CISC 484

Introduction to Machine Learning
Preliminaries

1

When do we need "learned" models?

- Make a cup of coffee?
- Classifying images?
- Recognize hand-written letters/digits?
- Driving on a specific route with no one else on the road?
- Drive in city traffic to go from any address to another address?

2

Classify: cat family or not?

3

And This

4

Generalization can cause errors

5

- Classification is a task where ML is often applied
- ML often involves generalization – generalize to *unseen* examples.
- Number of training data and representation of instances matters.
  - E.g., only number of legs and tails.

6

Machine Learning

- *A computer program is said to*
  - ***learn from experience E with respect to task T with performance measure P**, where*
  - *its performance on T as measured by P, improves with experience E.*

7

Supervised Learning

- Supervision in the form of lots of instances annotated with "answer"
- Classification: most common form of supervised learning.
  - The "answers" come from a finite set of classes/categories.

8

Training dataset

- $D=\{<\mathbf{x}^i, y^i> \mid 1 <= i <= N\}$ (note superscript)
- N training instances
- $\mathbf{x}^i$ is a training instance (single data point)
- y^i is called the target value/label

9

Classification vs Regression

- Classification when $y^i \in C = \{c_1,\ldots,c_k\}$
  - y^i is the target value/annotation for $\mathbf{x}^i$
  - k=2 : binary classification C={0,1} or {y.n} or {t,f}
  - k>2 : multi-class classification (different from multi-label classification) C={small, medium, large}
- Regression when y^i is a real-valued number.

10

Input Instance

- Typically, input specifies values for d attributes or features.
- Values can be discrete/categorical or numerical
- Each input instance can be thought of as a vector of dimension d
- ◆But input can also be an image, a text, an email,…
  - Though internally it can be represented in other ways

11

Input Instance

- $\mathbf{x}=<a_1=v_1, \ldots, a_d=v_d>$
- $\mathbf{x}=<v_1, \ldots, v_d>$
- $\mathbf{x}^t=<v^t_1, \ldots, v^t_d>$

12

Pima Indian Diabetes data point

- # pregnancies = 6
- Blood glucose level = 148
- Diastolic blood prossure =72
- Skin thickness triceps =35mm
- Serum insulin=0
- BMI =33.6
- Diabetes pedigree function =0.627
- Age = 50 years
- Outcome=1 (this is the target value y)

<6,148,72,35,0,33.6,0.627,50,1> or
<6,148,72,35,0,33.6,0.627,50>:1

13

Examples

- Spam Filtering (usually binary classification)
- High dimensional input
  - Presence/absence in the email body of each word in vocabulary
    - (bag of words)
  - Email address
  - Presence of figures/tables in email
  - …

14

Examples

- Hand-writing detection (MNIST)
- Multi-class classification
  - Output 0,1,…,9
  - sometimes probabilistic
- Input (black and white image)
  - 28x28 pixels (approximately 900 dimensions)
  - 0,1,…,255 gray-scale
- 60,000 instances in training and 10,000 for testing.

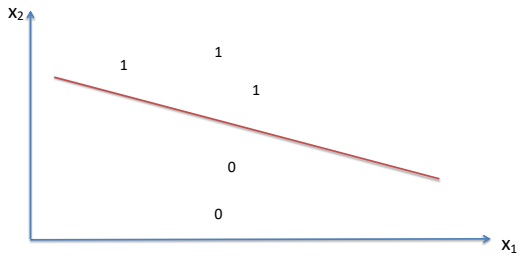
15

Target Function

- An *unknown* function *f()* such that y=f(**x**).
  - y is the target value and f is the target function
- Given a training data, we induce a model that is an approximation, say f' such that y'=f'(**x**)
  - f' represents the learned model and y' is the prediction (for input **x**)

16

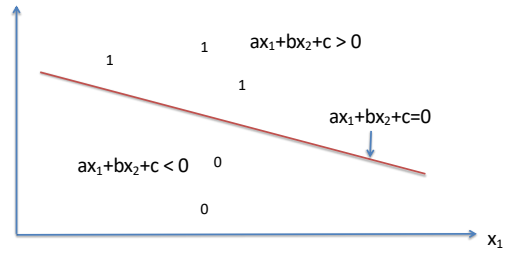
Classification and Boundaries



What if we learn a function f() such y=f(x) is a number that can be something other than 0 or 1?
For example we learn $y=(ax_1+bx_2+c)$?

17

Linear Classifier



18

What's a Linear Classifier

- In a 2-d space, we learn values a,b and c such that
- y =1 if $ax_1 + bx_2 + c >= 0$ and y=0 otherwise
  - Here y is the prediction for a given $<x_1, x_2>$
- y =1 if $I(ax_1 + bx_2 + c >= 0)$
  - I() is the indicator function
  - Also related to Kronecker delta function

19

Probabilistic Methods for Classification

- The learned model provides a probability distribution for each instance:
  - p(y | **x**, D) with
  - p(y=c_1 | **x**,D)
  - p(y=c_2 | **x**,D)
  - …
  - prediction y'= argmax_c p(y=c | **x**,D)
- p(y=0 | **x**,D) = 1 - p(y=1 | **x**,D) for binary classification

20

Inductive Bias

- Simplifying assumptions
  - Binary classification
  - Input instances from 2 dimensional space
  - Both attributes are binary valued

x_1	x_2	h_1	h_2	h_3		h_{16}
0	0	0	0	0	…	1
0	1	0	0	0	…	1
1	0	0	0	1	…	1
1	1	0	1	0	…	1

21

Need for a Bias

x_1	x_2	h_1	h_2	h_3		h_{16}
0	0	0	0	0	…	1
0	1	0	0	0	…	1
1	0	0	0	1	…	1
1	1	0	1	0	…	1

- First training instance <1,1>: 1
- Only h_2, h_4,…, h_{16} are now consistent with training data.
- 2nd training instance <1,0>: 0
- Now further cut down to h_2, h_6, h_{10}, h_{14}
- one more – down to 2
- Finally only 1 hypothesis consistent with training data.

22

Need for Bias

- If an instance is defined by d binary-valued attributes then number of hypothesis =

$$2^{(2^d)}$$

- If training data has N training examples then number of possible hypothesis consistent with training data =

$$2^{(2^d - N)}$$

23

Inductive Bias

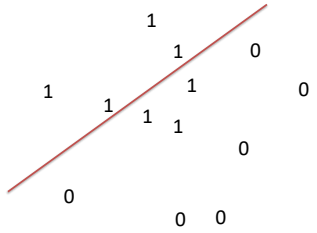
How do you pick one?

To pick one, there has to be some external criterion.

This external criterion is called the ***inductive bias*** of the training algorithm.

24

Inductive Bias = Linear Classifier



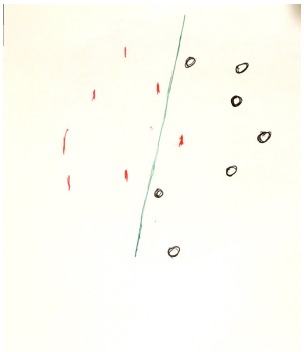
25

Bias and Fitting the Data

- Strong bias could result inability to fit the data
- The term *high bias* is often used to indicate the inability to fit training data.
- High bias leads to simple models.
- We expect simple models to not change much with small changes to training data (*low variance*)

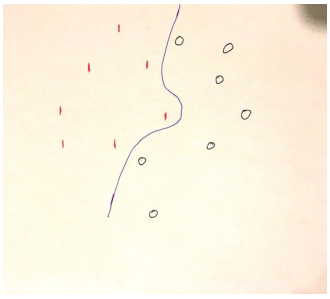
26

Over-fitting



27

Over-fitting



28

What is Noise?

1. Errors in labeling data – *teacher noise*
2. Imprecision in recording the input attributes (shifted data point)
   <…, 98.4, …>: infected=+ instead of
   <…, 100.4, …>: infected=+
3. There are additional factors/attributes that are not taken into account in input representation (*latent* or *hidden* attributes as these are unobserved)

29

DataSets for Evaluation

- Training Set -- large set
- Test Set – not small (statistically meaningful)
- Typical split (90:10 or 80:20)

30

n-fold Cross-Validation

- Especially when we don't have large amount of annotated data
- Partition annotated data into n *folds*
- Use (n-1) folds for training and remaining fold for testing. Repeat n times
- Common is 10 fold cross-validation – training/test split is 90% to 10%

31

Learning as Searching

- Each machine learning method defines its hypothesis space
- One hypothesis from this space will be selected after training
- This process of selection can be thought of as searching through the hypothesis, using the training data for navigation.

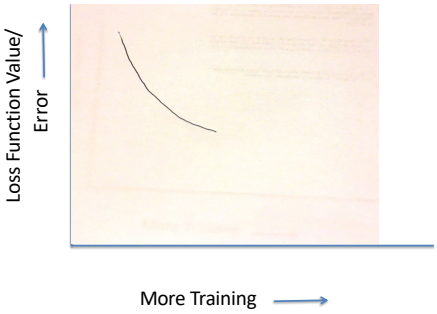
32

Loss Function

- Usual there is some well-developed principle for selection
- This includes the definition of a *loss function* that characterizes problems a chosen model has in fitting the data.
- This is often tied to *error* made by model on a certain data set.
- Goal of training would be to minimize the loss function.

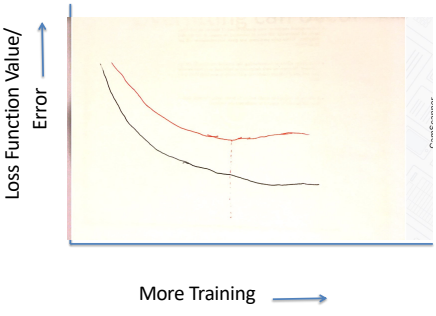
33

Error on Training set



34

Overfitting can occur



35

Model Selection

- Every x-axis value gives us a model.
- We have to *select* one of those models.
- Selection can't be made on the basis of the test set.
- Some amount of held-out data is used for model selection

36

Validation Set

- Held-out data used for model selection.
- *Validation set* (not to be confused with n-fold cross-validation process)
- Meets same requirements as test set.
- The annotated data is broken up into three partitions: training set, validation set and test set.

37

Linear Regression

1

House Prices

- Can we predict the price of a house based on
  - a_1: Floor area
  - a_2: number of rooms
  - a_3: number of bathrooms
  - a_4: availability of swimming pool
- Assumption: it is reasonable the price grows proportionately (linearly) with these attributes' values (x_1, x_2, x_3,x_4)

2

Using Linear Regression

- Prediction of house price → regression
- Representation of an instance (house) – $\mathbf{x} = < x_1, x_2, x_3, x_4>$
- Training set: {<$\mathbf{x}^1$, y^1>, <$\mathbf{x}^2$, y^2>, … , <$\mathbf{x}^N$, y^N>}

3

Example: training data

- <<2500, 5, 2, 0> , 150>
- <<3500, 7, 2, 0> , 250>
- <<2000, 4, 1, 1> , 160>
- <<2000, 4, 2, 1> , 180>

- $\mathbf{x}^1$ = <2500,5,2,0>
- $y^2 = 250$
- $x_2^3 = 4$

4

Using Linear Regression

- $f(\mathbf{x}) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_0$
  - Add $x_0 = 1$
- Training of the model = induce ("learn") the weights (w_0, w_1, w_2, w_3, w_4) to best *fit the training data*.
- Thus, the model is specified by $w = < w_0, w_1, w_2, w_3, w_4 >$

5

In General

- $D = \{<\mathbf{x}^i, y^i> \mid 1 <= I <= N\}$
- If data points come from a d-dimensional space then $\mathbf{x}^i = <x_0^i, x_1^i, \ldots, x_d^i>$
- The learned weights $\mathbf{w}=<w_0, w_1, \ldots w_d>$ specify the learned model

6

Predicted Value

- For a new datapoint $\mathbf{x} = <x_1, \ldots, x_d>$
- $o = w_0 + w_1x_1 + \ldots w_dx_d$
  $= w_0x_0 + w_1x_1 + \ldots w_dx_d$, where $x_0 = 1$
- $o = \mathbf{w} \cdot \mathbf{x}$
  - $\mathbf{x} = <x_0, x_1, \ldots, x_d> = <1, x_1, \ldots, x_d>$
  - $\mathbf{w} = <w_0, w_1, \ldots, w_d>$

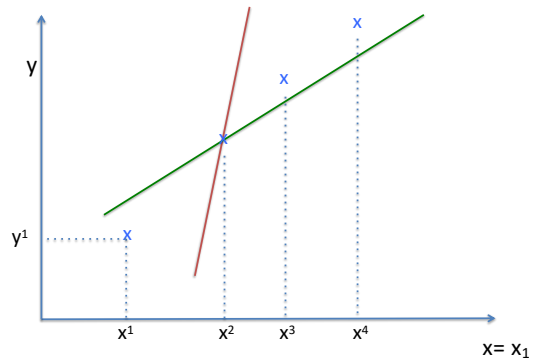
7

Predicted Value

- $o = w_0x_0 + w_1x_1 + \ldots w_dx_d$, where $x_0 = 1$
- $\mathbf{x} = \begin{bmatrix} x_0 \\ \\ x_d \end{bmatrix}$ then $o = [w_0, \ldots, w_d] \cdot \begin{bmatrix} x_0 \\ \\ x_d \end{bmatrix} = w^Tx$

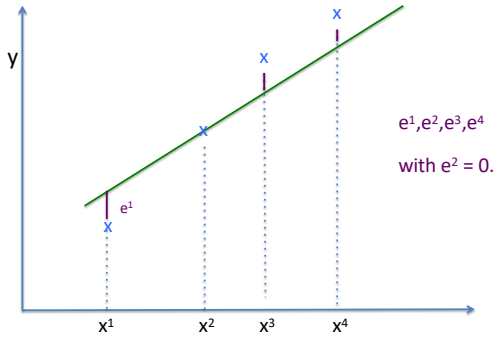
8

Which Line is a Better Fit



9

Loss in Terms of Error



10

Loss in terms of Error

- **w** is the model
- E(**w**) specifies error on the training set by **w**
- For a training instance $\mathbf{x}^t$,

  $e^t(\mathbf{w}) = y^t - o^t = y^t - \mathbf{w} \cdot \mathbf{x}^t \qquad (y^t - \mathbf{w}^T \cdot \mathbf{x}^t)$
- We need to compute E(**w**) based on the entire training set.

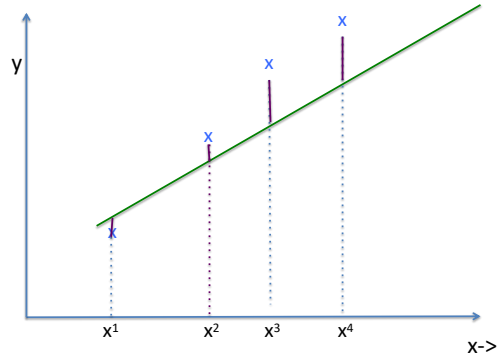
11

Sum of Squared Error

- $E(w) = \sum_t y^t - o^t = \sum_t y^t - w.x^t$, where $1 <= t <= N$
  - Just taking the sum of $e^t(\mathbf{w})$ for each training instance t won't work.
- $E(w) = \sum_t |y^t - o^t| = \sum_t |y^t - w.x^t|$
- $E(w) = \sum_t (y^t - o^t)^2 = \sum_t (y^t - w.x^t)^2$

12

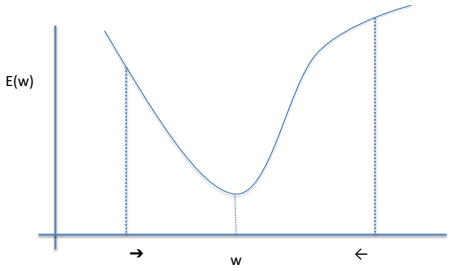
Approach to Training
Reduce Error (Loss Function)



13

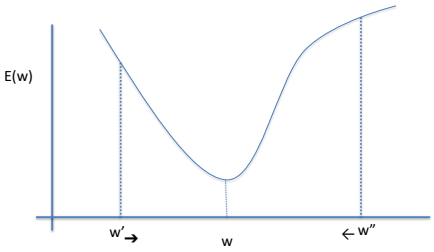
Error Surface

- Pretend **w** = w



14

Finding Model with Minimum Error



At w', increase w and at w", decrease w.

At w', gradient is negative and at w", it is positive

So in both cases, add Δw given by – η (∇E/∇w)

We choose η as a small positive number

15

What's the Gradient?

- We need to compute E(**w**) based on the entire training set.

- For a training instance $\mathbf{x}^t$, $e^t(\mathbf{w}) = y^t - o^t$
- $E(w) = \sum_t (y^t - o^t)^2$
- $o^t = w_0 x_0^t + w_1 x_1^t + \ldots w_d x_d^t$, where $x_0^t = 1$

16

Gradient Descent Update-1

- $\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$
- $\Delta\vec{w} = -\eta \frac{\nabla E(\vec{w})}{\nabla \vec{w}}$

Instead

- $w_i \leftarrow w_i + \Delta w_i$
- $w_i = -\eta \frac{\partial E(\vec{w})}{\partial w_i}$

17

Gradient Descent Update-2

- $E(\vec{w}) = \frac{1}{2} \sum_{t=1}^{N} (y^t - o^t)^2$
- $o^t = w_0 x_0^t + w_1 x_1^t + \cdots + w_d x_d^t$
- $\Delta w_i = -\eta \frac{\partial \frac{1}{2} \left(\sum_t y^t - (w_0 x_0^t + \cdots + w_i x_i^t + \cdots + w_d x_d^t)\right)^2}{\partial w_i}$

18

Gradient Descent Update-3

- $\frac{\partial E(\vec{w})}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_t (y^t - o^t)^2$
  $= \frac{1}{2} \sum_t 2 \cdot (y^t - o^t) \cdot \frac{\partial (y^t - o^t)}{\partial w_i}$
  $= \sum_t (y^t - o^t)(-x_i^t)$
- $\Delta w_i = -\eta \frac{\partial}{\partial w_i} E(\vec{w})$
  $= +\eta \sum_t (y^t - o^t) x_i^t$

19

Gradient Descent Update-4

- $\frac{\partial}{\partial w_i}(y^t - o^t) = \frac{\partial}{\partial w_i}(y^t - \vec{w} \cdot \vec{x}^t)$
  $= \frac{\partial}{\partial w_i}(y^t - (w_0 x_0^t + \cdots + w_i x_i^t + \cdots + w_d x_d^t))$
  $= -(x_i^t)$

20

Gradient Descent Update -- 5

$$\Delta w_i = -\eta \frac{\partial E(\vec{w})}{\partial w_i}$$

$$= +\eta \sum_t (y^t - o^t) x_i^t$$

$$o^t = \vec{w} \cdot \vec{x} = \sum_{j=0}^{d} w_j x_j^t$$

For incremental/stochastic version we will consider a single instance t

so $\Delta w_i = \eta (y^t - o^t) x_i^t$

21

Stochastic and Batch

- Start with a random set of weights (w_i's).
- Repeat
  - Compute loss (error as given by current **w**)
  - Take a small step in direction of negative gradient
  - Recompute model
- Compute loss/gradient for a single data point (stochastic/incremental) or a set of points, which is a batch or entire training set.

22

Gradient Descent Algorithm

In both the batch and incremental methods described next, we will assume that

- The training set is D = {<$\mathbf{x}^t$, y^t> | 1 <= t <= N}
- The weights of the learned model will be w_i where 0 <= i <= d
- The learning rate is η where 0 < η < 1

23

Gradient Descent Algorithm

(Batch = Training Set)

- Initiate each w_i to some small random number.
- Until termination condition is met, do
  - Initialize each Δw_i to zero. (Step A)
  - For each t between 1 and N do
    - Compute $\mathbf{w}.\mathbf{x}^t$
    - For each i between 0 and d do
      - $\Delta w_i \leftarrow \Delta w_i + \eta (y^t - \mathbf{w}.\mathbf{x}^t) \mathbf{x}^t_i$
  - For each i between 0 and d do
    - $w_i \leftarrow w_i + \Delta w_i$ (Step B)

24

Incremental Gradient Descent Algorithm

(i.e., batch size = 1)

- Initiate each w_i to some small random number.
- Until termination condition is met, do
  - For each t between 1 and N do
    - Initialize each Δw_i to zero. (Step A)
    - Compute $\mathbf{w}.\mathbf{x}^t$
    - For each i between 0 and d do
      - $\Delta w_i \leftarrow \Delta w_i + \eta (y^t - \mathbf{w}.\mathbf{x}^t)\, x^t_i$
      - $w_i \leftarrow w_i + \Delta w_i$ (Step B)

25

Batch vs Incremental

- Incremental mode can approximate batch model with an arbitrary small error.
- Incremental mode can better avoid local minima. Update **w** more frequently. It is faster. It deals with dynamic datasets. Learning rate is usually smaller in incremental mode.
- Batch mode is more stable. Easy to check convergence.

26

Regularization

- Don't make the models overly complex
- Regularization is a modification to a learning algorithm that is intended to reduce its *generalization error* but not its *training error -- Ian Goodfellow*
- Intended to avoid overfitting

27

Regularization

- Reduce overfitting: simple model, small parameters
- New cost function:

  $E^*(\mathbf{w}) = E(\mathbf{w}) +$ regularization

  $E^*(\mathbf{w}) = \sum_t (y^t - \mathbf{w}.\mathbf{x}^t)^2 + \boldsymbol{\lambda} \sum_i |w_i|^q$

- Minimizing $E^*(\mathbf{w})$ implicitly minimizes $\sum_i |w_i|^q$
- q = 1: L1 regularization (Lasso)
- q = 2: L2 regularization

28

Logistic Regression

1

Logistic Regression

- Linear Classifier
- Binary Classification
- Widely used especially in medical and social sciences
  - Medical diagnosis
  - Political science
  - Engineering – probability of failure
  - Marketing – customer satisfaction, halting subscription
  - Business – prob of Mortgage defaulting

2

One Way of explaining Logistic Regression

- Like regression we will consider
  - $w_0 + w_1x_1 + \ldots + w_d x_d$
  - Model is specified by $w_0, w_1, \ldots w_d$
- But we want to classify not to do regression
- $w_0 + w_1x_1 + \ldots + w_k x_k$ ranges from negative infinity to positive infinity
- Larger the number, more likely we have a positive sample (y=1) and more negative …

3

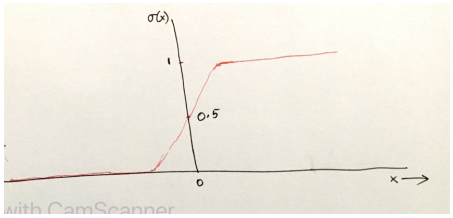
Squashing Function

- Thus, we want the model to compute P(y=1|**x**), where of course
  - P(y=0|**x**) = 1- P(y=1|**x**).
- P(y|**x**) is determined by first computing **w.x**
  1. Squash $w_0 + w_1x_1 + \ldots + w_d x_d$ to values between 0 and 1.
  2. Higher **w.x** means probability of prediction y=1 is higher.
  3. Large negative value means P(y=0|**x**) should be close to 1 (therefore P(y=1|**x**) should be close to 0)

4

Squashing Function

- One such function is sigmoid
- Range:



5

Probability of Success or Failure

$$P(y=1|\vec{x},\vec{w}) = \sigma(\vec{w}\cdot\vec{x}) = \sigma(w_0x_0 + w_1x_1 + \dots w_nx_n)$$

$x_0 = 1$

$$P(y=1|\vec{x},\vec{w}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x})}} = \frac{e^{\vec{w}\cdot\vec{x}}}{1+e^{\vec{w}\cdot\vec{x}}} = \frac{e^{\sum_i w_i x_i}}{1+e^{\sum_i w_i x_i}}$$

$$P(y=0|\vec{x},w) = 1 - \frac{1}{1+e^{-(\vec{w}\cdot\vec{x})}} = \frac{1+e^{-(\vec{w}\cdot\vec{x})}-1}{1+e^{-(\vec{w}\cdot\vec{x})}}$$

$$= \frac{e^{-\sum_i w_i x_i}}{1+e^{-\sum_i w_i x_i}}$$

6

Decision Rule

- P(y=1|**w, x**) is still a number. But we want a classifier.
- y=1 if P(y=1|**w, x**) ≥ 0.5
  - y=1 if P (y=0|**w, x**) < 0.5
- y=0 if P(y=1|**w, x**) < 0.5
  - y=0 if P (y=0|**w, x**) ≥ 0.5

7

Logistic Regression Training

- For probabilistic methods, maximum likelihood estimation (MLE) methods are used.
- MLE finds the parameter values that maximize the likelihood of making the observations given the parameters.
- That is, for a model θ, what is the likelihood of the observed data (training data)
  - E.g., coin tosses 8 of 10 tosses results in heads

8

Likelihood of Observed Data

- Suppose we have a training data with n instances given by $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N$ and we have a model given by parameters $\mathbf{w}$.
- Suppose the annotations/labels for this instances are given as $y^1, y^2, \ldots, y^N$.
- Then likelihood of the training data is given by
  $P(y^1, y^2, \ldots, y^N \mid \mathbf{w}, \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N)$
- If we assume independence among the observations then the likelihood is
  $P(y^1 \mid \mathbf{w}, \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N) \ldots P(y^N \mid \mathbf{w}, \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N)$

  $= \boldsymbol{\Pi}_t P(y^t \mid \mathbf{w}, \mathbf{x}^t)$

  Note that y^t is 0 or 1.

9

Log Likelihood

- $\boldsymbol{\Pi}_t P(y^t \mid \mathbf{w}, \mathbf{x}^t)$ can be extremely small and lead to underflow errors.
- So we take the log and maximize
  $\sum_t \ln(P(y^t \mid \mathbf{w}, \mathbf{x}^t))$
- Since y^t is 0 or 1, we can write this is as
  $\sum_t y^t \ln(P(y=1 \mid \mathbf{w}, \mathbf{x}^t)) + (1-y^t) \ln(P(y=0 \mid \mathbf{w}, \mathbf{x}^t))$
- We have to find w that maximizes this
- This can be done using gradient descent.

10

Wald Statistic

- With every "coefficient" (weight w_i), the training algorithm will produce another number – Wald statistic.
- The **Wald test** is a way to find out if explanatory variables (here the weight) in a model are significant.
  - "Significant" means that they add something to the model; variables that add nothing can be deleted without affecting the model in any meaningful way.

11

Second Way

- Instead of going from regression to probability, in statistics a link function maps probability to a regression.
- Logit is log of odds.
- Odds:
  - F(x) = probability of success.
  - Probability of failure = 1 – F(x),
  - Odds = F(x)/ (1- F(x))

12

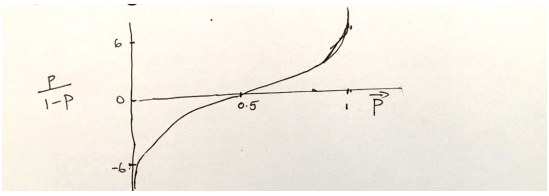
Odds

- Probability Eagles will win = ½
  - Odds = (1/2)/(1/2) = 1
- Probability is ¾
  - Odds = (3/4)/(1/4) = 3
- Probability is 1/3
  - Odds=(1/3)/(2/3) = 0.5

13

Range of Odds

- F(x) ranges between 0 and 1
- Odds ranges between 0 to infinity



- Log(odds) ranges from negative to positive infinity.

14

Range of Log(Odds)

- Log(odds) ranges from negative to positive infinity.
- Recall link function used to map probability to regression.
- Log(odds) = $w_0 + w_1x_1 + \ldots + w_k x_k$

$$\log \frac{F(x)}{1-F(x)} = w_0 + w_1 x_1 + \cdots + w_k x_k$$

$$F(x) = P(Y=1 \mid \vec{x}, \vec{w}).$$

15

P(Y=1|**X**) ?

$$P(Y=1 \mid \vec{x}, \vec{w}) = F(x).$$

$$\log \frac{F(x)}{1-F(x)} = w_0 + w_1 x_1 + \cdots + w_k x_k = \sum_i w_i x_i$$

$$\frac{F(x)}{1-F(x)} = e^{\sum_i w_i x_i} \quad \therefore F(x) = (1-F(x)) e^{\sum_i w_i x_i}$$

$$F(x) = e^{\sum w_i x_i} - F(x) e^{\sum w_i x_i} \quad \therefore F(x)\left(1 + e^{\sum w_i x_i}\right) = e^{\sum w_i x_i}$$

$$F(x) = \frac{e^{\sum w_i x_i}}{1 + e^{\sum w_i x_i}} = \frac{1}{1 + e^{-\sum w_i x_i}}$$

16

Odds? Probability of Success?

$$P(y=1 \mid \vec{w}, \vec{x}) = \sigma(\vec{w} \cdot \vec{x}) = \sigma\left(\sum_i w_i x_i\right)$$

$$P(y=0 \mid \vec{w}, \vec{x}) = 1 - \sigma(\vec{w}\,\vec{x}) = \frac{e^{-\vec{w}\cdot\vec{x}}}{1+e^{-\vec{w}\cdot\vec{x}}}$$

$$\text{odds} = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x})}} \Big/ \left(\frac{e^{-(\vec{w}\cdot\vec{x})}}{1+e^{-\vec{w}\cdot\vec{x}}}\right) = \frac{1}{e^{-(\vec{w}\cdot\vec{x})}}$$

$$\log \text{odds} = \log \frac{1}{e^{-\vec{w}\cdot\vec{x}}} = -\log e^{-(\vec{w}\cdot\vec{x})}$$

$$= \vec{w}\cdot\vec{x}$$

17

ENTROPY etc.

Random variable: X

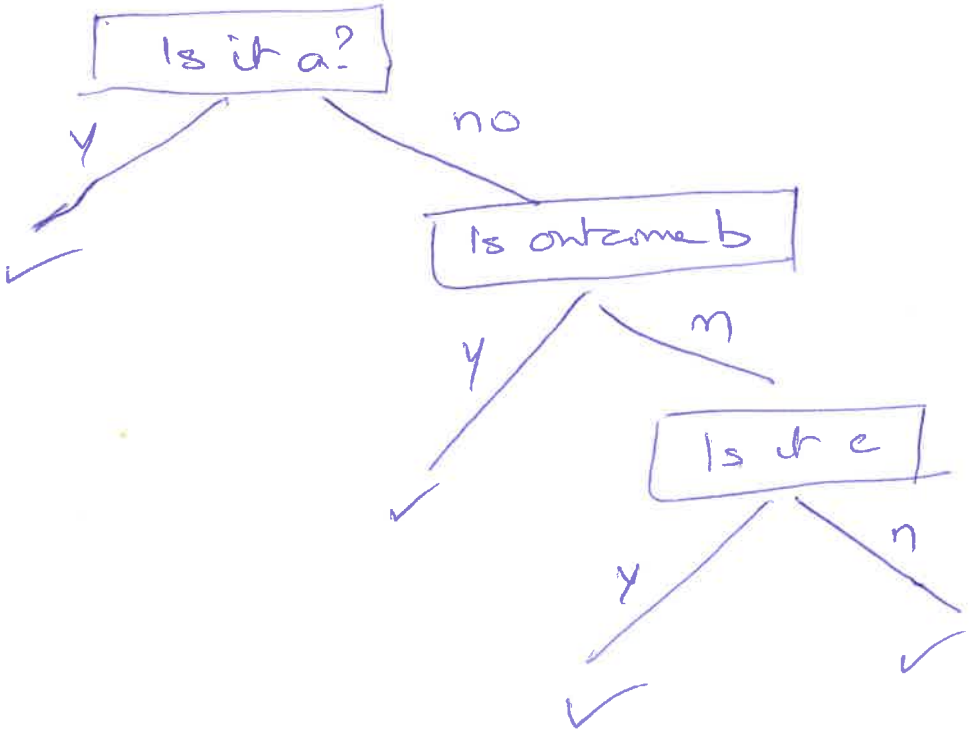
Outcomes: {a, b, c, d}.

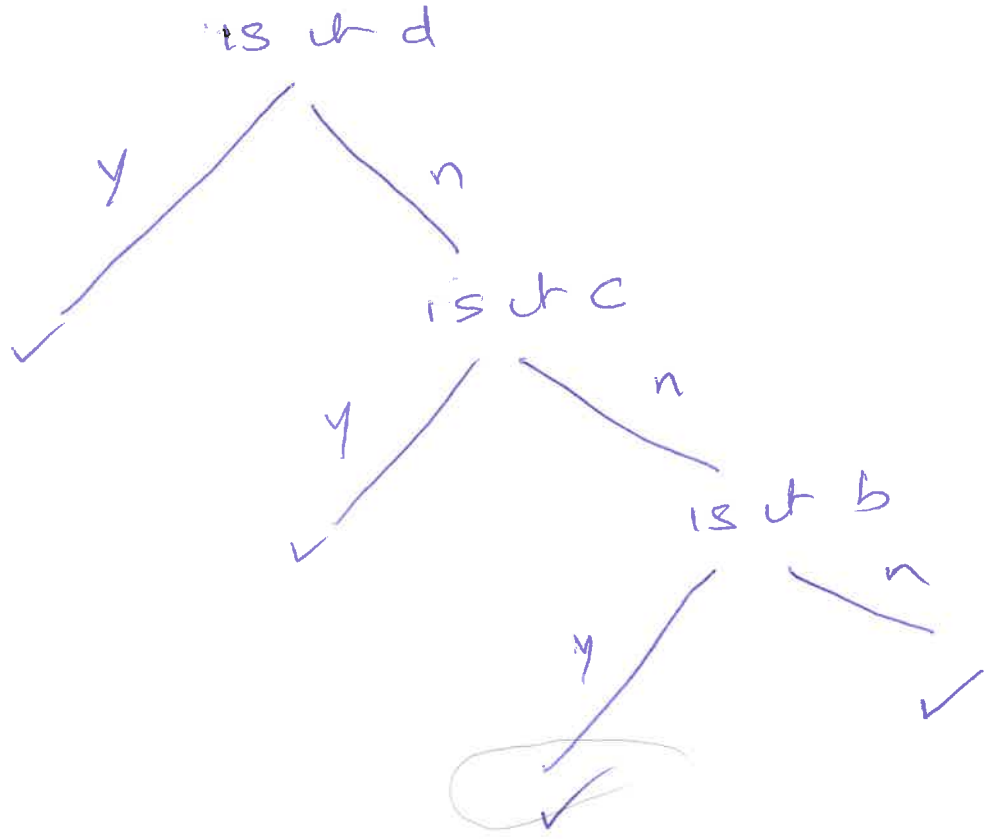
- How many yes-no questions do you need to ask to know the outcome?
- What if outcomes differed in how frequently they happen?
- How many yes-no questions on an average?
- When will this be high?
- What does a low number (such as 0) mean?

$p(x=a) = \frac{1}{2}$ $p(x=b) = \frac{1}{4}$ $p(x=c) =$

$p(x=d) =$

$\frac{1}{8}$.



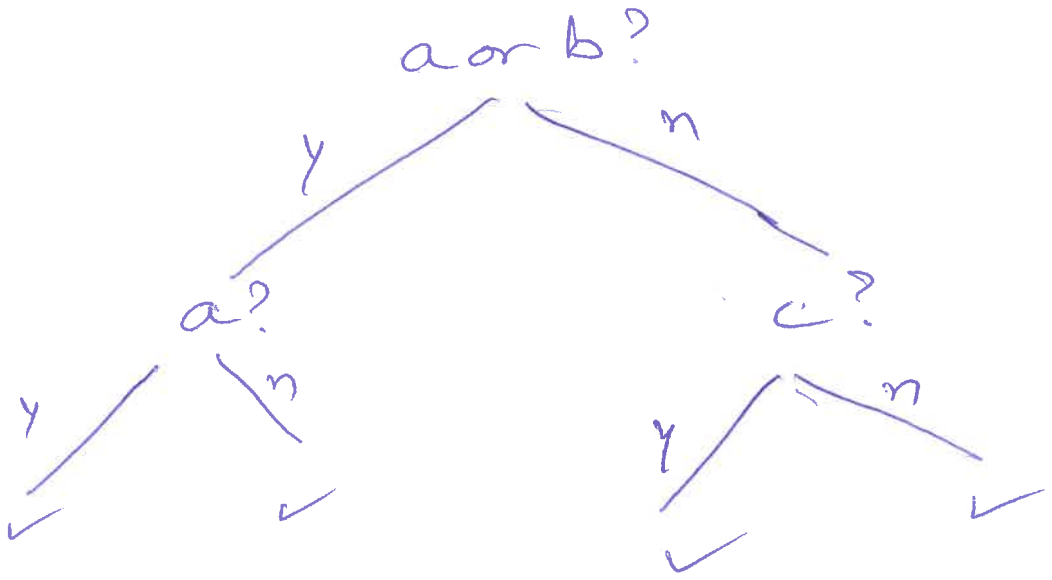


questions on an average?

Expected number of questions?

general strategy — with each question cover about half the remaining probability mass with any branch?

tree for equiprobable.



$$H(X) = \sum_{0} p(\text{outcome}) \#(\text{questions}) \text{ for outcome} = 0$$

$$= \sum_{x} p(x) \log_2 \frac{1}{p(x)}$$

$$= -\sum_{x} p(x) \log_2 p(x)$$

Properties. 1. $H(X) \geq 0$

2. $H(X)$ is maximum when all outcomes are equiprobable.

3. $0 \leq H(X) \leq \log_2 n.$

Joint Entropy $H(X,Y)$

$$= -\sum_{x,y} p(x,y) \log p(x,y)$$

Conditional Entropy $H(X|Y)$

$$= \sum_{y} p(y) H(X|Y=y) =$$

$$= H(X,Y) - H(Y)$$

$$p(x|y) = \frac{p(x,y)}{p(x)}$$

$$H(X|Y) = H(X,Y) - H(Y)$$ ~~$= H(X,Y)$~~

$$H(X,Y) = H(X|Y) + H(Y)$$
$$= H(Y|X) + H(X)$$

$$H(Y) + H(X|Y) = H(Y|X) + H(X)$$

$$H(Y) - H(Y|X) = H(X) - H(X|Y)$$

Mutual Information: $I(X;Y)$

- $I(X;Y) = H(Y) - H(Y|X) = H(X) - H(X|Y)$
- reduction in uncertainty in value of Y when X is known = reduction in uncertainty in X when Y is known.

- note if Y is independent of X

  $I(X;Y) = 0$.

- note if Y completely determined by X

  $I(X;Y) = H(Y)$

- $I(X;Y) \geq 0$

  $I(X;Y) = I(Y;X)$

Assume that $p(\,)$ & q are two probability ~~differ~~ distributions on same X.

Can we measure how similar or different are they?

Start assuming the real probability is given by $P(\,)$ & we guess that it is given by Q.

What penalty to we pay for this incorrect assumption.

∴ If p & q are identical there should be no penalty.

ie $D(p \| p) = 0$.

$D(p||q)$ — Kullback-Leibler divergence.

— KL divergence.

Since we think the probability is given by $q(\,)$, our questioning is based on $q(\,)$.

That is the number of questions to determine if outcome is x is

$$\log_2 \frac{1}{q(X=x)} = -\log_2 q(x)$$

Expected number of questions is

$$-\sum p(x) \log_2 q(x)$$

But if we know what the correct probability distribution was, then the ~~average~~ expected # of questions is

$$-\sum_x p(x) \log p(x).$$

To get the penalty, let's subtract:

$$D(p\|q) = -\sum_x p(x) \log q(x) - \left(-\sum_x p(x) \log p(x)\right)$$

$$= -\sum_x p(x) \log \frac{q(x)}{p(x)}$$

$$= +\sum_x p(x) \log \frac{p(x)}{q(x)}.$$

Decision Trees

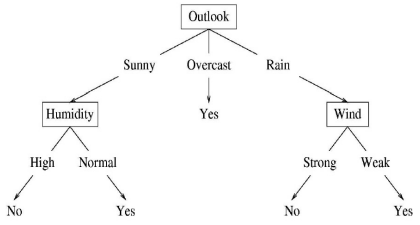
1

An Example Training Set

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

2

An Example Decision Tree



3

Characteristics of Decision Trees

- Internal nodes are labeled with attribute names
- Branches (edges) from an internal node labeled by attribute A are labeled by values of attribute A.
- Leaf Nodes are labeled by target values

4

An Example Dataset

	A	B	Target
Instance 1	a1	b1	0
Instance 2	a2	b2	1
Instance 3	a1	b2	1
Instance 4	a2	b1	0

Attribute to be checked at the root? What does the tree look like?
Which tree will we build?

5

A Different Example

- Current Situation: 8 + and 8 -
- Attribute A1 has 3 values:
  - for c1: 5+, 5-; for c2: 2+, 1-; and for c3: 1+, 2-.
- For attribute A2:
  - for d1: 2+, 2-; for d2: 4+, 2-; and for d3: 2+, 4-.
- Which attribute should we choose?
  - Expected value of entropy and gain in information

6

Entropy as a measure of uncertainity

7

Decision Trees

- An n-tuple: values for n attributes. E.g., <sunny, hot, …>
- Attribute values can be categorial.
  - In perceptrons, Log Reg, MLP etc. attribute values had to be numerical.
- Unlike SVM, logistic Regression etc, we are not constrained to binary classification.
  - Regression with CART

8

PlayTennis (from Mitchell's Book)

- A running example – binary classification with 14 training instances.
- Four attributes plus *target* attribute
  - Outlook
    - Sunny, Overcast, Rain
  - Temperature
    - Hot, Mild, Cool
  - Humidity
    - High, Normal
  - Wind
    - Strong, Weak

9

Running Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

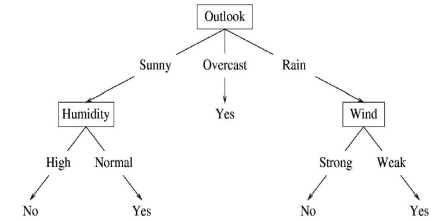
10

Decision Tree Nodes

- A decision tree classifies an instance by testing the attributes sequentially.
- Each internal (non-leaf) node will be labeled by an attribute.
- Branches of this node correspond to the possible values of the attribute.
- Leaf nodes are labeled by target values.

11

An Induced Decision Tree



- D5: Outlook=Rain, Temp=cool, humidity=normal and Wind=Weak. Prediction?
- D1: Outlook=Sunny, Temp= Hot, Humidity=High and wind=weak. ???

12

Inducing a Decision Tree

- We will discuss the classical Decision Tree Training algorithm called ID3.
- It builds the tree top-down
- The inductive bias – Make the tree as short as possible.

13

How to Train: Case 1

- Consider the node corresponding to Outlook= overcast (one level down from the root)
- Look at the instances in the training data that match this constraint. (Instances D3, D7, D12 and D13)

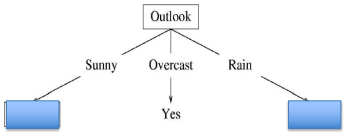
14

Running Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

15

Decision Tree



16

ID3 Algorithm

- ID3(S,$\mathcal{A}$)
- Create a root node
- Terminate this branch? (*overcast vs rain – next slide*)
- Otherwise
  - Choose the "best" A from $\mathcal{A}$
  - For each value v of A
    - Add new branch appropriate subset S_v of S
    - ID3(S_v, $\mathcal{A}$ - {A})
      - Sv is subset of instances of S which have v as value of A.

17

How to Train: Case 2

- Now consider outlook = Rain
- The training instances match this constraint
- The instances with Outlook = Rain are D4, D5, D6, D10, D14.
- Now we can restart decision tree induction from here but now the only instances we need to consider are given by D4, D5, D6, D10, D14.

18

Running Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

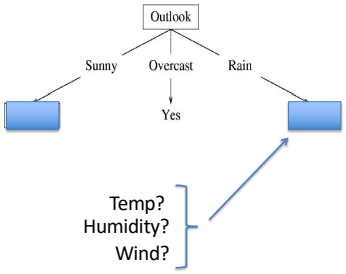
19

ID3 Algorithm

- ID3(S,$\mathcal{A}$)
- Create a root node
- Terminate this branch? (*overcast vs rain – next slide*)
- Otherwise
  - Choose the "best" A from $\mathcal{A}$
  - For each value v of A
    - Add new branch appropriate subset S_v of S
    - ID3(S_v, $\mathcal{A}$ - {A})
      - Sv is subset of instances of S which have v as value of A.

20

Decision Tree



21

Outlook = Rain

- The training instances (without Outlook) are
  - D4: Temp=mild,Humidity=high,wind=weak: YES
  - D5: Temp=cool,Humidity=normal,wind=weak: YES
  - D6: Temp=cool,Humidity=normal,wind=strong: NO
  - D10: Temp=mild,Humidity=normal,wind=weak:YES
  - D14: Temp=mild,Humidity=high,wind=strong: NO
- Temp?
  - Mild – D4, D10, D14 – 2y and 1n
  - Cool – D5, D6 – 1y and 1n

22

Outlook = Rain

- The training instances (without Outlook) are
  - D4: Temp=mild,Humi=high,wind=weak: YES
  - D5: Temp=cool,Humi=normal,wind=weak: YES
  - D6: Temp=cool,Humi=normal,wind=strong: NO
  - D10: Temp=mild,Humi=normal,wind=weak:YES
  - D14: Temp=mild,Humi=high,wind=strong: NO
- Humidity
  - High: D4: Y and D14:N
  - Normal: D5, D10: Y and D6: N

23

Outlook = Rain

- The training instances (without Outlook) are
  - D4: Temp=mild,Humi=high,wind=weak: YES
  - D5: Temp=cool,Humi=normal,wind=weak: YES
  - D6: Temp=cool,Humi=normal,wind=strong: NO
  - D10: Temp=mild,Humi=normal,wind=weak:YES
  - D14: Temp=mild,Humi=high,wind=strong: NO
- Wind
  - Weak: D4,D5, D10: Y
  - Strong: D6, D14: N

24

Choosing an Attribute

- Before looking at *wind* attribute, we were unsure (uncertain) about what the outcome (i.e., prediction) should be. Uncertainty can be measured by entropy of outcome (P(yes)= 3/5 and P(NO)=2/5).
- After knowing the value of *wind* attribute on these 5 instances, we will have no uncertainty on either branches.
- Information gain of *wind* is maximum possible since the reduction in uncertainty is highest possible.

25

What about Humidity

- Uncertainity when humidity is
  - high: entropy of P(yes) = ½ and P(no)= ½.
  - normal: entropy of P(yes) = 2/3 and P(no)= 1/3
- How do we combine these entropies?

26

Combining Entropies

- Attribute A1 has 3 choices:
  - for c1: 5+, 5-; for c2: 1+, 0-; and for c3: 0+, 1-.
- For attribute A2:
  - for d1: 3+, 3-; for d2: 3+, 0-; and for d3: 0+, 3-.
- Expected level of uncertainity is lower with A2.

27

Expected Value of Entropy

- Recall: humidity: High (D4, D14); normal (D5,D6,D10)
  - D4: Temp=mild,Humi=high,wind=weak: YES
  - D5: Temp=cool,Humi=normal,wind=weak: YES
  - D6: Temp=cool,Humi=normal,wind=strong: NO
  - D10: Temp=mild,Humi=normal,wind=weak:YES
  - D14: Temp=mild,Humi=high,wind=strong: NO
- Expected entropy, knowing value of humidity for this set is

  2/5(1/2 log2 + ½ log2) + 3/5 (2/3 log 3/2 + 1/3 log 3)
- Recall $H(X) = \Sigma_i (p_i \log 1/p_i)$

28

Information Gain Formula

- IG(S,A)= reduction in Entropy of S because of knowledge of values of A (therefore partitioning S according to this attribute).
- IG(S,A) = Entropy (S) – Σ_v ($|S_v|/|S|$) Entropy (S_v)

29

Idea behind IG(S,A) formula

- Let v be one of the possible values for A.
- S is the set of instances being considered
- Let S_v be the subset of S where instances have value v for A.
- Then we can compute the entropy of S_v based on the outcome of the distribution.
- Also we can estimate the probability an instance of S will belong to S_v. This can be computed as $|S_v|/|S|$.

30

Information Gain Formula

- IG(S,A)= reduction in Entropy of S because of knowledge of values of A (therefore partitioning S according to this attribute).
- IG(S,A) = Entropy (S) – Σ_v ($|S_v|/|S|$) Entropy (S_v)

31

Information Gain – Example 1

- S = {D4,D5,D6,D10,D14}
  - D4: Temp=mild,Humi=high,wind=weak: YES
  - D5: Temp=cool,Humi=normal,wind=weak: YES
  - D6: Temp=cool,Humi=normal,wind=strong: NO
  - D10: Temp=mild,Humi=normal,wind=weak:YES
  - D14: Temp=mild,Humi=high,wind=strong: NO
- IG(S,Humidity) = (3/5log 5/3 + 2/5 log 5/2) – (2/5(1/2 log2 + ½ log2) + 3/5 (2/3 log 3/2 + 1/3 log 3)

32

Information Gain – Example 2

- S = {D4,D5,D6,D10,D14}
  - D4: Temp=mild,Humi=high,wind=weak: YES
  - D5: Temp=cool,Humi=normal,wind=weak: YES
  - D6: Temp=cool,Humi=normal,wind=strong: NO
  - D10: Temp=mild,Humi=normal,wind=weak:YES
  - D14: Temp=mild,Humi=high,wind=strong: NO
- IG(S,Wind) = (3/5log 5/3 + 2/5 log 5/2) – (3/5(3/3 log1 + 0 log 0) + 2/5 (0 log 0 +2/2 log 1)

33

ID3 Algorithm

- ID3(S,$\mathcal{A}$)
- Create a root node
- Time to end this branch? (next slide)
- Otherwise
  - Choose A from $\mathcal{A}$ with highest IG(S,A)
  - For each value v of A
    - Add new branch appropriate subset S_v of S
    - ID3(S_v, $\mathcal{A}$ - {A})
      - Sv is subset of instances of S which have v as value of A.

34

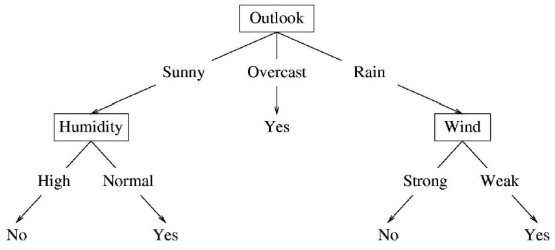
Finishing a branch

- In ID3(S, $\mathcal{A}$), we have created a node.
  - $\mathcal{A}$ Is empty (label root with most common target value)
  - All instances have same target value

    (sufficiently pure: proportion of instances in S having a value is higher than a threshold)
    - Label root with this target value

35

Resulting Tree



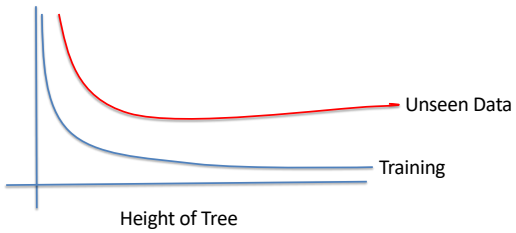
36

Tree as Set of Decision Rules

- Outlook=sunny & Humidity=high → playTennis=no
- Outlook=sunny&Humidity=normal →playTennis=yes
- Outlook = Overcast → playTennis=yes
- Outlook = Rain & Wind = strong → playTennis=no
- Outlook = Rain & Wind = weak → playTennis=yes

37

Overfitting



Number of instances at a node as tree depth increases?
Impact of Noise

38

Post-Pruning (Reduced Error Pruning)

- Pruning a (internal) node – removing subtree rooted at the node and replacing with leaf node with most common classification as label
- Use development set
- Start from leaf nodes
- Prune a node only if the resulting tree performs as good or better than original.

39

Post-Pruning Rules

- Remember each decision tree can be considered as a set of rules of the form
- If (condition 1) & … & (condition n) → class
- Pruning involves eliminating any condition
- Notice, in tree pruning we will consider the "lowest" condition first. Here any condition can be dropped.

40

Numerical Values

- Suppose Temperature value was numerical and not just hot, mild or cool.
- Suppose the 14 instances had temperature values of 90, 94, 90, 70, 54, 54, 52, 70, 56, 66, 70, 74, 94 and 66.
- Sort them: 52, 54, 56, 66,70,74,90 and 94
- Create 7 binary valued attributes based on average between consecutive values:
- temp>53, temp>55, temp>61, temp>68, temp>72, temp>82 and temp>92

41

Missing Values

Suppose there is a training instance x with a missing value, how do we compute Info Gain?

1. If S is current set of instances at node n, use the most common value for this attribute among the instances in S and use it as the value for the attribute in instance x.

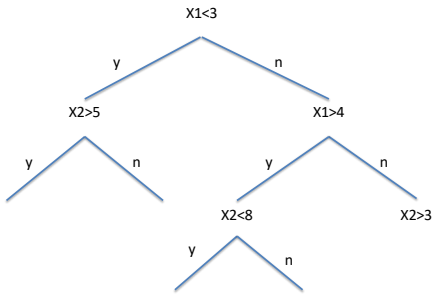
42

Missing values

- Method 2: Let |S|=s. Let the number of instances in S with value v_i of attribute is A is s_i. Then we assume that s_i/s fraction of the instance x has value v_i for the attribute for each v_i.
- For example, let a be an attribute with possible values of v_1 and v_2. Let us assume we are

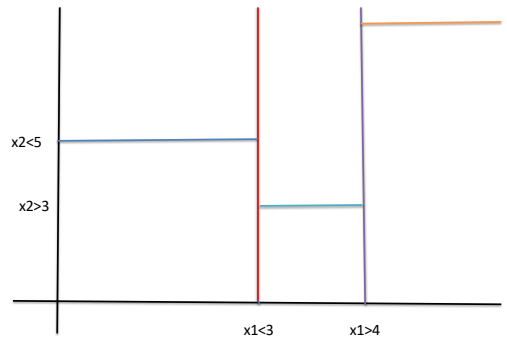
43

Decision Boundaries



44

Decision Boundaries



45

Summary

- How does a decision tree work?
  - Examine attributes sequentially.
- How to build a decision tree?
  - Select the next attribute to test.
  - ID3 algorithm: Information gain.
  - Many practical methods.

46

Summary

- Inductive bias
  - Short tree/information gain
- Overfitting
  - Pruning
- Real-valued Attributes
  - Use threshold
- Missing attribute values
  - Several common methods

47

Perceptrons

1

Modeling a Neuron

- Brain is an interconnection of nerve cells (neurons)
- Neurons have many inputs (from other neurons).
- If a neuron gets inputs such that the neuron's state value exceed a threshold, then it "fires".
- We assume that neuron's state is determined as a weighted sum of its inputs.

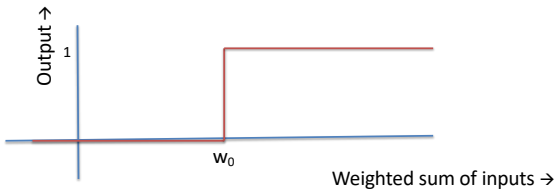
2

A Perceptron

- Has three parts:
  - Inputs
  - (Weighted) Summation
  - Threshold Function

3

Threshold Function

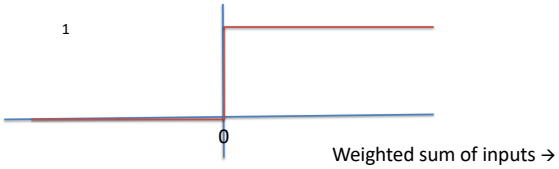


$$y = \begin{cases} 1 \text{ if } w_1x_1 + \ldots + w_kx_k > w_0 \\ 0 \text{ otherwise} \end{cases}$$

4

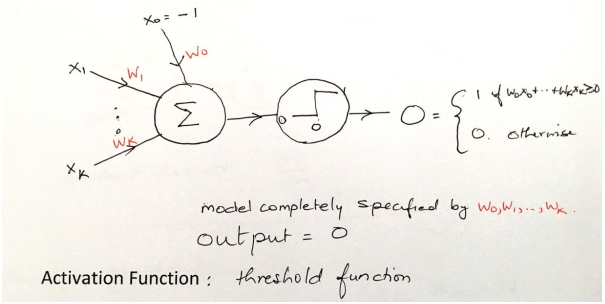
Alternatively

$$y = \begin{cases} 1 \text{ if } w_1x_1 + \ldots + w_kx_k - w_0 > 0 \\ 0 \text{ otherwise} \end{cases}$$



5

Pictorially



6

New Terms: Net and Sign

- Given an instance **x**, we define

$net(\mathbf{x}) = w_0x_0 + w_1x_1 + \ldots + w_kx_k = \mathbf{w}.\mathbf{x}$.

where $w_0 = -1$

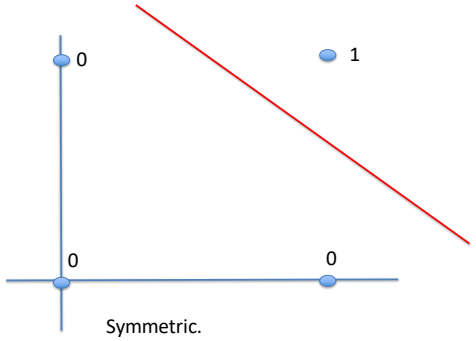
- Then output for x is

O=sign(net(x)) where

$$sign(z) = \begin{cases} 1 \text{ if } Z>0 \\ 0 \text{ otherwise} \end{cases}$$

7

AND



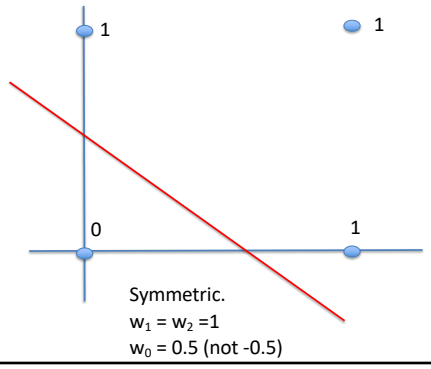
Symmetric.

$w_1 = w_2 = 1$

$w_0 = 1.5$ (not - 1.5)

8

OR (Inclusive)



Symmetric.
$w_1 = w_2 = 1$
$w_0 = 0.5$ (not -0.5)

9

Negation (not)

- not(0)=1 and not(1)=0
- Flip it by multiplying by -1 and use threshold of -0.5.

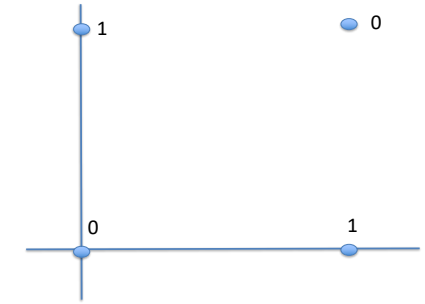
- W_0=-0.5 and w_1 = -1.

  When x_1 = 0: (-0.5)(-1) + (-1).0 = 0.5-0 >0 Hence out=1

  When x_1 = 1: (-0.5)(-1) + (-1).1 = 0.5-1 <0 Hence out=0

10

Exclusive OR (XOR)



11

Training a Perceptron

- Considering a training instance $\mathbf{x}^t$
- Based on current model ($\mathbf{w}$) let output = o^t.
- Let target value be y^t.
- If $y^t = o^t$ i.e., $(y^t - o^t)$
- No need to change the model (weights).
- Change only when $y^t - o^t > 0$ or $y^t - o^t < 0$

12

When $y^t - o^t > 0$

- y^t is fixed. Changing model can change o^t
- $o^t = \text{sign}(w_o x_0^t + w_1 x_1^t + \ldots + w_i x_i^t + \ldots w_k x_k^t)$
- Is current net^t too small or too big?
- We need to increase net^t. We will increase each of the (k+1) terms.
- We can only change the weights.
- So consider updating w_i by $w_i + \Delta w_i$.

13

Updating Weights

- Current net^t is $w_o x_0^t + w_1 x_1^t + \ldots + w_i x_i^t + \ldots w_k x_k^t$
- New $net^t = \ldots + (w_i + \Delta w_i)\, x_i^t + \ldots$
- New $net^t = \ldots + (w_i x_i^t + \Delta w_i x_i^t) + \ldots$

$(y^t - o^t)$	$w_i x_i^t + \Delta w_i x_i^t$	x_i^t	Δw_i	$(y^t - o^t)\, x_i^t$
positive	increase	positive	positive	positive
positive	increase	negative	negative	negative
negative	decrease	positive	negative	negative
negative	decrease	negative	positive	positive

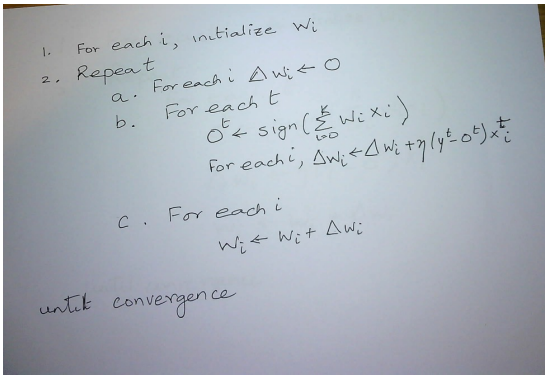
14

Perceptron Update Rule

- $\Delta w_i = \eta\, (y^t - o^t)\, x_i^t$
- Perceptron Update Rule:
  - update w_i by $w_i + \Delta w$
  - $w_i \leftarrow w_i + \eta\, (y^t - o^t)\, x_i^t$
- We are not able to use gradient descent and yet we have something like linear regression updates.
- Difference is that both y^t and o^t are 0/1.

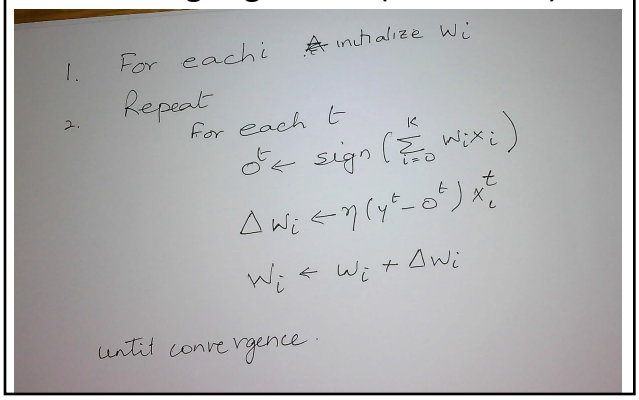
15

Training Algorithm



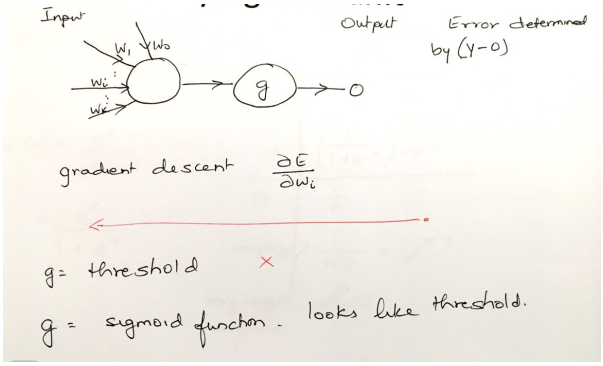
16

Training Algorithm (Stochastic)



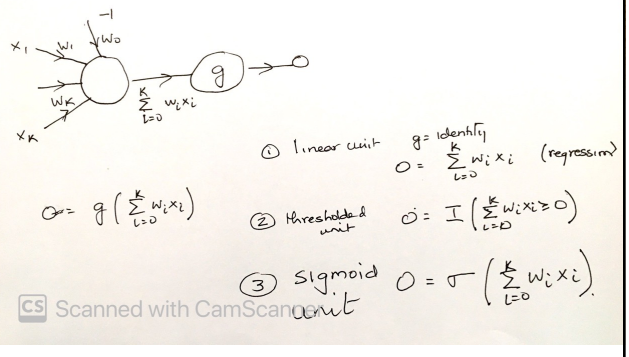
17

Why sigmoid unit



18

Activation Function and Types of Units



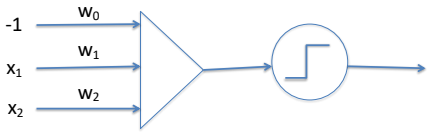
19

Multilayer Perceptron

Artificial Neural Network

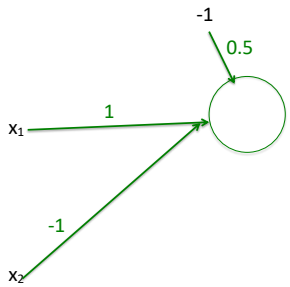
1

Perceptrons: Simple Boolean Operators



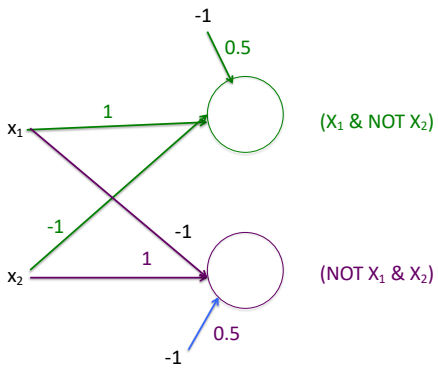
- OR: w_1=1, w_2=1 and w_0=0.5
- AND: w_1=1, w_2=1 and w_0=1.5
- NOT: w_1=-1 w_0=-0.5

2

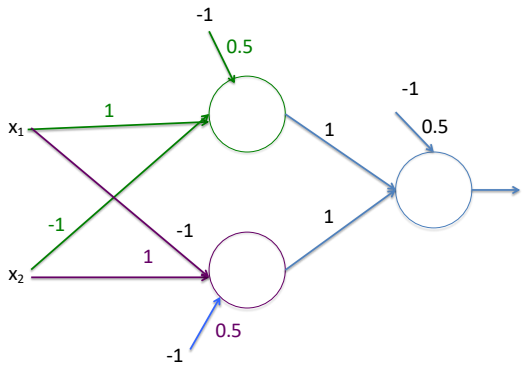


(X_1 & NOT X_2)

3



4



(X_1 & NOT X_2) OR (NOT X_1 & X_2) = (X_1 XOR X_2)

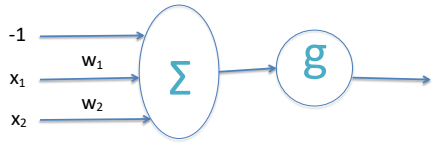
5

Non-Linearity

- Without non-linearity activation function in the hidden units, we can only get linear regression/classification – regardless of how many hidden layers there are.

7

Activation Function

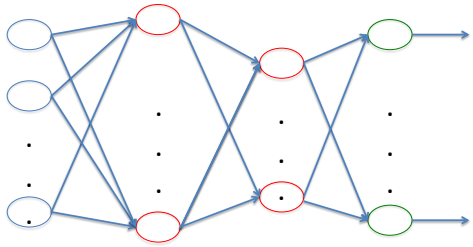


G represents the activation function:

Identity, threshold, sigmoid, tanh, relu

8

Feedforward Networks



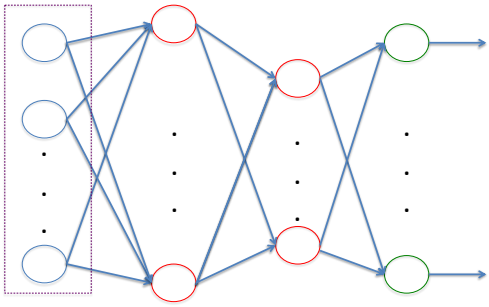
Information flow (feedforward networks)

Several Layers (no connections within a layer)

Fully connected from one layer to next

9

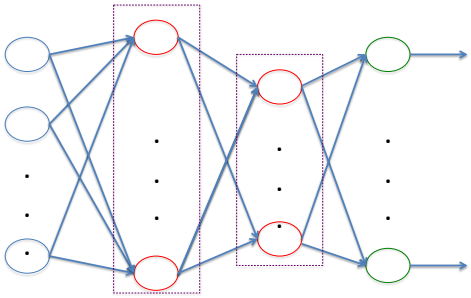
Feedforward Networks



Input Units are numbers (not real (neurons) units)

10

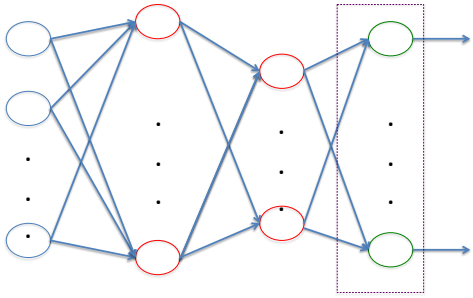
Feedforward Networks



Hidden Units are usually sigmoid/tanh units

11

Feedforward Networks



Output units: sigmoid/tanh, thresholded or linear units – latter for regression

12

ALVINN -- Autonomous Land Vehicle in Neural Network

13

ALVINN

History of Self-Driving Autos starts with ALVINN

14

ALVINN I/O

- An Autonomous Land Vehicle in Neural Networks (paper published in 1993)
- 30x32 grid of pixel intensities
- 960 input units
- 30 output units
- 4 hidden units

15

ALVINN Architecture

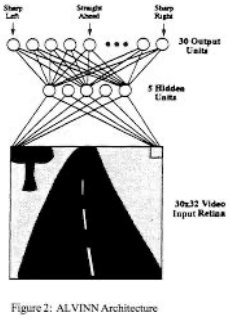


Figure 2: ALVINN Architecture

16

Another Well-Known Example of Work at CMU

17

Design Choices – An Example

- 20 human subjects, 32 images per person
- Expression: happy, sad, angry and neutral
- Direction: left, right, straight and up
- Sunglasses: yes and no

- 624 images
- Resolution 120 x 128 pixels
- Greyscale 0 (black) and 255 (white)

18

Face Recognition – Design Contd.

- 120X128 reduced to 30x32 pixels (input units)
- 256 greyscale intensities are input values between 0 and 1.
- 4 (or 2) output units with 1 of n encoding
- Each output target value is 0.1 and 0.9 instead of 0 and 1.
- One hidden layer
  - 3 hidden units – about 90% accuracy
  - 30 hidden units – about one-two % higher

19

Local Minima

- Gradient descent methods will stop at any minimum
- Issue: minimum is a local one and not a global one. So the model produced is not necessarily the best one to fit the training set.
- One possible solution: Try with several random initializations and choose the one based on development set.

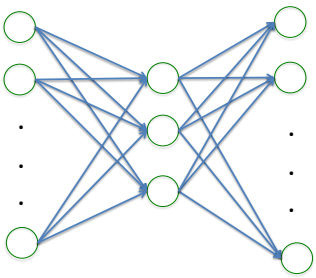
20

Representational Power

- Hidden layers allows us to beyond linear functions
- Any bounded function can be approximated with arbitrary small error by a network with two layers.
- Led to shallow (1-2 hidden layers) and wide (several hidden units) networks

22

Hidden Units



8 Training instances:
8-tuple Inputs: <1,0,0…0>, <0,1,0,…0>, … <0,0,0,…1>
Output always same as input.

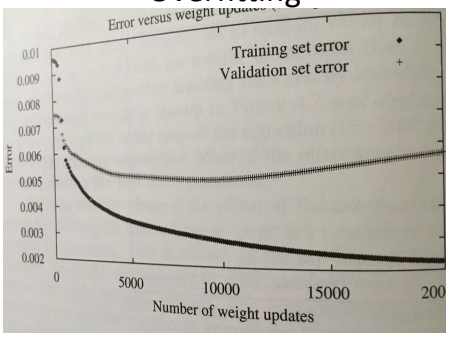
23

Auto-Encoder (Mitchell)

Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.15	.99	.99	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.01	.11	.88	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

24

Problems
Overfitting



From Mitchell's book

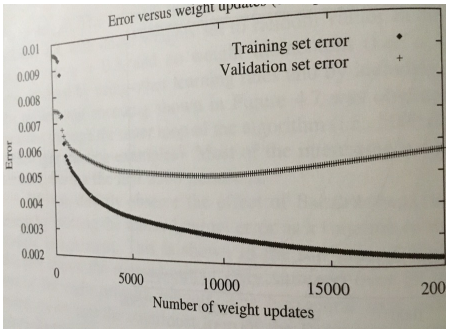
25

Overfitting & Stopping

- Why? Initial low values – linear
- Use validation set to decide when to stop
  - Use two models to track performance

26

Overfitting (Mitchell)



27

Deciding When to Stop

- **Best**, **current**: two models so far.

1. Train more to get new **current** model
2. Test on validation set and compare with **best** model's performance on validation set.
3. If **current** is better than **best**
   - Then **best** = **current**
   - Else stop?
     - Few times in a row, difference more than a threshold

28

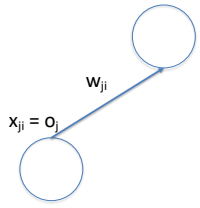
Overfitting & Stopping

- How to use n-fold cross-validation to decide when to stop.
- $Fold_1$ as validation set → M_1, … , $Fold_n$ as validation set → M_n
- How to combine them? Each model is a set of weights. Can't take average because of the dependencies of the weights.
- $Fold_1$ as validation set → $\#iterations_1$, … , $Fold_n$ as validation set → $\#iterations_n$
- Take average # of iterations and train on full data (union of all folds) for that many iterations.

29

Training of MLP

- Backpropagation Algorithm
- Page 98 Mitchell's book
- Notations:
  - w_{ji} -- weight from unit i to unit j.
  - x_{ji} -- input from unit i to unit j.
  - o_i – output of unit i



30

BackPropagation Algorithm

- For each $<\mathbf{x}^t, y^t>$ in training set, feed information forward to compute output of each unit.
- For each output unit k, compute error term δ_k
  - $\delta_k \leftarrow o_k(1 - o_k)(y_k - o_k)$
- For each hidden unit h, compute error term δ_h
  - $\delta_h \leftarrow o_h(1 - o_h)\sum_{k \in \text{outputs}} w_{kh}\, \delta_k$
- For each i,j
  - $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$ where $\Delta w_{ji} = \eta\, \delta_j\, x_{ji}$

31

For each output unit k, compute error term δ_k

$\delta_k \leftarrow - o_k(1 - o_k)(y_k - o_k)$

- $E(\mathbf{w}) = \frac{1}{2} \sum_{k \in \text{outputs}} (y_k - o_k)^2$
- $\delta_k = - \partial E/\partial net_k = - \partial E/\partial o_k \,.\, \partial o_k/\partial net_k$
- $\partial o_k/\partial net_k = \partial \sigma(net_k)/\partial net_k = o_k(1-o_k)$
- $\partial E/\partial o_k = \partial/\partial o_k \, (\frac{1}{2} \sum_{j \in \text{outputs}} (y_j - o_j)^2)$
  $= - (y_k - o_k)$
- $\delta_k = - \partial E/\partial o_k \,.\, \partial o_k/\partial net_k = (y_k - o_k)\, o_k(1-o_k)$
- $\Delta w_{kh} = - \eta\, \partial E/\partial w_{kh} = - \eta \partial E/\partial net_k \,.\, \partial net_k/\partial w_{kh}$
  $= - \eta\, (- \delta_k)\, x_{kh} = \eta\, \delta_k\, x_{kh}$

32

$\delta_h \leftarrow o_h(1 - o_h)\sum_{k \in \text{outputs}} w_{kh}\, \delta_k$

- $\delta_h = \partial E/\partial net_h$
  $= \sum_{k \in \text{down(h)}} \partial E/\partial net_k \,.\, \partial net_k/\partial net_h$
- $\partial net_k/\partial net_h = \partial net_k/\partial o_h \,.\, \partial o_h/\partial net_h$
  $= w_{kh} \,.\, \partial \sigma(net_h)/\partial net_h$
  $= w_{kh} \,.\, o_h\,(1-o_h)$
- $\delta_h = \sum_{k \in \text{down(h)}} - \delta_k\, w_{kh} \,.\, o_h\,(1-o_h)$

33