

10-601 Introduction to Machine Learning

Machine Learning Department School of Computer Science Carnegie Mellon University

Feature Engineering + Regularization + Neural Networks

Matt Gormley Lecture 11 Feb. 19, 2020

Reminders

- Homework 4: Logistic Regression
 - Out: Wed, Feb. 19
 - Due: Fri, Feb. 28 at 11:59pm
- Today's In-Class Poll
 - http://p11.mlcourse.org

MULTINOMIAL LOGISTIC REGRESSION



Multinomial Logistic Regression

Chalkboard

- Background: Multinomial distribution
- Definition: Multi-class classification
- Geometric intuitions
- Multinomial logistic regression model
- Generative story
- Reduction to binary logistic regression
- Partial derivatives and gradients
- Applying Gradient Descent and SGD
- Implementation w/ sparse features



Debug that Program!

In-Class Exercise: Think-Pair-Share

Debug the following program which is (incorrectly) attempting to run SGD for multinomial logistic regression

Buggy Program:

while not converged:

for i in shuffle([1,...,N]):

for k in [1,...,K]:

theta[k] = theta[k] - gamma * grad(x[i], y[i], theta, k)

9[4]

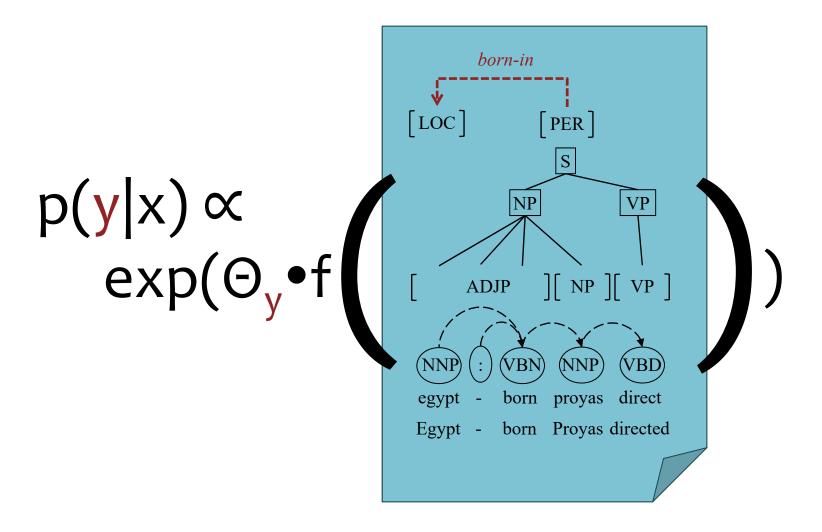
Assume: grad(x[i], y[i], theta, k) returns the gradient of the negative log-likelihood of the training example (x[i],y[i]) with respect to vector theta[k]. gamma is the learning rate. N = # of examples. K = # of output classes. M = # of features. theta is a K by M matrix.

Ofor kin 1. 1. 1.

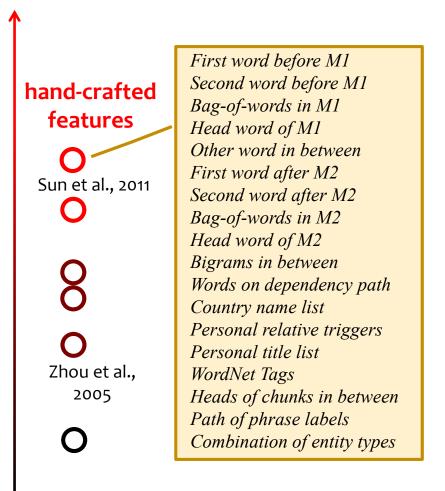
- g[k] = grad(...,,k)

FEATURE ENGINEERING

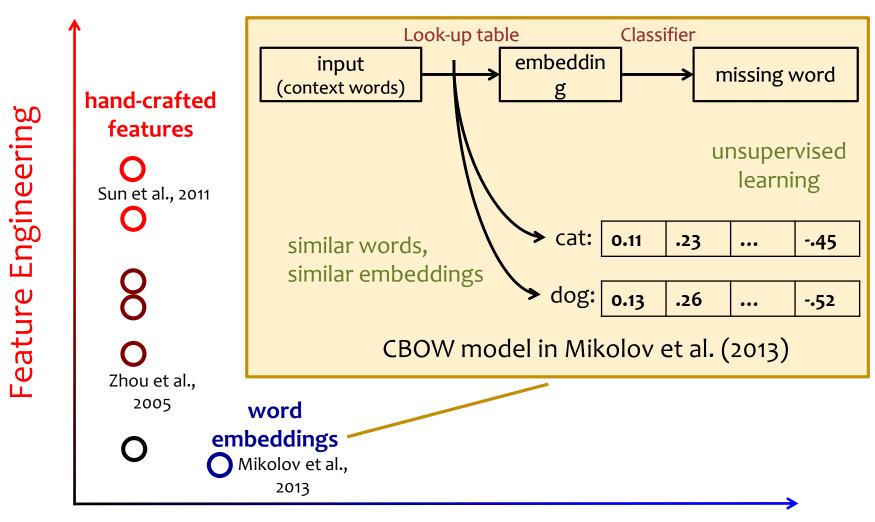
Handcrafted Features



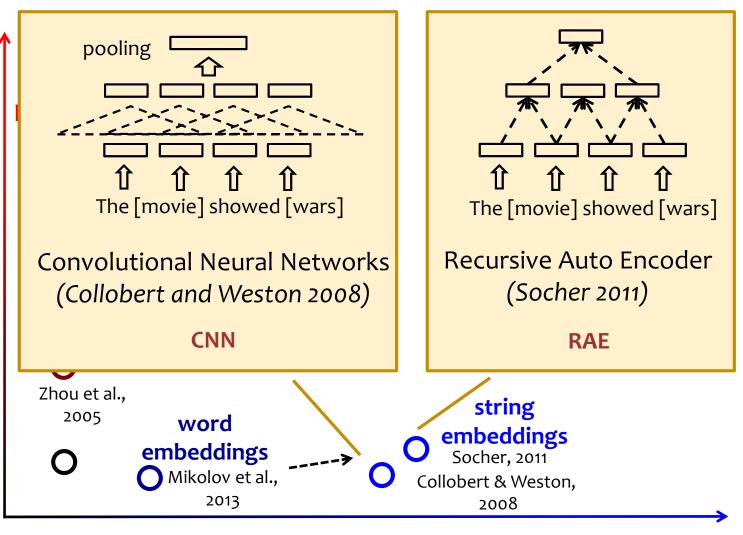
Feature Engineering



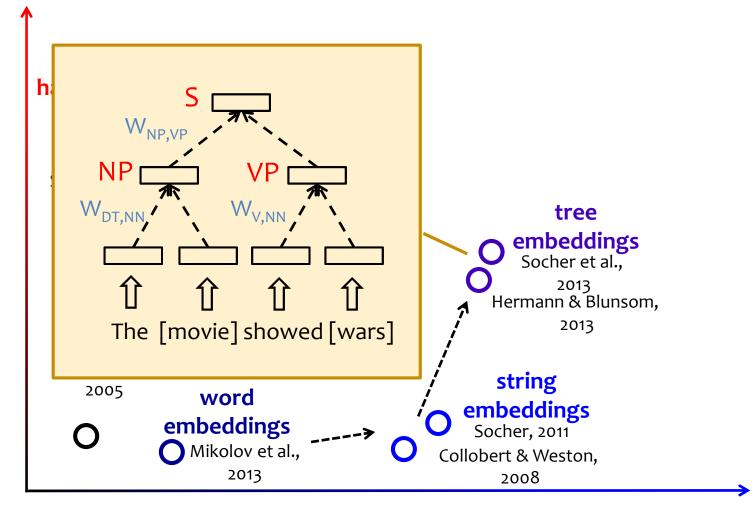
Feature Learning



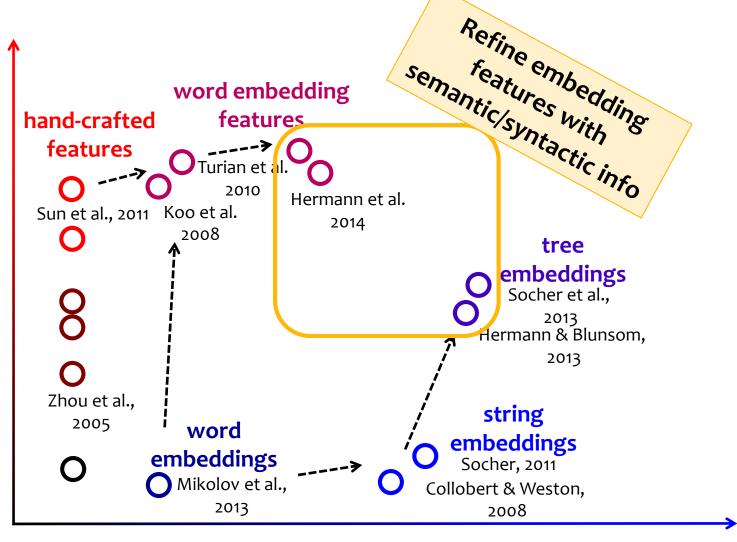
Feature Learning



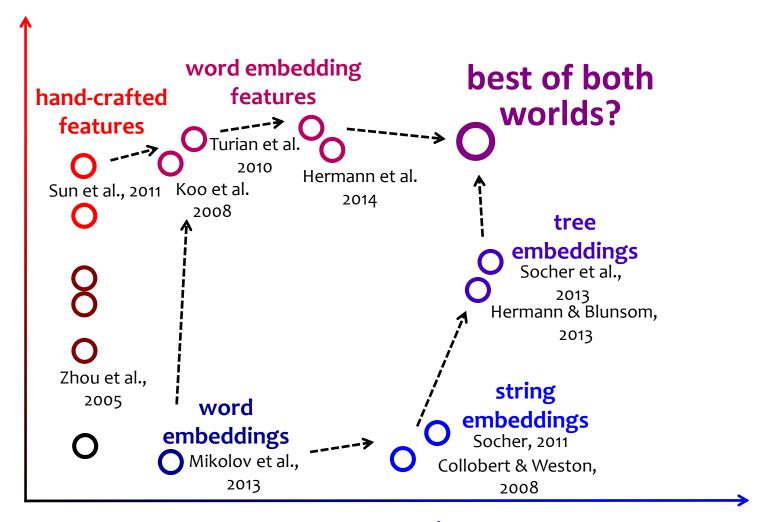
Feature Learning



Feature Learning



Feature Learning



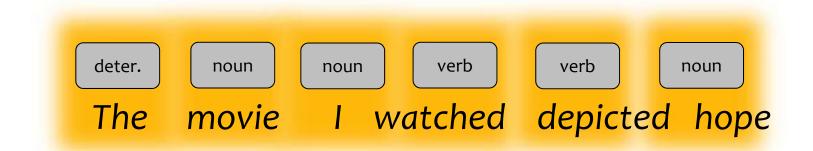
Feature Learning

Suppose you build a logistic regression model to predict a part-of-speech (POS) tag for each word in a sentence.

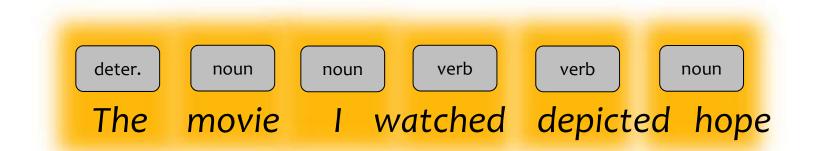
What features should you use?



Per-word Features:



Context Features:



Context Features:

$$w_{i} == "I"$$
 $w_{i+1} == "I"$
 $w_{i-1} == "I"$
 $w_{i+2} == "I"$
 $w_{i-2} == "I"$

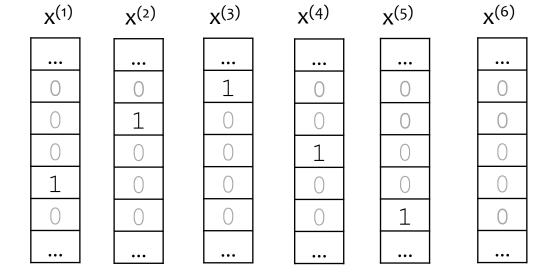
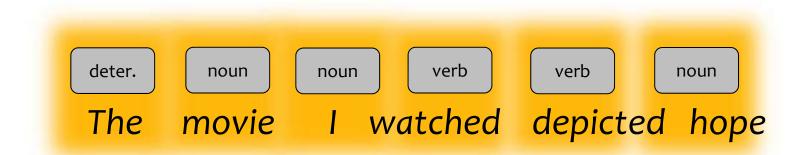


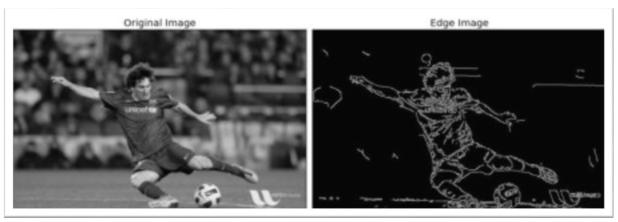


Table 3. Tagging accuracies with different feature templates and other changes on the WSJ 19-21 development set.

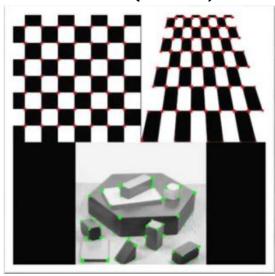
M odel	Feature Templates	#	Sent.	Token	Unk.
		Feats	A cc.	A.cc.	A cc.
3gr amM emm	See text	248,798	52.07%	96.92%	88.99%
naacl 2003	See text and [1]	460,552	55.31%	97.15%	88.61%
Replication	See text and [1]	460,551	55.62%	97.18%	88.92%
Replication '	+rareFeatureThresh = 5	482,364	55.67%	97.19%	88.96%
5w	+ \Box ₁₀ , w ₋₂ \Box , \Box ₁₀ , w ₂ \Box	730,178	56.23%	97.20%	89.03%
5wShapes	+ \square_0 , s ₋₁ \square , \square_0 , s ₀ \square , \square_0 , s ₊₁		56.52%	97.25%	89.81%
5w ShapesDS	+ distributional similarity	737,955	56.79%	97.28%	90.46%



Edge detection (Canny)



Corner Detection (Harris)



Scale Invariant Feature Transform (SIFT)



Figure 3: Model images of planar objects are shown in the top row. Recognition results below show model outlines and image keys used for matching.

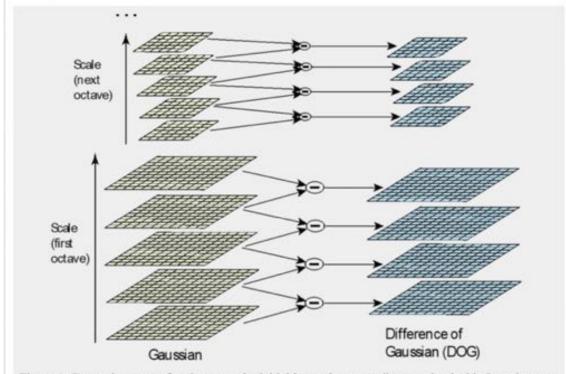


Figure 1: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated.

NON-LINEAR FEATURES

Nonlinear Features

- aka. "nonlinear basis functions"
- So far, input was always $\mathbf{x} = [x_1, \dots, x_M]$
- **Key Idea:** let input be some function of **x**
 - original input: $\mathbf{x} \in \mathbb{R}^M$ where M' > M (usually) new input: $\mathbf{x}' \in \mathbb{R}^{M'}$

 - define $\mathbf{x}' = b(\mathbf{x}) = [b_1(\mathbf{x}), b_2(\mathbf{x}), \dots, b_{M'}(\mathbf{x})]$ where $b_i: \mathbb{R}^M \to \mathbb{R}$ is any function
- Examples: (M = 1)

$$b_j(x) = x^j \quad \forall j \in \{1, \dots, J\}$$

radial basis function

$$b_j(x) = \exp\left(\frac{-(x - \mu_j)^2}{2\sigma_j^2}\right)$$

sigmoid

$$b_j(x) = \frac{1}{1 + \exp(-\omega_j x)}$$

log

$$b_j(x) = \log(x)$$

For a linear model: still a linear function of b(x) even though a nonlinear function of

X

Examples:

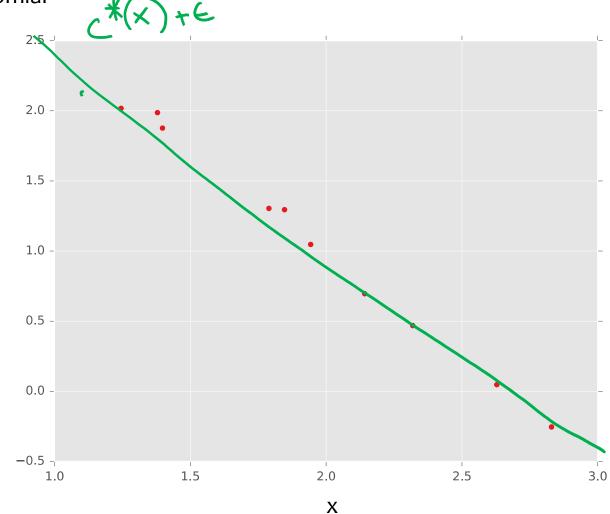
- Perceptron
- Linear regression
- Logistic regression

Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + \mathbf{b}$

where f(.) is a polynomial

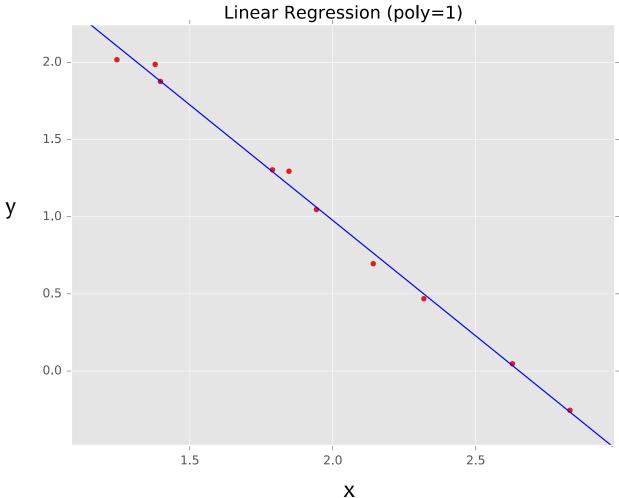
basis function

у	х
2.0	1.2
1.3	1.7
0.1	2.7
1.1	1.9



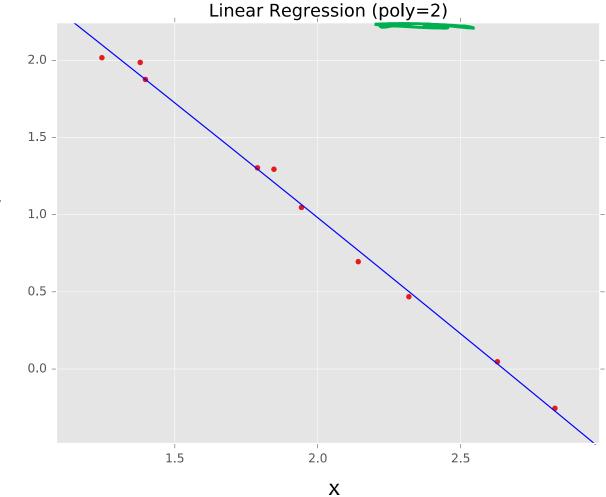
Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + \mathbf{b}$ where f(.) is a polynomial basis function

у	х
2.0	1.2
1.3	1.7
0.1	2.7
1.1	1.9



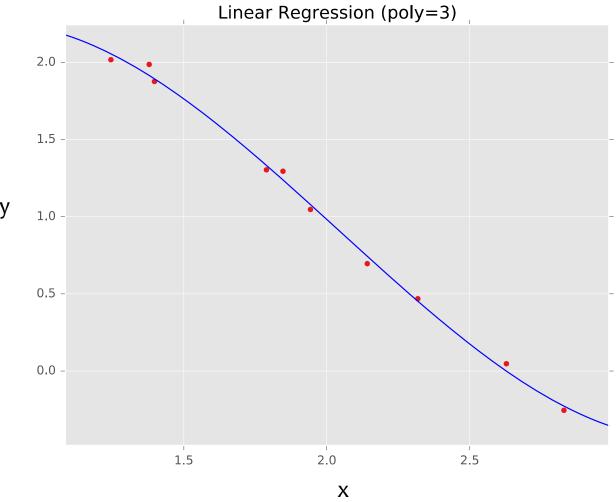
Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + \mathbf{b}$ where f(.) is a polynomial basis function

у	х	X ²	
2.0	1.2	(1.2)2	
1.3	1.7	(1.7)2	
0.1	2.7	(2.7)2	,
1.1	1.9	(1.9)2	
		~	



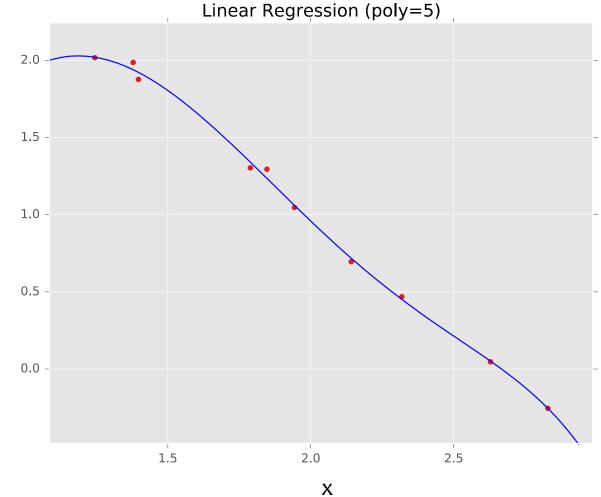
Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + \mathbf{b}$ where f(.) is a polynomial basis function

.,	v	X ²	x ³
У	Х	X-	X
2.0	1.2	(1.2)2	(1.2)3
1.3	1.7	(1.7)2	(1.7) ³
0.1	2.7	(2.7)2	(2.7) ³
1.1	1.9	(1.9)2	(1.9)3



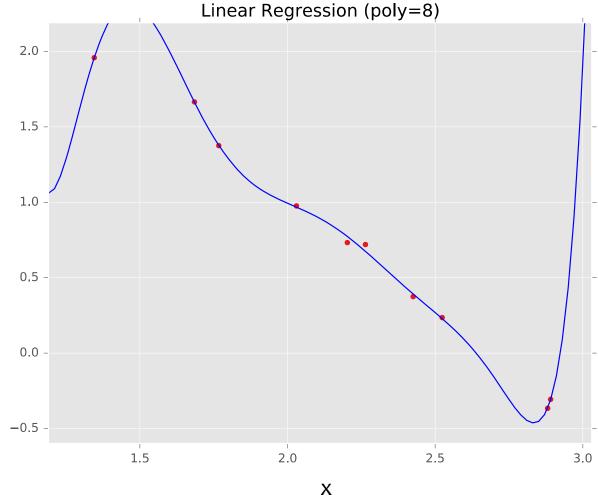
Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + \mathbf{b}$ where f(.) is a polynomial basis function

у	х	X ²	•••	x ⁵	
2.0	1.2	(1.2)2	•••	(1.2)5	
1.3	1.7	(1.7)2		(1.7)5	
0.1	2.7	(2.7)2		(2.7)5	y
1.1	1.9	(1.9)2		(1.9)5	-



Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + \mathbf{b}$ where f(.) is a polynomial basis function

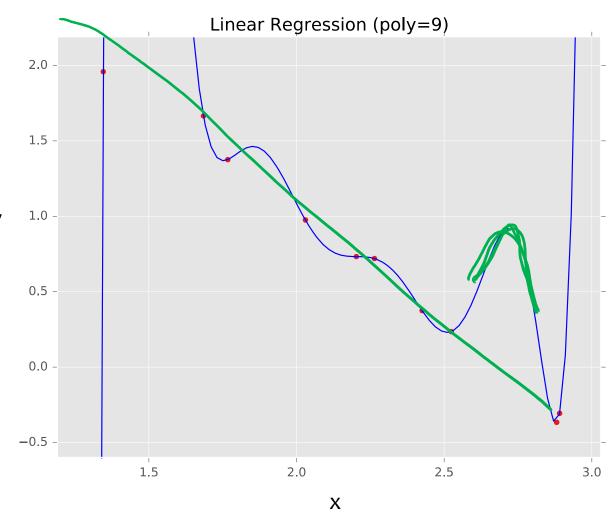
у	х	X ²	•••	x ⁸	
2.0	1.2	(1.2)2		(1.2)8	
1.3	1.7	(1.7)2	•••	(1.7)8	
0.1	2.7	(2.7)2	•••	(2.7)8	y
1.1	1.9	(1.9)2		(1.9)8	



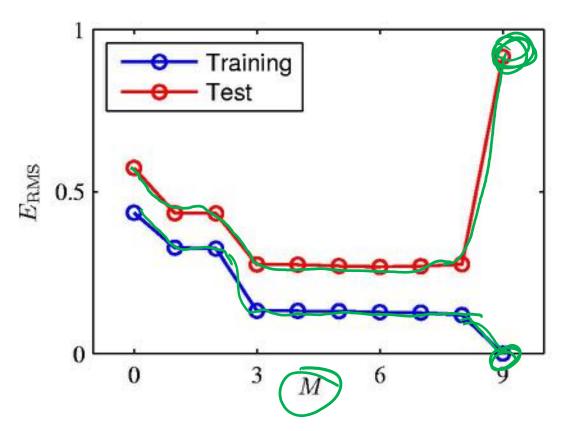
Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + b$ where f(.) is a polynomial

basis function

у	х	X ²		x ⁹	
2.0	1.2	(1.2)2		(1.2)9	
1.3	1.7	(1.7)2		(1.7)9	
0.1	2.7	(2.7)2	•••	(2.7)9	y
1.1	1.9	(1.9)2		(1.9)9	



Over-fitting



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

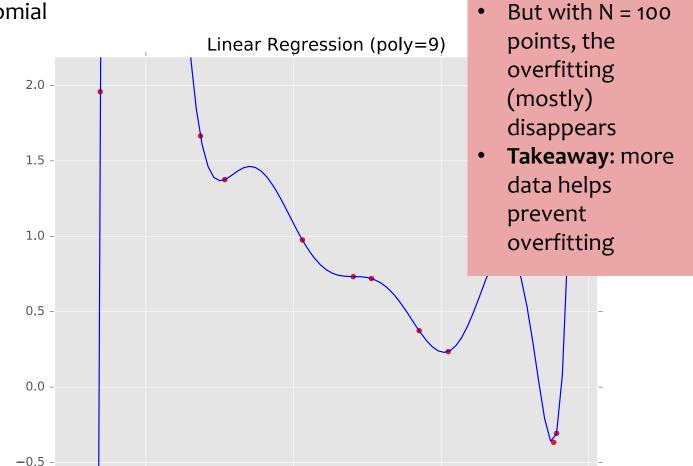
$$E_{\rm RMS} = \sqrt{2E(\mathbf{w}^{\star})/N}$$

Polynomial Coefficients

	M=0	M = 1	M = 3	M = 9
$\overline{\theta_0}$	0.19	0.82	0.31	0.35
$ heta_1$		-1.27	7.99	232.37
$ heta_2$			-25.43	-5321.83
$ heta_3$			17.37	48568.31
$ heta_4$				-231639.30
$ heta_5$				640042.26
$ heta_6$				-1061800.52
$ heta_7$				1042400.18
$ heta_8$				-557682.99
$ heta_9$				125201.43

Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + b$ where f(.) is a polynomial basis function

i	у	х		x ⁹	
1	2.0	1.2		(1.2)9	
2	1.3	1.7	•••	(1.7)9	
			•••		У
10	1.1	1.9	•••	(1.9)9	



2.5

2.0

Χ

1.5

3.0

With just N = 10

points we overfit!

Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + \mathbf{b}$ where f(.) is a polynomial basis function

2.5

2.0

1.5

1.0

0.5

0.0

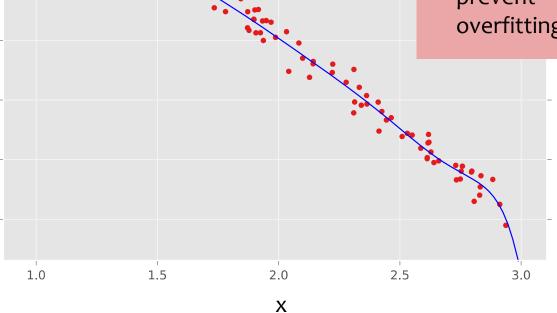
-0.5

i	у	х	•••	x ⁹	
1	2.0	1.2	•••	(1.2)9	
2	1.3	1.7		(1.7)9	
3	0.1	2.7	•••	(2.7)9	у
4	1.1	1.9	•••	(1.9)9	
•••	•••	•••	•••		
•••	•••	•••	•••		
•••	•••	•••	•••	•••	
98	•••	•••	•••		
99	•••	•••	•••	•••	
100	0.9	1.5	•••	(1.5)9	



- With just N = 10 points we overfit!
- But with N = 100
 points, the
 overfitting
 (mostly)
 disappears
- Takeaway: more data helps prevent overfitting

41



REGULARIZATION

Overfitting

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

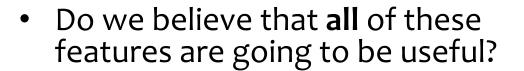
Overfitting can occur in all the models we've seen so far:

- Decision Trees (e.g. when tree is too deep)
- KNN (e.g. when k is small)
- Perceptron (e.g. when sample isn't representative)
- Linear Regression (e.g. with nonlinear features)
- 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





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)
 - small number of "important" features (shrinkage)

$$\overrightarrow{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \overrightarrow{\triangle} = \begin{bmatrix} 00 \\ 0 \\ -200 \end{bmatrix}$$

$$\overrightarrow{X} = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} \qquad \overrightarrow{\triangle} = \begin{bmatrix} 00 \\ -200 \end{bmatrix}$$

Regularization

- **Given** objective function: $J(\theta)$
- Goal is to find: $\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \underbrace{J(\theta)}_{\text{fit}} + \lambda \underline{r(\theta)}_{\text{combat}}$
- **Key idea:** Define regularizer $r(\theta)$ s.t. we tradeoff between fitting the data and keeping the model simple
- Choose form of $r(\theta)$:

 Example: q-norm (usually p-norm) $r(\theta) = ||\theta||_q = \left[\sum_{m=1}^M ||\theta_m||^q\right]^{(\frac{1}{q})}$

V				
J Ge q	$r(oldsymbol{ heta})$	yields parame- ters that are	name	optimization notes
90	$ \boldsymbol{\theta} _0 = \sum \mathbb{1}(\theta_m \neq 0)$	zero values	Lo reg.	no good computa- tional solutions
1 2	$ oldsymbol{ heta} _1 = \sum heta_m \ (oldsymbol{ heta} _2)^2 = \sum heta_m^2$	zero values small values	L1 reg. L2 reg.	subdifferentiable SGD differentiable

Plun Course . erg Regularization

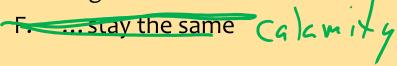
Question:

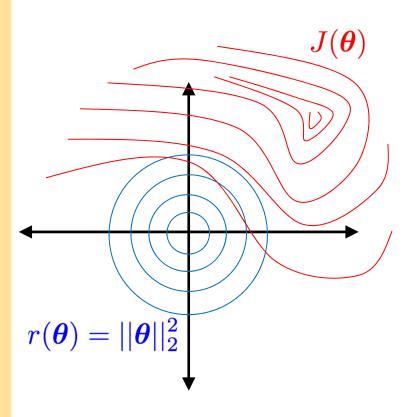
Suppose we are minimizing $J'(\theta)$ where

$$J'(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \lambda r(\boldsymbol{\theta})$$

As λ increases, the minimum of J'(θ) will...

- A. ... move towards the midpoint between $J'(\theta)$ and $r(\theta)$
- B. ... move towards the minimum of $J(\theta)$
- C. ... move towards the minimum of $r(\theta)$
- 69%
- D. ... move towards a theta vector of positive infinities
- E. ... move towards a theta vector of negative infinities

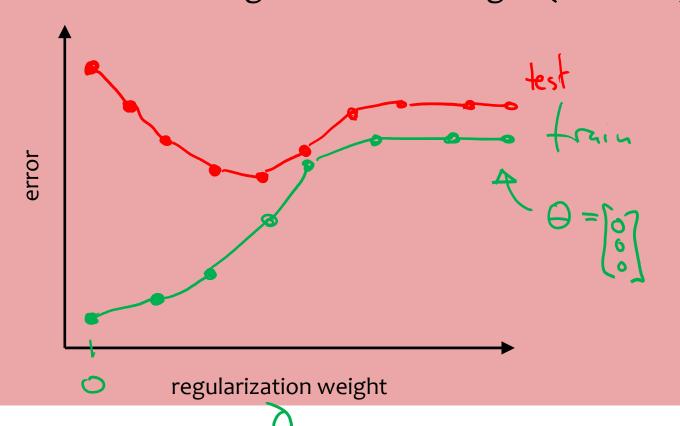




Regularization Exercise

In-class Exercise

- 1. Plot train error vs. regularization weight (cartoon)
- 2. Plot test error vs . regularization weight (cartoon)



Regularization

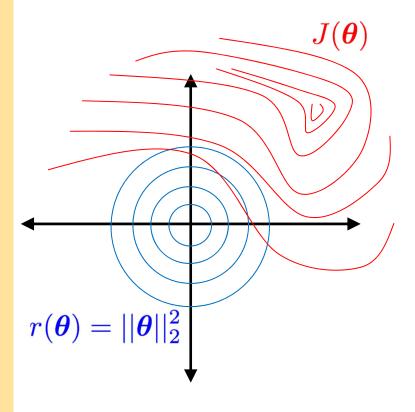
Question:

Suppose we are minimizing $J'(\theta)$ where

$$J'(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \lambda r(\boldsymbol{\theta})$$

As we increase λ from 0, the the validation error will...

- A. ...increase
- B. ... decrease
- C. ... first increase, then decrease
- D. ... first decrease, then increase
- E. ... stay the same Calamity



Regularization

Don't Regularize the Bias (Intercept) Parameter!

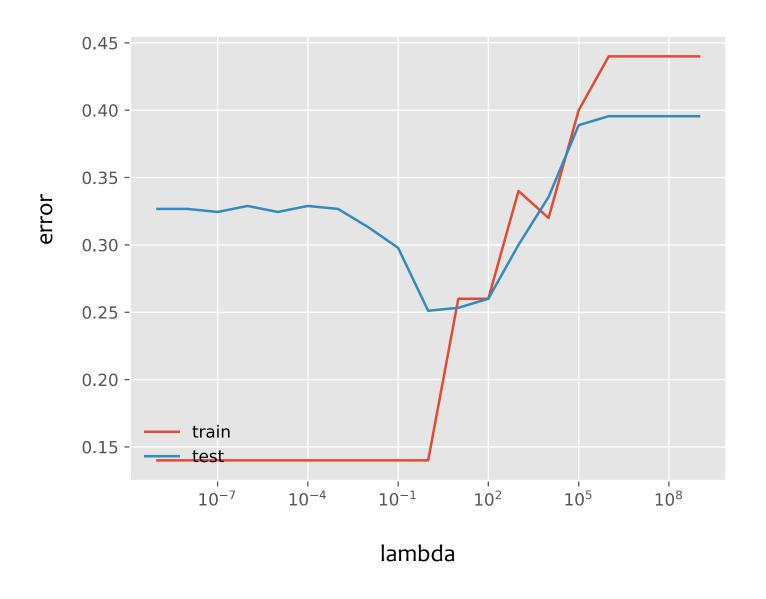
- In our models so far, the bias / intercept parameter is usually denoted by θ_0 -- that is, the parameter for which we fixed $x_0=1$
- Regularizers always avoid penalizing this bias / intercept parameter
- Why? Because otherwise the learning algorithms wouldn't be invariant to a shift in the y-values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

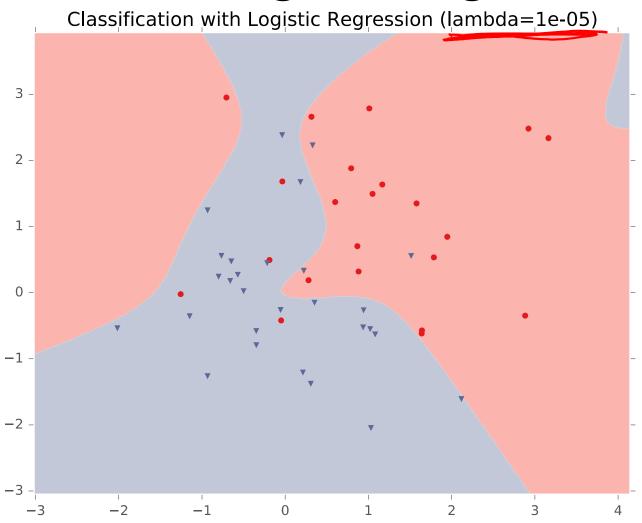
Whitening Data

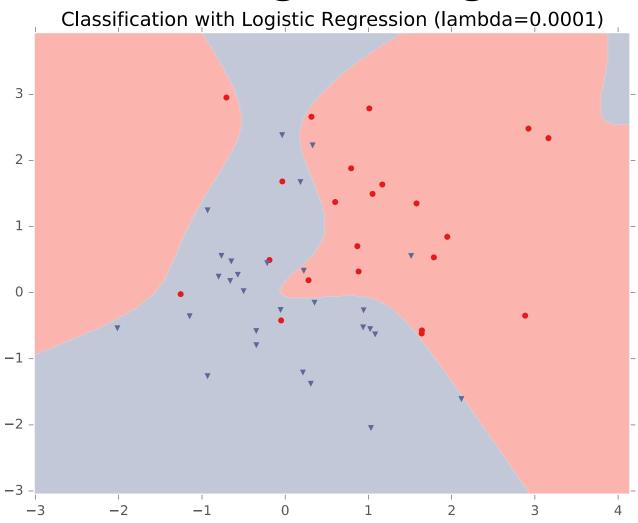
- It's common to whiten each feature by subtracting its mean and dividing by its variance
- For regularization, this helps all the features be penalized in the same units (e.g. convert both centimeters and kilometers to z-scores)

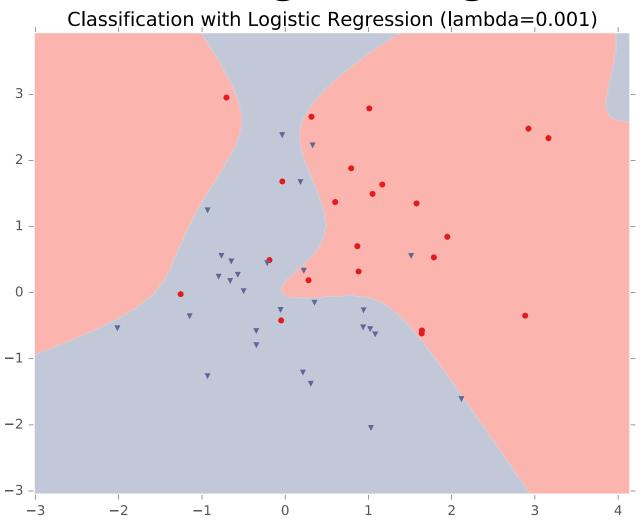


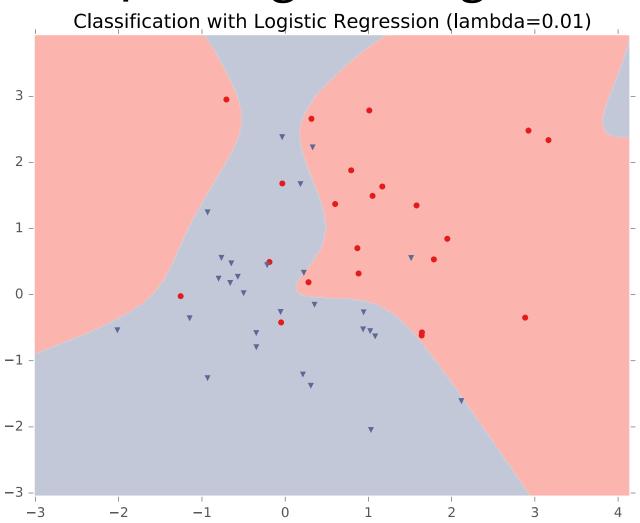
- For this example, we construct nonlinear features (i.e. feature engineering)
- Specifically, we add
 polynomials up to order 9 of
 the two original features x₁
 and x₂
- Thus our classifier is linear in the high-dimensional feature space, but the decision boundary is nonlinear when visualized in low-dimensions (i.e. the original two dimensions)

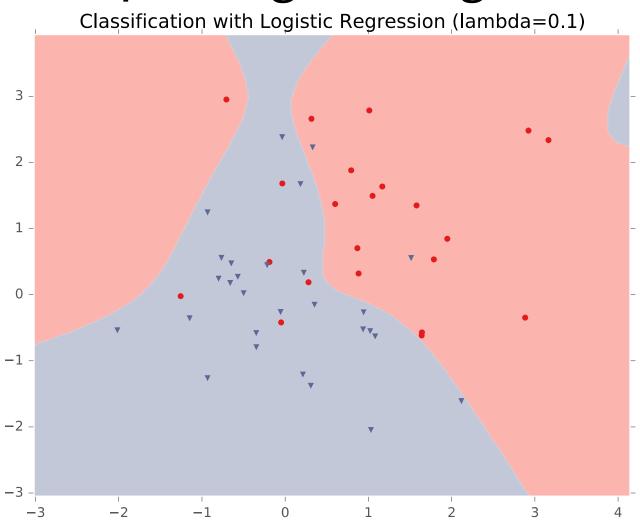


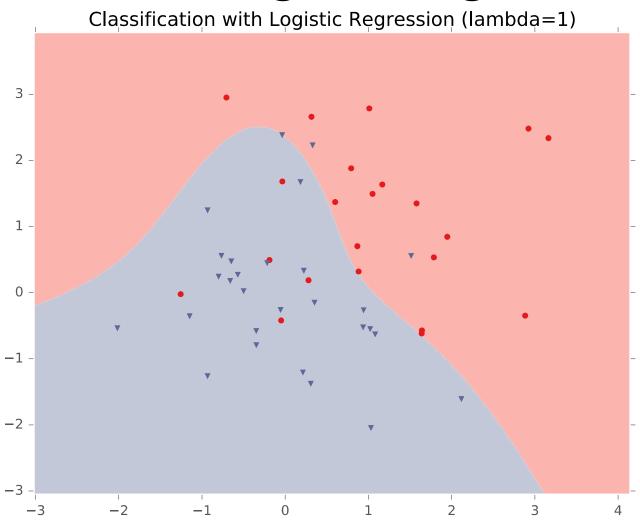


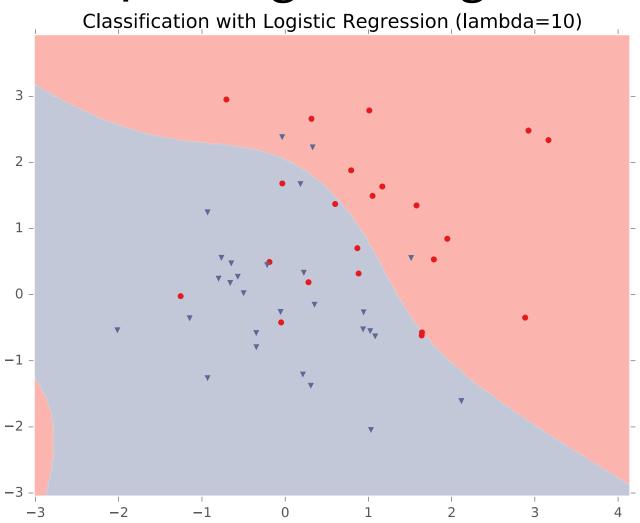


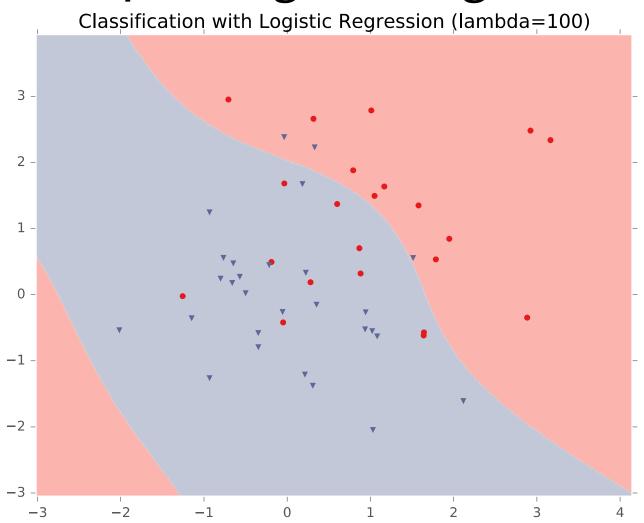


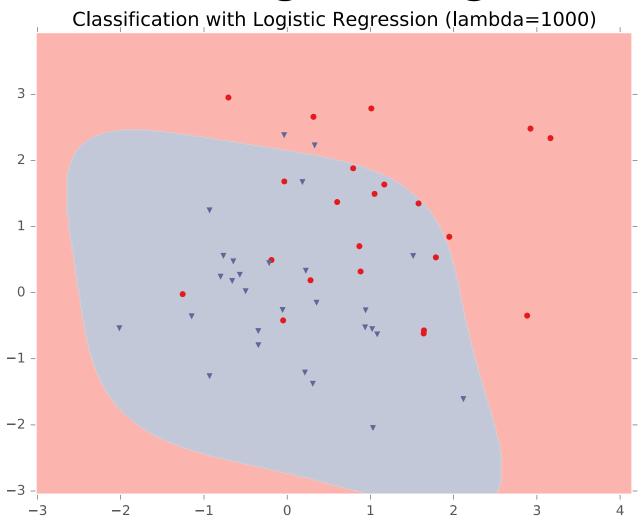


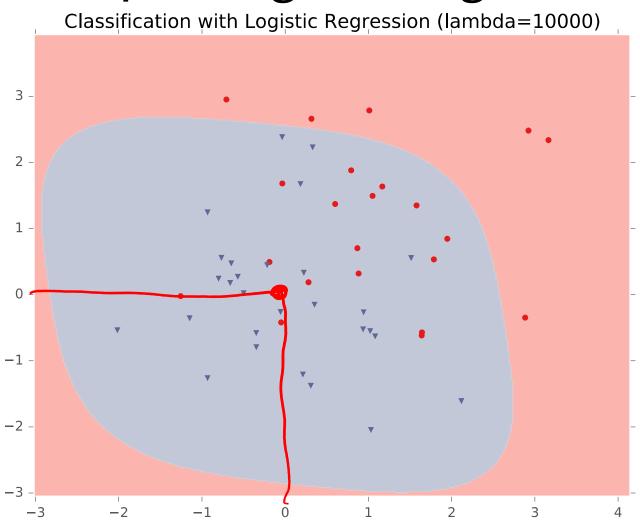


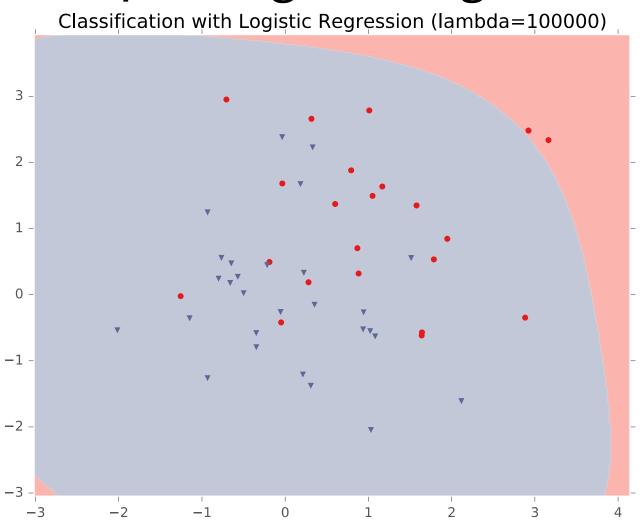


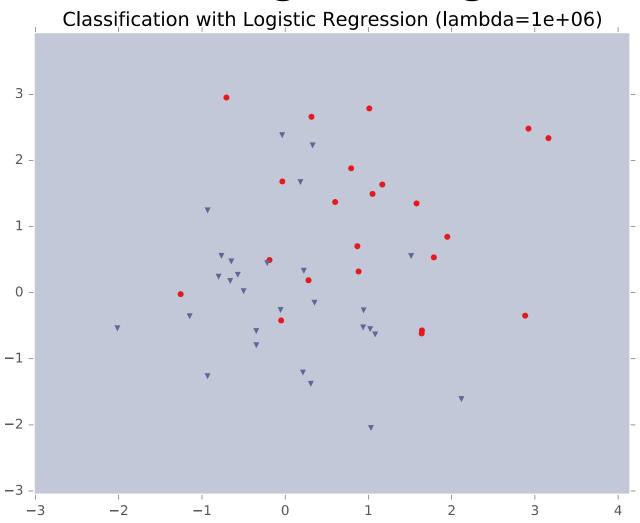


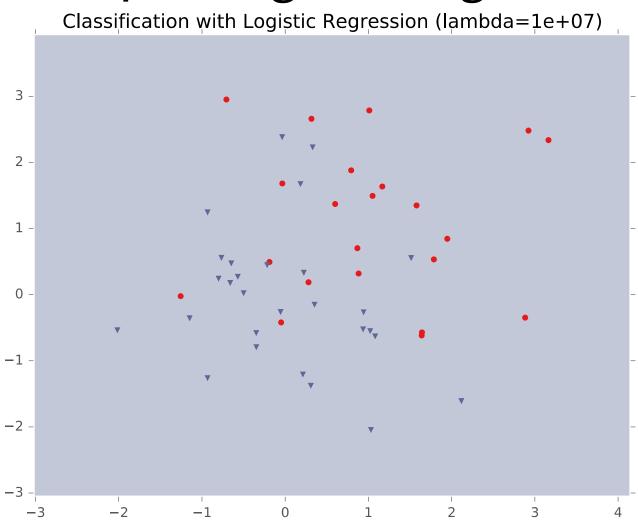


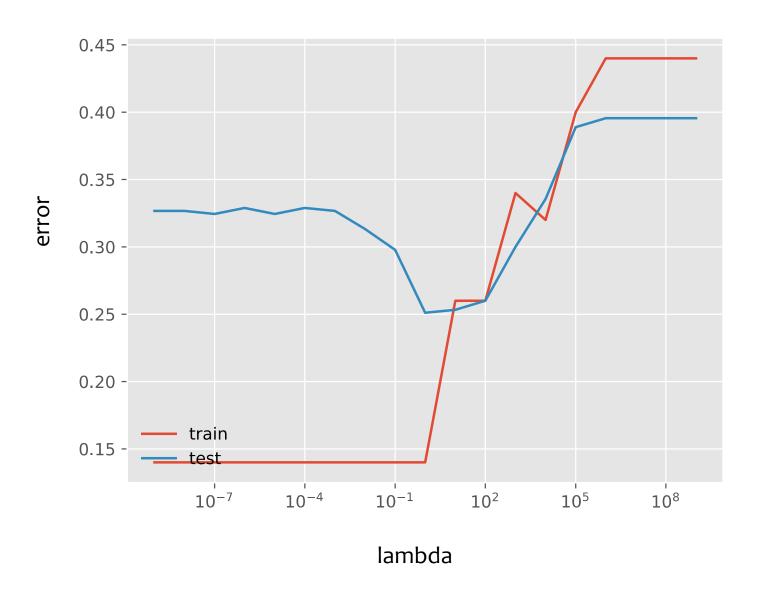












Regularization as MAP

- L1 and L2 regularization can be interpreted as maximum a-posteriori (MAP) estimation of the parameters
- To be discussed later in the course...

Takeaways

- 1. Nonlinear basis functions allow linear models (e.g. Linear Regression, Logistic Regression) to capture nonlinear aspects of the original input
- 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

Feature Engineering / Regularization Objectives

You should be able to...

- Engineer appropriate features for a new task
- Use feature selection techniques to identify and remove irrelevant features
- Identify when a model is overfitting
- Add a regularizer to an existing objective in order to combat overfitting
- Explain why we should not regularize the bias term
- Convert linearly inseparable dataset to a linearly separable dataset in higher dimensions
- Describe feature engineering in common application areas

Neural Networks Outline

Logistic Regression (Recap)

Data, Model, Learning, Prediction

Neural Networks

- A Recipe for Machine Learning
- Visual Notation for Neural Networks
- Example: Logistic Regression Output Surface
- 2-Layer Neural Network
- 3-Layer Neural Network

Neural Net Architectures

- Objective Functions
- Activation Functions

Backpropagation

- Basic Chain Rule (of calculus)
- Chain Rule for Arbitrary Computation Graph
- Backpropagation Algorithm
- Module-based Automatic Differentiation (Autodiff)

NEURAL NETWORKS

Background

A Recipe for Machine Learning

1. Given training data:

$$\{oldsymbol{x}_i, oldsymbol{y}_i\}_{i=1}^N$$

- 2. Choose each of these:
 - Decision function

$$\hat{\boldsymbol{y}} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

Loss function

$$\ell(\hat{oldsymbol{y}}, oldsymbol{y}_i) \in \mathbb{R}$$



Examples: Linear regression, Logistic regression, Neural Network

Examples: Mean-squared error, Cross Entropy

Background

A Recipe for Machine Learning

1. Given training data:

$$\{oldsymbol{x}_i, oldsymbol{y}_i\}_{i=1}^N$$

- 2. Choose each of these:
 - Decision function

$$\hat{m{y}} = f_{m{ heta}}(m{x}_i)$$

Loss function

$$\ell(\hat{oldsymbol{y}}, oldsymbol{y}_i) \in \mathbb{R}$$

3. Define goal: empirical risk $oldsymbol{ heta}^* = rg \min_{oldsymbol{ heta}} \sum_{i=1}^N \ell(f_{oldsymbol{ heta}}(oldsymbol{x}_i), oldsymbol{y}_i)$

4. Train with SGD:(take small steps opposite the gradient)

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

Background

A Recipe for Gradients

1. Given training dat

$$\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$$

- 2. Choose each of the
 - Decision function

$$\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

Loss function

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

Backpropagation can compute this gradient!

And it's a special case of a more general algorithm called reversemode automatic differentiation that can compute the gradient of any differentiable function efficiently!

opposite the gradient) $heta^{(t)} - \eta_t
abla \ell(f_{m{ heta}}(m{x}_i), m{y}_i)$

 $oldsymbol{y}_i)$

A Recipe for

Goals for Today's Lecture

- 1. Explore a **new class of decision functions** (Neural Networks)
 - 2. Consider variants of this recipe for training

 (\boldsymbol{y}_i)

2. enoose each of these.

- Decision function

$$\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

- Loss function

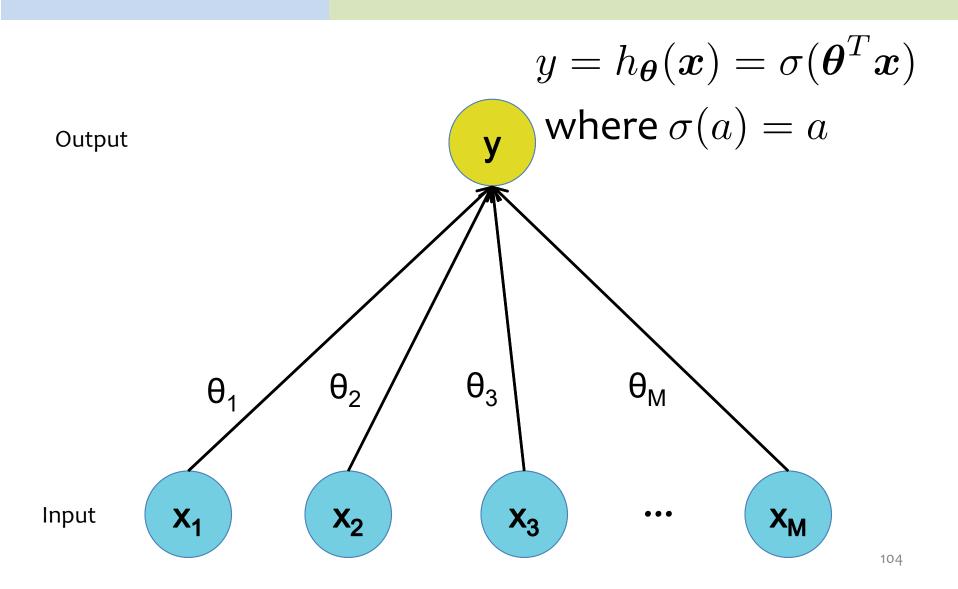
$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

Train with SGD:

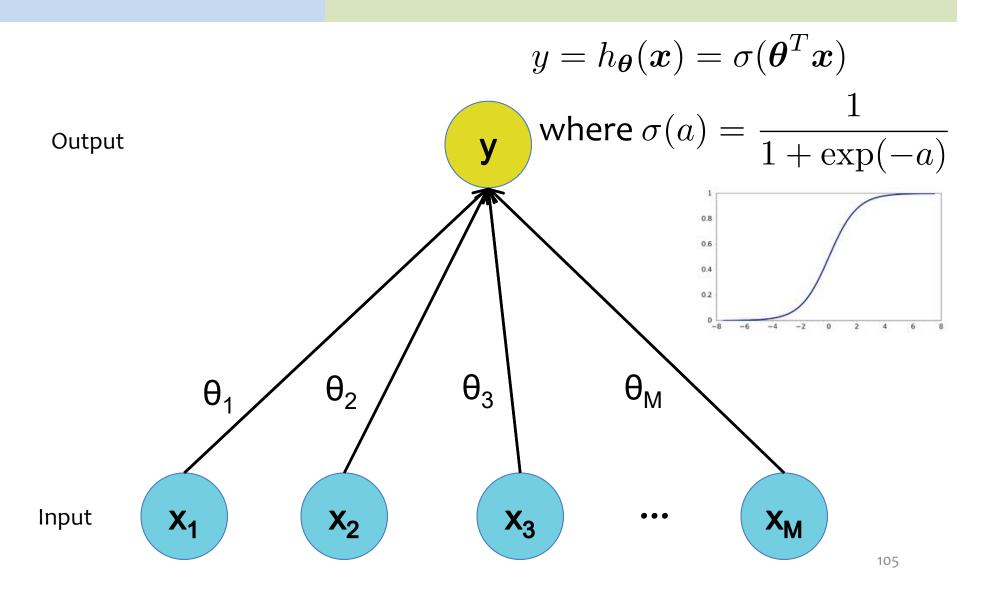
ke small steps
opposite the gradient)

$$oldsymbol{ heta}^{(t+1)} = oldsymbol{ heta}^{(t)} - \eta_t
abla \ell(f_{oldsymbol{ heta}}(oldsymbol{x}_i), oldsymbol{y}_i)$$

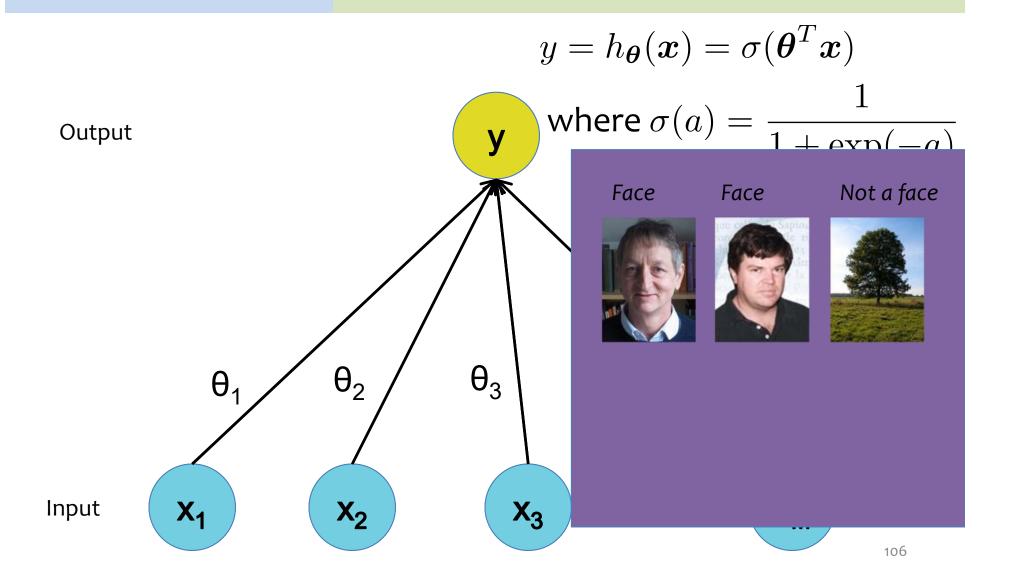
Linear Regression



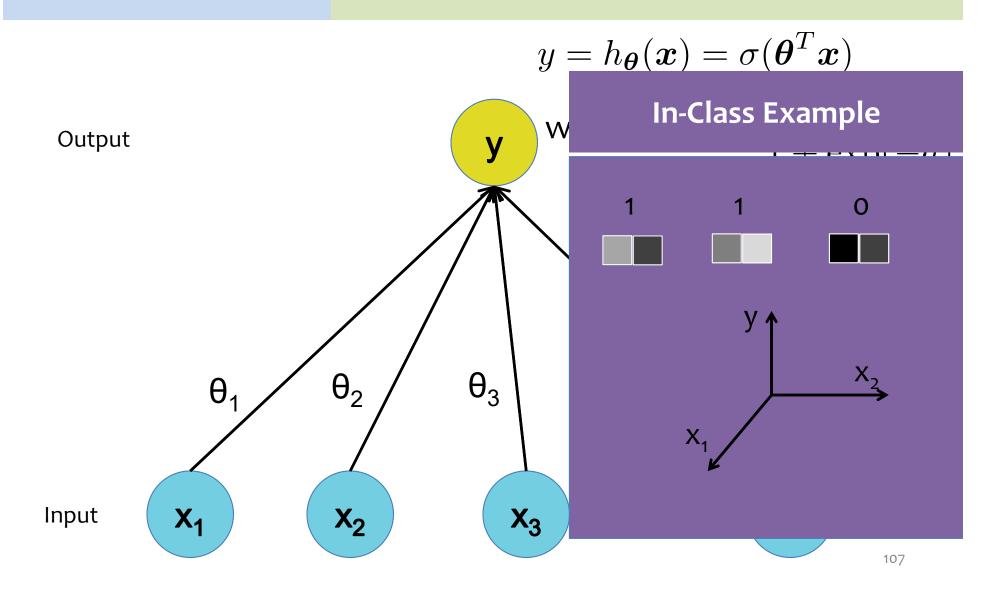
Logistic Regression



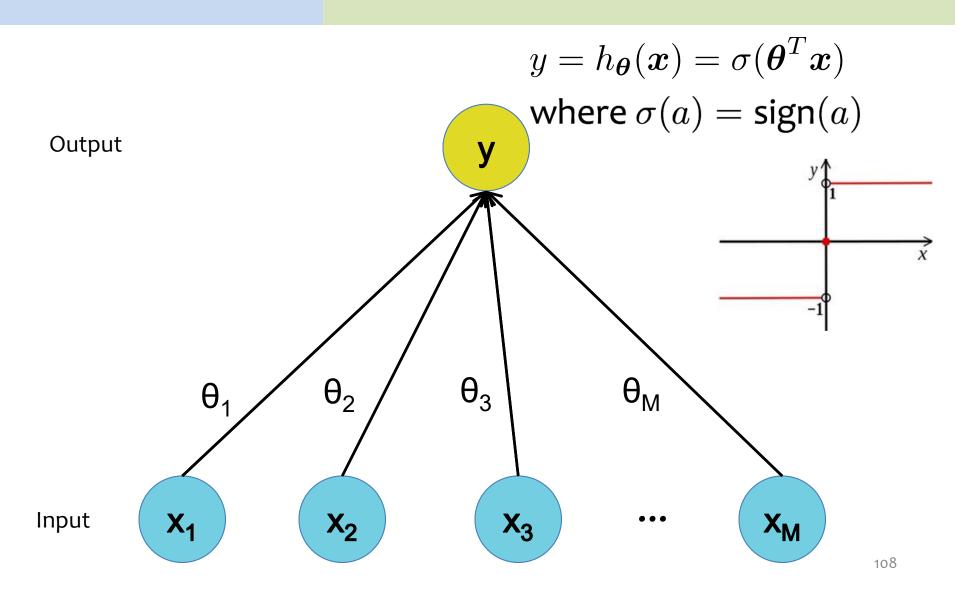
Logistic Regression



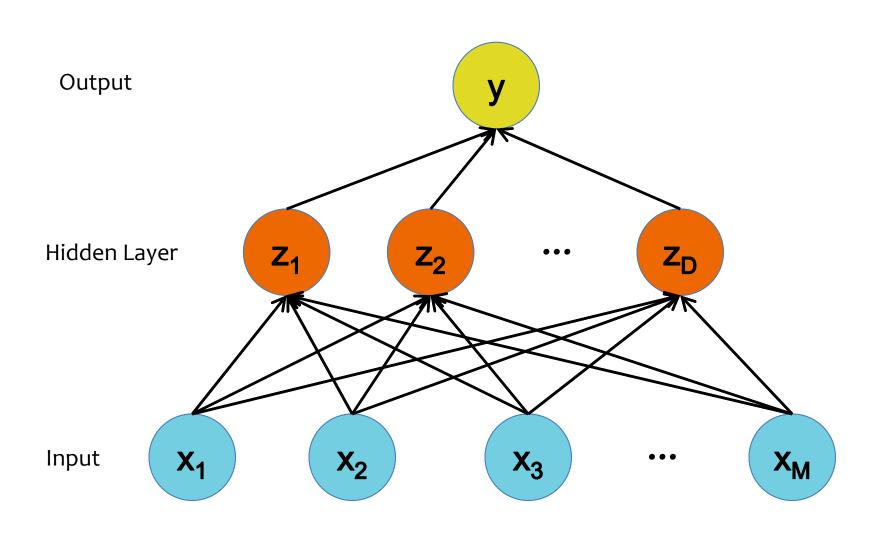
Logistic Regression



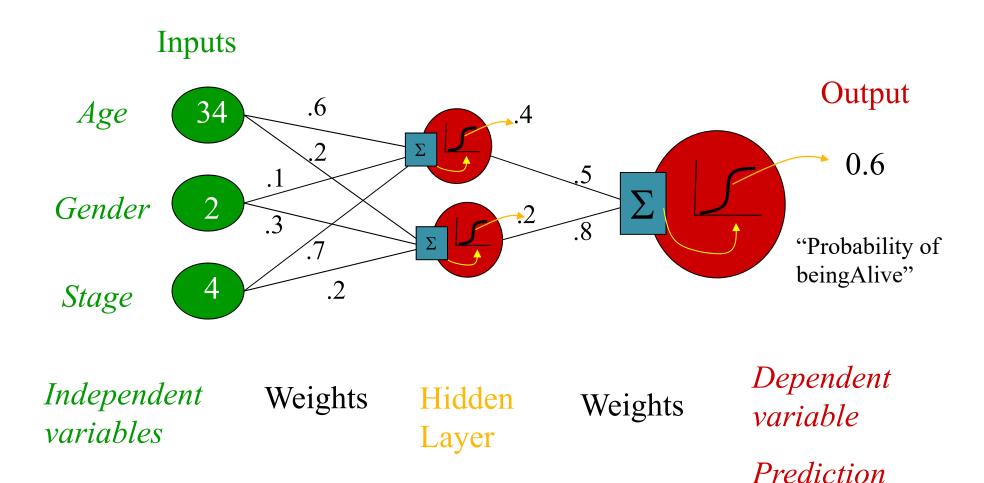
Perceptron



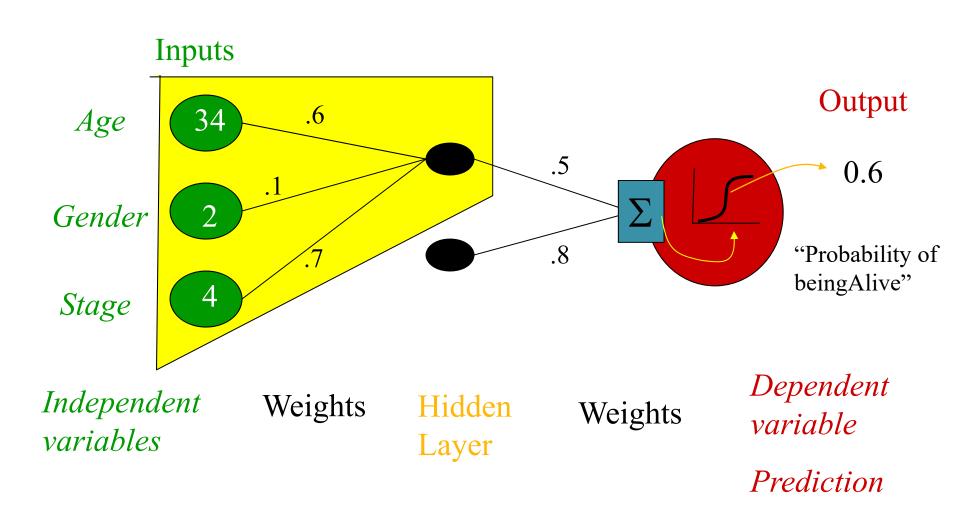
Neural Network

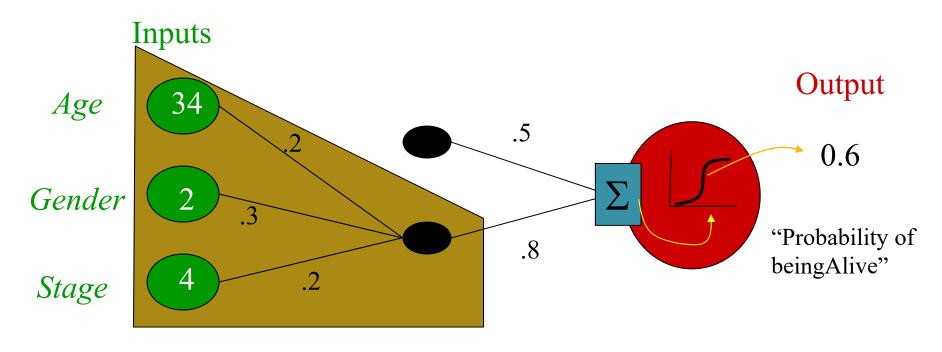


Neural Network Model



"Combined logistic models"





Independent variables

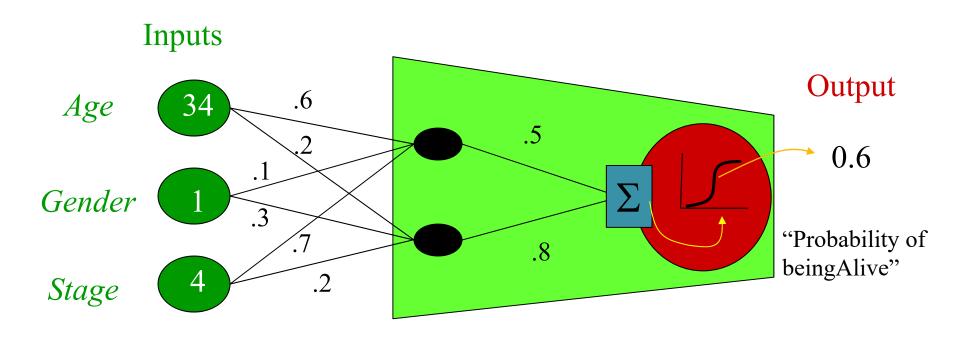
Weights

Hidden Layer

Weights

Dependent variable

Prediction



Independent variables

Weights

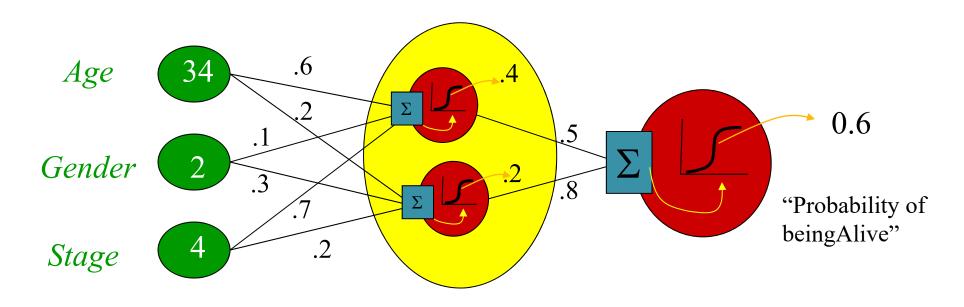
Hidden Layer

Weights

Dependent variable

Prediction

Not really, no target for hidden units...



Independent variables

Weights

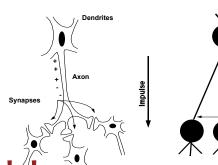
Hidden Layer Weights

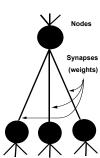
Dependent variable

Prediction

From Biological to Artificial

The motivation for Artificial Neural Networks comes from biology...





Biological "Model"

- Neuron: an excitable cell
- Synapse: connection between neurons
- A neuron sends an electrochemical pulse along its synapses when a sufficient voltage change occurs
- Biological Neural Network: collection of neurons along some pathway through the brain

Artificial Mode

- Neuron: node in a directed acyclic graph (DAG)
- Weight: multiplier on each edge
- Activation Function: nonlinear thresholding function, which allows a neuron to "fire" when the input value is sufficiently high
- Artificial Neural Network: collection of neurons into a DAG, which define some differentiable function

Biological "Computation"

- Neuron switching time: ~ 0.001 sec
- Number of neurons: ~ 10¹⁰
- Connections per neuron: ~ 10⁴⁻⁵
- Scene recognition time: ~ 0.1 sec

Artificial Computation

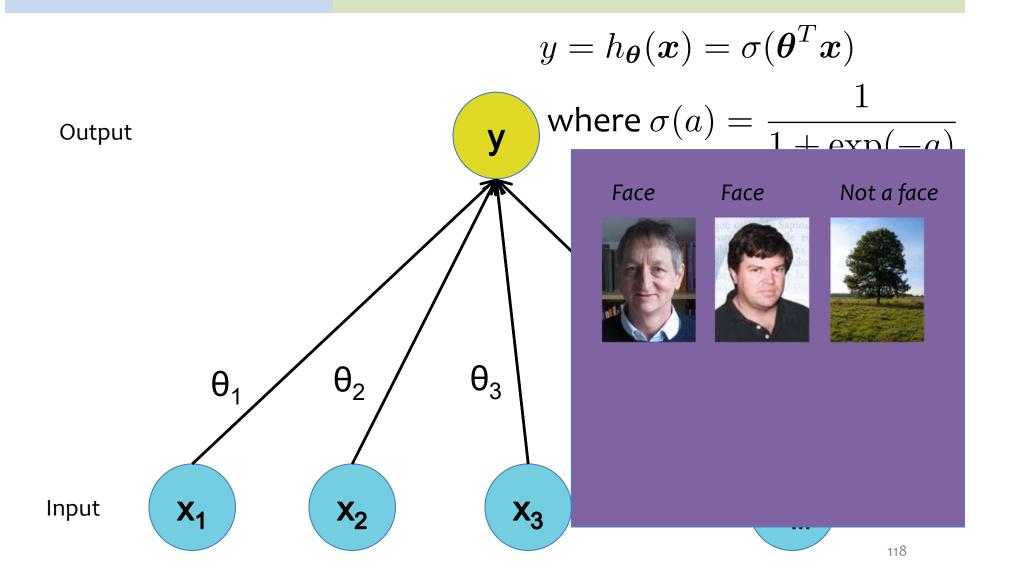
- Many neuron-like threshold switching units
- Many weighted interconnections among units
- Highly parallel, distributed processes

Neural Networks

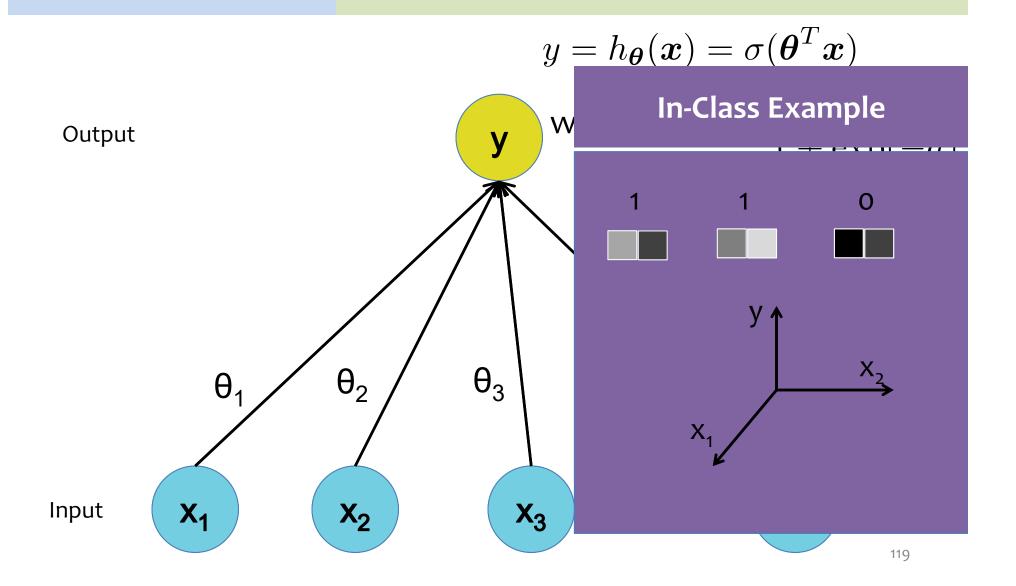
Chalkboard

- Example: Neural Network w/1 Hidden Layer
- Example: Neural Network w/2 Hidden Layers
- Example: Feed Forward Neural Network

Logistic Regression

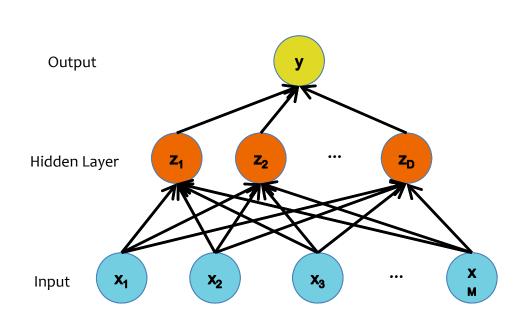


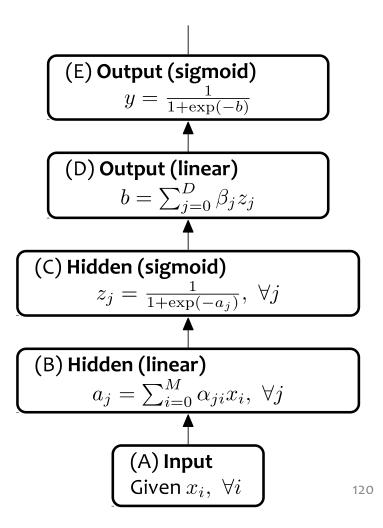
Logistic Regression



Neural Network

Neural Network for Classification





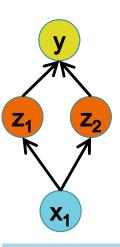
Neural Network Parameters

Question:

Suppose you are training a one-hidden layer neural network with sigmoid activations for binary classification.



True or False: There is a unique set of parameters that maximize the likelihood of the dataset above.



Answer: