CSE 446 Machine Learning

Neural Networks

Daniel Weld

Adapted from slides by Carlos Guestrin & Tom Dietterich

1

Midterm Wed

- Closed book
- No notes
- No calculators
- No Internet ☺

2

Overview of Learning

Type of Supervision
(eg, Experience, Feedback)

What is Being Learned?

	Labeled Examples	Reward	Nothing
Discrete Function	Classification		Clustering
Continuous Function	Regression		
Policy	Apprenticeship Learning	Reinforcement Learning	

3

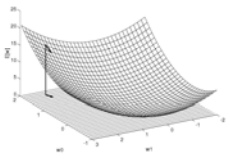
Machine Learning

- Supervised Learning
  - Parametric
    - Y Continuous
      - Gaussians: Learned in closed form
      - Linear Functions: 1. Learned in closed form 2. Using gradient descent
    - Y Discrete
      - Decision Trees: Greedy search; pruning
      - Probability of class | features: 1. Learn P(Y), P(X|Y); apply Bayes 2. Learn P(Y|X) w/ gradient descent
      - Non-probabilistic Linear Classifier: Learn w/ gradient descent
  - Non-parametric
- Unsupervised Learning
- Reinforcement Learning

4

Learning

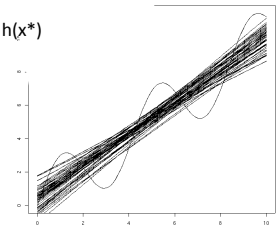
- Learning as function approximation
- Learning as optimization
  - Closed form
  - Greedy search
  - Gradient ascent
- Loss Function
  - Error + regularization



5

Bia / Variance Tradeoff

- Variance: $E[(h(x^*) - \underline{h(x^*)})^2]$
  - How much h(x*) varies between training sets
  - Reducing variance risks underfitting
- Bias: $[\underline{h(x^*)} - f(x^*)]$
  - Describes the ***average*** error of h(x*)
  - Reducing bias risks overfitting



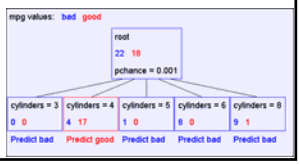
Slide from T Dietterich

Ensembles of Classifiers

- Traditional approach: Use one classifier
- Can one do better?
- Approaches:
  - Cross-validated committees
  - Bagging
  - Boosting
  - Stacking

Decision Trees

- One of the most popular ML tools
  - Easy to understand, implement, and use
  - Computationally cheap (to solve heuristically)
- Information gain to select attributes
  - Gain ratio, threshold splits
- Decision trees very expressive -> will overfit!
  - Must use tricks to find "simple trees", e.g.,
    - Fixed depth/Early stopping
    - Reduced error & chi^2 pruning

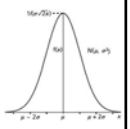


Probabilities

- Marginal, joint, conditional
- Independence & conditional independence
- Bayes Theorem, product rule, sum rule
- Prior, posterior distributions
- MLE, MAP, Bayesian inference

The Distributions We Love

	Discrete		Continuous
	Binary {0, 1}	k Values	
Single Event	Bernouilli		
Sequence (N trials) $N= \alpha_H+\alpha_T$	Binomial $P(\mathcal{D} \mid \theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T}$	Multinomial	
Conjugate Prior	Beta	Dirichlet	

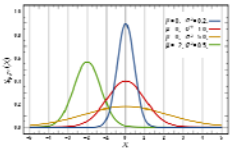


Learning Gaussian Parameters

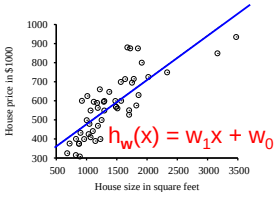
MLE:

$$\hat{\mu}_{MLE} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

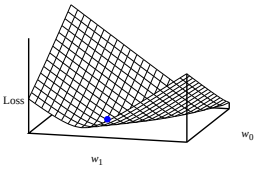
$$\hat{\sigma}^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N} (x_i - \hat{\mu})^2$$



Linear Regression



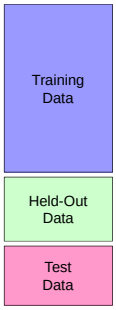
$\text{Argmin}_w \text{Loss}(h_w)$



$$w_1 = \frac{N\sum(x_jy_j)-(\sum x_j)(\sum y_j)}{N\sum(x_j^2)-(\sum x_j)^2}$$

$$w_0 = (\sum(y_j)-w_1(\sum x_j)/N$$

Three Views of Classification



- Naïve Bayes:
  - Parameters from data statistics
  - Parameters: probabilistic interpretation
  - Training: one pass through the data
- Logistic Regression:
  - Parameters from gradient ascent
  - Parameters: linear, probabilistic model, and discriminative
  - Training: one pass through the data per gradient step, use validation to stop
- The Perceptron:
  - Parameters from reactions to mistakes
  - Parameters: discriminative interpretation
  - Training: go through the data until held-out accuracy maxes out

Naïve Bayes *vs.* Logistic Regression

Learning: h:**X** ↦ Y — **X** – features, Y – target classes

Generative

- Assume functional form for
  - P(X|Y) **assume cond indep**
  - P(Y)
  - Est params from train data
- Gaussian NB for cont features
- Bayes rule to calc. P(Y|X= x)
  - P(Y | **X**) ∝ P(**X** | Y) P(Y)
- **Indirect** computation
  - Can also generate a sample of the data

Discriminative

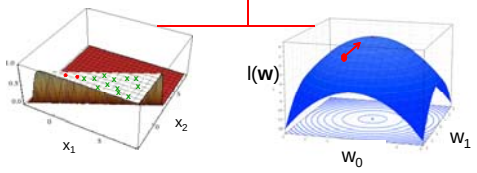
- Assume functional form for
  - P(Y|X) **no assumptions**
  - Est params from training data
- Handles discrete & cont features
- Directly calculate P(Y|X=x)
  - Can't generate data sample

Logistic w/ Initial Weights

w_0=20 w_1= -5 w_2 =10

Loss(H_w) = Error(H_w, data)

Minimize error → Maximize l(**w**) = ln P(D_Y | D_x, H_w)



Update rule:

$$\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$$

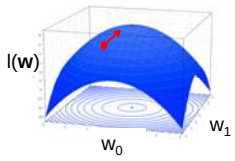
$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{\partial l(\mathbf{w})}{\partial w_i}$$

Details

Update rule:

$$\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{\partial l(\mathbf{w})}{\partial w_i}$$



$$\frac{\partial l(w)}{\partial w_i} = \sum_j x_i^j \left(y^j - P(Y^j = 1 | x^j, w)\right)$$

$$P(Y=1|X,W) = \frac{exp(w_0 + \sum_i w_i X_i)}{1 + exp(w_0 + \sum_i w_i X_i)}$$

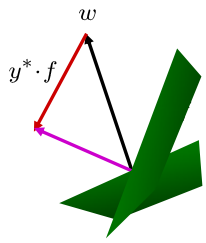
Binary Perceptron Algorithm

- Start with zero weights
- For each training instance (x,y*):
  - Classify with current weights

$$y = \begin{cases} +1 & \text{if } w \cdot f(x) \geq 0 \\ -1 & \text{if } w \cdot f(x) < 0 \end{cases}$$

  - If correct (i.e., y=y*), no change!
  - If wrong: update

$$w = w + y^* \cdot f$$

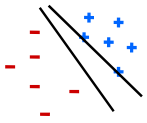


Properties of Perceptrons

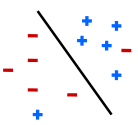
- Separability: some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

$$\text{mistakes} < \frac{k}{\delta^2}$$

Separable

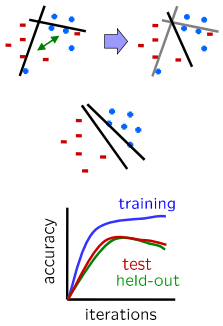


Non-Separable

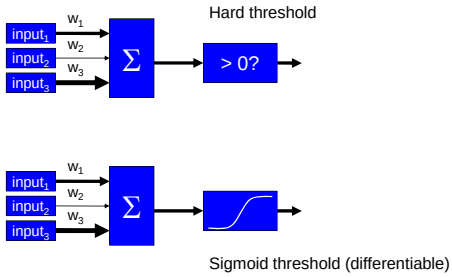


Problems with the Perceptron

- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)
- Mediocre generalization: finds a "barely" separating solution
- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining is a kind of overfitting



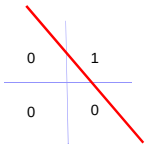
Perceptrons *vs.* Sigmoid Perceptrons



Sigmoid threshold (differentiable)

Percepton, linear classification, Boolean functions

- Can learn $x_1 \wedge x_2$



$f(x)$

```
BIAS  :  1
A     :  1
B     :  1
...
```

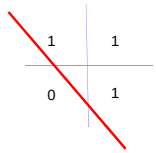
w

```
BIAS  : -0.8
free  :  .5
money :  .5
...
```

$$\sum_i w_i \cdot f_i(x)$$

Percepton, linear classification, Boolean functions

- Can learn $x_1 \wedge x_2$
- Can learn $x_1 \vee x_2$



$f(x)$

```
BIAS  :  1
A     :  1
B     :  1
...
```

w

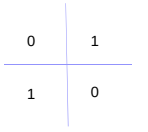
```
BIAS  : -0.3
free  :  .5
money :  .5
...
```

$$\sum_i w_i \cdot f_i(x)$$

- Can learn any conjunction or disjunction

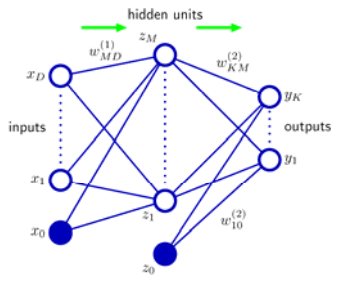
Percepton, linear classification, Boolean functions

- Can learn majority
- Can learn m of n
- Can perceptrons do everything?



Going beyond linear classification

- Solving the XOR problem

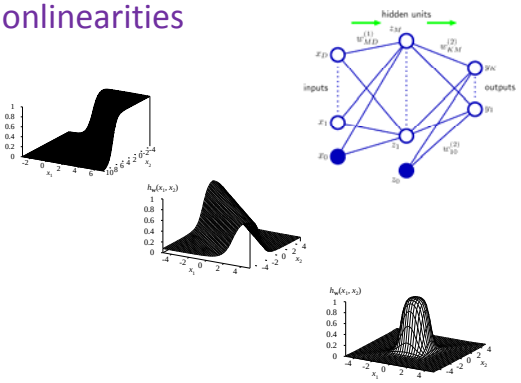


Hidden layer

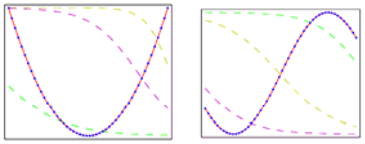
- Perceptron: $out(\mathbf{x}) = g(w_0 + \sum_i w_i x_i)$
- 1-hidden layer:

$$out(\mathbf{x}) = g\left(w_0 + \sum_k w_k g(w_0^k + \sum_i w_i^k x_i)\right)$$

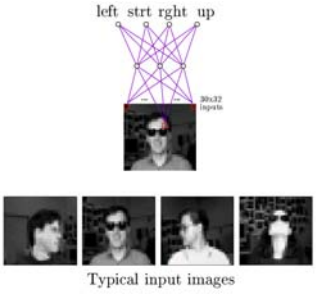
Nonlinearities



Can Approximate Any Continuous F



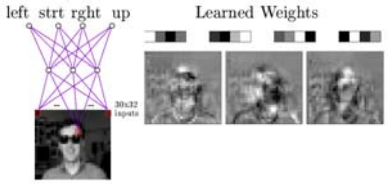
NN for images



Typical input images

90% accurate learning head pose, and recognizing 1-of-20 faces

Weights in NN for images



Typical input images

Forward propagation for 1-hidden layer - Prediction

- 1-hidden layer:

$$out(\mathbf{x}) = g\left(w_0 + \sum_k w_k g(w_0^k + \sum_i w_i^k x_i)\right)$$

Gradient descent for 1-hidden layer Back-propagation: Computing $\frac{\partial \ell(W)}{\partial w_k}$

Dropped w_0 to make derivation simpler

$$\ell(W) = \frac{1}{2}\sum_j [y^j - out(\mathbf{x}^j)]^2$$

$$out(\mathbf{x}) = g\left(\sum_{k'} w_{k'} g\left(\sum_{i'} w_{i'}^{k'} x_{i'}\right)\right)$$

$$\frac{\partial \ell(W)}{\partial w_k} = \sum_{j=1}^{m} -[y^j - out(\mathbf{x}^j)]\frac{\partial out(\mathbf{x}^j)}{\partial w_k}$$

Gradient descent for 1-hidden layer Back-propagation: Computing $\frac{\partial \ell(W)}{\partial w_i^k}$

Dropped w_0 to make derivation simpler

$$\ell(W) = \frac{1}{2}\sum_j [y^j - out(\mathbf{x}^j)]^2$$

$$out(\mathbf{x}) = g\left(\sum_{k'} w_{k'} g\left(\sum_{i'} w_{i'}^{k'} x_{i'}\right)\right)$$

$$\frac{\partial \ell(W)}{\partial w_i^k} = \sum_{j=1}^{m} -[y - out(\mathbf{x}^j)]\frac{\partial out(\mathbf{x}^j)}{\partial w_i^k}$$

Forward propagation – prediction

- Recursive algorithm
- Start from input layer
- Output of node V_k with parents $U_1, U_2, \ldots$:

$$V_k = g\left(\sum_i w_i^k U_i\right)$$

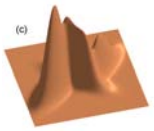
Back-propagation – learning

- Just gradient descent!!!
- Recursive algorithm for computing gradient
- For each example
  - Perform forward propagation
  - Start from output layer
  - Compute gradient of node V_k with parents $U_1, U_2, \ldots$
  - Update weight w_i^k

Many possible response functions

- Sigmoid
- Linear
- Exponential
- Tanh
- …

Convergence of Backprop



- Single sigmoid perceptron
  - Convex optimization
  - Gradient descent reaches **global minima**
- Multilayer neural nets **not convex**
  - Gradient descent gets stuck in local minima
  - Hard to set learning rate
  - Selecting number of hidden units and layers = fuzzy process
  - NNs falling in disfavor in last few years
  - We'll see later in semester, *kernel trick* is a good alternative
  - Nonetheless, neural nets are one of the most used ML approaches
    - Plus, neural nets are back with a new name!!!!
      - Deep belief networks
        - (and a probabilistic interpretation & slightly different learning procedure)

Overfitting?

- Neural nets represent complex functions
  - Output becomes more complex with gradient steps

Overfitting

- Output fits training data "too well"
  - Poor test set accuracy
- Overfitting the training data
  - Related to bias-variance tradeoff
  - One of central problems of ML
- Avoiding overfitting?
  - More training data
  - Regularization
  - Early stopping

What you need to know about neural networks

- Sigmoid Perceptron:
  - Like logistic regression
- Multilayer neural nets
  - Representation
  - Learning rule
- Overfitting
  - Definition
  - Training set versus test set
  - Learning curve

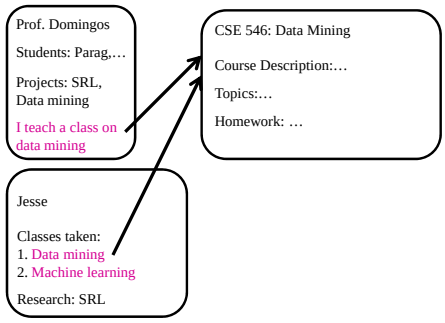
Co-Training Motivation

- Learning methods need labeled data
  - Lots of <x, f(x)> pairs
  - Hard to get… (who wants to label data?)
- But unlabeled data is usually plentiful…
  - Could we use this instead??????
- Semi-supervised learning

Co-training Suppose

- Have ***little*** labeled data + ***lots*** of unlabeled
- Each instance has two parts:
  x = [x1, x2]
  x1, x2 conditionally independent given f(x)
- Each half can be used to classify instance
  $\exists f1, f2$ such that $f1(x1) \sim f2(x2) \sim f(x)$
- Both f1, f2 are learnable
  $f1 \in H1$, $f2 \in H2$, $\exists$ learning algorithms A1, A2

Co-training Example

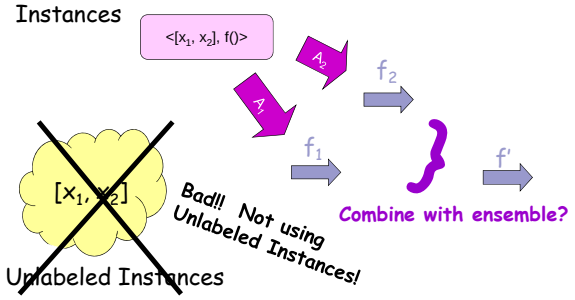


Without Co-training

$f_1(x_1) \sim f_2(x_2) \sim f(x)$

A_1 learns f_1 from x_1
A_2 learns f_2 from x_2

A *Few* Labeled Instances



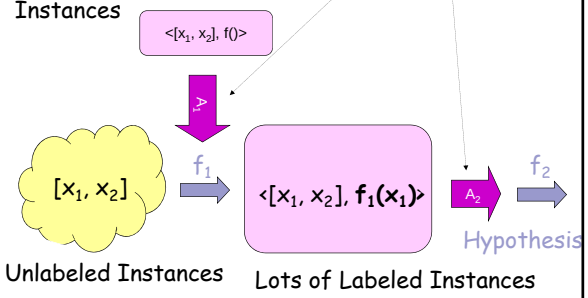
50

Co-training

$f_1(x_1) \sim f_2(x_2) \sim f(x)$

A_1 learns f_1 from x_1
A_2 learns f_2 from x_2

A *Few* Labeled Instances



51

Observations

- Can apply A_1 to generate as much training data as one wants
  - If x_1 is conditionally independent of x_2 / f(x),
  - then the error in the labels produced by A_1
  - ***will look like random noise to A_2 !!!***
- Thus ***no limit*** to quality of the hypothesis A_2 can make

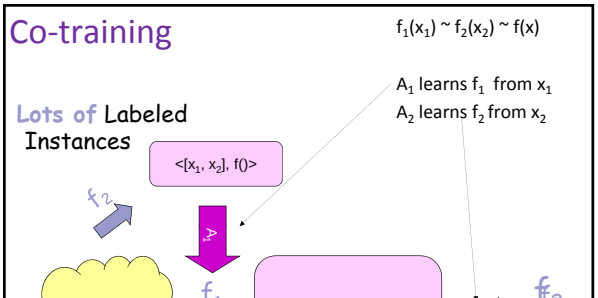
52

Co-training

$f_1(x_1) \sim f_2(x_2) \sim f(x)$

A_1 learns f_1 from x_1
A_2 learns f_2 from x_2

Lots of Labeled Instances



53

It really works!

- Learning to classify web pages as course pages
  - x1 = bag of words on a page
  - x2 = bag of words from all anchors pointing to a page
- Naïve Bayes classifiers
  - 12 labeled pages
  - 1039 unlabeled

	Page-based classifier	Hyperlink-based classifier	Combined classifier
Supervised training	12.9	12.4	11.1
Co-training	6.2	11.6	5.0

Table 2: Error rate in percent for classifying web pages as course home pages. The top row shows errors when training on only the labeled examples. Bottom row shows errors when co-training, using both labeled and unlabeled examples.

54