

Summary: Supervised Learning

Greg Grudic

Machine Learning

1

Summary Topics

- Supervised Learning
 - Model Selection (i.e. Learning Parameters)
 - Frequentist and Bayesian
 - Learning algorithm evaluation
 - Assumptions on Data
 - Generative and Discriminative Classifiers
- Supervise Learning Algorithms

Machine Learning

2

Supervised Learning

- Given: Training examples
 $\{(\mathbf{x}_1, \mathbf{f}(\mathbf{x}_1)), (\mathbf{x}_2, \mathbf{f}(\mathbf{x}_2)), \dots, (\mathbf{x}_p, \mathbf{f}(\mathbf{x}_p))\}$
of some unknown function (system) $\mathbf{y} = \mathbf{f}(\mathbf{x})$
- Find $\hat{\mathbf{f}}(\mathbf{x})$ (i.e. an approximation)
 - Predict $\mathbf{y}' = \hat{\mathbf{f}}(\mathbf{x}')$, where \mathbf{x}' is not in the training set

Machine Learning

3

Two Types of Supervised Learning

- **Classification** $\mathbf{y} \in \{c_1, c_2, \dots, c_N\}$
 - Model output is a prediction that the input belongs to some class
 - If the input is an image, the output might be chair, face, dog, boat, ... etc.
- **Regression** $\mathbf{y} \in \mathfrak{R}$
 - The output has infinitely many values
 - If the input is stock features, the output could be a prediction of tomorrow's stock price

Machine Learning

4

Goal of Supervised Learning

- Build a model that does best on Future Data!

Assumptions on Regression Data

- Data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ where $\mathbf{x}_i \in \mathbb{R}^d$ are independently identically distributed (iid) from $D(\mathbf{x})$ and $y \in \mathbb{R}$ are generated from

$$y_i = f(\mathbf{x}_i) + \rho$$

- where $f(\mathbf{x}) \in \mathbb{R}$ is a real valued function defined on $\mathbf{x} \in \mathbb{R}^d$ and ρ is a random variable

$$E[\rho] = 0, \quad V[\rho] = c, \quad c \in \mathbb{R}, \quad 0 \leq c < \infty$$

Assumptions on Classification Data

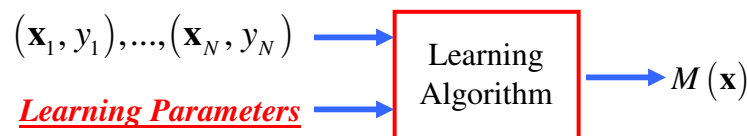
- Assume data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y \in \{c_1, \dots, c_K\}$ (i.e. K classes). The prior probability of each class is p_k and each class is iid from a pdf $h_k(\mathbf{x})$. Then the posterior probability of class c_k given \mathbf{x} is

$$\Pr(y = c_k | \mathbf{x}) = \frac{p_k h_k(\mathbf{x})}{\sum_{i=1}^K p_i h_i(\mathbf{x})}$$

Machine Learning

7

Building Supervised Learning Models: Frequentist Model Selection



Model is used to make predictions! $\hat{y} = M(\mathbf{x})$

Machine Learning

8

Learning Parameters

- These dictate how the learning algorithm will build a model
- Changing the learning parameters changes how good the model is
- **Goal:** Choose the learning parameters that produce the best model

Machine Learning

9

Measuring Model Accuracy: Regression

- Assume a set of data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_K, y_K)$
- Regression accuracy
 - Two commonly used metrics

- Mean Square Error

$$error_{M(\mathbf{x})} = \frac{1}{K} \sum_{i=1}^K (y_i - M(\mathbf{x}_i))^2 = \frac{1}{K} \sum_{i=1}^K (y_i - \hat{y}_i)^2$$

- Relative Error

$$error_{M(\mathbf{x})} = \frac{\sum_{i=1}^K (y_i - M(\mathbf{x}_i))^2}{\sum_{i=1}^K (y_i - \bar{y})^2}$$

Machine Learning

10

Measuring Model Accuracy: Classification

- Assume a set of data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_K, y_K)$
- Classification accuracy

$$error_{M(\mathbf{x})} = \frac{1}{K} \sum_{i=1}^K c(\mathbf{x}_i, y_i, M(\mathbf{x}_i))$$

$$\text{Where } c(\mathbf{x}_i, y_i, M(\mathbf{x}_i)) = \begin{cases} 0 & \text{if } y_i = M(\mathbf{x}_i) \\ 1 & \text{otherwise} \end{cases}$$

Picking the Best Learning Parameters

- Partition learning data into **disjoint sets**
 - Training Set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)$
 - Used to build the model
 - Validation Set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_V, y_V)$
 - Used to evaluate model
- Pick the Learning Parameters that give the lowest error on the Validation Set

$$error_{M(\mathbf{x})} = \frac{1}{V} \sum_{i=1}^V c(\mathbf{x}_i, y_i, M(\mathbf{x}_i))$$

How Big Should the Training and Validation Sets Be?

- It Depends...
- If you have **Lots** of data for learning
 - Randomly putting half the data into each set is often sufficient
- If you only have a **Small** data set for learning
 - Usually do N-Fold Cross Validation

N-Fold Cross-Validation

- Partition the data $D_0 = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_M, y_M)\}$ into N disjoint sets T_1, \dots, T_N
- For i from 1 to N, do
 - Use T_i for validation and the remaining S_i for training
 - Training Set: $S_i = \{D_0 - T_i\}$
 - Error on validation T_i : $error_{T_i}$
- Return the average error on validation sets

$$error_{M(x)} = \frac{1}{N} \sum_{i=1}^N error_{T_i}$$

Pick the learning parameters that minimize this error!

Does My Cross Validation Error Reflect the True Error of My Model?

- **No!!!!!!!!!!!!!!!!!!!!!!**
- Need to do randomized experiments
 - e.g. 100 experiments
 - 90% data for learning (use cross validation on this set to pick learning parameters)
 - 10% for testing
 - Report average test error over the 100 experiments

Bayesian Model Selection

- Pick the hypothesis that has maximum probability given the data – **Bayes Theorem:**

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- $P(h)$ = prior probability of hypothesis h
 - $P(D)$ = prior probability of training data D
 - $P(h|D)$ = probability of h given D
 - $P(D|h)$ = probability of D given h
- **Learning parameters are chosen to maximize the probability of the hypothesis given the data**

Generative and Discriminative Classifiers

- **Generative Classifier Models:** model the distributions that generate the data (e.g. Bayesian density models.)

$$\hat{y} = \arg \max_k \{ \hat{p}_k \hat{h}_k(\mathbf{x}) \}$$

- **Discriminative classifier Models:** model only the boundaries (e.g. trees, SVMs, Nearest Neighbor, Neural Networks, etc.)

Supervised Learning Algorithms I

- Linear Regression
- Ridge Regression
 - Linear and Kernel
- Lasso Regression
 - Linear and Kernel
- Perceptron Classification
- Support Vector Machines
 - Classification and Regression

Supervised Learning Algorithms II

- K Nearest Neighbors
 - Classification and Regression
- Decision Trees
 - 1-R stump
- Neural Networks
 - Classification and Regression
- Bagging
 - Classification and Regression
- Random Forests
 - Classification and Regression
- Boosting Classifiers

Machine Learning

19

Linear Regression

- Main Assumptions:
 - Linear weighted sum of attribute values.
 - Attributes and target values are real valued.
- Hypothesis Space
 - Fixed size (parametric) : Limited modeling potential
- Can be made non-linear using basis functions (now linear in basis function space)

$$y = \sum_{i=1}^d \beta_i x_i + \beta_0$$

$$y = \sum_{i=1}^K \beta_i \phi_i(x) + \beta_0$$

Machine Learning

20

Linear Regression Learning Algorithms

- Minimum Least Square Error

$$\hat{\beta}^{MSE} = \arg \min_{\beta} \left\{ \sum_{i=1}^N \left(y_i - \beta_0 - \sum_{j=1}^d \beta_j x_{ij} \right)^2 \right\}$$

- Ridge Regression

$$\hat{\beta}^{ridge} = \arg \min_{\beta} \left\{ \sum_{i=1}^N \left(y_i - \beta_0 - \sum_{j=1}^d \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^d \beta_j^2 \right\}$$

- Lasso

$$\hat{\beta}^{lasso} = \arg \min_{\beta} \left\{ \sum_{i=1}^N \left(y_i - \beta_0 - \sum_{j=1}^d \beta_j x_{ij} \right)^2 \right\}$$

$$\text{subject to: } \sum_{j=1}^d |\beta_j| \leq s, \quad s > 0$$

Machine Learning

21

Linear Regression Summary

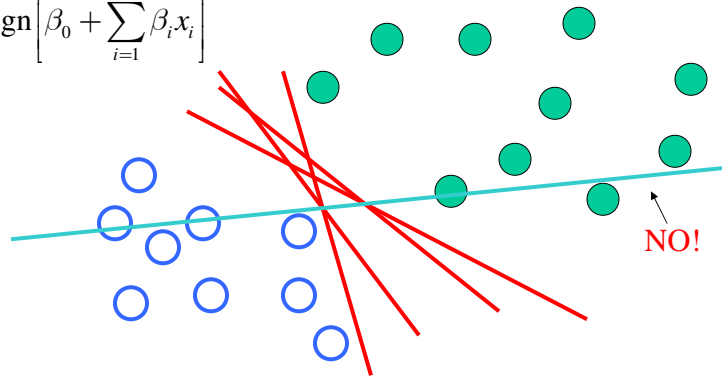
- Good points
 - Does feature selection - LASO
- Bad points
 - Slow learning on very large datasets (>20,000)
- Software
 - WEKA:
<http://www.cs.waikato.ac.nz/~ml/weka/index.html>
 - LARS: <http://www-stat.stanford.edu/~hastie/Papers/LARS/>

Machine Learning

22

Perceptron Algorithm: Finds a Linear Separating Hyper-Plane

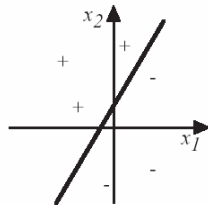
$$y = \text{sgn} \left[\beta_0 + \sum_{i=1}^d \beta_i x_i \right]$$



Machine Learning

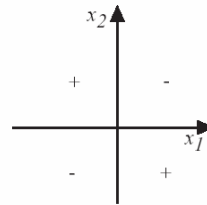
23

Linear Hyperplanes



(a)

Linearly Separable



(b)

Not Linearly Separable

Machine Learning

24

Nonlinear Perceptron Algorithm

- Use a nonlinear basis function space

$$y = \text{sgn} \left[\beta_0 + \sum_{i=1}^K \beta_i \varphi_i(\mathbf{x}) \right]$$

- Basis functions can be kernels

Perceptron Algorithm

- Works by gradient descent

$$\boldsymbol{\beta} = \boldsymbol{\beta} - \rho \left[\frac{\partial L}{\partial \boldsymbol{\beta}} \right]$$

$$L(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_d) = - \sum_{i \in M} y_i \left(\hat{\beta}_0 + (\hat{\beta}_1, \dots, \hat{\beta}_d) \mathbf{x}^T \right)$$

where M is the set of misclassified training examples

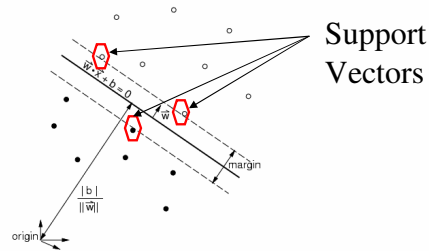
Perceptron Summary

- Good points
 - Convergence guaranteed if problem is separable
 - In basis function space or linear space
 - Works on large data sets
 - Algorithm works by gradient descent
- Bad points
 - Won't converge if data isn't separable
- Learning Parameters
 - Learning rate, choice of nonlinear basis functions,...

Support Vector Machines

- Main Assumption:
 - Build a model using minimal number of training instances (Support Vectors).
- Hypothesis Space
 - Variable size (nonparametric): Can model any function given the right kernels
 - e.g. Gaussian

Linear Support Vector Machines



We'd like the hyperplane with maximum margin

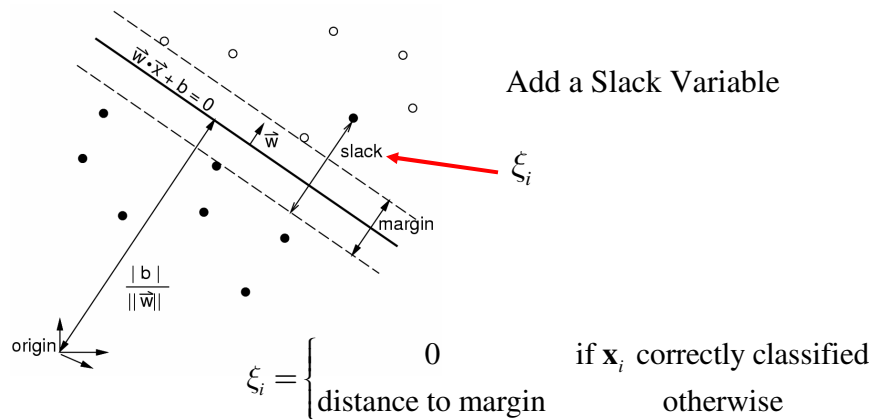
- size of margin is $\frac{2}{\|\vec{w}\|}$

So view our problem as a constrained optimization problem:

Minimize $\|\vec{w}\|^2$, subject to

$$y_i(\vec{x}_i \cdot \vec{w} + b) - 1 \geq 0, (\forall i)$$

What Happens When Data is Not Separable: **Soft Margin SVM**



Soft Margin SVM: Constraint Optimization Problem

- Given data:

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$$

- Minimize $\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^N \xi_i$ subject to:

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i = (1, \dots, N)$$

Dual Problem (Non-separable data)

- Maximize

$$W(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

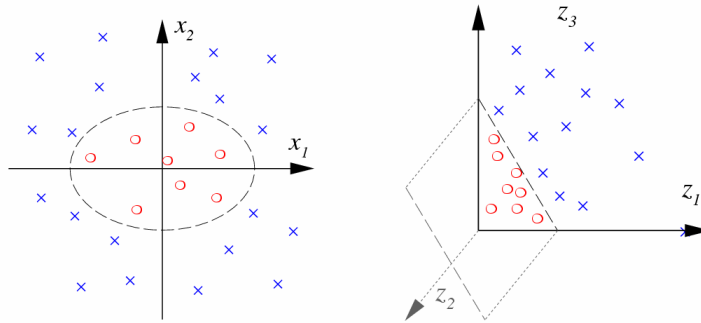
- Subject to

$$0 \leq \alpha_i \leq C, \quad i = 1, \dots, N$$

$$\sum_{i=1}^N \alpha_i y_i = 0$$

Mapping into Nonlinear Space

$$\begin{aligned}\Phi: \mathbb{R}^2 &\rightarrow \mathbb{R}^3 \\ (x_1, x_2) &\mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)\end{aligned}$$



Machine Learning

33

Kernel Trick

Replace $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$

with

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$$

Can use the same algorithms in nonlinear kernel space!

Machine Learning

34

Nonlinear SVMs

Maximize:

$$W(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

Boundary:

$$f(\mathbf{x}) = \text{sgn} \left(\sum_{i=1}^N \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right)$$

Need Mercer Kernels

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \\ &= \langle \Phi(\mathbf{x}_j), \Phi(\mathbf{x}_i) \rangle \\ &= K(\mathbf{x}_j, \mathbf{x}_i) \end{aligned}$$

Gram (Kernel) Matrix

Training Data: $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$

$$K = \begin{pmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & \dots & K(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x}_N, \mathbf{x}_1) & \dots & K(\mathbf{x}_N, \mathbf{x}_N) \end{pmatrix}$$

Properties:

- Positive Definite Matrix
- Symmetric
- Positive on diagonal
- N by N

Machine Learning

37

Commonly Used Mercer Kernels

- Polynomial

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + c)^d$$

- Sigmoid

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \theta)$$

- Gaussian

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

Machine Learning

38

SV Regression: ε -Insensitive Loss

[64]

Goal: generalize SV pattern recognition to regression, preserving the following properties:

- formulate the algorithm for the linear case, and then use kernel trick
- sparse representation of the solution in terms of SVs

ε -Insensitive Loss:

$$|y - f(\mathbf{x})|_{\varepsilon} := \max\{0, |y - f(\mathbf{x})| - \varepsilon\}$$

Estimate a linear regression $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$ by minimizing

$$\frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{m} \sum_{i=1}^m |y_i - f(\mathbf{x}_i)|_{\varepsilon}.$$

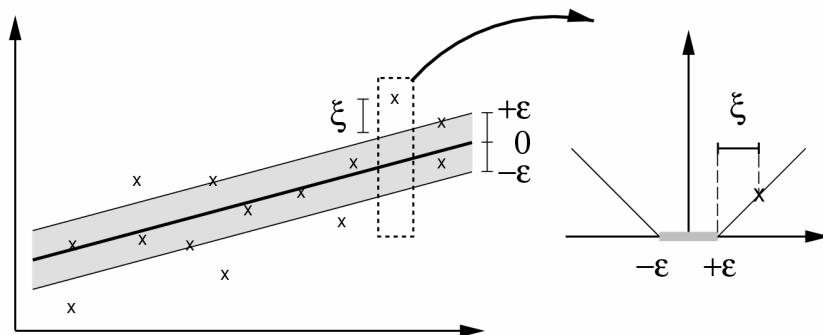
B. Schölkopf, Canberra, February 2002

Machine Learning

39

ε -SV Regression Estimation

[64]



Machine Learning

40

Formulation as an Optimization Problem

Estimate a linear regression

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

with precision ε by minimizing

$$\begin{aligned} \text{minimize} \quad & \tau(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\xi}^*) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m (\xi_i + \xi_i^*) \\ \text{subject to} \quad & (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - y_i \leq \varepsilon + \xi_i \\ & y_i - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0 \end{aligned}$$

for all $i = 1, \dots, m$.

B. Schölkopf, Canberra, February 2002

Machine Learning

41

Dual Problem, In Terms of Kernels

For $C > 0, \varepsilon \geq 0$ chosen a priori,

$$\begin{aligned} \text{maximize} \quad & W(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = -\varepsilon \sum_{i=1}^m (\alpha_i^* + \alpha_i) + \sum_{i=1}^m (\alpha_i^* - \alpha_i) y_i \\ & - \frac{1}{2} \sum_{i,j=1}^m (\alpha_i^* - \alpha_i) (\alpha_j^* - \alpha_j) k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} \quad & 0 \leq \alpha_i, \alpha_i^* \leq C, \quad i = 1, \dots, m, \quad \text{and} \quad \sum_{i=1}^m (\alpha_i - \alpha_i^*) = 0. \end{aligned}$$

The regression estimate takes the form

$$f(\mathbf{x}) = \sum_{i=1}^m (\alpha_i^* - \alpha_i) k(\mathbf{x}_i, \mathbf{x}) + b,$$

B. Schölkopf, Canberra, February 2002

Machine Learning

42

SVM Summary

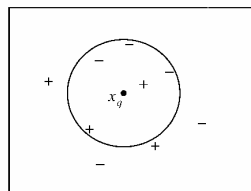
- Good points
 - Picks a subset of the data that explains it all
 - Gives good models
- Bad points
 - Slow on large datasets (<20,000)
 - Difficult to pick good kernels
- Software
 - LIBSVM, SVMlight, etc

Machine Learning

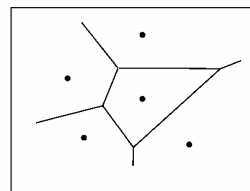
43

K Nearest Neighbor

- Main Assumption:
 - An effective distance metric exists.
- Hypothesis Space
 - Variable size (nonparametric): Can model any function



*Classify according to
Nearest Neighbors*



*Separates the input
space*

Machine Learning

44

Nearest Neighbor Algorithm

- Given training data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$
- Define a distance metric between points in inputs space. Common measures are:
 - Euclidean (squared) $D(\mathbf{x}, \mathbf{x}_i) = \sum_{j=1}^d (x_j - x_{i,j})^2$
 - Weighted Euclidean $w_j \geq 0$

$$D(\mathbf{x}, \mathbf{x}_i) = \sum_{j=1}^d w_j (x_j - x_{i,j})^2$$

Machine Learning

45

K-Nearest Neighbor Model

- Given test point \mathbf{x}
- Find the K nearest training inputs $\mathbf{x}_1, \dots, \mathbf{x}_N$ to \mathbf{x} given the distance metric $D(\mathbf{x}, \mathbf{x}_i)$

- Denote these points as

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_K, y_K)$$

Machine Learning

46

K-Nearest Neighbor Model

- Regression:

$$\hat{y} = \frac{1}{K} \sum_{k=1}^K y_k$$

- Classification:

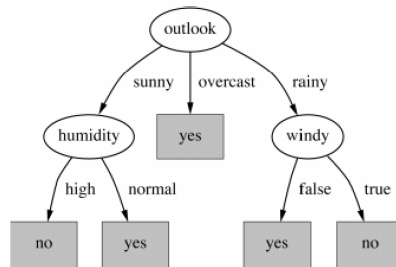
$\hat{y} =$ most common class in set $\{y_1, \dots, y_K\}$

K Nearest Neighbor Summary

- Good points
 - Fast learning
 - Can learn anything
 - Trivial to implement
- Bad points
 - Slow predictions for large datasets
 - Distance metric is critical
- Software
 - WEKA:
<http://www.cs.waikato.ac.nz/~ml/weka/index.html>

Decision Trees

- Main Assumption:
 - Data effectively modeled via decision splits on attributes.
- Hypothesis Space
 - Variable size (nonparametric): Can model any function

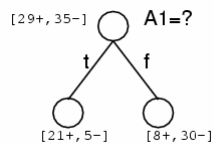


Machine Learning

49

1-R (A Decision Tree Stump)

- Main Assumptions
 - Only one attribute is necessary.
 - Finite number of splits on the attribute.
- Hypothesis Space
 - Fixed size (parametric): Limited modeling potential



Machine Learning

50

Decision Tree Summary

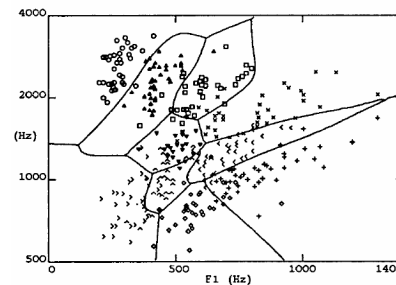
- Good points
 - Does feature selection
 - Models can be interpreted
 - Works on large data sets
- Bad points
 - Model performance often not best
- Learning Parameters
 - Depth of tree, choice of splitting function, pruning, etc...
- Software
 - WEKA: <http://www.cs.waikato.ac.nz/~ml/weka/index.html>

Machine Learning

51

Neural Networks

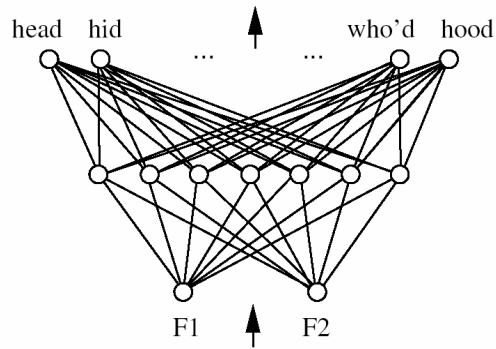
- Main Assumption:
 - Many simple functional units, combined in parallel, produce effective models.
- Hypothesis Space
 - Variable size (nonparametric): Can model any function



Machine Learning

52

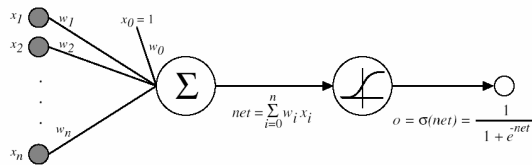
Neural Networks



Machine Learning

53

Sigmoid Unit



$\sigma(x)$ is the sigmoid function $\frac{1}{1+e^{-x}}$

Nice property: $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

We can derive gradient decent rules to train

- One sigmoid unit
- *Multilayer networks* of sigmoids \rightarrow Backpropagation

Machine Learning

54

Neural Networks Summary

- Good points
 - Can work on very large data sets
 - Gradient descent
- Bad points
 - Not easy to interpret
 - Can be difficult to choose a topology
- Software
 - WEKA: <http://www.cs.waikato.ac.nz/~ml/weka/index.html>
 - LUSH: <http://lush.sourceforge.net/>

Bagging

- Main Assumption:
 - Combining many unstable predictors to produce a ensemble (stable) predictor.
 - Unstable Predictor: small changes in training data produce large changes in the model.
 - e.g. Neural Nets, trees
 - Stable: SVM (sometimes), nearest Neighbor.
- Hypothesis Space
 - Variable size (nonparametric): Can model any function

Bagging (continued)

- Each predictor in ensemble is created by taking a bootstrap sample of the data.
- Bootstrap sample of N instances is obtained by drawing N example at random, with replacement.
- On average each bootstrap sample has 63% of instances
 - Encourages predictors to have uncorrelated errors.

Bagging Reborn: Random Forests

- Injecting the right kind of randomness makes accurate models
 - As good as SVMs and sometimes better
 - As good as boosting
- **Very little playing with learning parameters is needed to get very good models**
- Traditional tree algorithms spend a lot of time choosing how to split at a node
 - Random forest trees put very little effort into this

R.I. (Random Input) Forests

- For K trees:
 - Build each tree by:
 - Selecting, at random, at each node a small set of features (F) to split on. Common values of F are
$$F = 1$$
$$F = \log(M) + 1$$
 - For each node split on the best of this subset
 - Grow tree to full length

Regression: average over trees

Classification: vote

R.C. (Random Combination) Forests

- For K trees:
 - Build each tree by:
 - Create F random linear sums of L variables

$$A_f = \sum_{i=1}^L b_{fi} x_{ind(i)}, \dots, A_F = \sum_{i=1}^L b_{Fi} x_{ind(i)}$$

$$b_{fi} = \text{uniform random } [-1, 1]$$

- At each node split on the best of these linear boundaries
- Grow tree to full length

Regression: average over trees

Classification: vote

Random Forests Summary

- Good points
 - Fast learning
 - Can learn anything
 - Feature weighting
 - Very little parameter tuning
 - Gives a good estimate of model accuracy
 - Uses out of bag examples
- Bad points
 - Models are huge
- Software
 - <http://www.stat.berkeley.edu/users/breiman/RandomForests/>

Boosting

- Main Assumption:
 - Combining many weak predictors (e.g. tree stumps or 1-R predictors) to produce an ensemble predictor.
- Hypothesis Space
 - Variable size (nonparametric): Can model any function
 - However, some functions may not be learnable

Boosting (Continued)

- Each predictor is created by using a biased sample of the training data
 - Instances (training examples) with high error are weighted higher than those with lower error
- Difficult instances get more attention

Boosting Summary

- Good points
 - Fast learning
 - Can learn anything
 - Feature weighting
 - Very little parameter tuning
- Bad points
 - Can overfit data
- Software
 - <http://www-stat.stanford.edu/~jhf/R-MART.html>