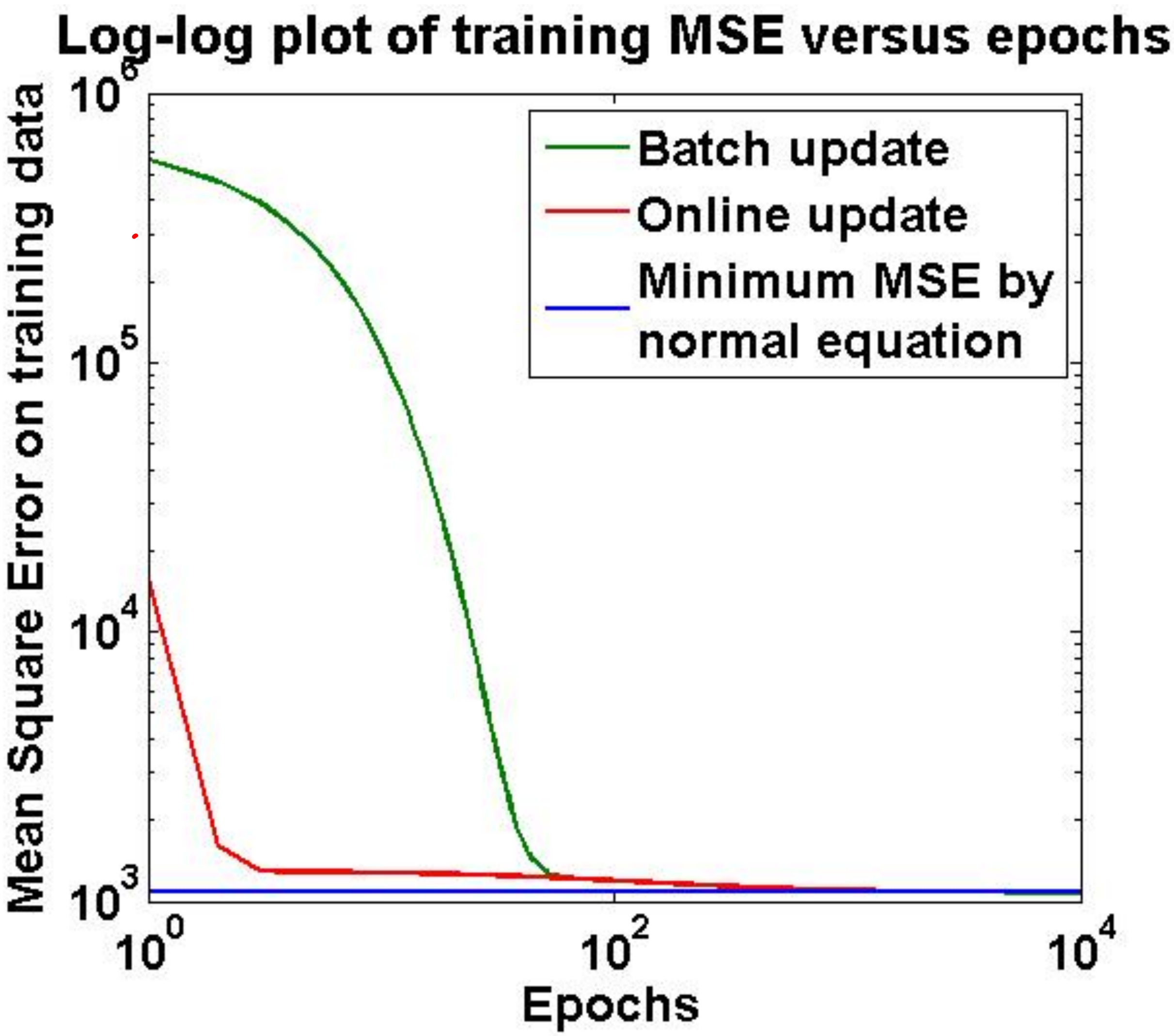
- Objective function: Mean squared error
- Hypothesis space: Linear Functions

Optimization for Linear Regression

Whiteboard

- Normal Equations (Closed-form solution)
  - Computational complexity
  - Stability
- SGD for Linear Regression
  - Partial derivatives
  - Update rule
- Gradient Descent for Linear Regression

Convergence Curves



- For the batch method, the training MSE is initially large due to uninformed initialization
- In the online update, N updates for every epoch reduces MSE to a much smaller value.

Summary

- Linear regression **predicts** its output as a **linear function** of its inputs
- Learning **optimizes** a function (equivalently likelihood or mean squared error) using **standard techniques** (gradient descent, SGD, closed form)

LOGISTIC REGRESSION

Logistic Regression Outline

- **Motivation:**
  - Choosing the right classifier
  - Example: Image Classification
- **Logistic Regression**
  - Background: Hyperplanes
  - Data, Model, Learning, Prediction
  - Log-odds
  - Bernoulli interpretation
  - Maximum Conditional Likelihood Estimation
- **Gradient descent for Logistic Regression**
  - Stochastic Gradient Descent (SGD)
  - Computing the gradient
  - Details (learning rate, finite differences)
- **Nonlinear Features**

MOTIVATION:
LOGISTIC REGRESSION

Classifiers

Which classification method should we use?

1. The one that gives the best predictions…
   - on the training data
   - on the (unseen) test data
   - on the (held-out) validation data
2. The one that is computationally efficient…
   - during training
   - during classification
3. The most interpretable one…
   - in terms of its parameters
   - as a model
4. The one that is easiest to implement…
   - for learning
   - for classification

Classifiers

Which classification method should we use?

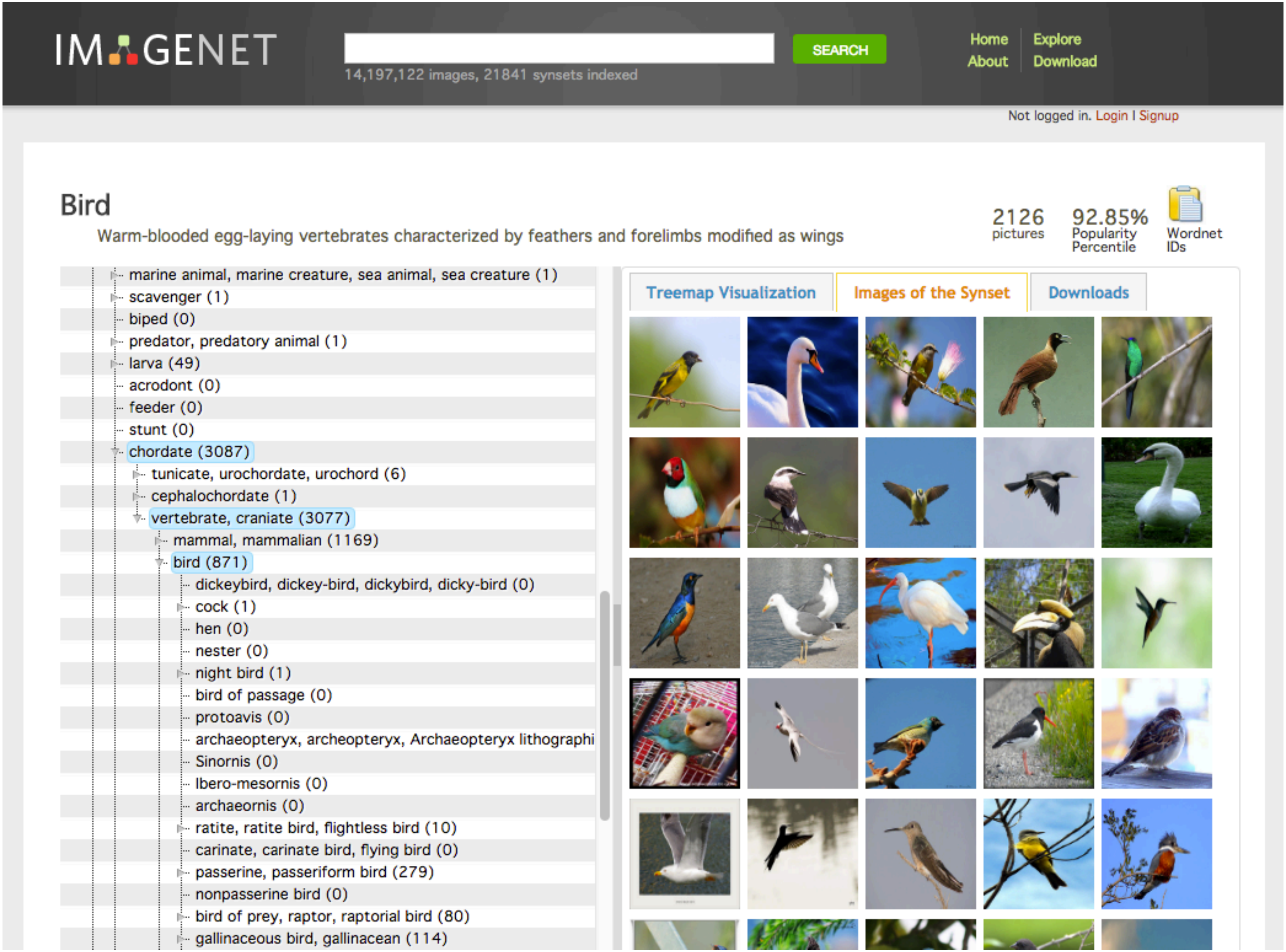
Naïve Bayes defined a generative model $p(\mathbf{x}, y)$ of the features $\mathbf{x}$ and the class y.

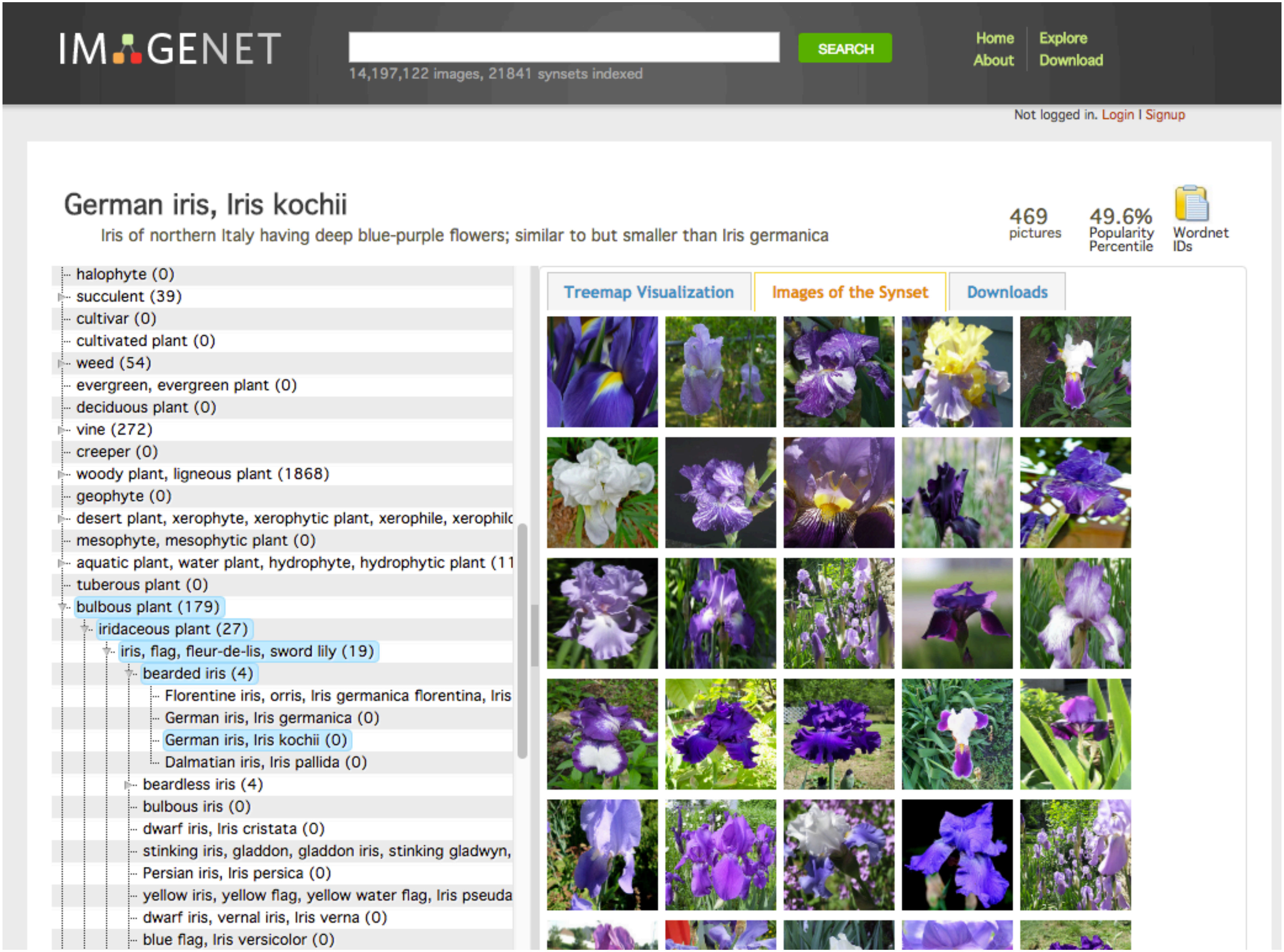
Why should we define a model of $p(\mathbf{x}, y)$ at all?

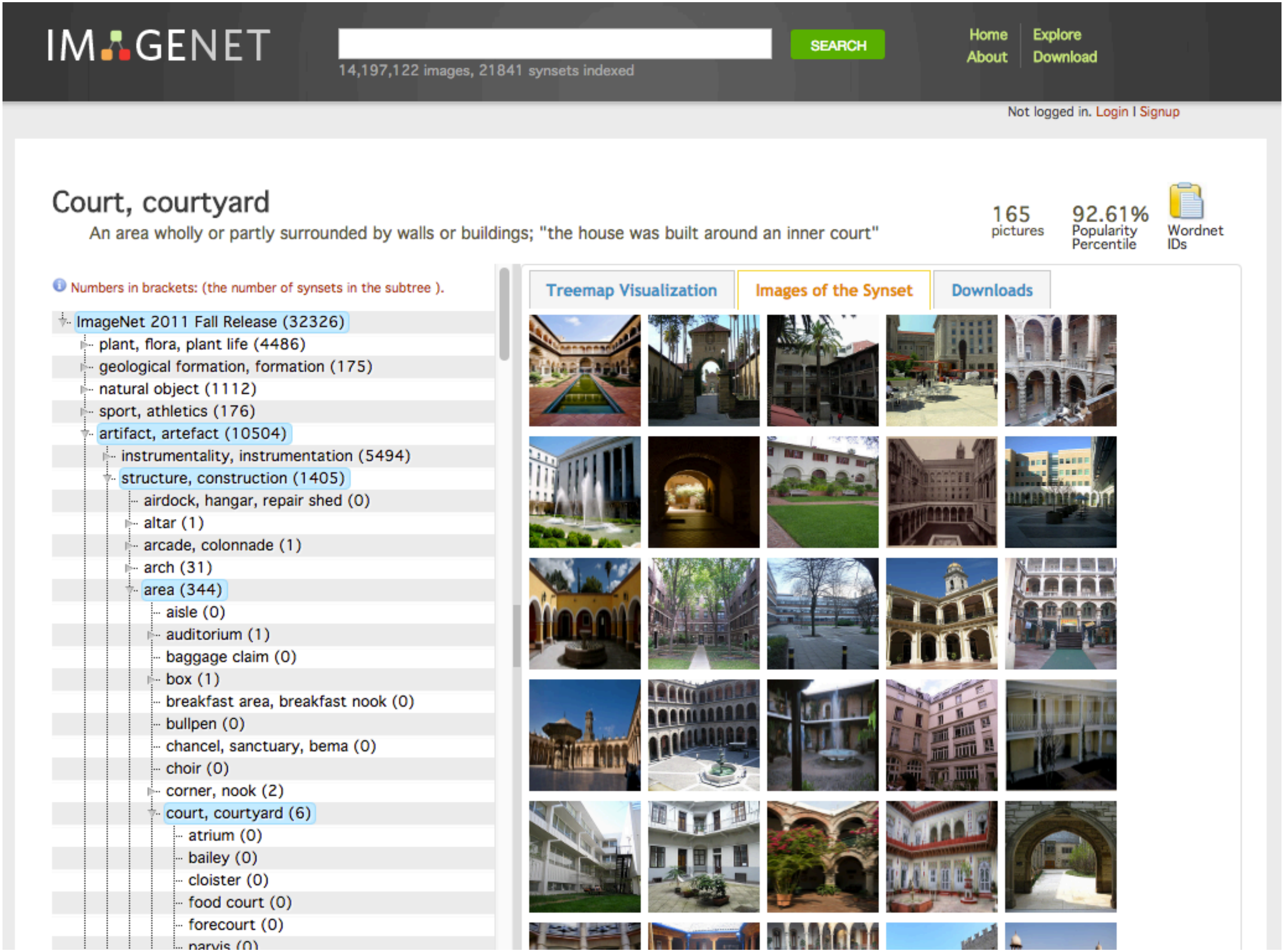
Why not directly model $p(y \mid \mathbf{x})$?

Example: Image Classification

- ImageNet LSVRC-2010 contest:
  - **Dataset:** 1.2 million labeled images, 1000 classes
  - **Task**: Given a new image, label it with the correct class
  - **Multiclass** classification problem
- Examples from http://image-net.org/


IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
About
Explore
Download
Not logged in. Login I Signup
Bird
Warm-blooded egg-laying vertebrates characterized by feathers and forelimbs modified as wings
2126 pictures
92.85% Popularity Percentile
Wordnet IDs
marine animal, marine creature, sea animal, sea creature (1)
scavenger (1)
biped (0)
predator, predatory animal (1)
larva (49)
acrodont (0)
feeder (0)
stunt (0)
chordate (3087)
tunicate, urochordate, urochord (6)
cephalochordate (1)
vertebrate, craniate (3077)
mammal, mammalian (1169)
bird (871)
dickeybird, dickey-bird, dickybird, dicky-bird (0)
cock (1)
hen (0)
nester (0)
night bird (1)
bird of passage (0)
protoavis (0)
archaeopteryx, archeopteryx, Archaeopteryx lithographi
Sinornis (0)
Ibero-mesornis (0)
archaeornis (0)
ratite, ratite bird, flightless bird (10)
carinate, carinate bird, flying bird (0)
passerine, passeriform bird (279)
nonpasserine bird (0)
bird of prey, raptor, raptorial bird (80)
gallinaceous bird, gallinacean (114)
Treemap Visualization
Images of the Synset
Downloads


IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
About
Explore
Download
Not logged in. Login I Signup
German iris, Iris kochii
Iris of northern Italy having deep blue-purple flowers; similar to but smaller than Iris germanica
469 pictures
49.6% Popularity Percentile
Wordnet IDs
halophyte (0)
succulent (39)
cultivar (0)
cultivated plant (0)
weed (54)
evergreen, evergreen plant (0)
deciduous plant (0)
vine (272)
creeper (0)
woody plant, ligneous plant (1868)
geophyte (0)
desert plant, xerophyte, xerophytic plant, xerophile, xerophilc
mesophyte, mesophytic plant (0)
aquatic plant, water plant, hydrophyte, hydrophytic plant (11
tuberous plant (0)
bulbous plant (179)
iridaceous plant (27)
iris, flag, fleur-de-lis, sword lily (19)
bearded iris (4)
Florentine iris, orris, Iris germanica florentina, Iris
German iris, Iris germanica (0)
German iris, Iris kochii (0)
Dalmatian iris, Iris pallida (0)
beardless iris (4)
bulbous iris (0)
dwarf iris, Iris cristata (0)
stinking iris, gladdon, gladdon iris, stinking gladwyn,
Persian iris, Iris persica (0)
yellow iris, yellow flag, yellow water flag, Iris pseuda
dwarf iris, vernal iris, Iris verna (0)
blue flag, Iris versicolor (0)
Treemap Visualization
Images of the Synset
Downloads


IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
About
Explore
Download
Not logged in. Login I Signup
Court, courtyard
An area wholly or partly surrounded by walls or buildings; "the house was built around an inner court"
165 pictures
92.61% Popularity Percentile
Wordnet IDs
Numbers in brackets: (the number of synsets in the subtree).
ImageNet 2011 Fall Release (32326)
plant, flora, plant life (4486)
geological formation, formation (175)
natural object (1112)
sport, athletics (176)
artifact, artefact (10504)
instrumentality, instrumentation (5494)
structure, construction (1405)
airdock, hangar, repair shed (0)
altar (1)
arcade, colonnade (1)
arch (31)
area (344)
aisle (0)
auditorium (1)
baggage claim (0)
box (1)
breakfast area, breakfast nook (0)
bullpen (0)
chancel, sanctuary, bema (0)
choir (0)
corner, nook (2)
court, courtyard (6)
atrium (0)
bailey (0)
cloister (0)
food court (0)
forecourt (0)
parvis (0)
Treemap Visualization
Images of the Synset
Downloads

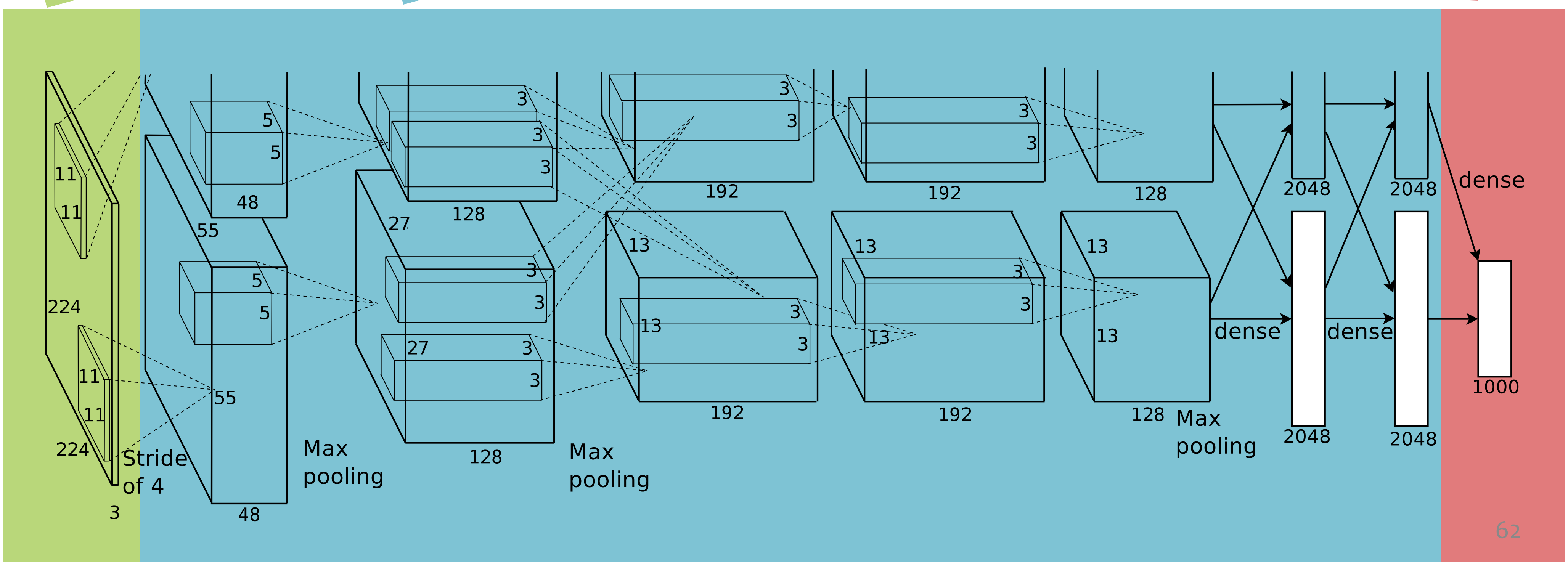
Example: Image Classification

CNN for Image Classification
(Krizhevsky, Sutskever & Hinton, 2011)
17.5% error on ImageNet LSVRC-2010 contest

Input image (pixels)

- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax



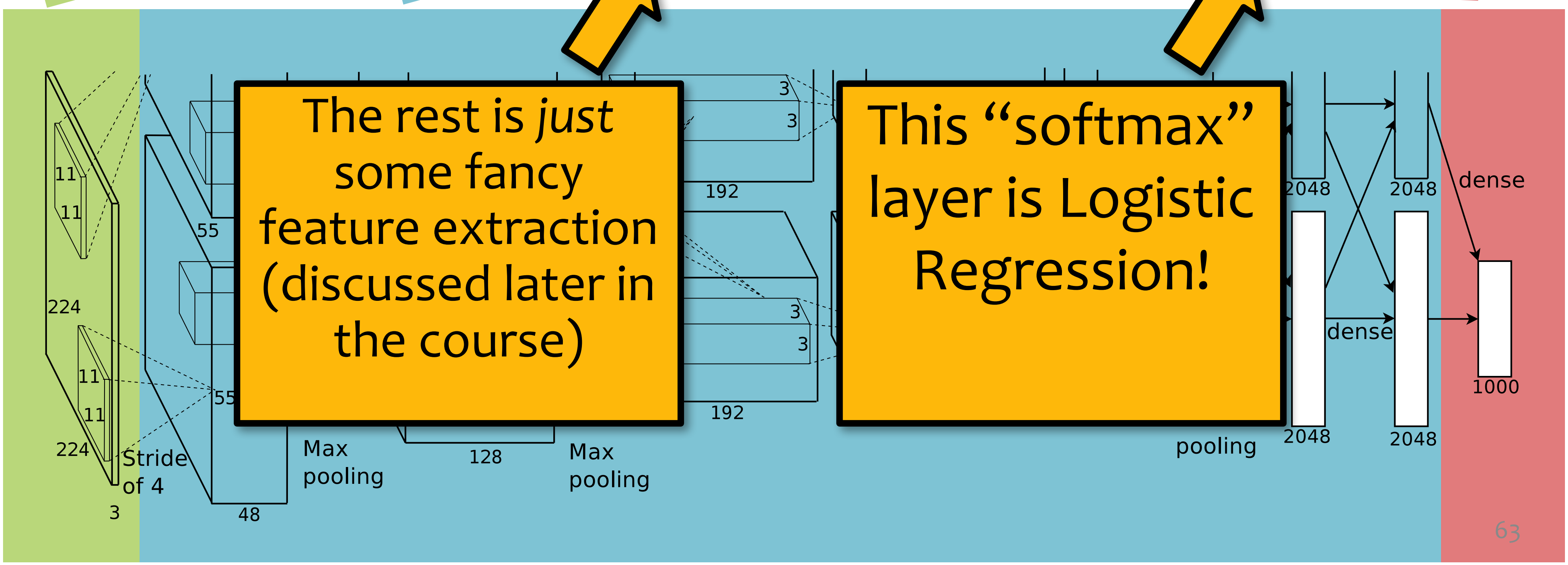
Example: Image Classification

CNN for Image Classification
(Krizhevsky, Sutskever & Hinton, 2011)
17.5% error on ImageNet LSVRC-2010 contest

Input image (pixels)

- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax



The rest is *just* some fancy feature extraction (discussed later in the course)

This "softmax" layer is Logistic Regression!

LOGISTIC REGRESSION

Logistic Regression

Data: Inputs are continuous vectors of length K. Outputs are discrete.

$$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N} \text{ where } \mathbf{x} \in \mathbb{R}^M \text{ and } y \in \{0, 1\}$$

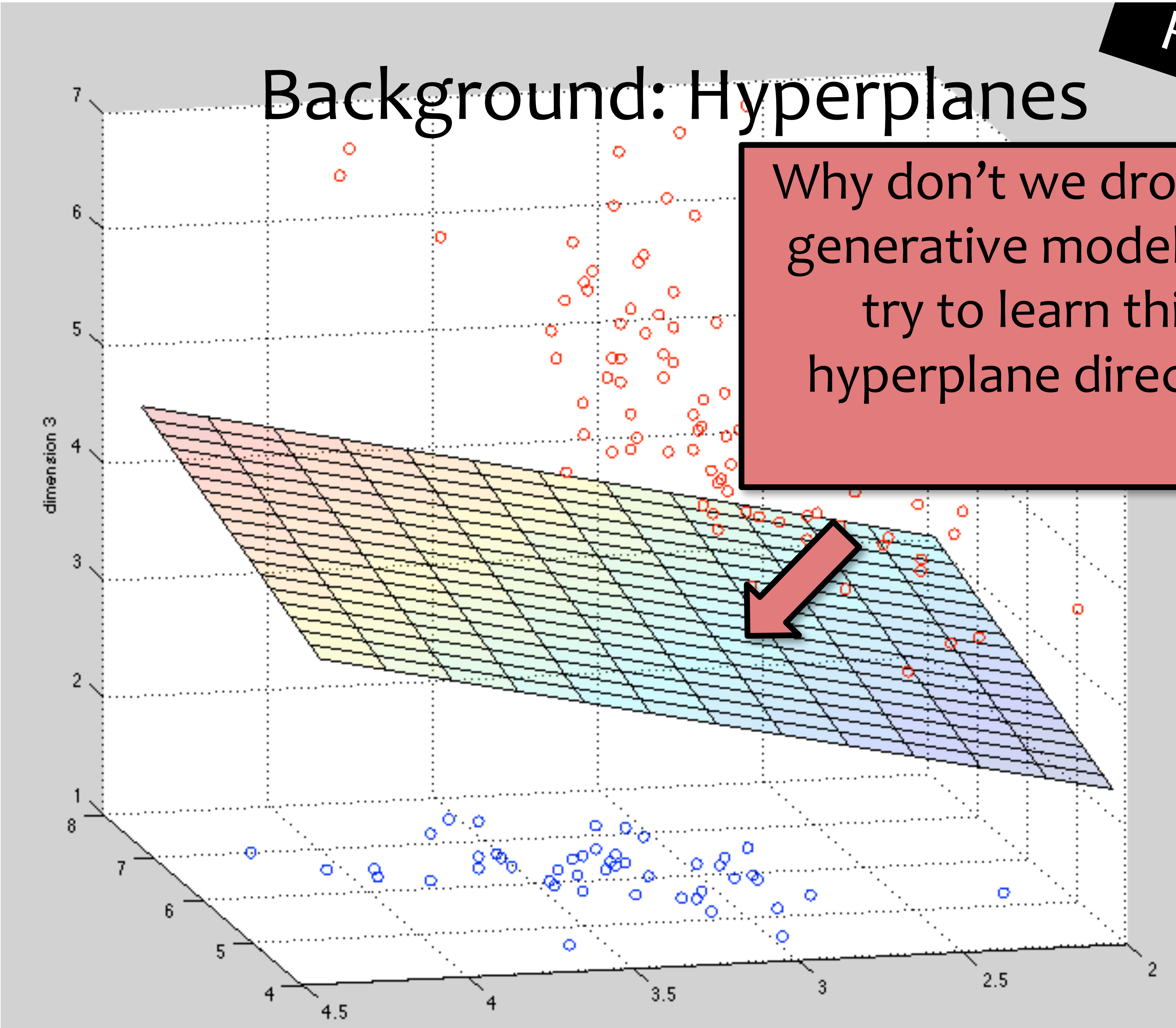
We are back to classification.

Despite the name logistic **regression.**

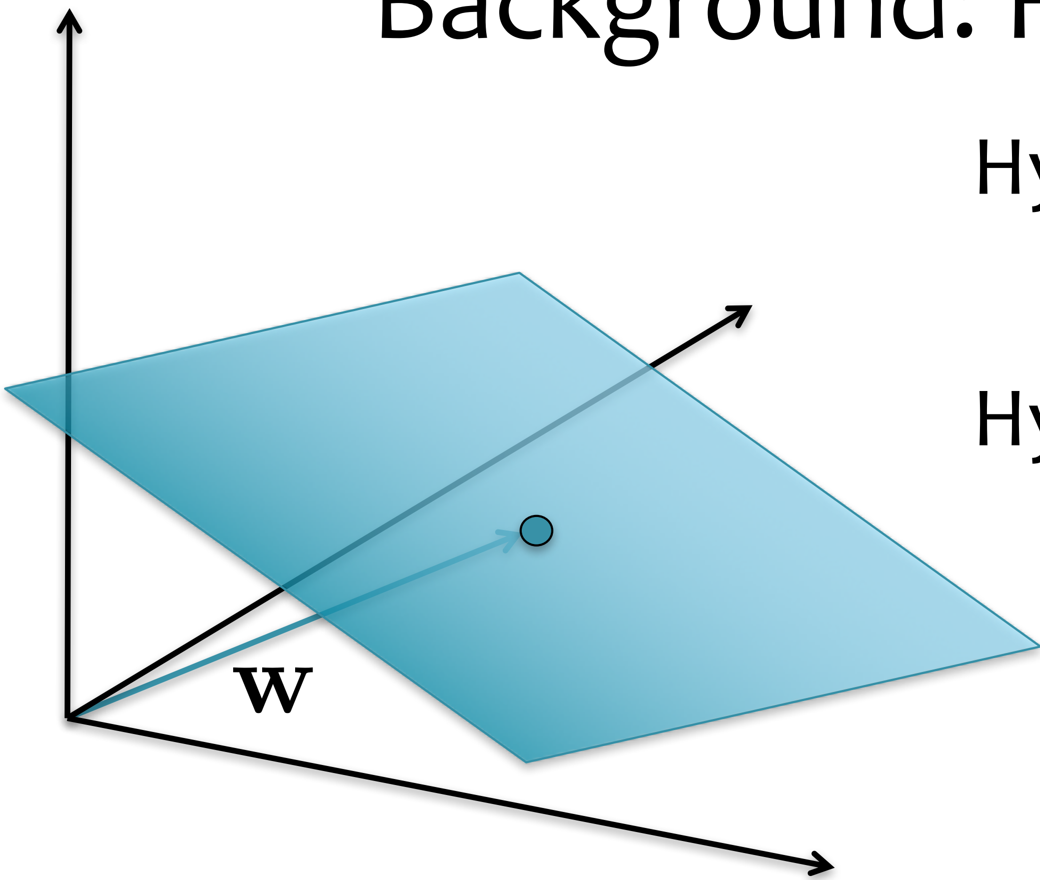
Background: Hyperplanes

Recall…

Why don't we drop the generative model and try to learn this hyperplane directly?



Background: Hyperplanes



Hyperplane (Definition 1):

$$\mathcal{H} = \{\mathbf{x} : \mathbf{w}^T \mathbf{x} = b\}$$

Hyperplane (Definition 2):

$$\mathcal{H} = \{\mathbf{x} : \mathbf{w}^T \mathbf{x} = 0 \text{ and } \mathrm{x}_0 = 1\}$$

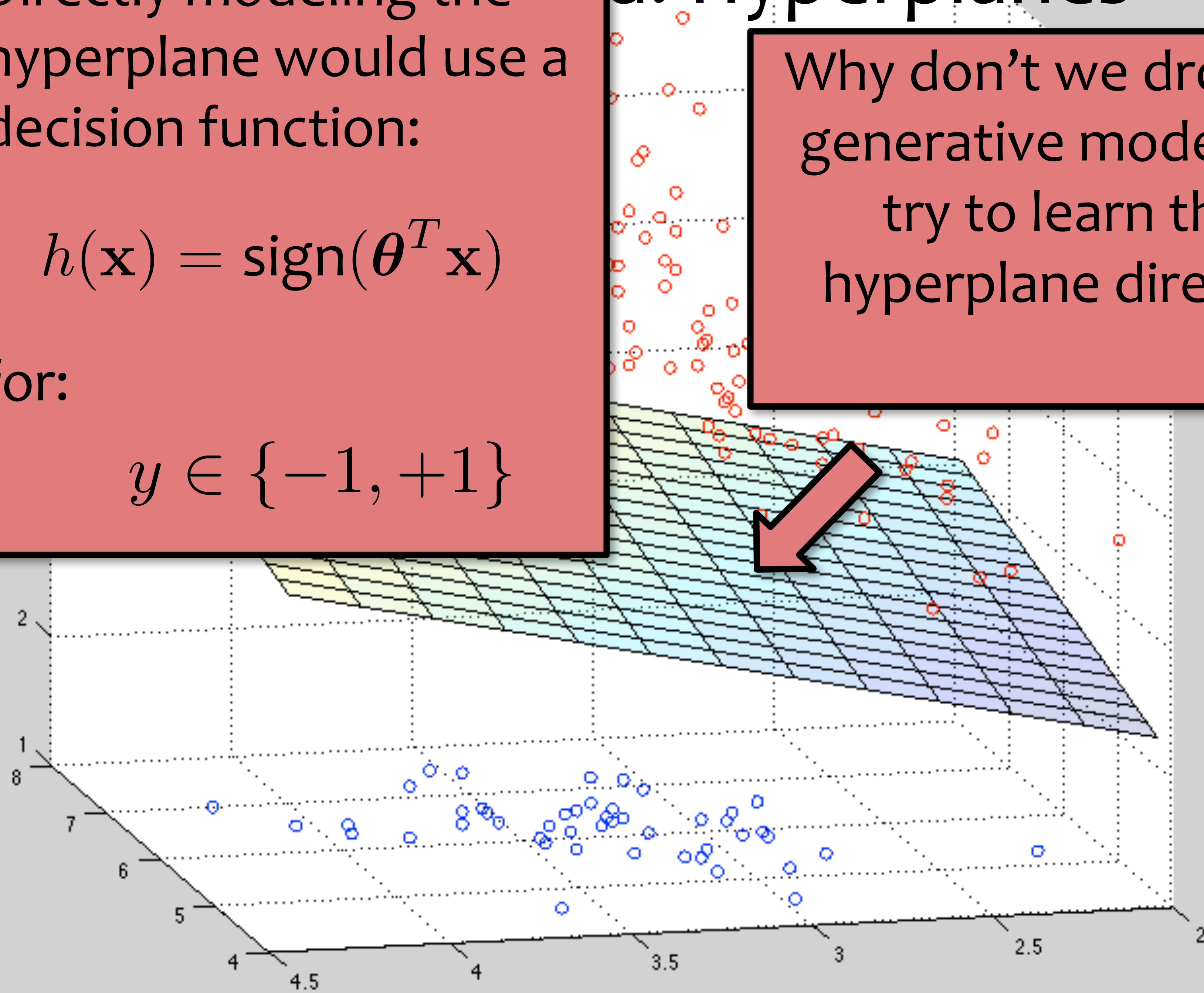
Half-spaces:

$$\mathcal{H}^+ = \{\mathbf{x} : \mathbf{w}^T \mathbf{x} > 0 \text{ and } \mathrm{x}_0 = 1\}$$

$$\mathcal{H}^- = \{\mathbf{x} : \mathbf{w}^T \mathbf{x} < 0 \text{ and } \mathrm{x}_0 = 1\}$$

Recall...

d: Hyperplanes



Directly modeling the hyperplane would use a decision function:

$$h(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$

for:

$$y \in \{-1, +1\}$$

Why don't we drop the generative model and try to learn this hyperplane directly?

Using gradient ascent for linear classifiers

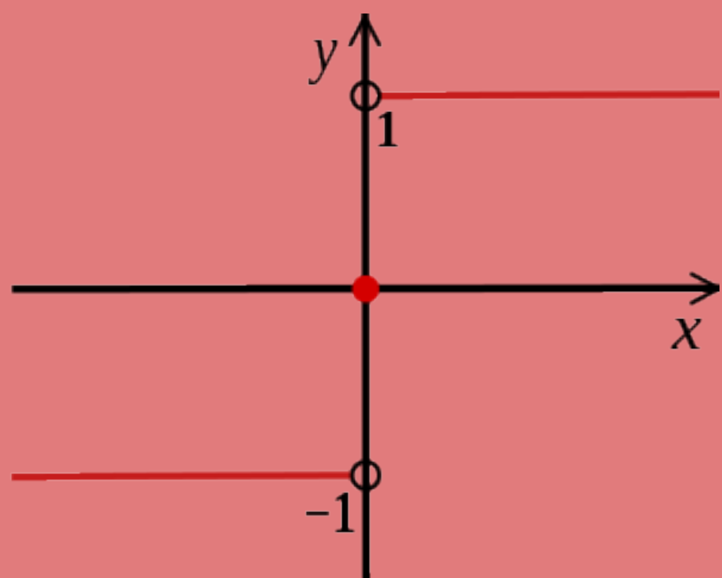
Key idea behind today's lecture:

1. Define a linear classifier (logistic regression)
2. Define an objective function (likelihood)
3. Optimize it with gradient descent to learn parameters
4. Predict the class with highest probability under the model

Using gradient ascent for linear classifiers

This decision function isn't differentiable:

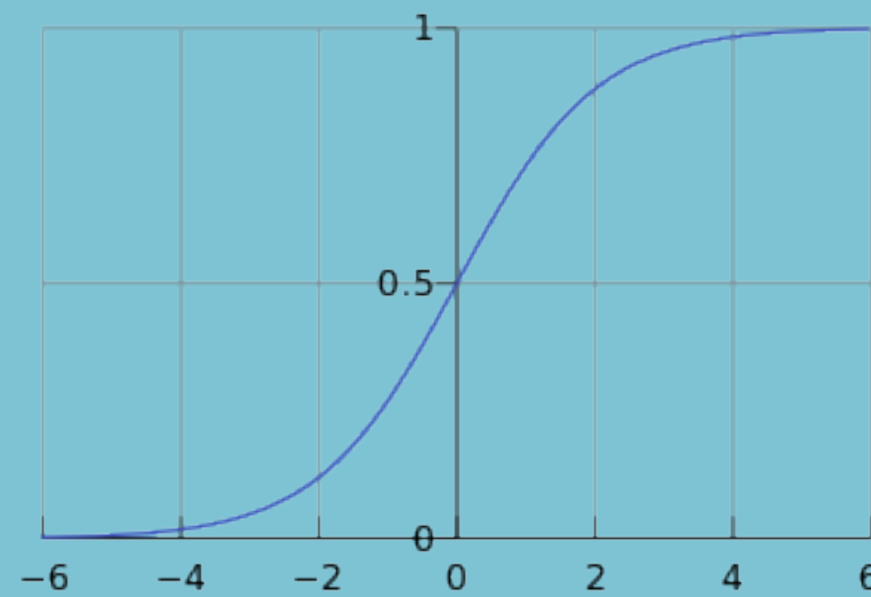
$$h(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$



sign(x)

Use a differentiable function instead:

$$p_{\boldsymbol{\theta}}(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{x})}$$

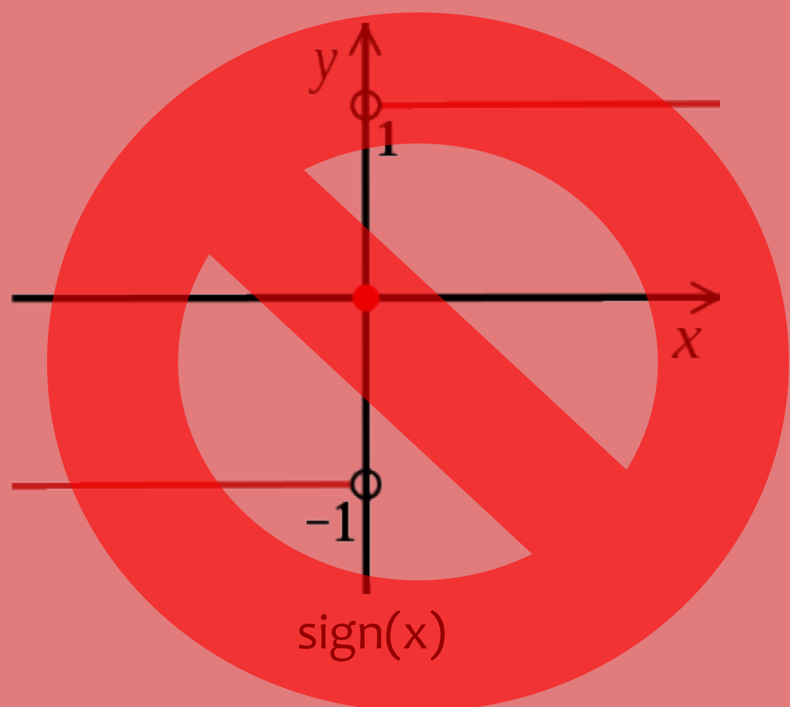


$$\text{logistic}(u) \equiv \frac{1}{1 + e^{-u}}$$

Using gradient ascent for linear classifiers

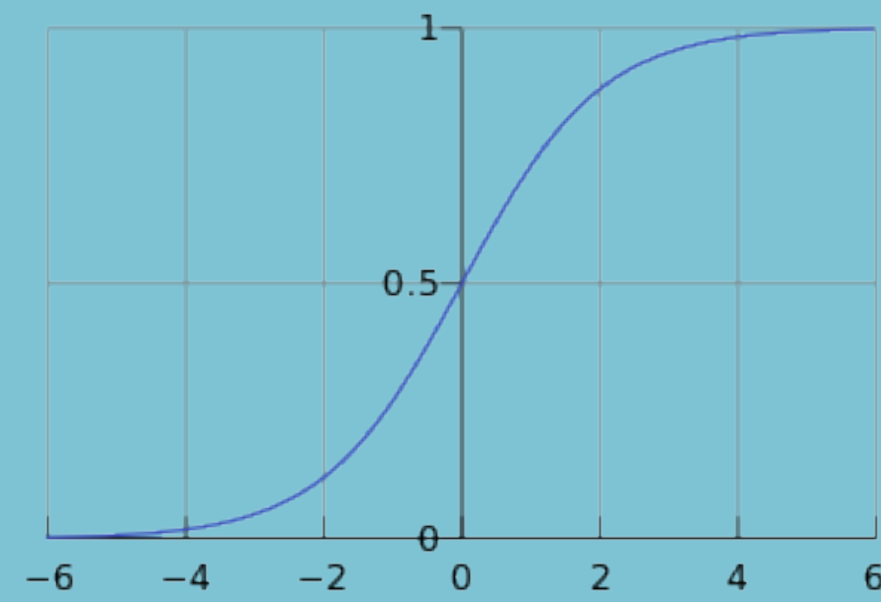
This decision function isn't differentiable:

$$h(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$



Use a differentiable function instead:

$$p_{\boldsymbol{\theta}}(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{x})}$$



$$\text{logistic}(u) \equiv \frac{1}{1 + e^{-u}}$$

Logistic Regression

Data: Inputs are continuous vectors of length K. Outputs are discrete.

$$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N \text{ where } \mathbf{x} \in \mathbb{R}^M \text{ and } y \in \{0, 1\}$$

Model: Logistic function applied to dot product of parameters with input vector.

$$p_{\boldsymbol{\theta}}(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{x})}$$

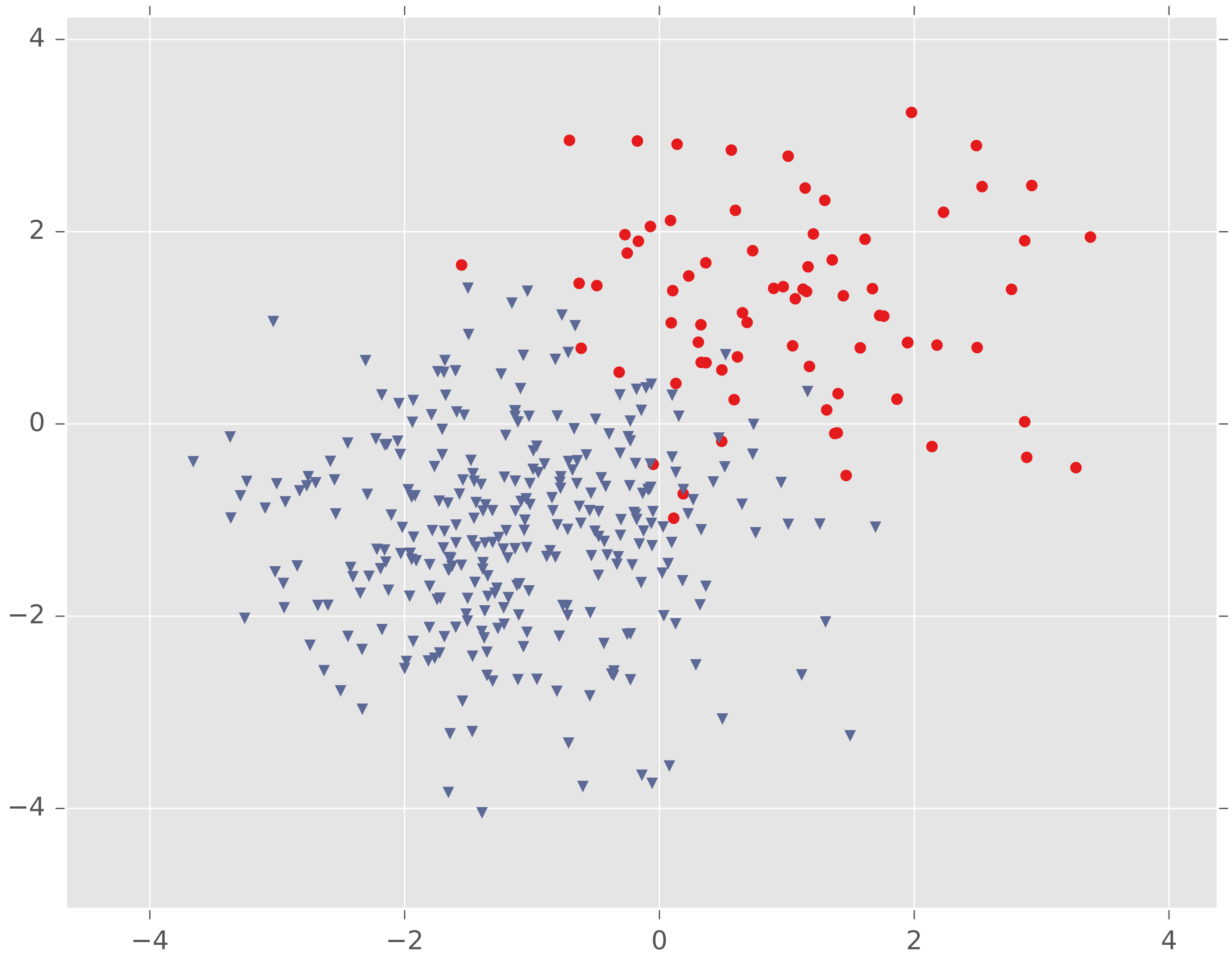
Learning: finds the parameters that minimize some objective function.

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \, J(\boldsymbol{\theta})$$

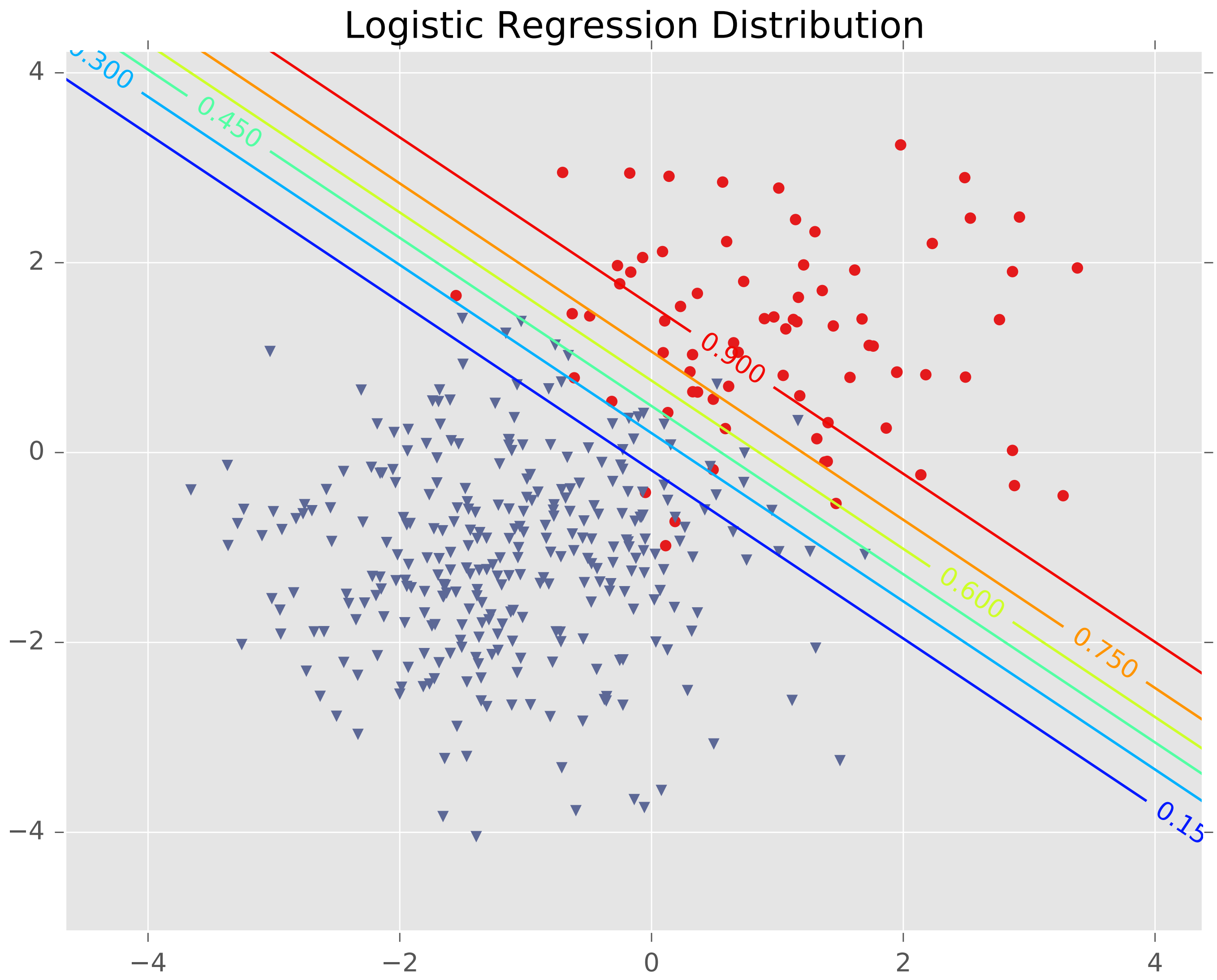
Prediction: Output is the most probable class.

$$\hat{y} = \underset{y \in \{0,1\}}{\operatorname{argmax}} \, p_{\boldsymbol{\theta}}(y|\mathbf{x})$$

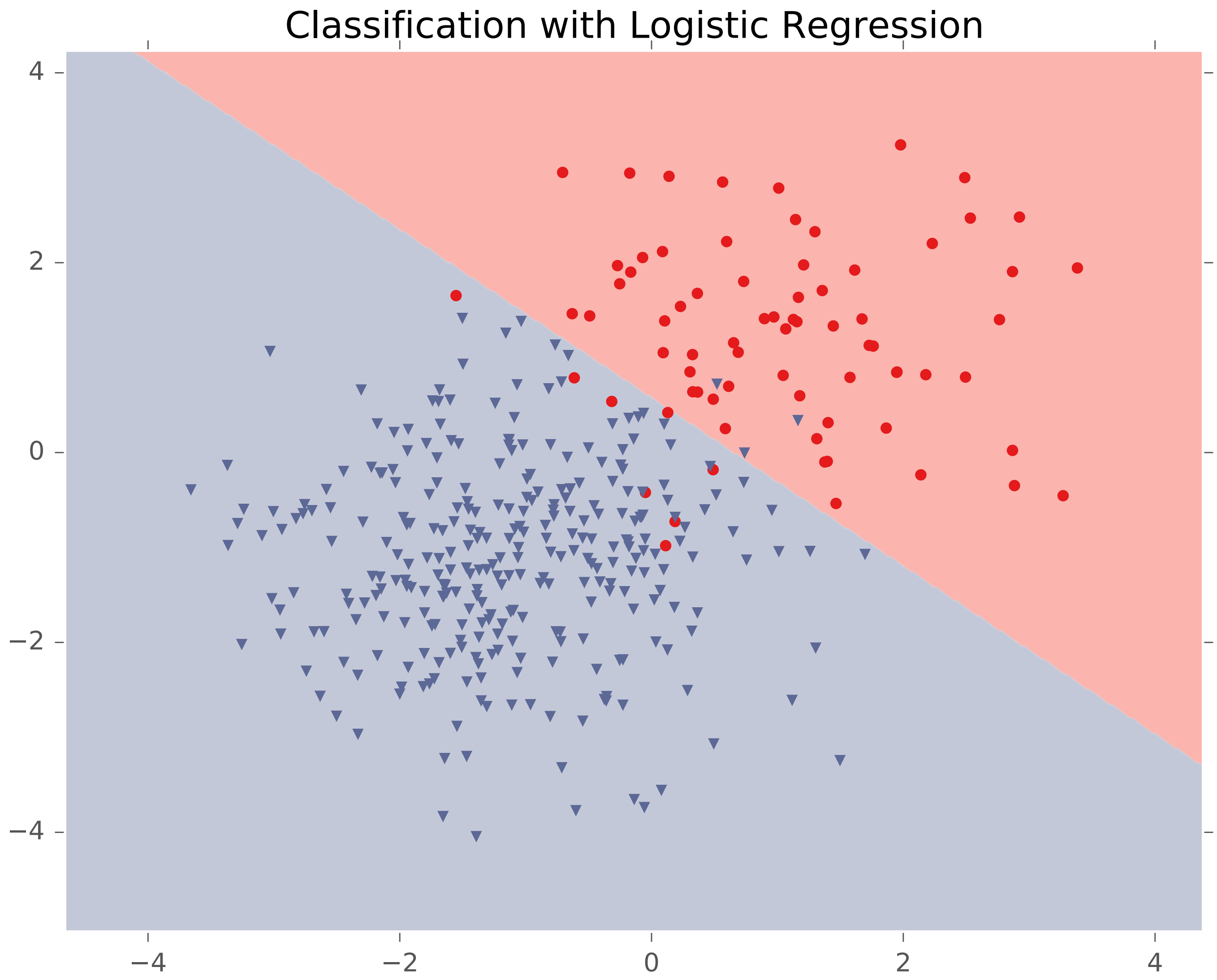
Logistic Regression



Logistic Regression



Logistic Regression



LEARNING LOGISTIC REGRESSION

Maximum **Conditional** Likelihood Estimation

Learning: finds the parameters that minimize some objective function.

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

We minimize the *negative* log conditional likelihood:

$$J(\boldsymbol{\theta}) = -\log \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(y^{(i)}|\mathbf{x}^{(i)})$$

Why?

1. We can't maximize likelihood (as in Naïve Bayes) because we don't have a joint model p(x,y)
2. It worked well for Linear Regression (least squares is MCLE)

Maximum **Conditional** Likelihood Estimation

Learning: Four approaches to solving $\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

Approach 1: Gradient Descent
(take larger – more certain – steps opposite the gradient)

Approach 2: Stochastic Gradient Descent (SGD)
(take many small steps opposite the gradient)

Approach 3: Newton's Method
(use second derivatives to better follow curvature)

Approach 4: Closed Form???
(set derivatives equal to zero and solve for parameters)

Maximum **Conditional** Likelihood Estimation

Learning: Four approaches to solving $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}}\, J(\boldsymbol{\theta})$

Approach 1: Gradient Descent
(take larger – more certain – steps opposite the gradient)

Approach 2: Stochastic Gradient Descent (SGD)
(take many small steps opposite the gradient)

Approach 3: Newton's Method
(use second derivatives to better follow curvature)

~~**Approach 4:** Closed Form???~~
~~(set derivatives equal to zero and solve for parameters)~~

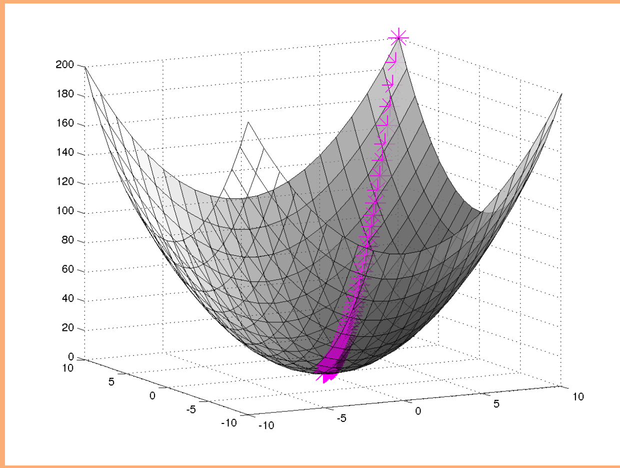
Logistic Regression does not have a closed form solution for MLE parameters.

Gradient Descent

Recall…

Algorithm 1 Gradient Descent

1: **procedure** GD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
2: $\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
3: $\quad$**while** not converged **do**
4: $\quad\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
5: $\quad$**return** $\boldsymbol{\theta}$



In order to apply GD to Logistic Regression all we need is the **gradient** of the objective function (i.e. vector of partial derivatives).

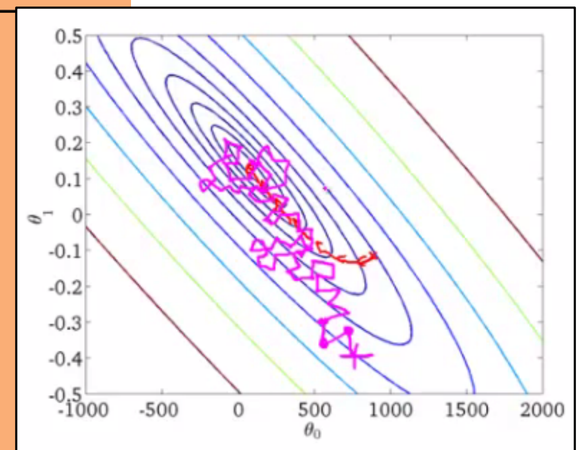
$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J(\boldsymbol{\theta}) \end{bmatrix}$$

Stochastic Gradient Descent (SGD)

Recall...

Algorithm 1 Stochastic Gradient Descent (SGD)

```
procedure SGD(𝒟, θ^(0))
    θ ← θ^(0)
    while not converged do
        for i ∈ shuffle({1, 2, ..., N}) do
            θ ← θ − λ∇_θ J^(i)(θ)
    return θ
```



We can also apply SGD to solve the MCLE problem for Logistic Regression.

We need a per-example objective:

$$\text{Let } J(\boldsymbol{\theta}) = \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$
$$\text{where } J^{(i)}(\boldsymbol{\theta}) = -\log p_{\boldsymbol{\theta}}(y^i | \mathbf{x}^i).$$

GRADIENT FOR LOGISTIC REGRESSION

Whiteboard

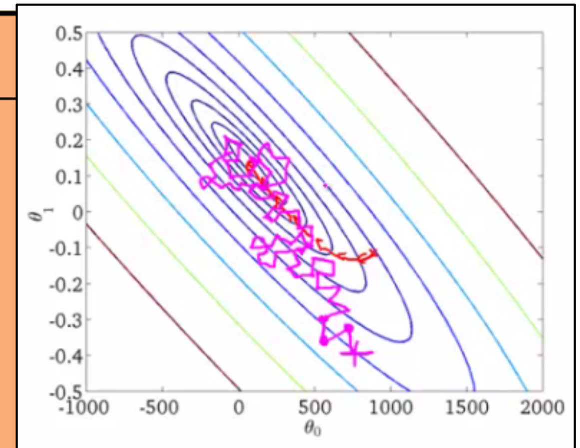
- Partial derivative for Logistic Regression
- Gradient for Logistic Regression

Details: Picking learning rate

- Use grid-search in log-space over small values on a tuning set:
  - e.g., 0.01, 0.001, …
- Sometimes, decrease after each pass:
  - e.g factor of $1/(1 + dt)$, t=epoch
  - sometimes $1/t^2$
- Fancier techniques I won't talk about:
  - Adaptive gradient: scale gradient differently for each dimension (Adagrad, ADAM, ….)

Slide courtesy of William Cohen

SGD for Logistic Regression



Algorithm 1 SGD for Logistic Regression

```
procedure SGD(𝒟, θ^(0))
    θ ← θ^(0)
    while not converged do
        for i ∈ shuffle({1, 2, ..., N}) do
            θ ← θ − λ(y^(i) − ρ^(i)) x^(i)
            where ρ^(i) := 1/(1 + exp(−θ^T x))
    return θ
```

procedure SGD($\mathcal{D}, \boldsymbol{\theta}^{(0)}$)
- $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
- **while** not converged **do**
  - **for** $i \in$ shuffle($\{1, 2, \ldots, N\}$) **do**
    - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda(y^{(i)} - \rho^{(i)})\mathbf{x}^{(i)}$
    - where $\rho^{(i)} := 1/(1 + \exp(-\boldsymbol{\theta}^T\mathbf{x}))$
- **return** $\boldsymbol{\theta}$

We can also apply SGD to solve the MCLE problem for Logistic Regression.

We need a per-example objective:

$$\text{Let } J(\boldsymbol{\theta}) = \textstyle\sum_{i=1}^N J^{(i)}(\boldsymbol{\theta})$$
$$\text{where } J^{(i)}(\boldsymbol{\theta}) = -\log p_{\boldsymbol{\theta}}(y^i|\mathbf{x}^i).$$

Takeaways

1. Discriminative classifiers directly model the **conditional,** p(y|x)
2. Logistic regression is a **simple linear classifier**, that retains a **probabilistic** semantics
3. Parameters in LR are learned by **iterative optimization** (e.g. SGD)

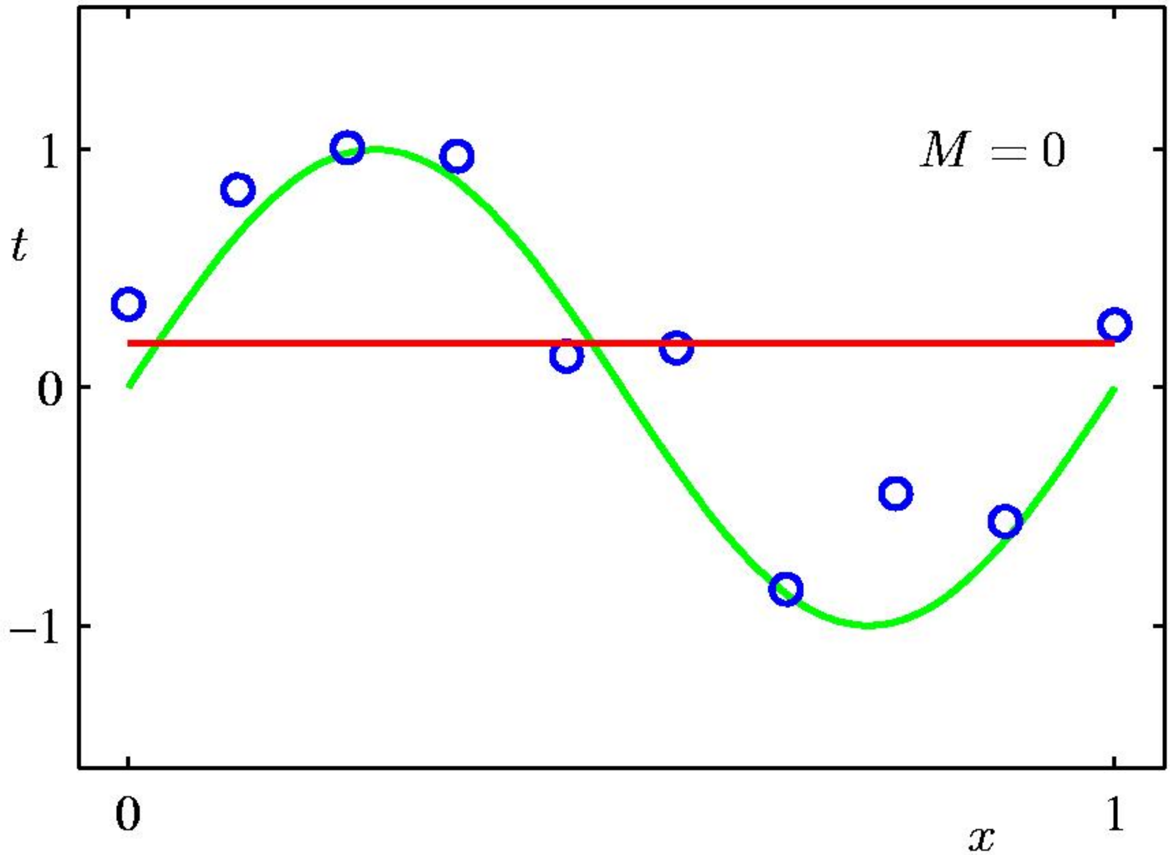
NON-LINEAR FEATURES

Example: Linear Regression Nonlinear Features

Polynomial basis vectors on a small dataset

– From Bishop Ch 1

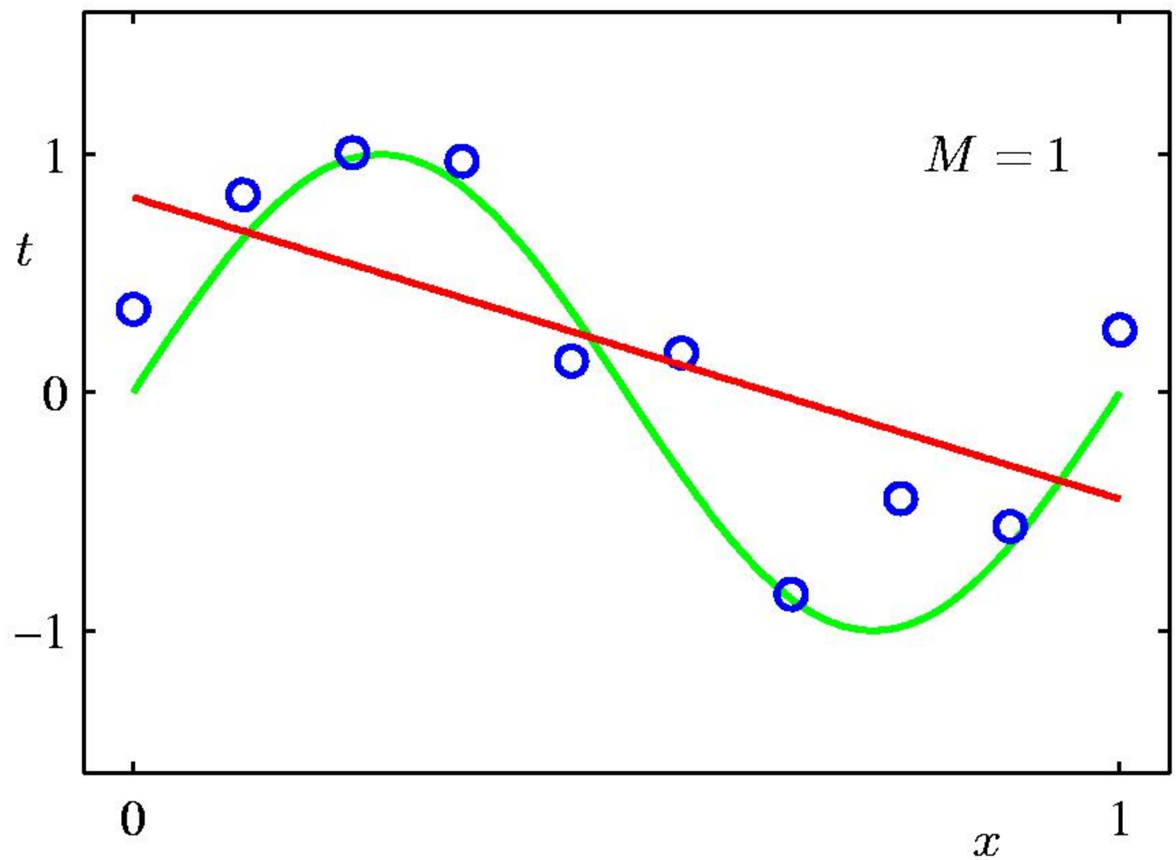
0^{th} Order Polynomial



n=10

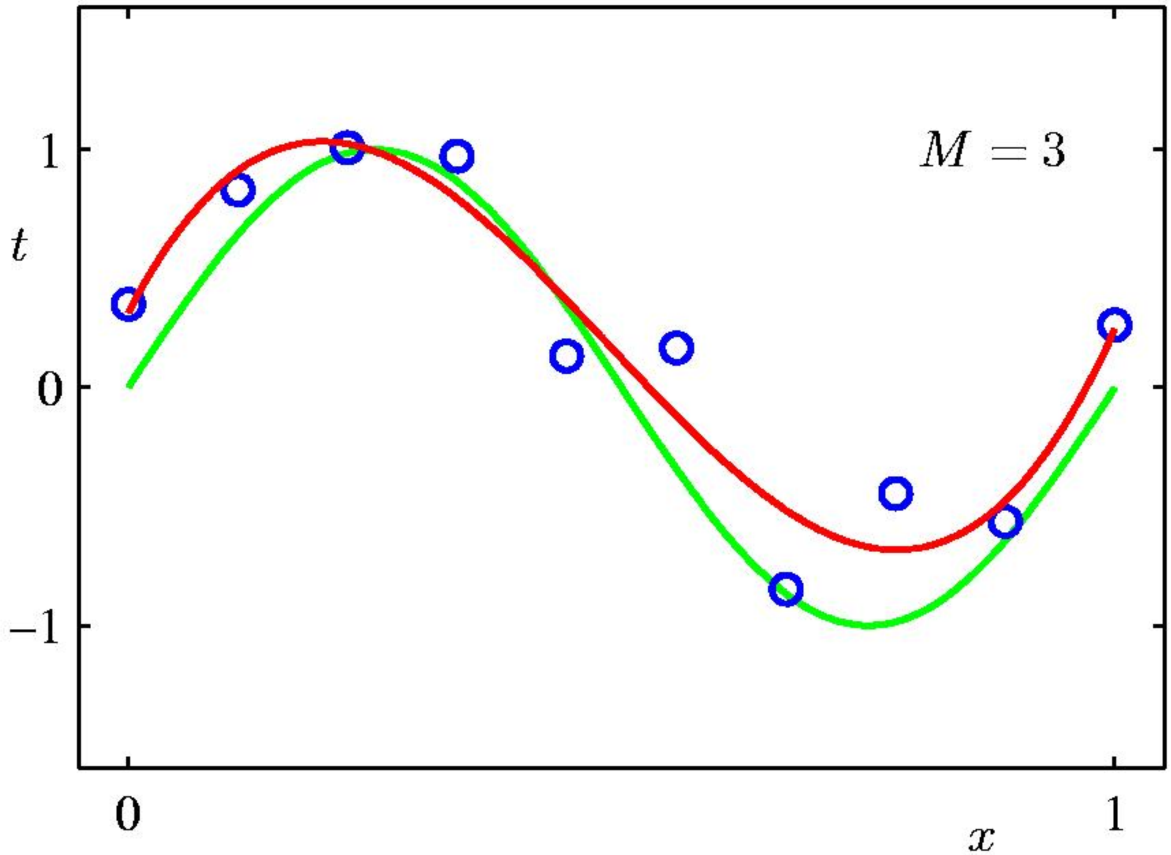
Slide courtesy of William Cohen

1st Order Polynomial



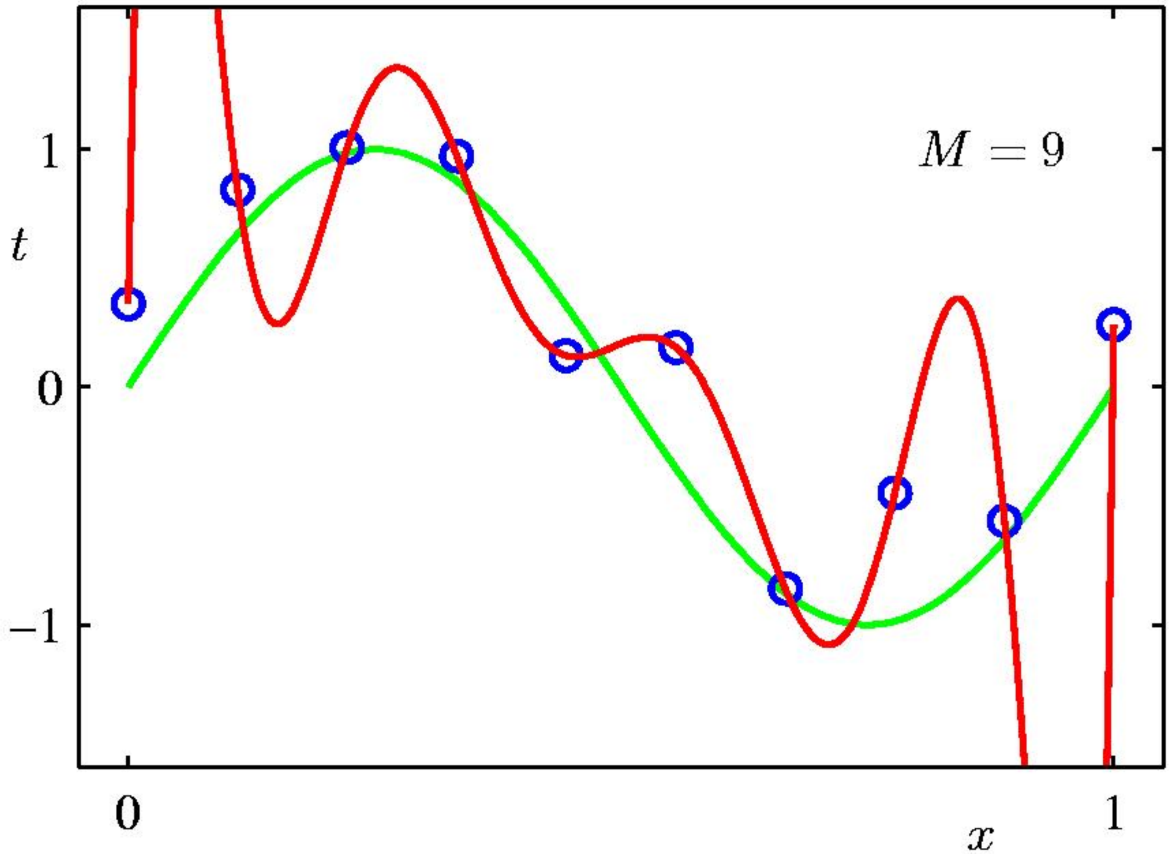
Slide courtesy of William Cohen

3rd Order Polynomial



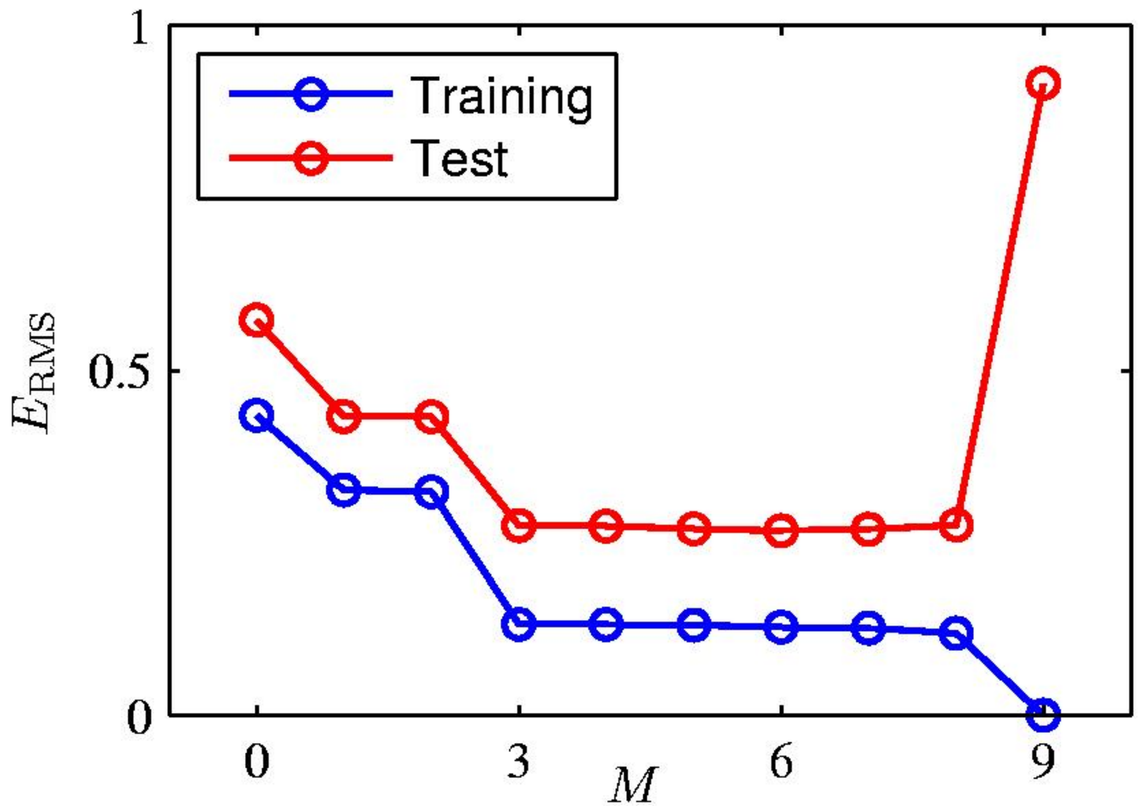
Slide courtesy of William Cohen

9^{th} Order Polynomial



Slide courtesy of William Cohen

Over-fitting



Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$

Slide courtesy of William Cohen

Polynomial Coefficients

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
θ_0	0.19	0.82	0.31	0.35
θ_1		-1.27	7.99	232.37
θ_2			-25.43	-5321.83
θ_3			17.37	48568.31
θ_4				-231639.30
θ_5				640042.26
θ_6				-1061800.52
θ_7				1042400.18
θ_8				-557682.99
θ_9				125201.43

Slide courtesy of William Cohen

Overfitting

Definition: The problem of **overfitting** is when the model captures the noise in the training data instead of the underlying structure

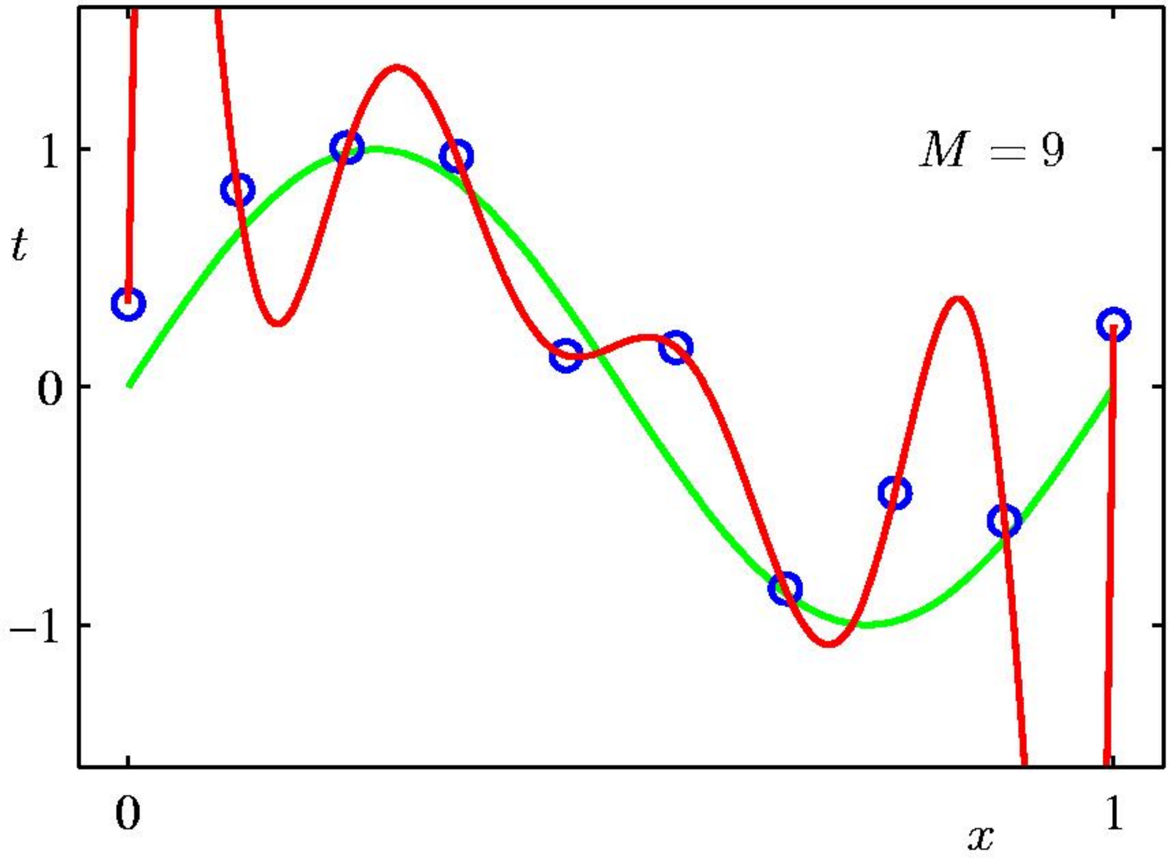
Overfitting can occur in all the models we've seen so far:

- KNN (e.g. when k is small)
- Naïve Bayes (e.g. without a prior)
- Linear Regression (e.g. with basis function)
- Logistic Regression (e.g. with many rare features)

9^{th} Order Polynomial

(Small # of examples)

$$N = 10$$

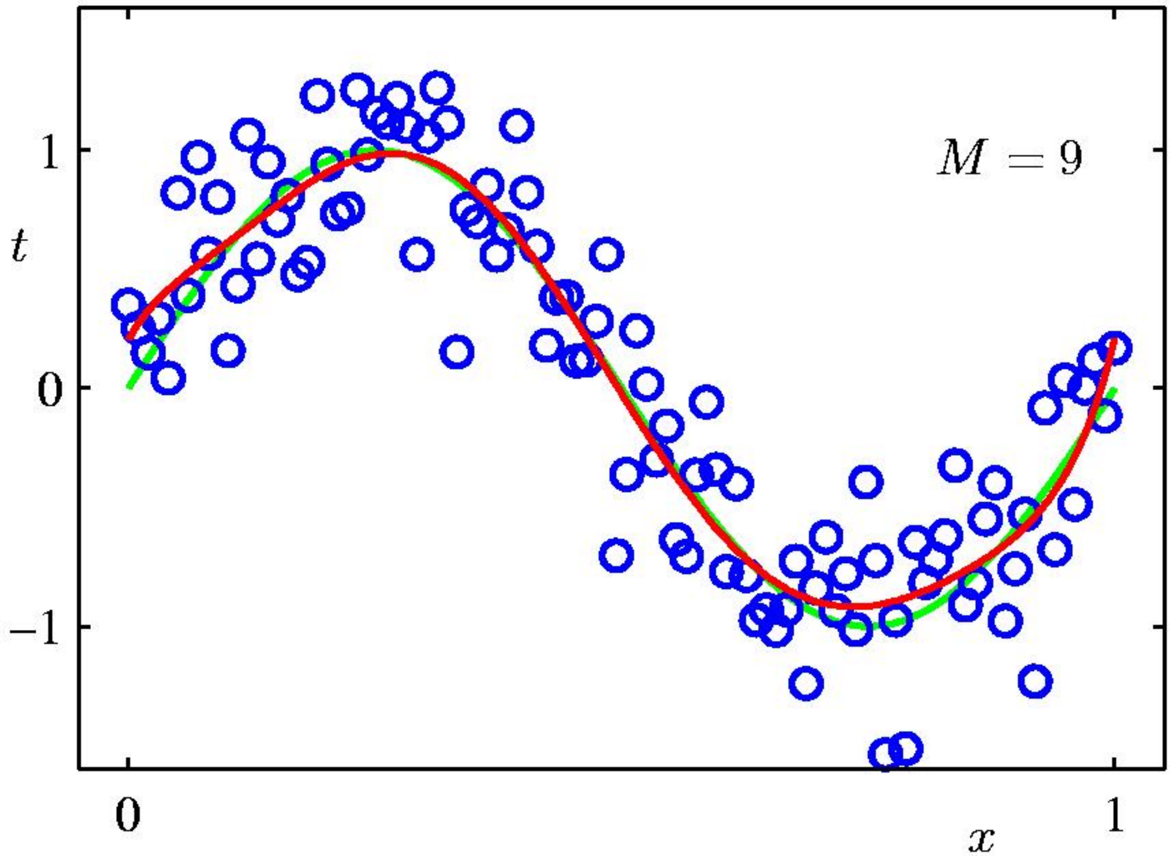


Slide courtesy of William Cohen

9^{th} Order Polynomial

(Large # of examples)

$$N = 100$$



Slide courtesy of William Cohen

REGULARIZATION

Regularization Outline

- **Regularization**
  - Motivation: Overfitting
  - L2, L1, L0 Regularization
  - Relation between Regularization and MAP Estimation

Recall…

Overfitting

Definition: The problem of **overfitting** is when the model captures the noise in the training data instead of the underlying structure

Overfitting can occur in all the models we've seen so far:

- KNN (e.g. when k is small)
- Naïve Bayes (e.g. without a prior)
- Linear Regression (e.g. with basis function)
- Logistic Regression (e.g. with many rare features)

Motivation: Regularization

Example: Stock Prices

- Suppose we wish to predict Google's stock price at time t+1
- **What features should we use?** (putting all computational concerns aside)
  - Stock prices of all other stocks at times t, t-1, t-2, … , t - k
  - Mentions of Google with positive / negative sentiment words in all newspapers and social media outlets

- Do we believe that **all** of these features are going to be useful?

Motivation: Regularization

- **Occam's Razor:** prefer the simplest hypothesis

- What does it mean for a hypothesis (or model) to be **simple**?
  1. small number of features (**model selection**)
  2. small number of "important" features (**shrinkage**)

Regularization

Whiteboard

- L2, L1, L0 Regularization
- Example: Linear Regression
- Probabilistic Interpretation of Regularization

Regularization

Don't Regularize the Bias (Intercept) Parameter!

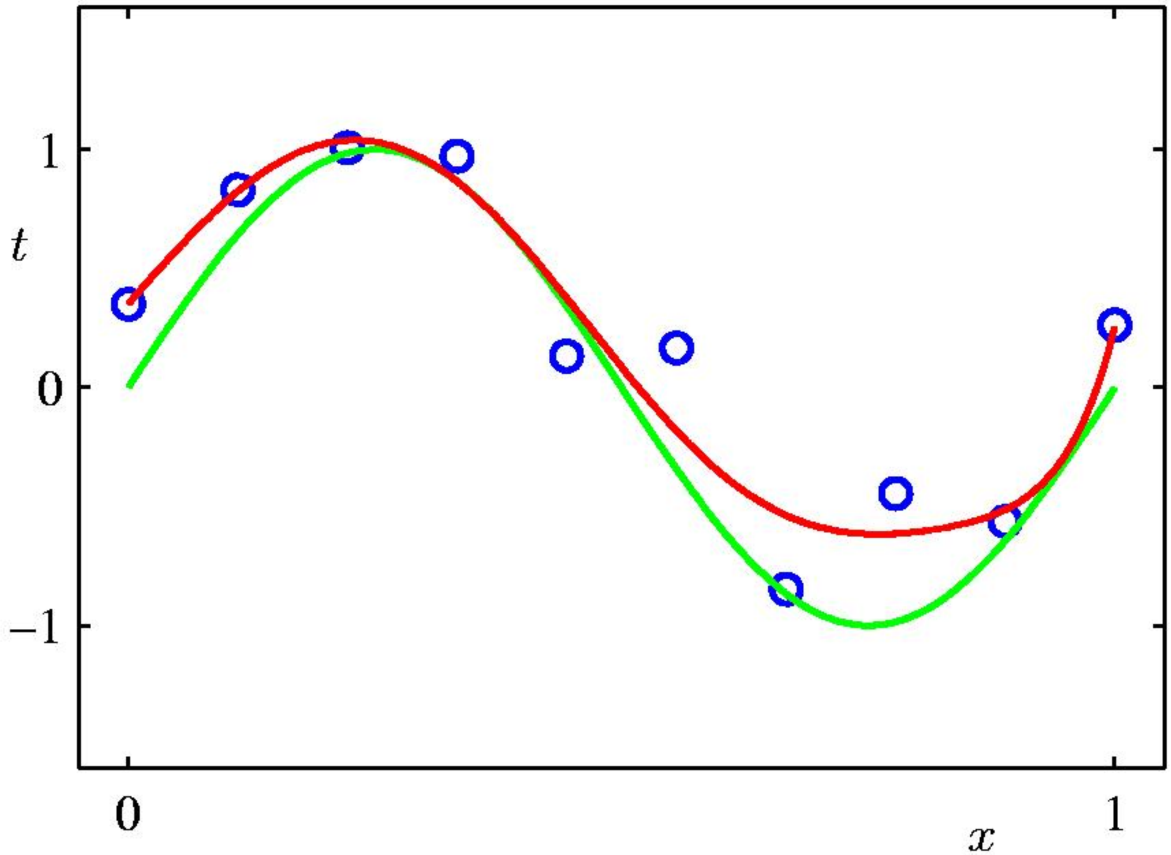
- In our models so far, the bias / intercept parameter is usually denoted by θ_0 -- that is, the parameter for which we fixed $x_0 = 1$
- Regularizers always avoid penalizing this bias / intercept parameter
- Why? Because otherwise the learning algorithms wouldn't be invariant to a shift in the y-values

Whitening Data

- It's common to *whiten* each feature by subtracting its mean and dividing by its variance
- For regularization, this helps all the features be penalized in the same units
(e.g. convert both centimeters and kilometers to z-scores)

Regularization:

$$\ln \lambda = +.18$$



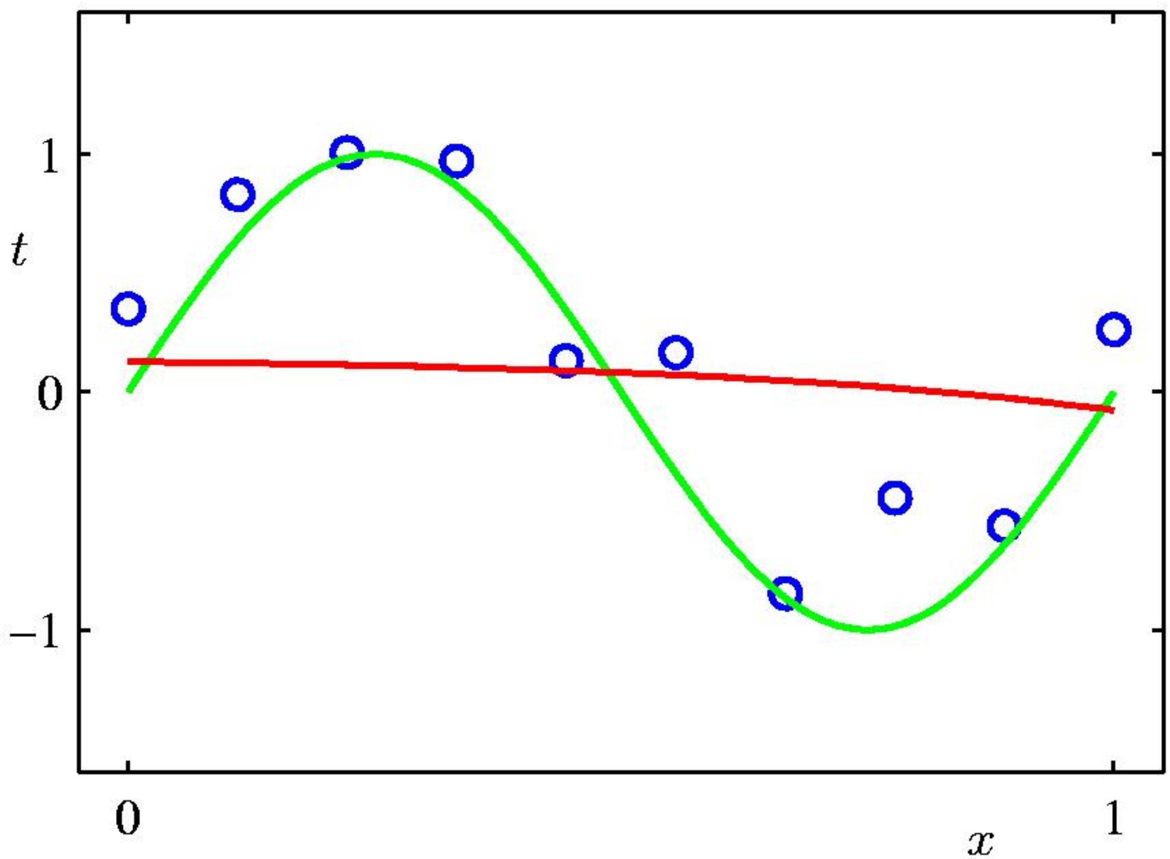
Slide courtesy of William Cohen

Polynomial Coefficients

	none	exp(18)	huge
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Slide courtesy of William Cohen

Over Regularization:

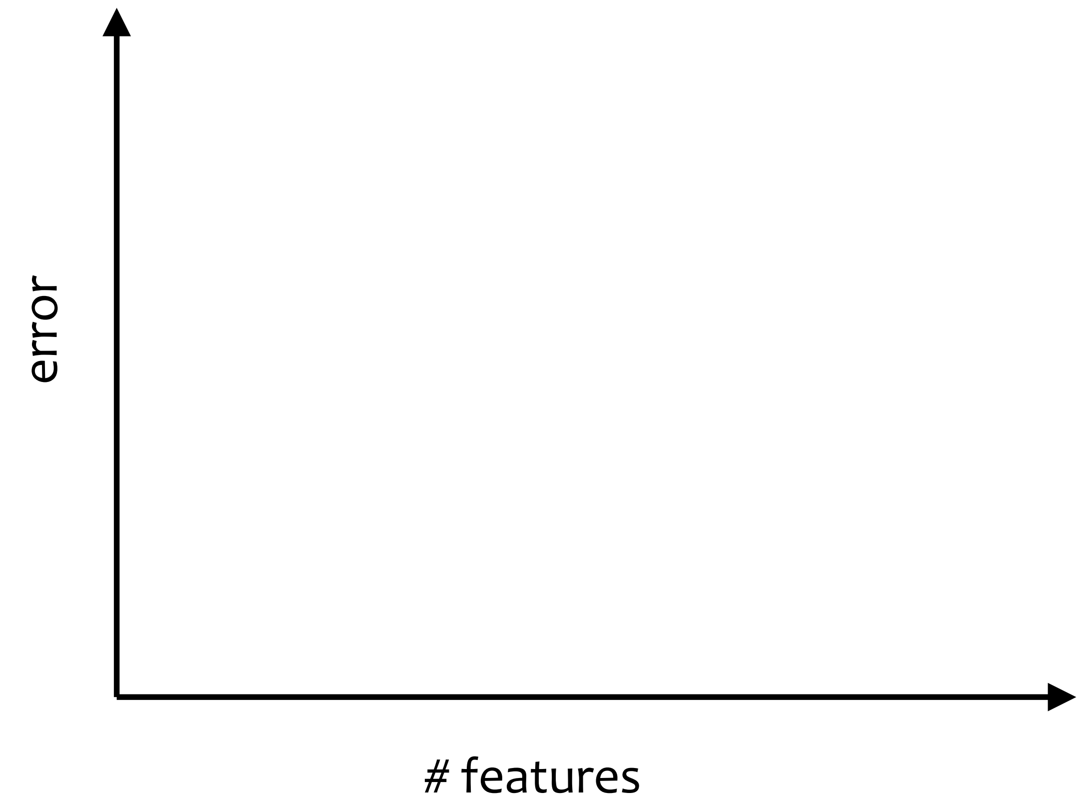


Slide courtesy of William Cohen

Regularization Exercise

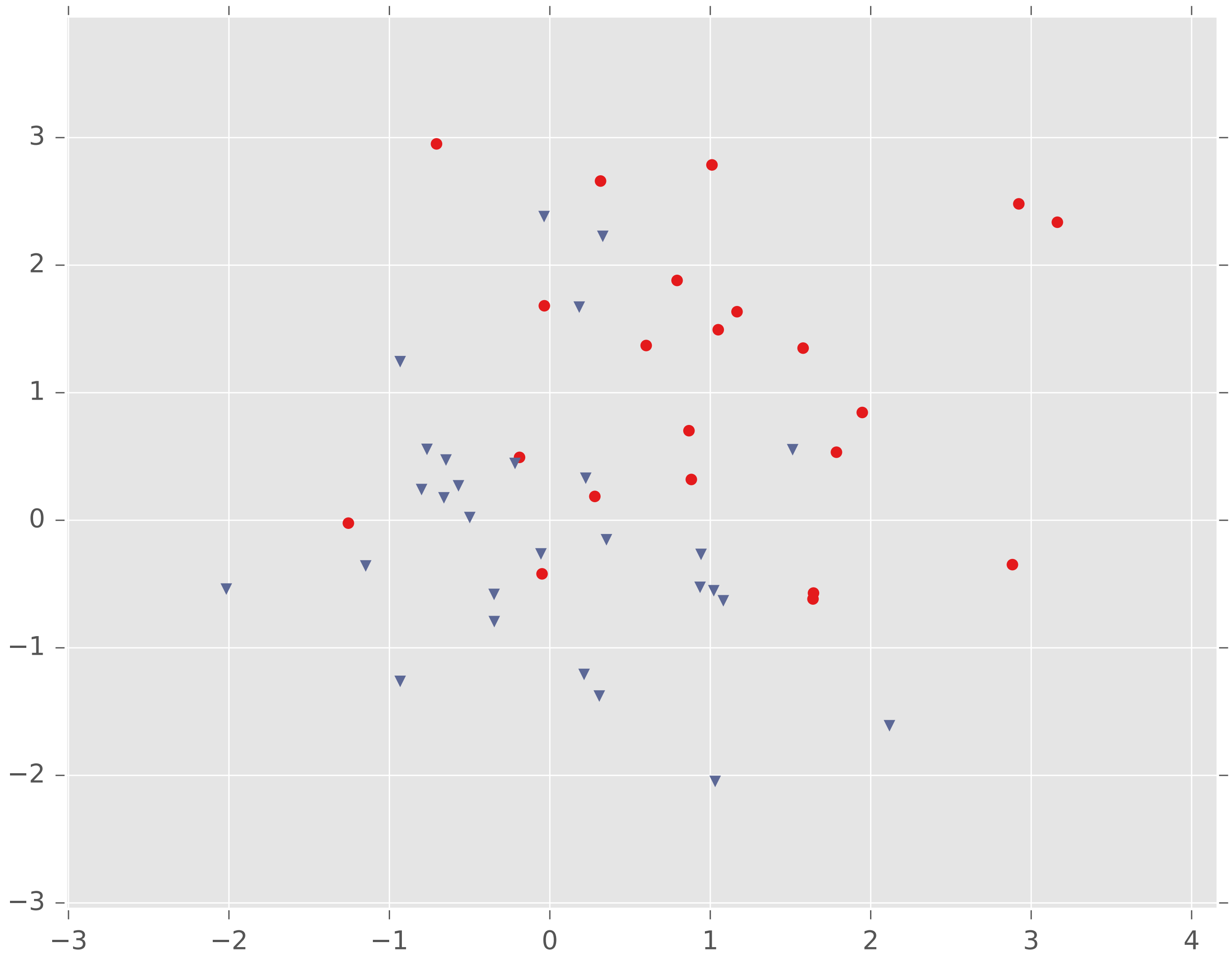
In-class Exercise

1. Plot train error vs. # features (cartoon)
2. Plot test error vs. # features (cartoon)



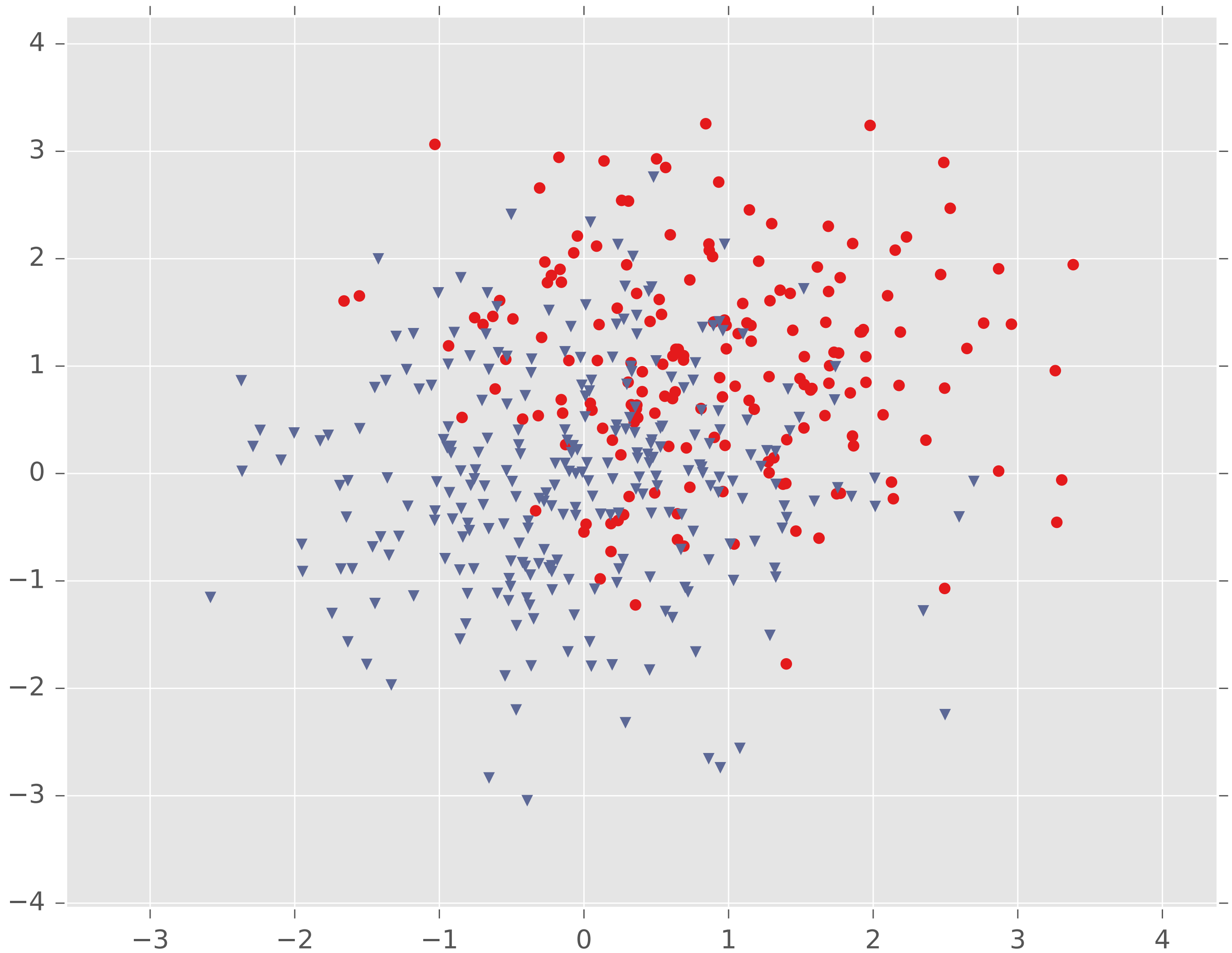
Example: Logistic Regression

Training Data

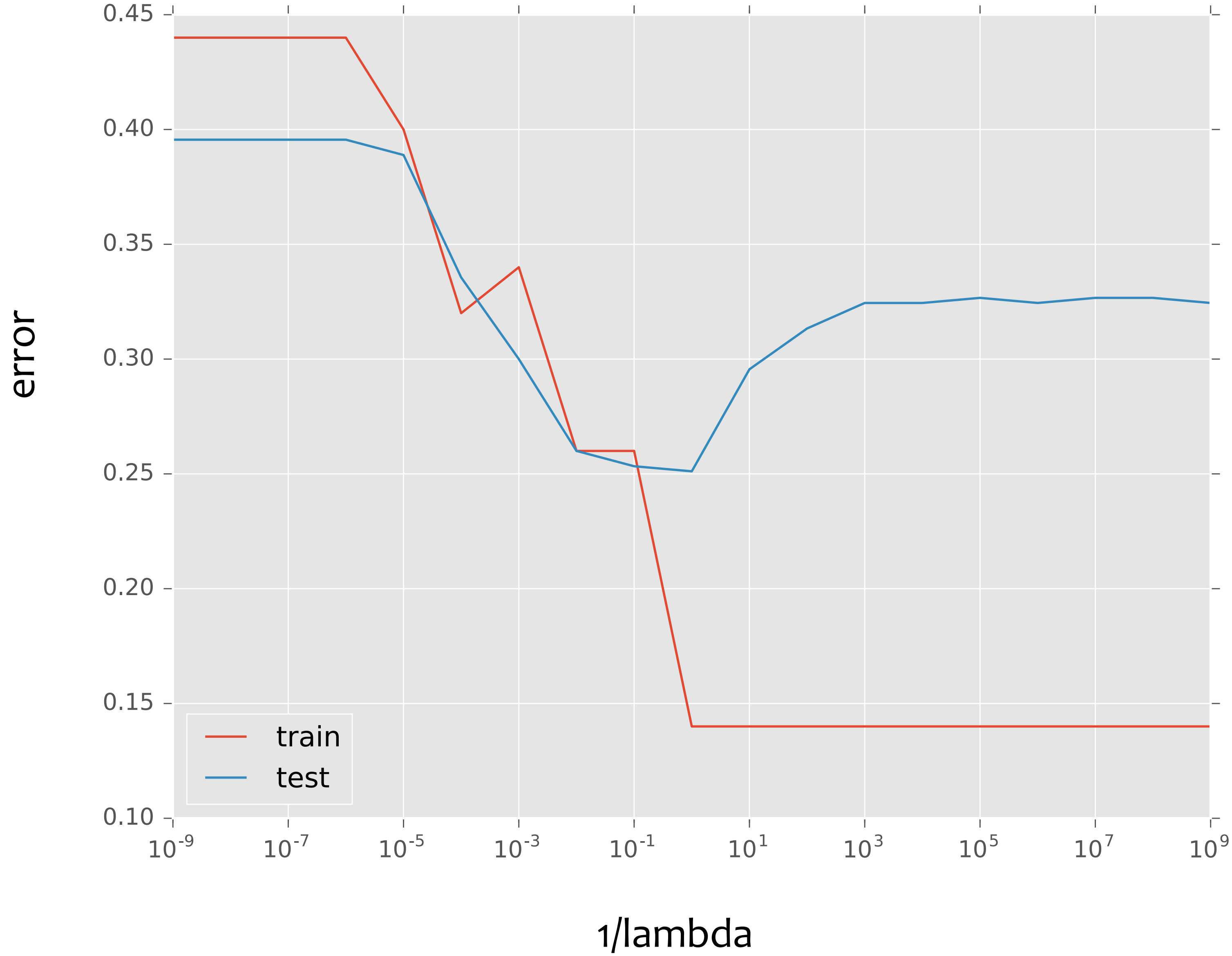


Example: Logistic Regression

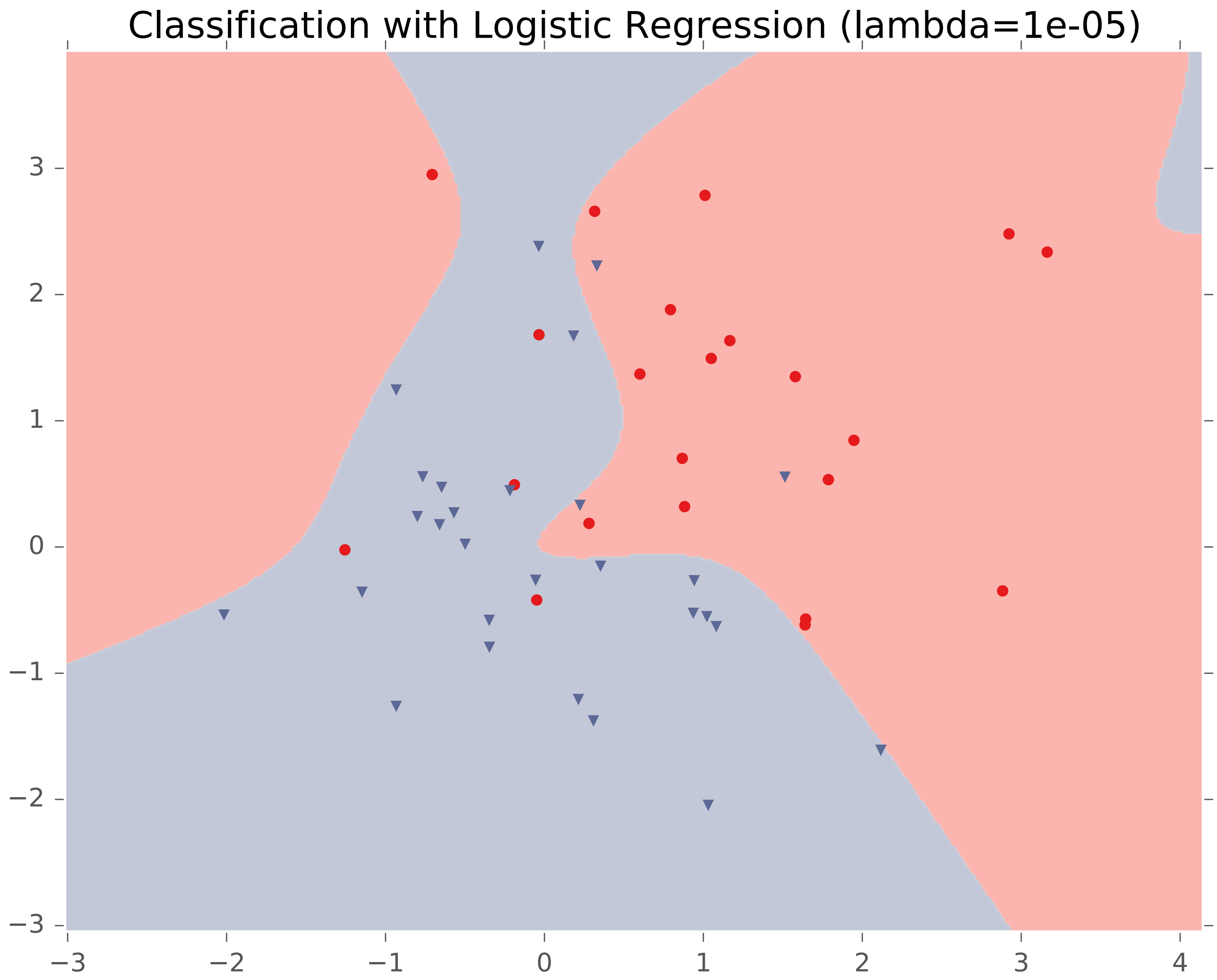
Test Data



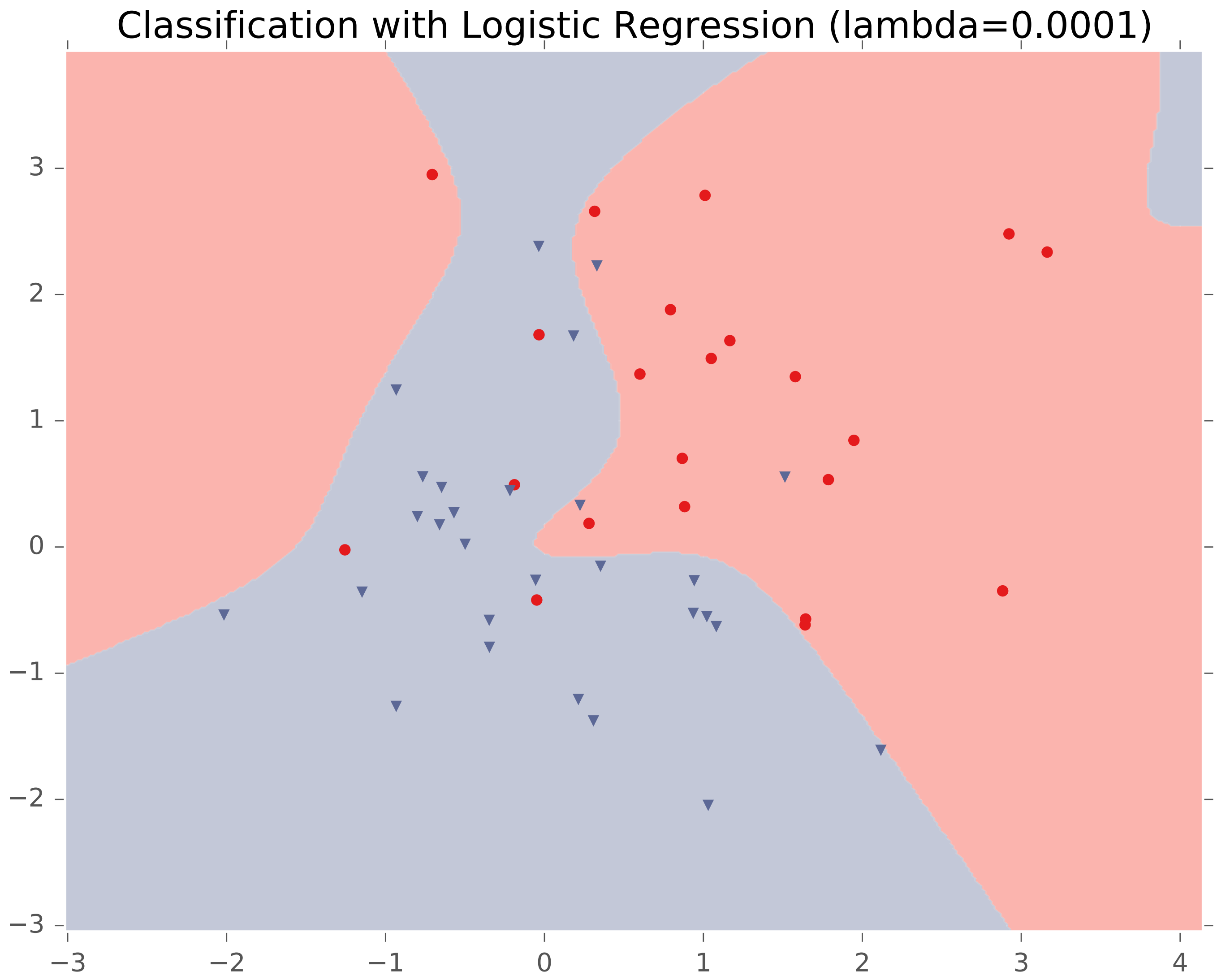
Example: Logistic Regression



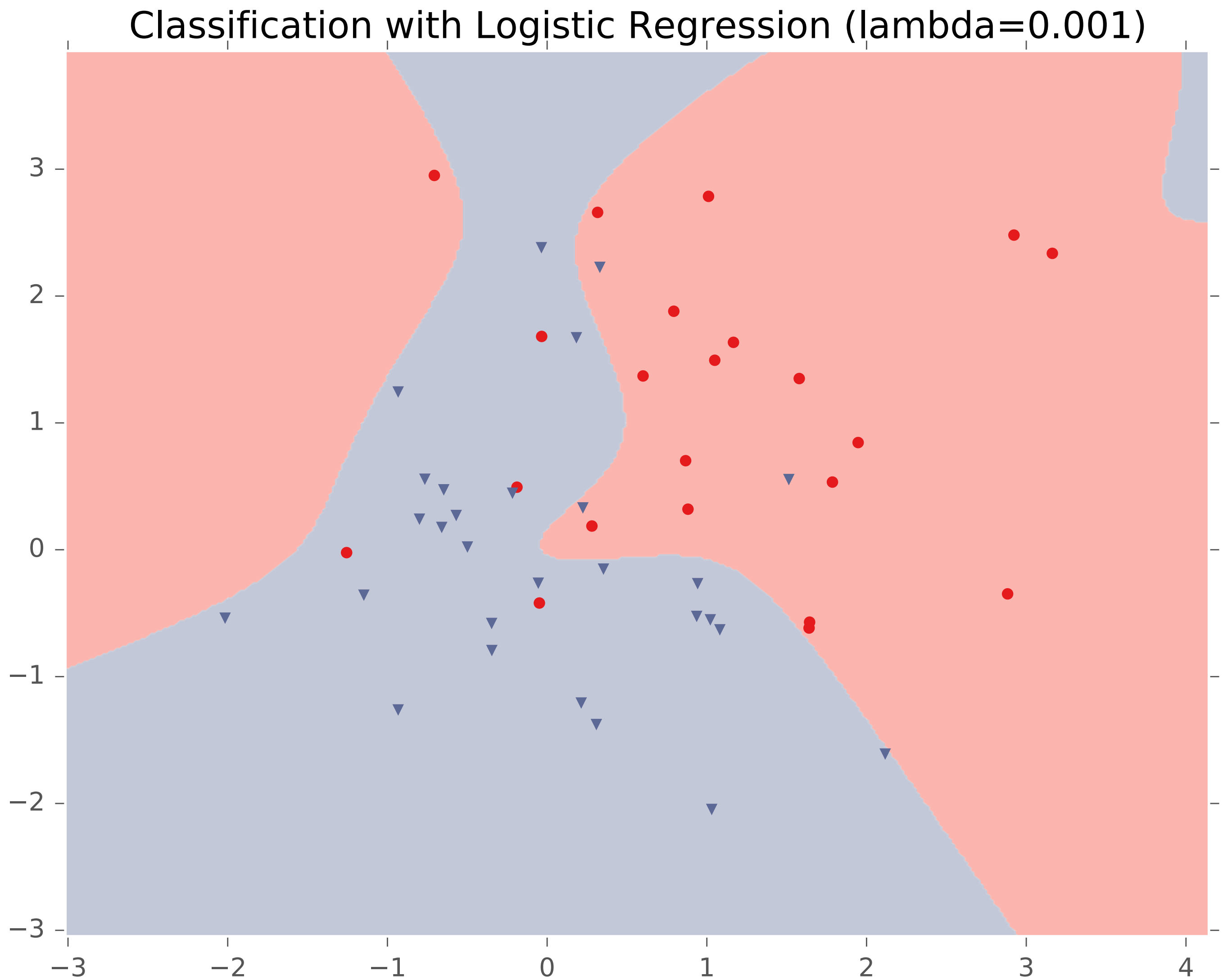
Example: Logistic Regression



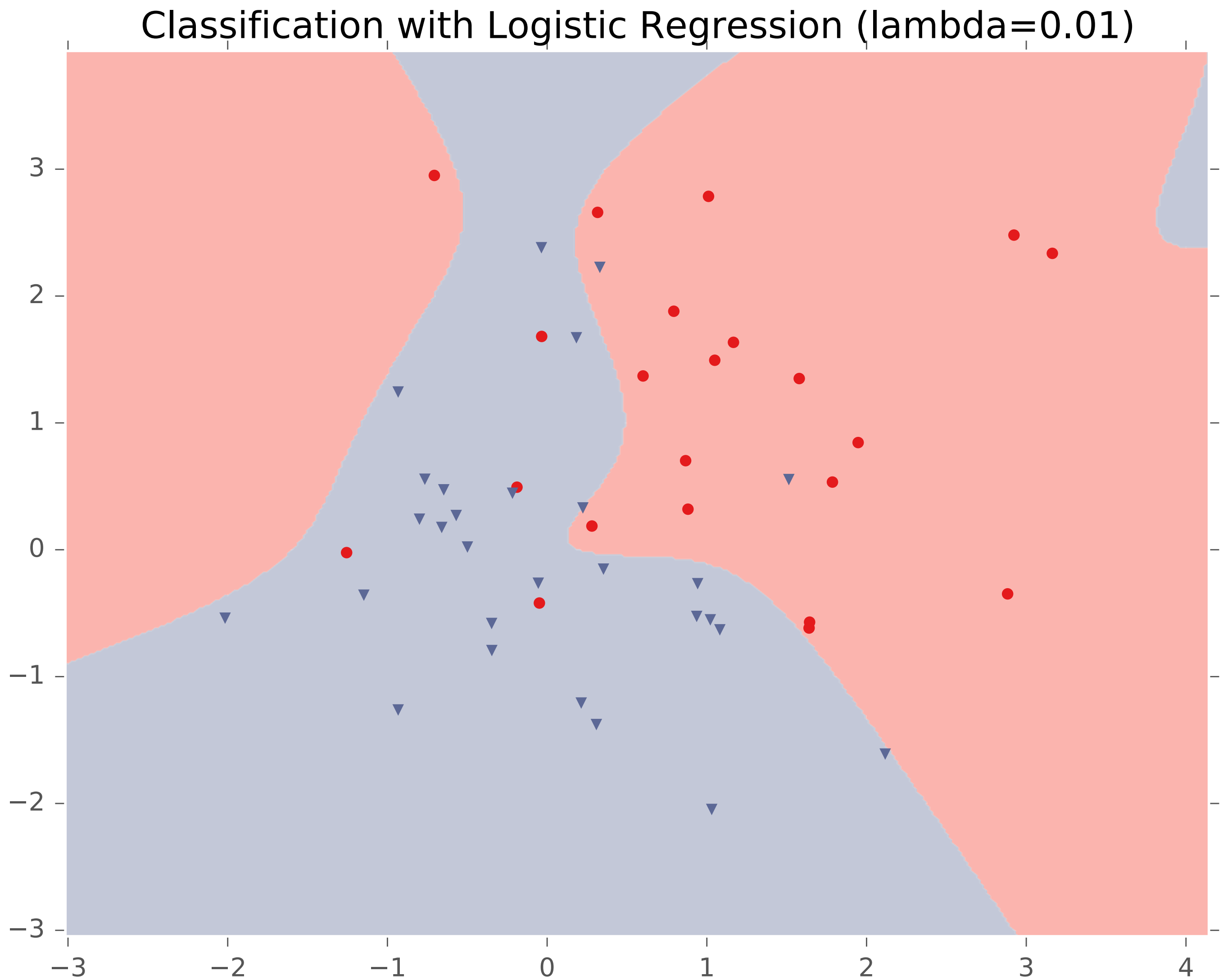
Example: Logistic Regression



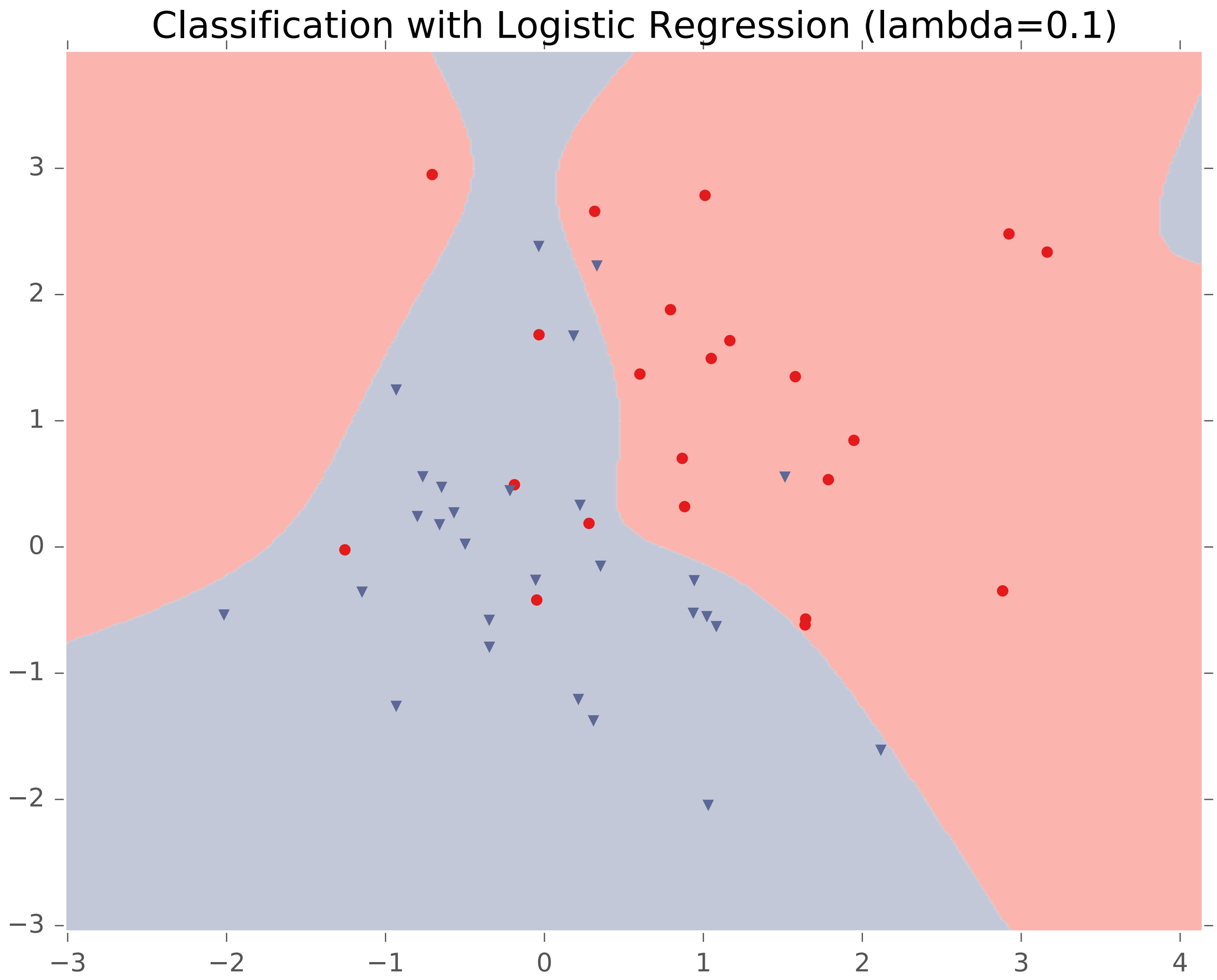
Example: Logistic Regression



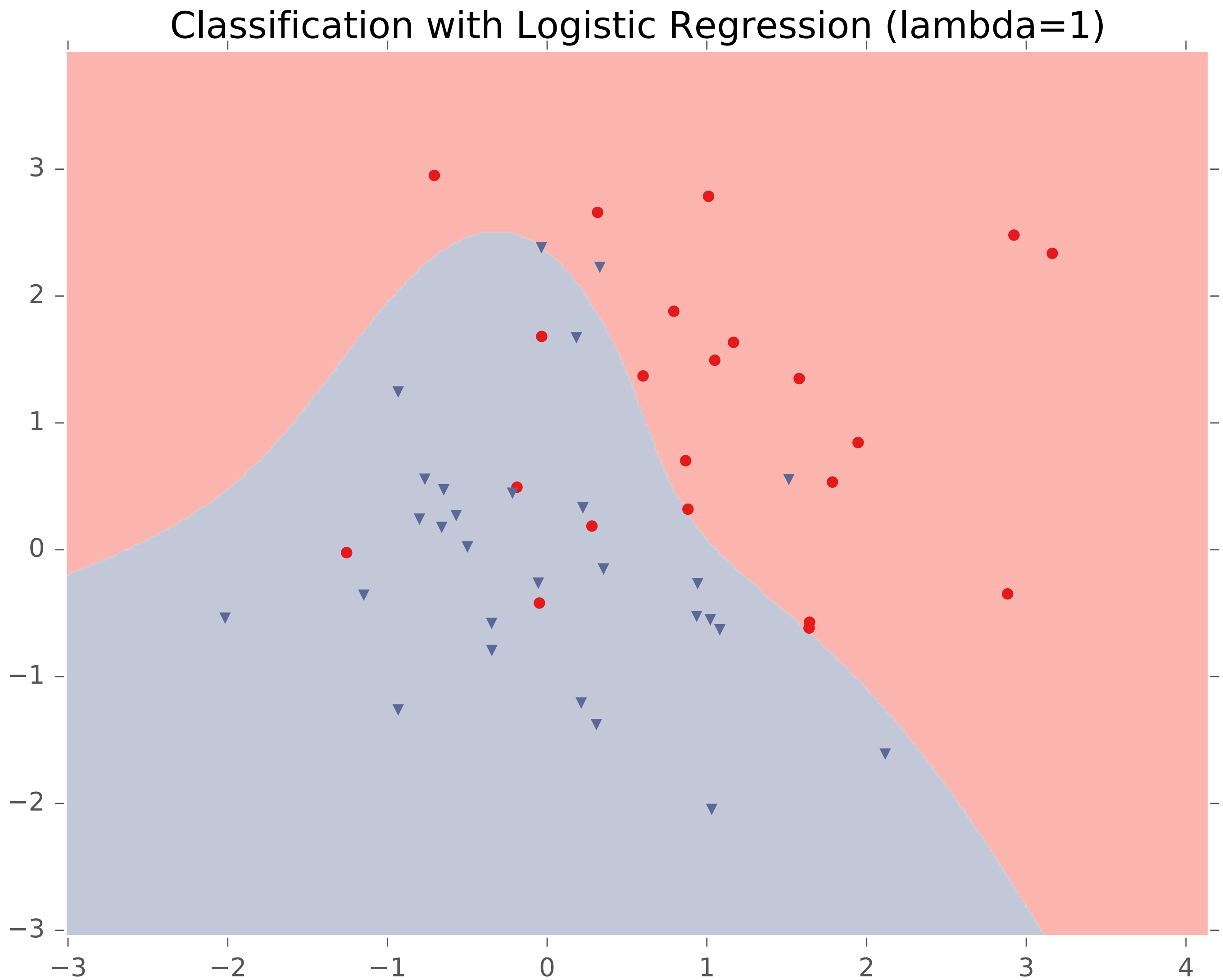
Example: Logistic Regression



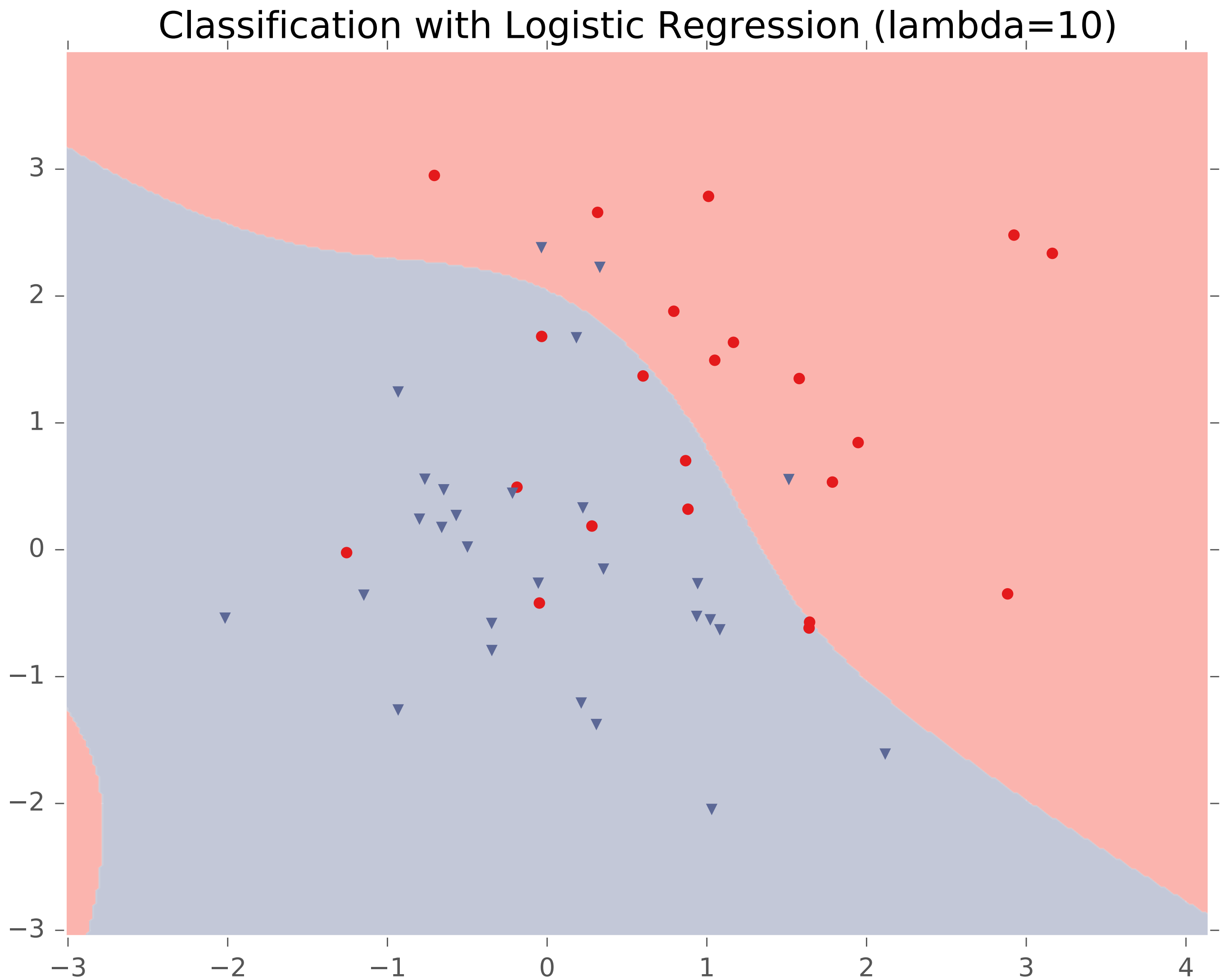
Example: Logistic Regression



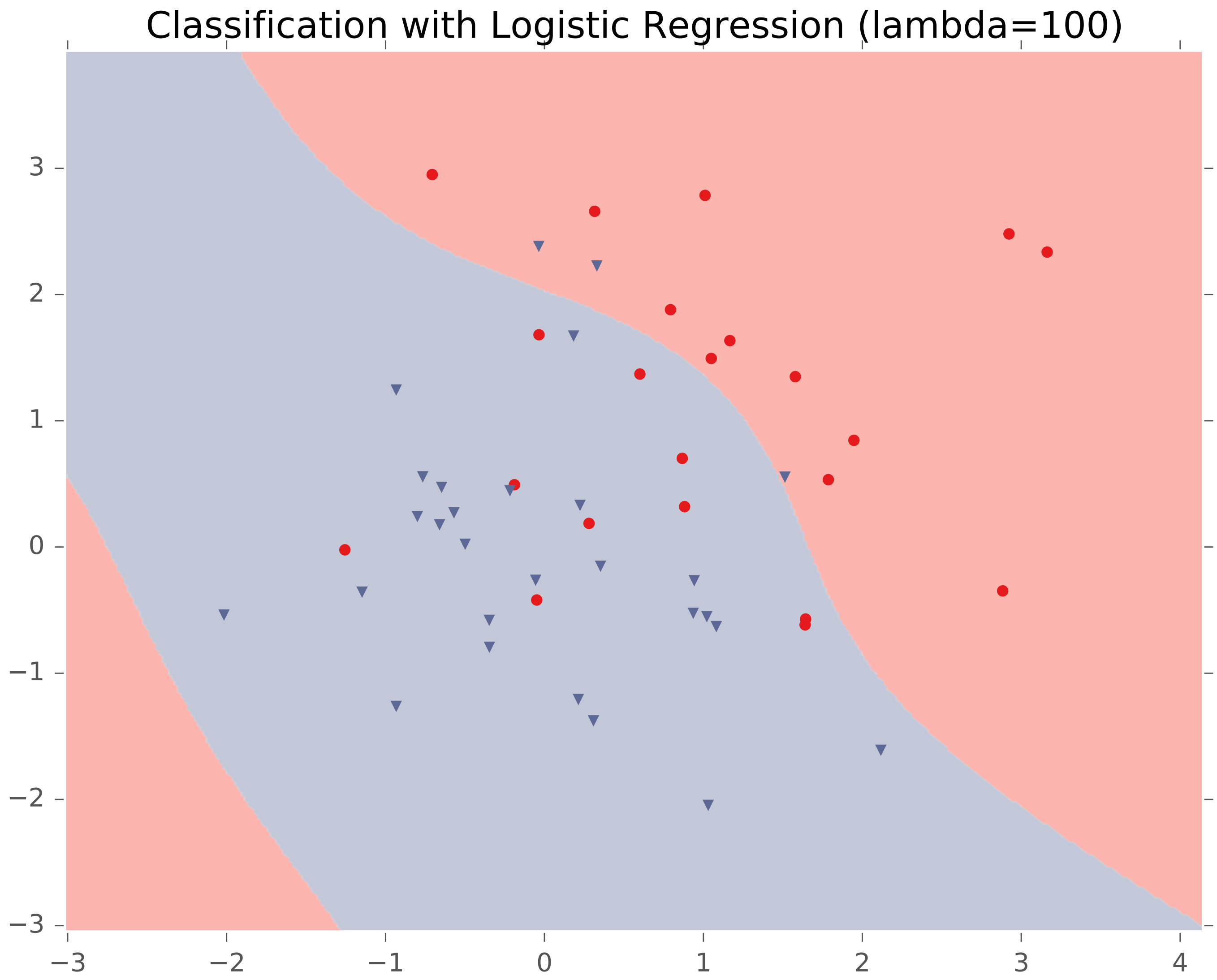
Example: Logistic Regression



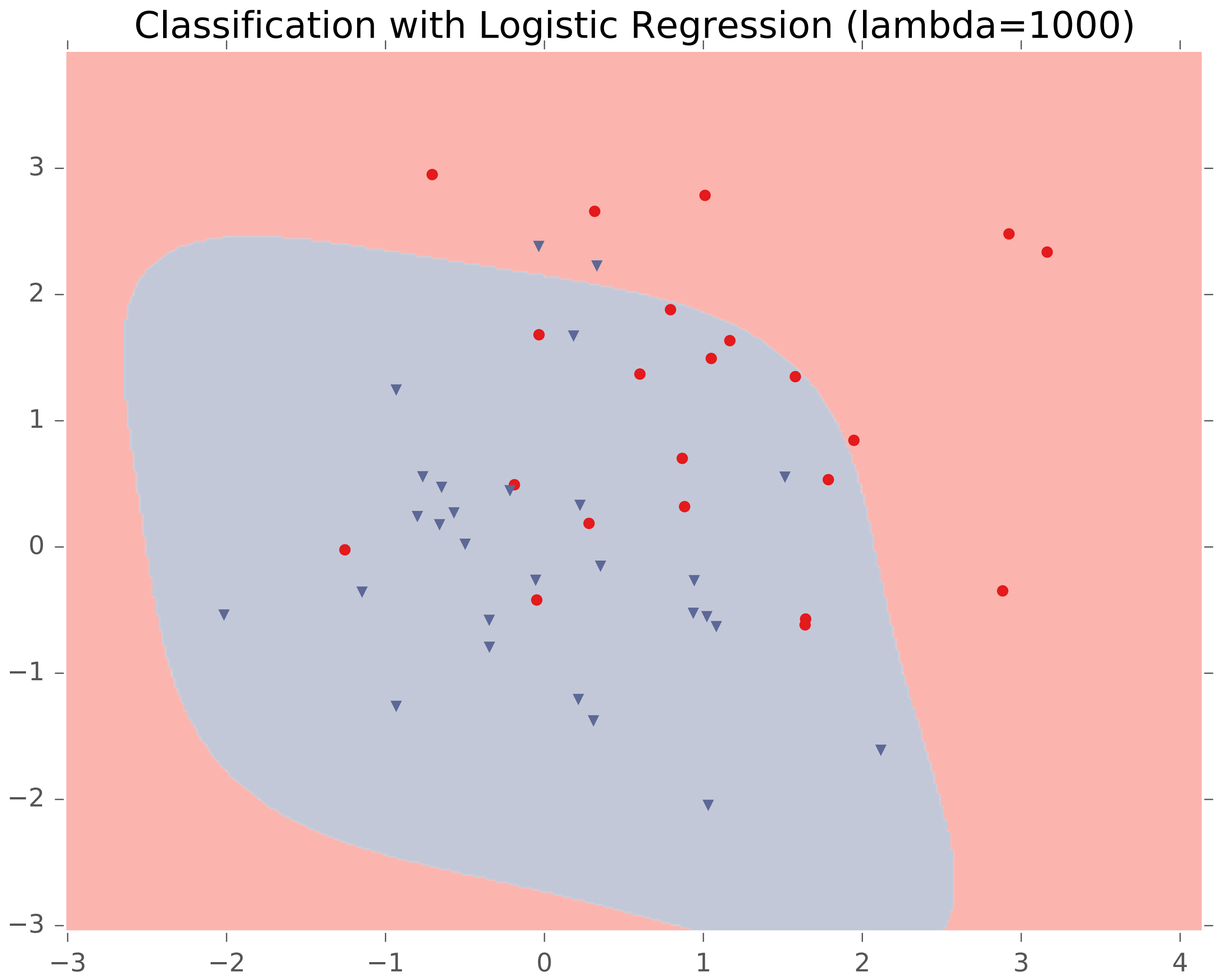
Example: Logistic Regression



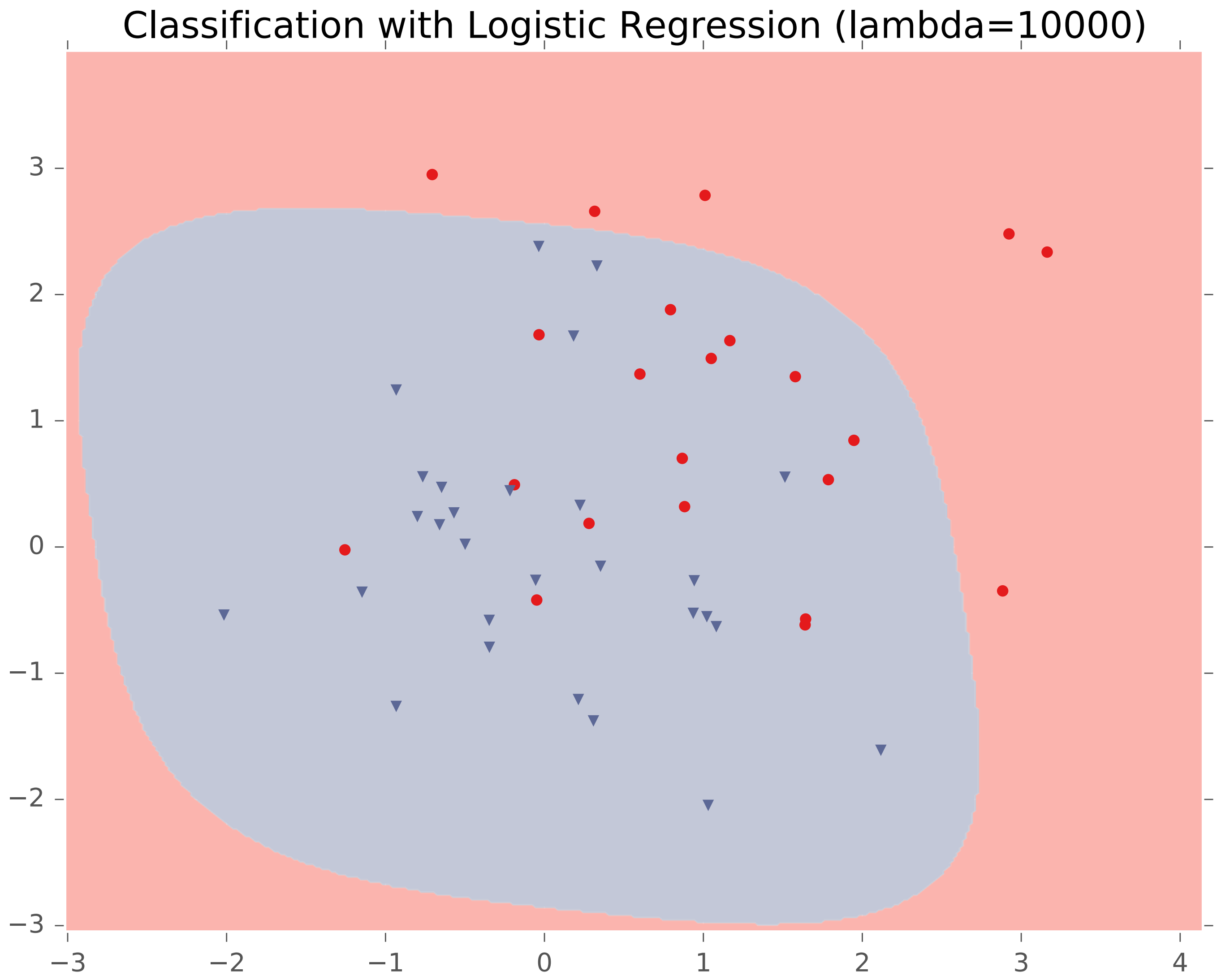
Example: Logistic Regression



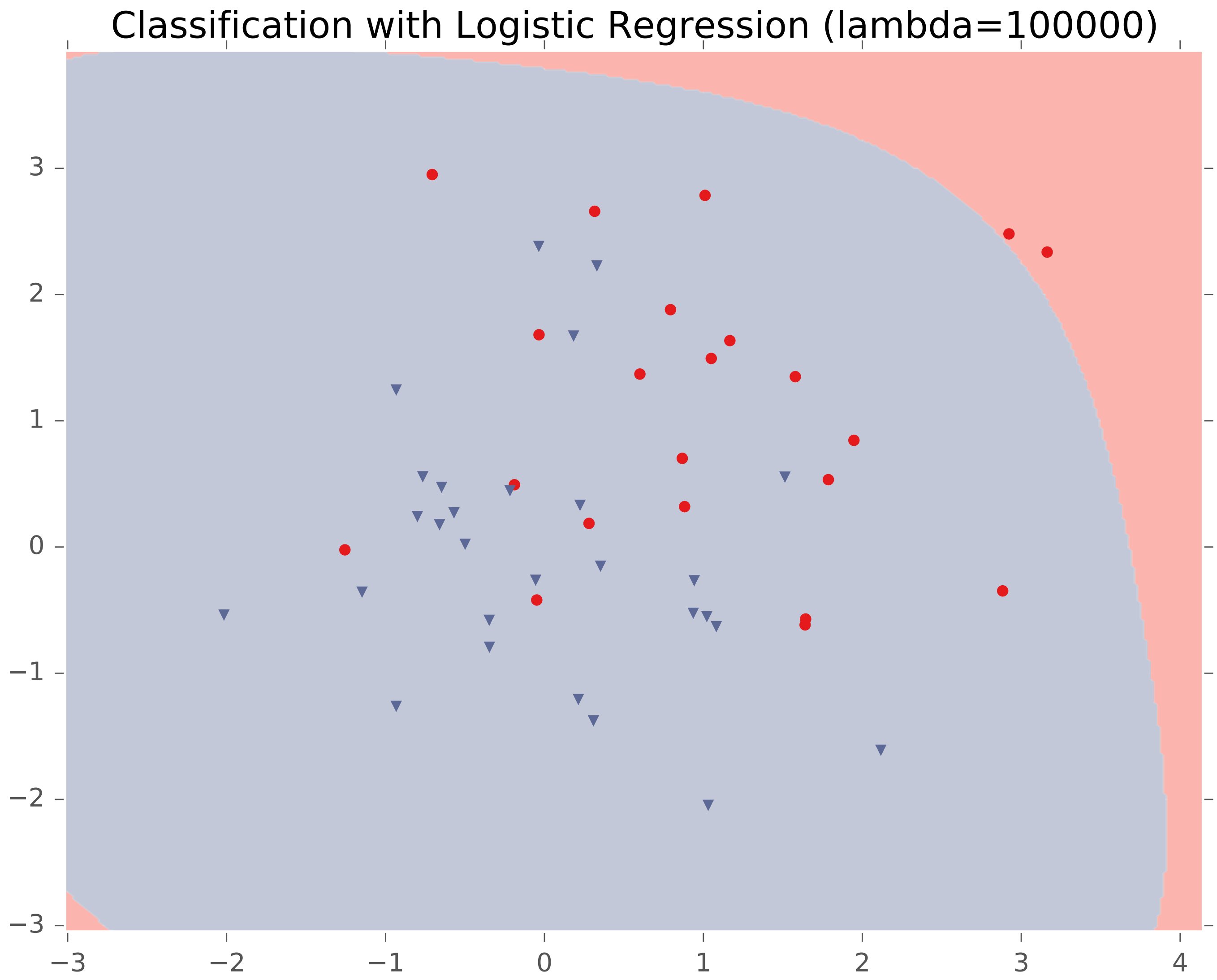
Example: Logistic Regression



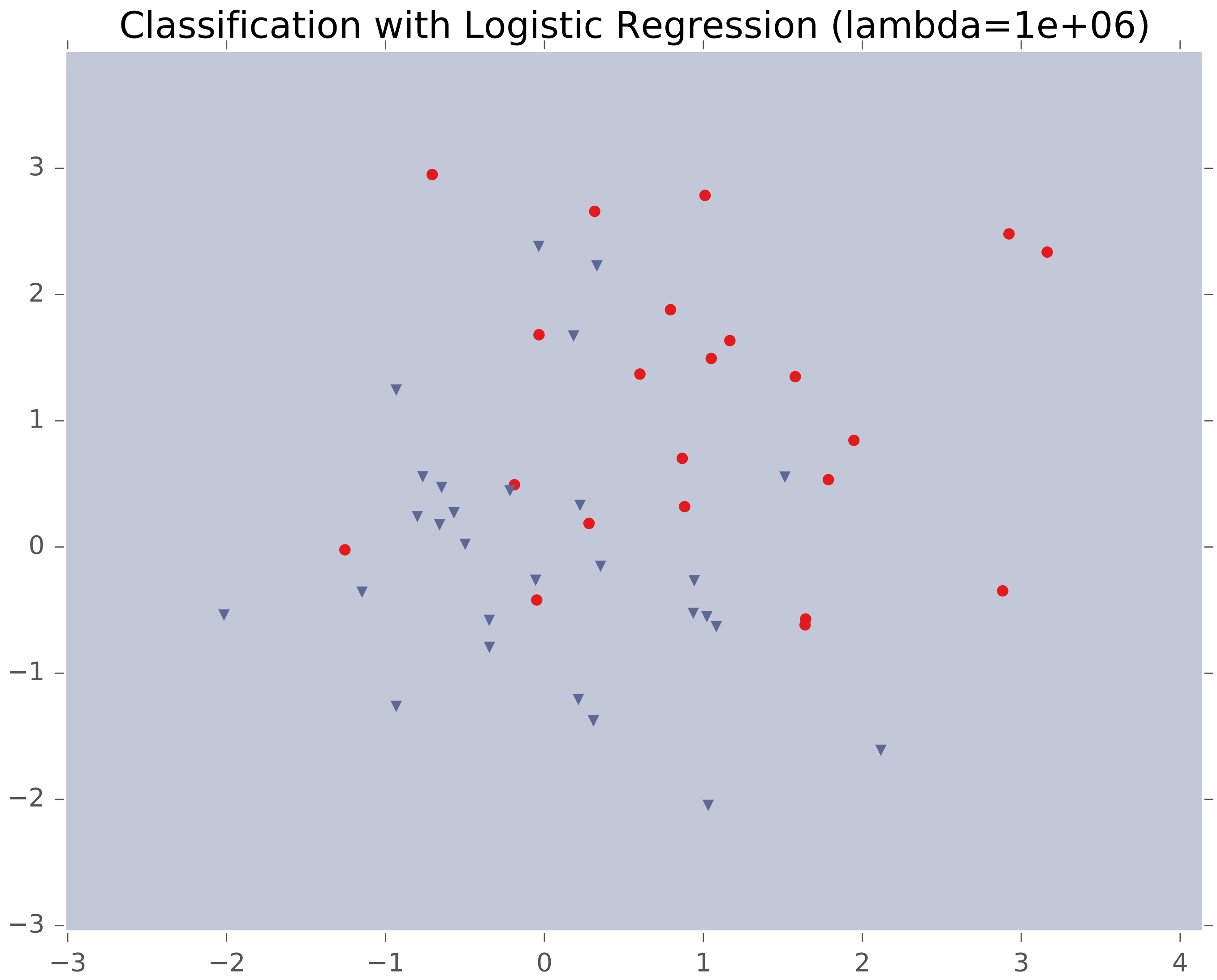
Example: Logistic Regression



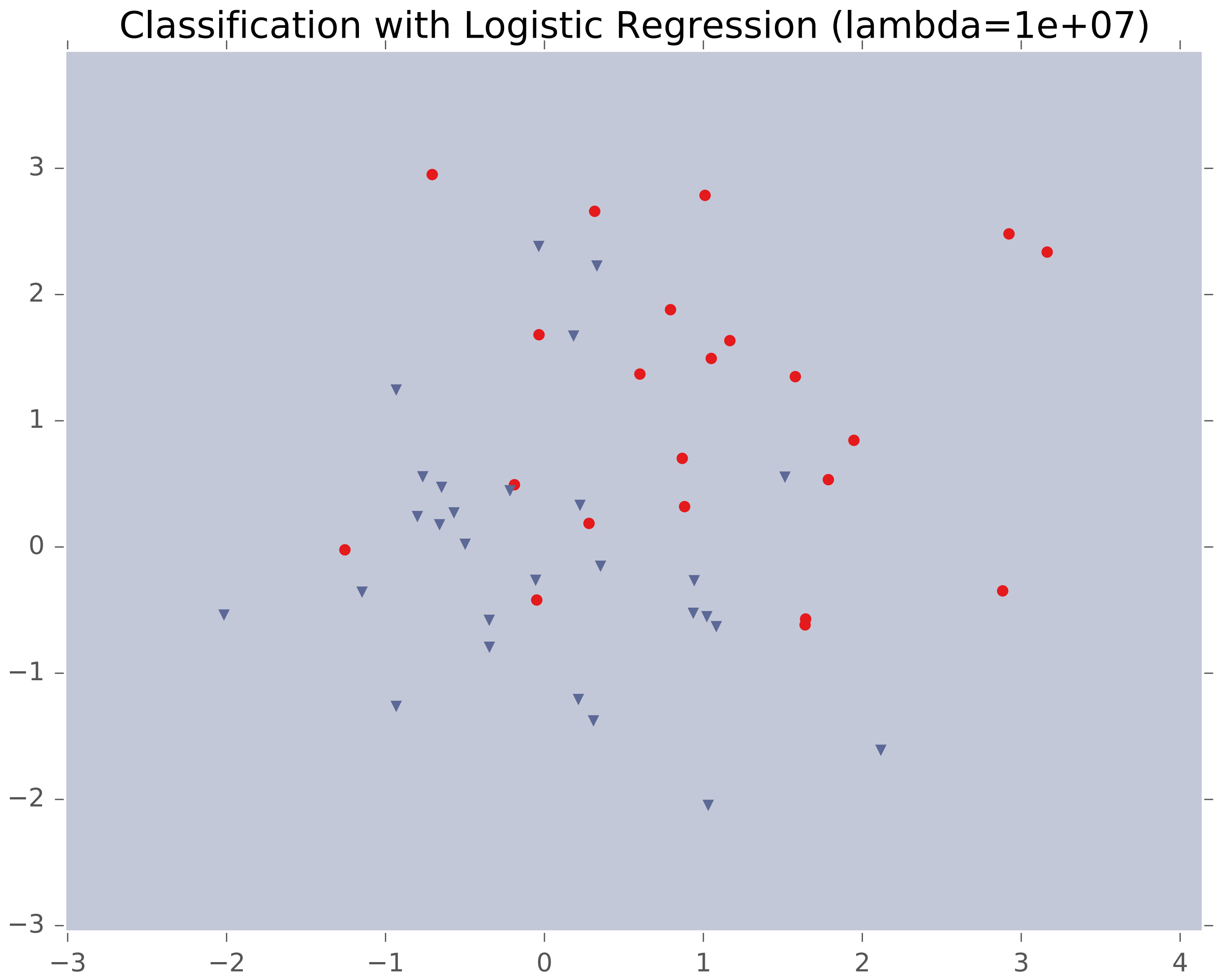
Example: Logistic Regression



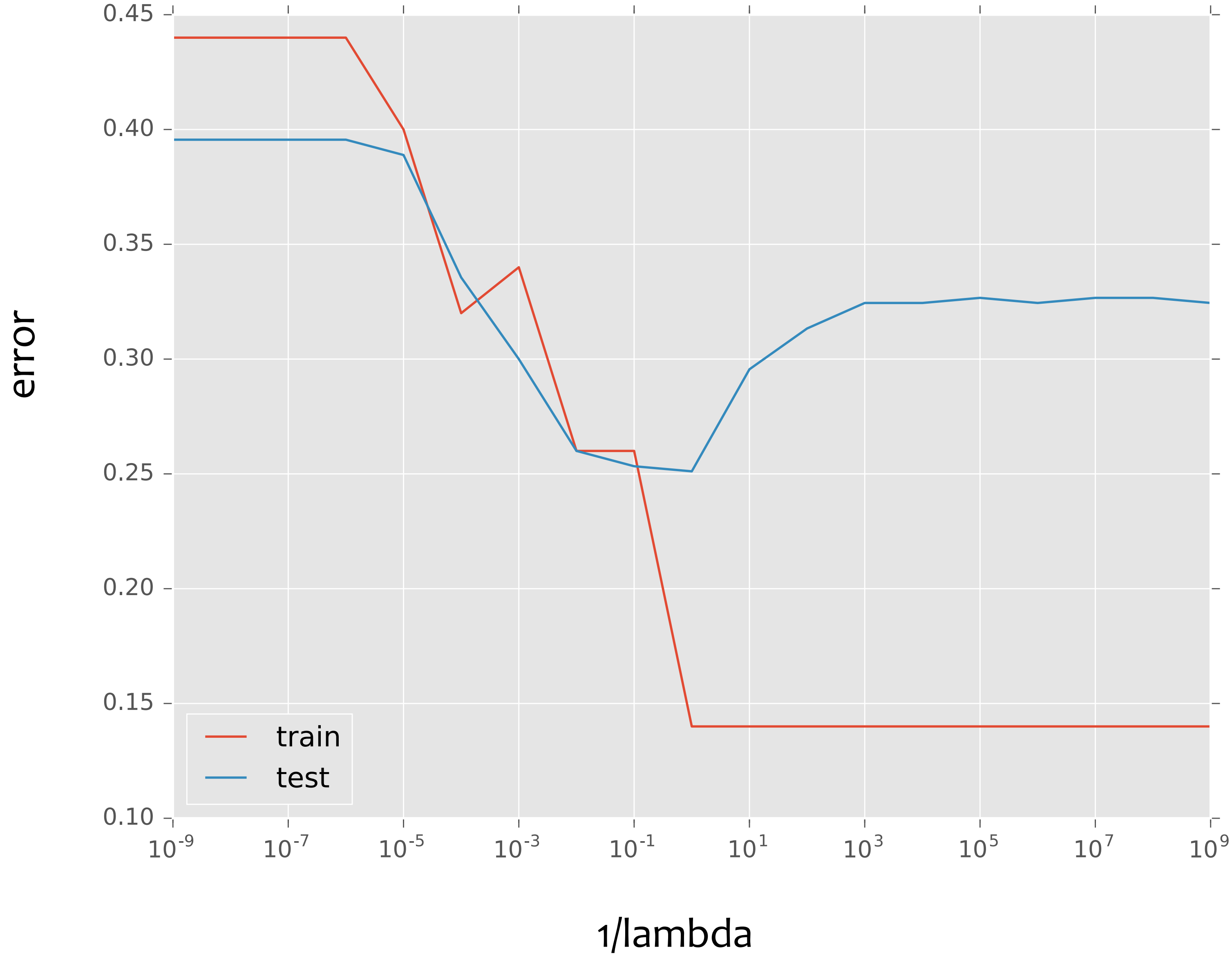
Example: Logistic Regression



Example: Logistic Regression



Example: Logistic Regression



Takeaways

1. **Nonlinear basis functions** allow **linear models** (e.g. Linear Regression, Logistic Regression) to capture **nonlinear** aspects of the original input
2. Nonlinear features are **require no changes to the model** (i.e. just preprocessing)
3. **Regularization** helps to avoid **overfitting**
4. **Regularization** and **MAP estimation** are equivalent for appropriately chosen priors