CSE 446 Machine Learning

Review

with some

Semi-Supervised Learning

& a hint of

SVMs

Daniel Weld

1

Exam

- Much like midterm, but a bit easier
- Will include one problem from midterm
- Will also include
  - Unsupervised learning
  - Reinforcement learning
  - Instance-based learning

2

Machine Learning

Study of algorithms that

- improve their performance
- at some task
- with experience

Exponential Growth in Data

Data → Machine Learning → **Understanding**

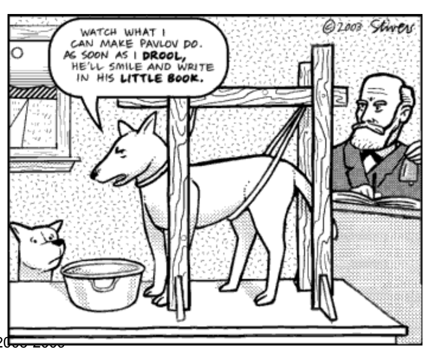
3

Supremacy of Machine Learning

- Machine learning is preferred approach to
  - Speech recognition, Natural language processing
  - Web search – result ranking
  - Computer vision
  - Medical outcomes analysis
  - Robot control
  - Computational biology
  - Sensor networks
  - …
- This trend is accelerating
  - Improved machine learning algorithms
  - Improved data capture, networking, faster computers
  - Software too complex to write by hand
  - New sensors / IO devices
  - Demand for self-customization to user, environment

4

Reinforcement Learning



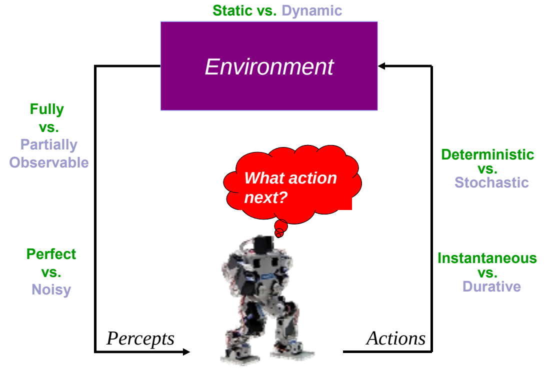
©20

5

Applications

- Robotic control
  - helicopter maneuvering, autonomous vehicles
  - Mars rover - path planning, oversubscription planning
  - elevator planning
- Game playing - backgammon, tetris, checkers
- Neuroscience
- Computational Finance, Sequential Auctions
- Assisting elderly in simple tasks
- Spoken dialog management
- Communication Networks – switching, routing, flow control
- War planning, evacuation planning

Planning Agent

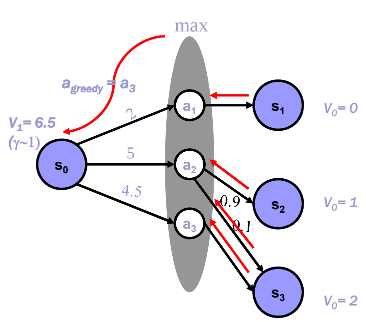


Bellman Equations for MDP

- <**S**, **A**, **Pr**, **R**, s_0, γ>
- Define **V*(s)** {optimal **value**} as the **maximum** expected **discounted reward** from this state.
- V* should satisfy the following equation:

$$V^*(s) \;=\; \max_{a \in Ap(s)} \sum_{s' \in \mathcal{S}} \mathcal{P}r(s'|s,a)\left[\mathcal{R}(s,a,s') + \gamma V^*(s')\right]$$

Bellman Backup



$Q_1(s,a_1) = 2 + \gamma\, 0$
~ 2

$Q_1(s,a_2) = 5 + \gamma\, 0.9$£ 1
+ $\gamma\, 0.1$£ 2
~ 6.1

$Q_1(s,a_3) = 4.5 + \gamma\, 2$
~ 6.5

Summary RL

- Bellman Equation
- Value iteration
- Credit assignment problem
- Explorqtion / exploitation tradeoff
  - Greedy in limit of infinite exploration
  - Optomistic exploration

Space of ML Problems

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	

12

Generalization

- Hypotheses must ***generalize*** to correctly classify instances not in the training data.
- Simply memorizing training examples is a consistent hypothesis ***that does not generalize***.

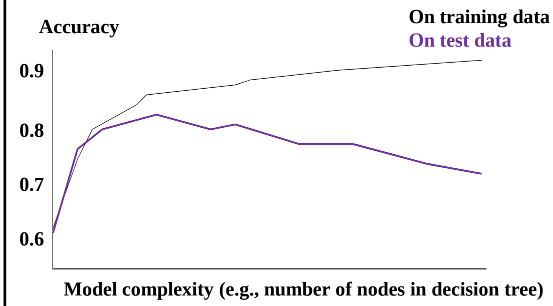
13

Learning as function approximation

- What's a ***good*** approximation?

14

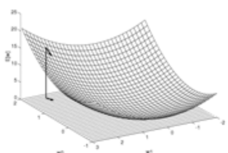
Overfitting



15

Learning as Optimization

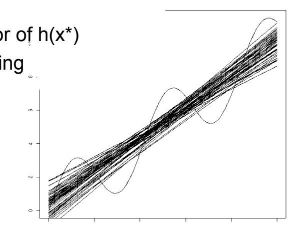
- Methods
  - Closed form
  - Greedy search
  - Gradient ascent
- Loss Function
  - Minimize ***loss*** over training data (test data)
  - Loss(h,data) = error(h, data) + complexity(h)
  - Error + regularization



16

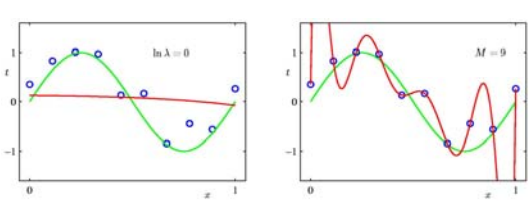
Bia / Variance Tradeoff

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

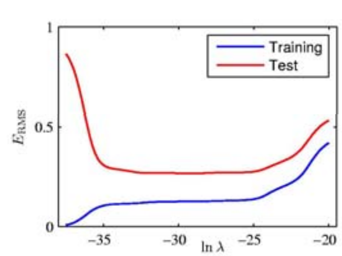


Slide from T Dietterich

Regularization



Regularization: E_{RMS} *vs.* $\ln\lambda$



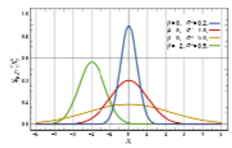
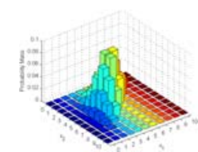
Machine Learning

- 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
- Reinforcement Learning
- Unsupervised Learning

20

Probabilities

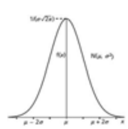
- Random variables, distributions
- Axioms of probability
- Marginal, joint & conditional probabilities
- Sum rule, product rule, Bayes rule
- Independence, conditional independence

21

Our Favorite Distributions

	Discrete: Binary {0, 1}	Discrete: M Values	Continuous
Single Event	Bernouilli		Gaussian ~ Normal
Sequence (N trials)	Binomial	Multinomial	
Conjugate Prior	↓ Beta	↓ Dirichlet	



22

Inference

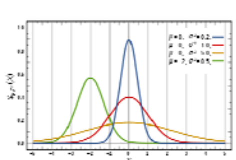
	Prior	Hypothesis
Maximum Likelihood Estimate	Uniform	The most likely
Maximum A Posteriori Estimate	Any	The most likely
Bayesian Estimate	Any	Weighted combination

Learning Gaussian Parameters

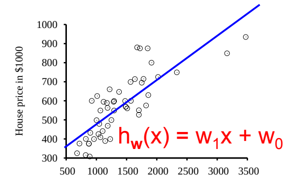
MLE:

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

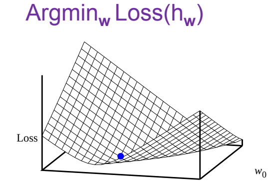
$$\hat{\sigma}^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N} (x_i - \hat{\mu})^2$$



Linear Regression



Argmin$_w$ Loss(h_w)



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

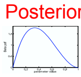
$$w_0 = (\sum(y_j) - w_1(\sum x_j))/N$$

Bayesian Learning

- Let set of categories be $\{c_1, c_2, \ldots c_n\}$
- Let E be description of an instance.
- Determine category of E by determining for each c_i

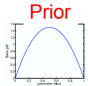
Use Bayes rule:

Posterior



Data Likelihood

Prior



$$P(c_i \mid E) = \frac{P(c_i)P(E \mid c_i)}{P(E)}$$

Normalization
Can ignore

Optimal classification

- Theorem: Bayes classifier h_{Bayes} is optimal!

$$error_{true}(h_{Bayes}) \leq error_{true}(h), \quad \forall h$$

- Why?

$$p_h(error) = \int_x p_h(error|x)p(x)$$

$$= \int_x \int_y \delta(h(x), y)p(y|x)p(x)$$

Naïve Bayes

- Naïve Bayes assumption:
  - Features are independent given class:

$$P(X_1, X_2|Y) = P(X_1|X_2, Y)P(X_2|Y)$$

  - More generally: $= P(X_1|Y)P(X_2|Y)$

$$P(X_1 \ldots X_n|Y) = \prod_i P(X_i|Y)$$

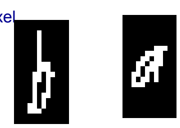
- How many parameters now?
  - Suppose **X** is composed of n binary features

Bag of Words Approach

aardvark	0
about	2
all	2
Africa	1
apple	0
anxious	0
...	
gas	1
...	
oil	1
…	
Zaire	0

What if we have continuous X_i ?

Eg., character recognition: X_i is i^{th} pixel



Gaussian Naïve Bayes (GNB):

$$P(X_i = x \mid Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} \; e^{\frac{-(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

Sometimes assume variance

- is independent of Y (i.e., σ_i),
- or independent of X_i (i.e., σ_k)
- or both (i.e., σ)

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

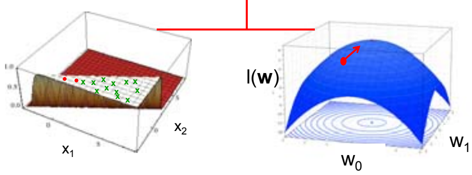
31

Logistic w/ Initial Weights

$w_0=20 \quad w_1=-5 \quad w_2=10$

Loss(H_w) = Error(H_w, data)

Minimize error → Maximize $l(\mathbf{w}) = \ln P(D_Y \mid D_x, H_w)$



Update rule:

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

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

32

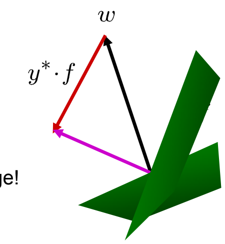
Binary Perceptron Algorithm

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

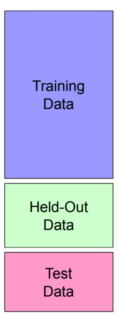
$$y = \begin{cases} +1 & \text{if } w \cdot f(x) \geq 0 \\ -1 & \text{if } w \cdot f(x) < 0 \end{cases}$$

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

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



Three Views of Classification

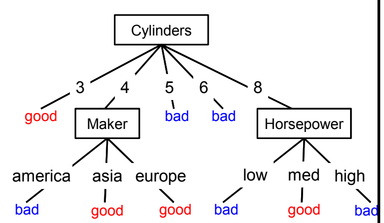


- Naïve Bayes:
  - Parameters from data statistics
  - Parameters: probabilistic interpretation
  - Training: one pass through the data
- Logistic Regression:
  - Parameters from gradient ascent
  - Parameters: linear, probabilistic model, and discriminative
  - Training: one pass through the data per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

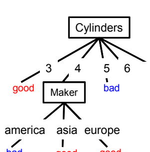
Hypotheses: decision trees $f : X \rightarrow Y$

- Each internal node tests an attribute x_i
- Each branch assigns an attribute value $x_i = v$
- Each leaf assigns a class y
- To classify input x?

  traverse the tree from root to leaf, output the labeled y



What functions can be represented?



. . .

cyl=3 ∨ (cyl=4 ∧ (maker=asia ∨ maker=europe)) ∨ …

Two Questions

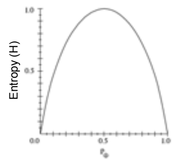
Greedy Algorithm:
- Start from empty decision tree
- Split on the **best attribute (feature)**
- Recurse

1. Which attribute gives the best split?
2. When to stop recursion?

Which attribute gives the best split?

Many answers (accuracy, misclassification rate, etc),
Most common method is:
"Attribute with the highest ***information gain,*** IG"

$$IG(X) = H(Y) - H(Y \mid X)$$



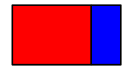
$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

$$H(Y \mid X) = -\sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i \mid X = x_j) \log_2 P(Y = y_i \mid X = x_j)$$

39

Reduced Error Pruning

Split data into ***training*** & ***validation*** sets (10-33%)



Train on training set (overfitting)
Do until further pruning is harmful:

1) Evaluate effect on validation set of pruning ***each*** possible node (and tree below it)
2) Greedily remove the node that ***most improves accuracy of validation set***

40

Ensembles of Classifiers

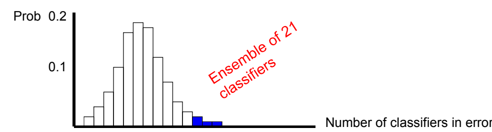
- Traditional approach: Use one classifier
- Can one do better?
- Approaches:
  - Cross-validated committees
  - Bagging
  - Boosting
  - Stacking

41

Ensembles of Classifiers

- Assume
  - Errors are independent (suppose 30% error)
  - Majority vote
- Probability that majority is wrong…

= area under binomial distribution



- If individual area is 0.3
- Area under curve for ≥11 wrong is 0.026
- Order of magnitude improvement!

42

Fighting the bias-variance tradeoff

- **Simple (a.k.a. weak) learners are good**
  - e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)
  - Low variance, don't usually overfit
- **Simple (a.k.a. weak) learners are bad**
  - High bias, can't solve hard learning problems
- Can we make weak learners always good???
  - **No!!!**
  - **But often yes…**

Boosting

[Schapire, 1989]

- Idea: given a weak learner, run it multiple times on (reweighted) training data, then let learned classifiers vote
- On each iteration t:
  - weight each training example by how incorrectly it was classified
  - Learn a hypothesis – h_t
  - A strength for this hypothesis – α_t
- Final classifier:

$$h(x) = \text{sign}\left(\sum_i \alpha_i h_i(x)\right)$$

Machine Learning

Supervised Learning | Reinforcement Learning | Unsupervised Learning

Supervised Learning → Parametric, Non-parametric

Non-parametric:

Nearest neighbor
Kernel density estimation
Support vector machines

45

k-Nearest Neighbor

Instance-based learning, four things to specify:

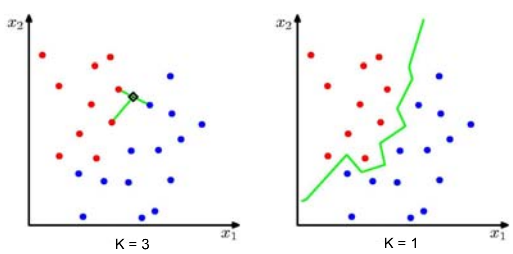
1. *A distance metric*
   Euclidian (and many more)

2. *How many nearby neighbors to look at?*
   k

1. *A weighting function (optional)*
   Unused

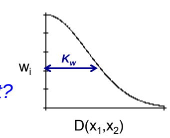
2. *How to fit with the local points?*
   Return the average output
   predict: (1/k) Σy_i (summing over k nearest neighbors)

K-Nearest-Neighbours for Classification



Kernel Regression

Instance-based learning:



1. *A distance metric*
   Euclidian (and many more)
2. *How many nearby neighbors to look at?*
   All of them
3. *A weighting function*
   $w_i = \exp(-D(x_i, query)^2 / K_w^2)$
   Nearby points to the query are weighted strongly,
   Far points weighted weakly.
   The K_W parameter is the **Kernel Width**. Very important.
4. *How to fit with the local points?*
   Predict the weighted average of the outputs:

   predict = $\Sigma w_i y_i / \Sigma w_i$

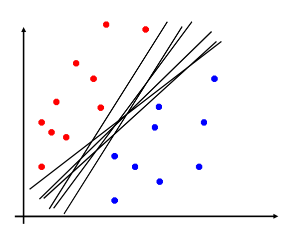
Support Vector Machines

- Key insight
  - Max Margin
- Clever trick
  - Kernel trick

49

Linear Separators

- Which of these linear separators is optimal?

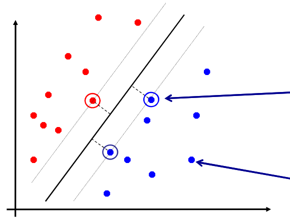


Support Vector Machines

- Maximizing the margin:
  - good according to intuition, theory, practice
- SVMs find separator with max margin
  - Convex optimization
  - Quadratic programming (off the shelf solns)
- Reduced set of features!

$$\min_w \frac{1}{2}||w||^2$$

$$\forall i, y\ w_{y^*} \cdot f(x_i) \geq w_y \cdot f(x_i) + 1$$



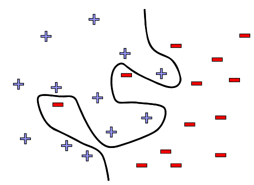
Support Vectors:
- data points on the canonical lines

Non-support Vectors:
- everything else
- moving them will not change w

What if the data is not linearly separable?

$\left\langle x_i^{(1)}, \ldots, x_i^{(m)} \right\rangle$ — m features

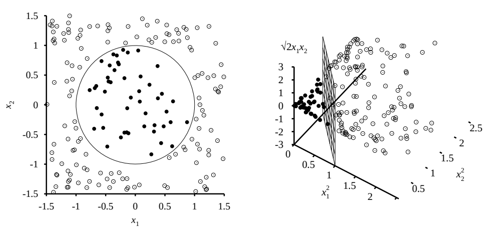
$y_i \in \{-1, +1\}$ — class



Add More Features!!!

$$\phi(x) = \begin{pmatrix} x^{(1)} \\ \ldots \\ x^{(n)} \\ x^{(1)}x^{(2)} \\ x^{(1)}x^{(3)} \\ \ldots \\ e^{x^{(1)}} \\ \ldots \end{pmatrix}$$

What if the data is not linearly separable?



2D → 3D, using new features: $F(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}\, x_1 x_2)$

Dual SVM Formulation

Derivation requires computing Lagrangian & some advanced math

Notes:
- One α for each training example

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

Sums over all training examples

scalars

dot product

Kernel trick:
Can compute F(x)·F(x') without computing F(x) or F(x') in many cases

Overfitting?

- Huge feature space with kernels, what about overfitting???
  - Maximizing margin leads to sparse set of support vectors
  - Some interesting theory says that SVMs search for simple hypothesis with large margin
  - Often robust to overfitting
    - But everything overfits sometimes!!!
    - Can control by choice of Kernel

Machine Learning

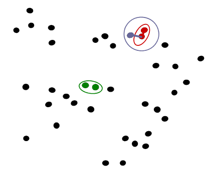
- Supervised Learning
  - Parametric
  - Non-parametric
- Reinforcement Learning
- **Unsupervised Learning**
  - Agglomerative Clustering
  - K-means
  - Expectation Maximization (EM)
  - Principle Component Analysis (PCA)

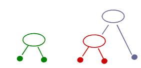
56

Example: K-Means for Segmentation

Agglomerative Clustering

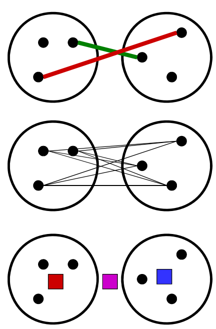
- Agglomerative clustering:
  - First merge very similar instances
  - Incrementally build larger clusters out of smaller clusters
- Algorithm:
  - Maintain a set of clusters
  - Initially, each instance in its own cluster
  - Repeat:
    - Pick the two closest clusters
    - Merge them into a new cluster
    - Stop when there's only one cluster left
- Produces not one clustering, but a family of clusterings represented by a dendrogram





Agglomerative Clustering

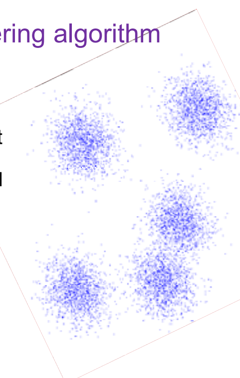
- How should we define "closest" for clusters with multiple elements?
- Many options:
  - Closest pair (single-link clustering)
  - Farthest pair (complete-link clustering)
  - Average of all pairs
  - Ward's method (min variance, like k-means)
- Different choices create different clustering behaviors



K-Means

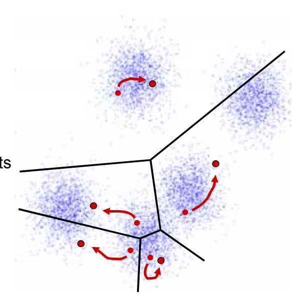
another iterative clustering algorithm

- Pick K random cluster models
- Alternate:
  - Assign data instances to different models
  - Revise each cluster model based on its assigned points
- Stop when no changes



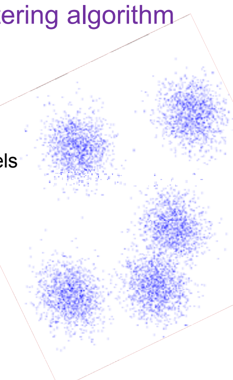
K-Means

- An iterative clustering algorithm
  - Pick K random points as cluster centers (means)
  - Alternate:
    - Assign data instances to closest mean
    - Assign each mean to the average of its assigned points
  - Stop when no points' assignments change

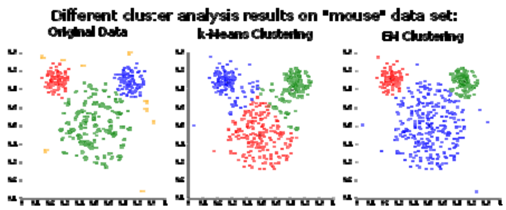


EM another iterative clustering algorithm

- Pick K random cluster models
- Alternate:
  - Assign data instances **proportionately** to different models
  - Revise each cluster model based on its **proportionately** assigned points
- Stop when no changes



Preference on Cluster Sizes

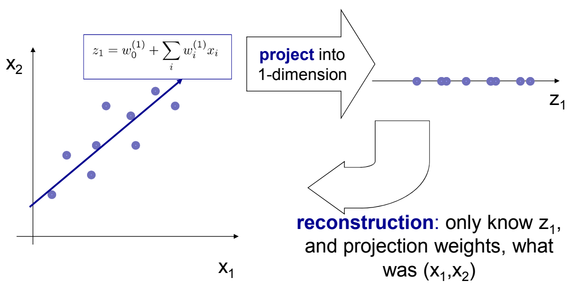


63

Feature Selection

- Want to learn f:**X**→Y
  - **X**=<X_1,…,X_n>
  - but some features are more important than others
- **Approach**: select subset of features to be used by learning algorithm
  - **Score** each feature (or sets of features)
  - **Select** set of features with best score

Linear projection and reconstruction



Basic PCA algorithm

- Start from m by n data matrix **X**
- **Recenter**: subtract mean from each row of **X**
  - $\mathbf{X}_c \leftarrow \mathbf{X} - \bar{\mathbf{X}}$
- **Compute covariance** matrix:
  - $\Sigma \leftarrow 1/m\ \mathbf{X}_c^T\ \mathbf{X}_c$
- Find **eigen vectors and values** of Σ
- **Principal components:** k eigen vectors with highest eigen values

Machine Learning

Supervised Learning
Unsupervised Learning
Reinforcement Learning

67

Co-Training Motivation

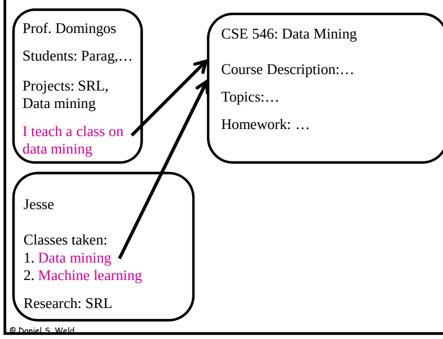
- Learning methods need labeled data
  - Lots of <x, f(x)> pairs
  - Hard to get… (who wants to label data?)
- But unlabeled data is usually plentiful…
  - Could we use this instead??????
- Semi-supervised learning

68

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
  - ∃f1, f2 such that f1(x1) ~ f2(x2) ~ f(x)
- Both f1, f2 are learnable
  - f1 ∈ H1, f2 ∈ H2, ∃ learning algorithms A1, A2

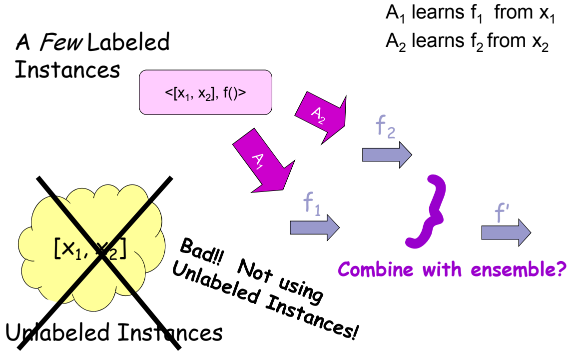
Co-training Example



Without Co-training

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

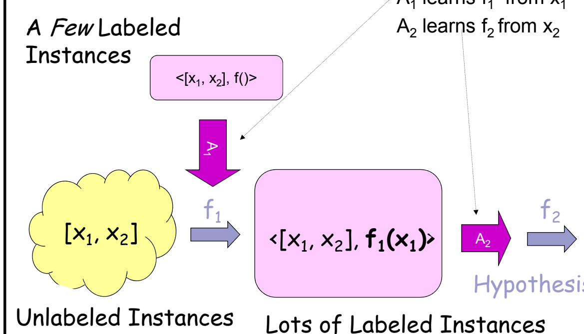
A_1 learns f_1 from x_1
A_2 learns f_2 from x_2



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



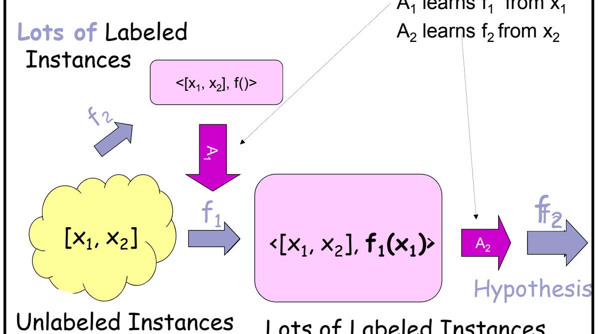
Observations

- 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

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



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.