Quick Primer on Machine Learning: Supervised Learning

Giri Iyengar

Cornell University

gi43@cornell.edu

Jan 29, 2018

Overview

1. Machine Learning
2. Regression
   - kNN
   - Linear Regression
   - Kernel Regression
3. Classification
   - Naive Bayes
   - Logistic Regression
   - Support Vector Machines
   - Random Forests
   - Gradient Boosting Machines
4. Assignments, Weekly Reading

Overview

Machine Learning

- Typically 3 different types of tasks
- Regression: $y = f(\mathbf{x}), y \in \mathbb{R}$
- Classification: $y = f(\mathbf{x}), y \in \Omega$
- Unsupervised Learning: Infer hidden structured of unlabeled data

Machine Learning: Common Considerations

- Training Dataset
- Validation of the model. What cost function?
- Test Data
- Dealing with Model Complexity
- Tuning parameters

Overview

1. Machine Learning
2. Regression
   - kNN
   - Linear Regression
   - Kernel Regression
3. Classification
   - Naive Bayes
   - Logistic Regression
   - Support Vector Machines
   - Random Forests
   - Gradient Boosting Machines
4. Assignments, Weekly Reading

kNN Regression

- One of the simplest models. Can be used as a classifier too.
- Regresses/Classifies every new point by *querying* k Nearest Neighbors
- Returns average prediction (for regression) or majority vote (for classification)
- Has strong consistency proofs
- **What are some design factors with this approach?**
- **Any issues/problems?**

Linear Regression

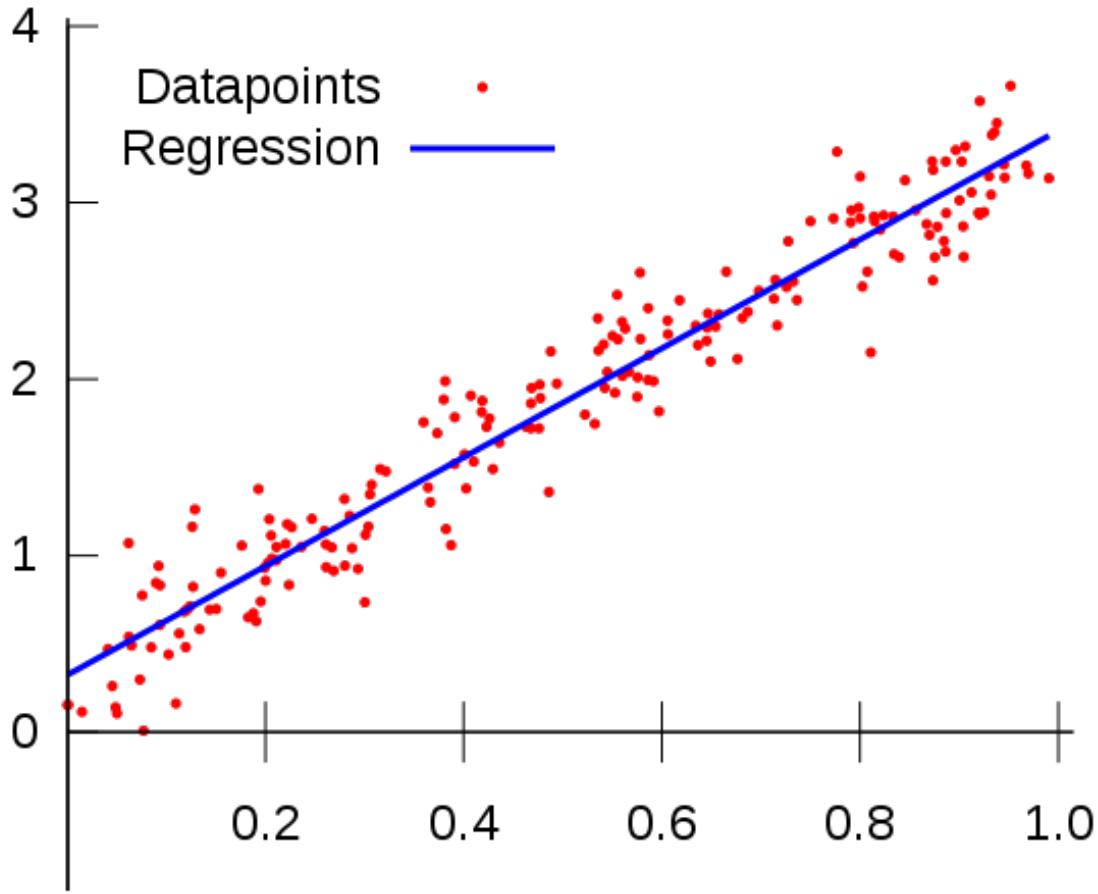

Figure: Simple Linear Regression

Linear Regression

- $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k$
- RMSE: $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2)}$
- **When does the error reach its minimum?**

Kernels: Cubic Spline Interpolation

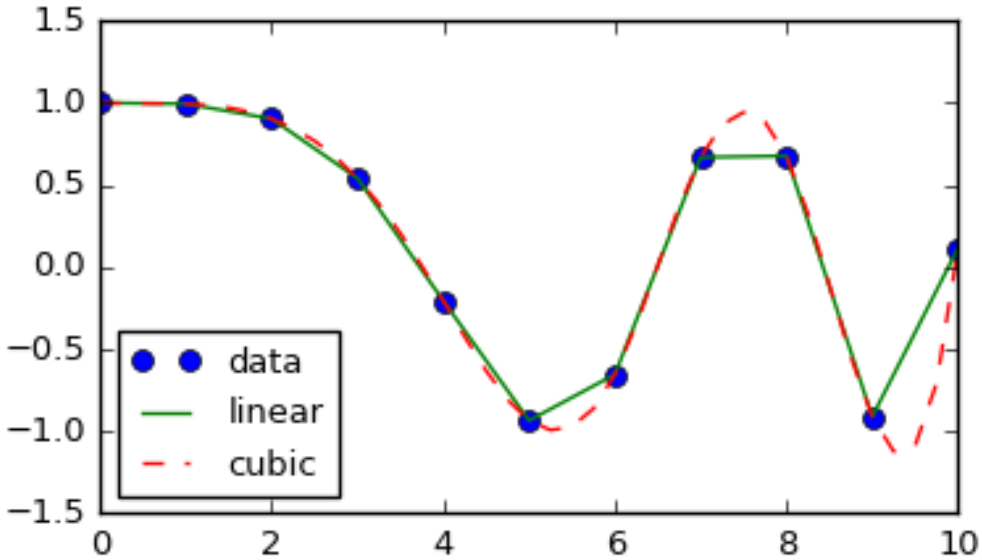

Figure: Cubic Interpolation. Courtesy Scipy

Kernel Regression

- Given (x_i, y_i), $i = 1, \ldots, n$
- Formulate $y_i = r(x_i) + \epsilon_i$
- Goal is to estimate $r(x)$ with $\hat{r}(x)$
- $\hat{r}(x) = \sum_{i=1}^{n} w(x_i, x) \cdot y_i$
- We need to choose the set of weights w. Both kNN and Linear Regression are special cases of this
- Many choices for kernels. E.g. Cubic Splines, Gaussian, Box

Overview

1. Machine Learning
2. Regression
   - kNN
   - Linear Regression
   - Kernel Regression
3. Classification
   - Naive Bayes
   - Logistic Regression
   - Support Vector Machines
   - Random Forests
   - Gradient Boosting Machines
4. Assignments, Weekly Reading

Naive Bayes

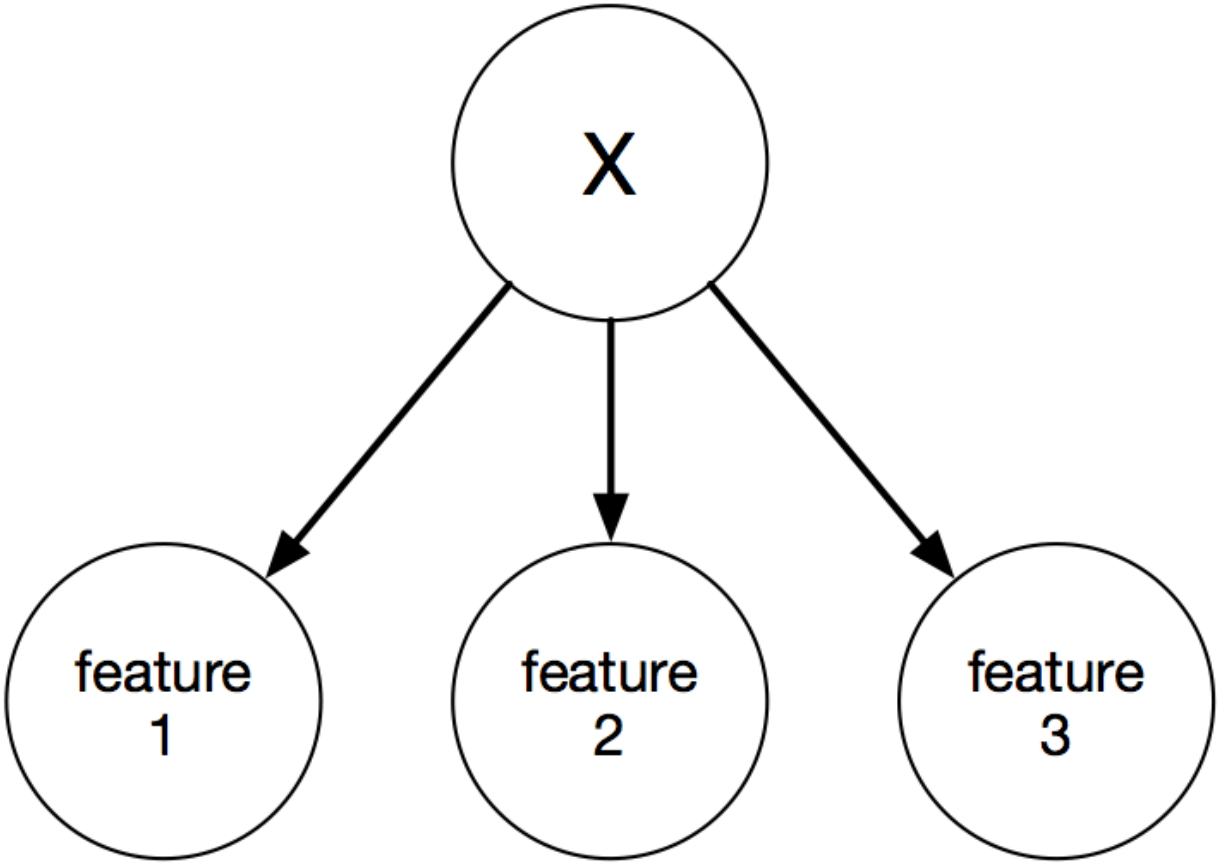

Figure: Naive Bayes Classifier

Naive Bayes

- Naive because it *assumes* all features are independent
- $P(C_i|x_1, \ldots, x_n) \propto P(C_i) \times \prod_{k=1}^{n} P(x_k|C_i)$
- $\hat{y} = argmax_i P(C_i) \times \prod_{k=1}^{n} P(x_k|C_i)$
- To learn the model, all you have to do is to learn the individual feature distributions
- Requires much less data compared to other models that make better structural assumptions

Logistic Regression

- Goal: $y = f(\mathbf{x}), y \in \Omega$
- Want to keep things linear, if possible
- That is, We prefer something like $y = W^t\mathbf{x}$ or close to it

Backstory of Logistic Regression

- It was discovered back in 19th century when mathematicians tried to explain observed population growth
- It fits very well with observed population growths of countries (e.g. US, Belgium, France)
- In the 60s, it was discovered again and was fitted to populations of fruit flies, cantaloupes, Humans in North Africa etc

Logistic Regression

- $Y = W^t\mathbf{x}$ results in a linear function
- Assume $Y \sim B(P(Y = 1|\mathbf{x}))$, where B is a Bernoulli distribution
- That is, Y takes values $\{0, 1\}$ with probability $\{p, 1 - p\}$ respectively
- We can write this as $Y \sim p^k(1 - p)^{1-k}, k \in \{0, 1\}$
- The Logistic Regression Model: $\ln(\frac{P(Y=1|\mathbf{x})}{P(Y=0|\mathbf{x})}) = W^t\mathbf{x}$

Logistic Regression

- The Logistic Regression Model: $\ln(\frac{P(Y=1|\mathbf{x})}{P(Y=0|\mathbf{x})}) = W^t\mathbf{x}$
- Why does this make sense?
- We want a linear combination of features. That explains $W^t\mathbf{x}$
- We need to turn that linear combination which can take any real value in something takes ranges between 0 and 1
- $\frac{P(Y=1|\mathbf{x})}{P(Y=0|\mathbf{x})}$, is the odds-ratio (is positive real number)
- Log of the odds-ratio converts this positive value into a proper real number!

Logistic Regression Formula

- The Logistic Regression Model: $\ln(\frac{P(Y=1|\mathbf{x})}{P(Y=0|\mathbf{x})}) = W^t\mathbf{x}$
- From this, we get $\frac{P(Y=1|\mathbf{x})}{P(Y=0|\mathbf{x})} = e^{W^t\mathbf{x}}$
- $P(Y=1|\mathbf{x}) = \frac{e^{W^t\mathbf{x}}}{1+e^{W^t\mathbf{x}}}$

Fitting the Logistic Model

- We need to **learn** W, the weights that explain the data best
- We'll use **Maximum Likelihood** approach to fit the model
- $P(Y_1 = y_1, Y_2 = y_2, \ldots, Y_n = y_n | W, \mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n})$ is what we are trying to maximize
- iid assumption. Each data point is independent. Class depends only on that data point
- $L(W) = \prod_i P(Y_i = y_i | W, \mathbf{x_i})$
- $W^* = argmax_W \log L(W)$. Log because it converts the $\prod$ to a $\sum$ and it doesn't affect the $argmax$
- Let's assume that $y_i = \{-1, 1\}$. This simplifies the next steps

Fitting the Logistic Model

- If $y_i = 1$, we will use $P(Y = 1|W, \mathbf{x_i}) = \frac{e^{W^t \mathbf{x}}}{1+e^{W^t \mathbf{x}}} = \frac{1}{1+e^{-y_i W \mathbf{x_i}}}$
- If $y_i = -1$, we will use
  $1 - P(Y = 1|W, \mathbf{x_i}) = 1 - \frac{e^{W^t \mathbf{x}}}{1+e^{W^t \mathbf{x}}} = \frac{1}{1+e^{-y_i W \mathbf{x_i}}}$
- $W^* = argmax_W \log L(W) = argmin_W \sum_i \log(1 + e^{-y_i W^t \mathbf{x_i}})$

Support Vector Machines

Let's start with some intuition. We want to separate the two classes. Which is a better classifier?

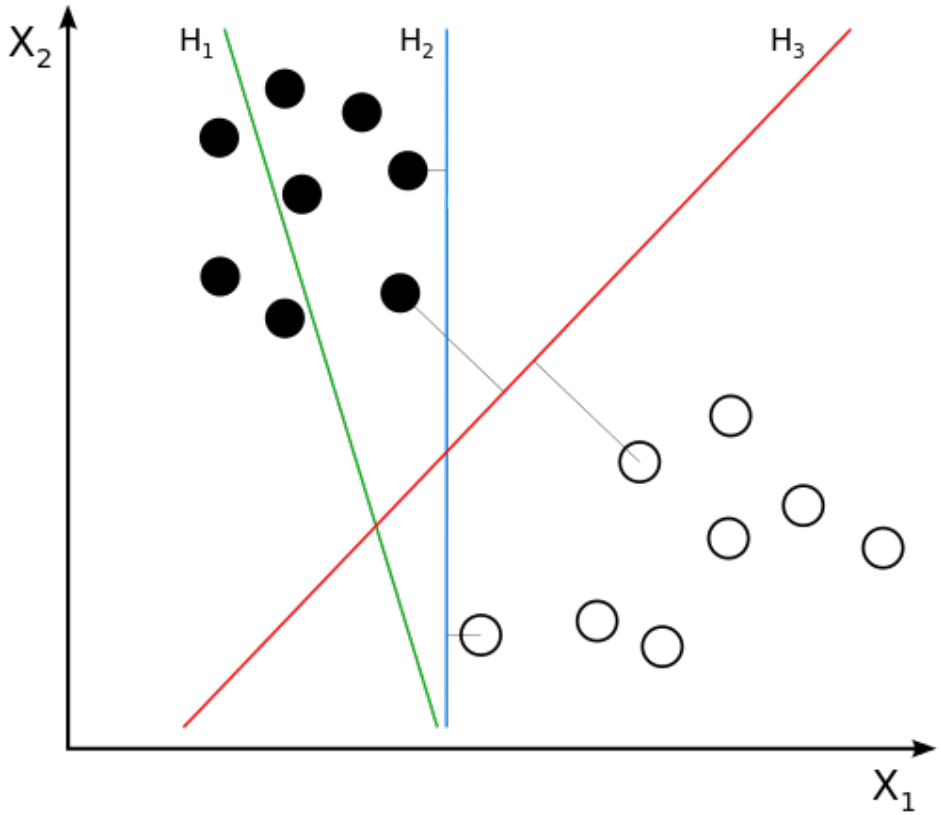

Figure: Separating Hyperplanes. Image courtesy: Zack Weinberg

Support Vector Machines

- Define **margin** as $y_i f(\mathbf{x_i})$. So, margin is positive if we are on the **correct** side of the decision boundary
- We want all examples as far away from the boundary as possible. Margin as large as possible

Support Vector Machines

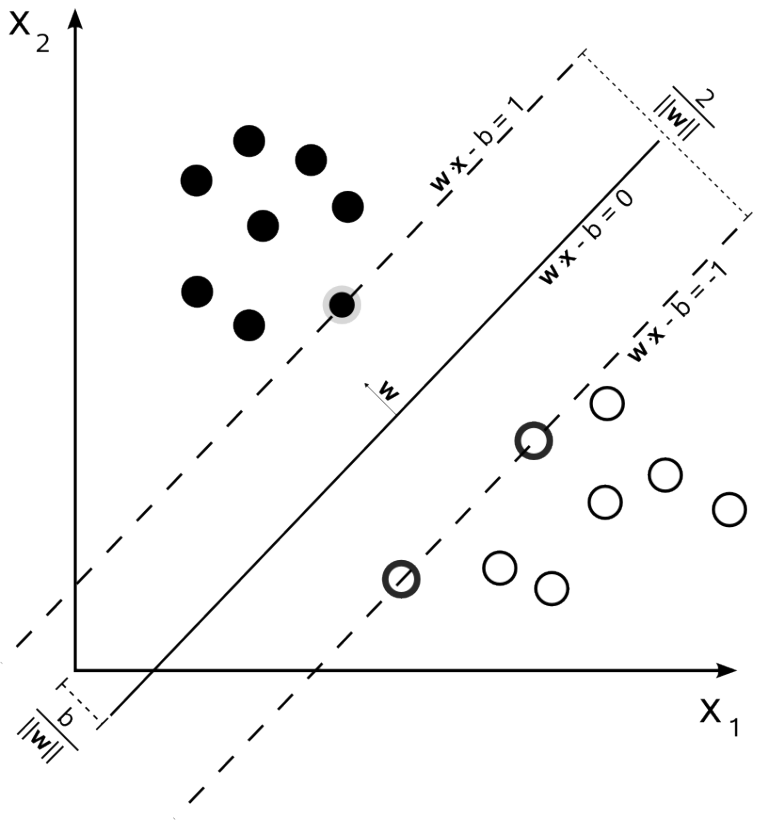

Figure: Separating Hyperplanes. Image courtesy: Cyc

Note: They conventionally add an intercept term, but it can be swept into $\mathbf{x}$ by assuming $x_0 = 1$.

Support Vector Machines

- As in Logistic Regression, the function that we fit is linear
- $f(\mathbf{x}) = \sum_{j=1}^{m} w_j x_j + b$
- We classify samples by looking at the sign of $f(\mathbf{x})$
- Support vector machines maximize the minimum margin
- With some math (Hastie 2001), it is equivalent to minimizing $||W||$ subject to a constraint $y_i(W\mathbf{x_i} + b) \geq 1$

Support Vector Machines

- In real-life, the classes are not always separable. They overlap. So, we end up with a loss minimization scenario
- $\phi(W) = \frac{1}{2}||W||^2 + \frac{\gamma}{2}\sum_{i=1}^{N} L(y_i, W\mathbf{x_i} + b)$
- Minimize the above function, given some Loss function $L()$ and user-defined regularization parameter γ
- Hinge Loss: $max\{0, 1 - y_i f(\mathbf{x_i})\}$
- Leads to minimizing $\frac{1}{2}||W||^2 + \frac{\gamma}{2}\sum_{i=1}^{N} \zeta_i$, ζ_i are non-negative
- The constraint is modified to $y_i(W\mathbf{x_i} + b) \geq 1 - \zeta_i$
- We try to reduce the mis-classification by holding $\sum_i \zeta_i \leq C$

Support Vector Machines

- SVMs were very successful. Some of the best classifiers of their times
- Turns out that you can generalize by applying a **kernel** to each $\mathbf{x}$ and transform to a high dimensional space
- Relatively slow to train and with the emergence of Big Data, started falling out of favor
- Deep Learning models (see later) offer better performance and are trainable using simpler algorithms

Ensemble Methods

- So far, we looked at individual classifiers
- These classifiers are what are called as **Strong** learners
- An alternative approach is to take a collection of much simpler techniques and combine them in intelligent ways to build a **Strong** ensemble
- Wisdom of the crowds approach

Random Forests

- Given a dataset of size N, sample N cases at random, with replacement
- This sample is used to build a decision tree
- From M features randomly select m, such that $m << M$
- Using these m variables, and the bootstrapped data sample, build a decision tree
- Repeat the above procedure several hundreds of times
- The collection of decision trees you get is the Random Forest

Random Forests: Decision Tree

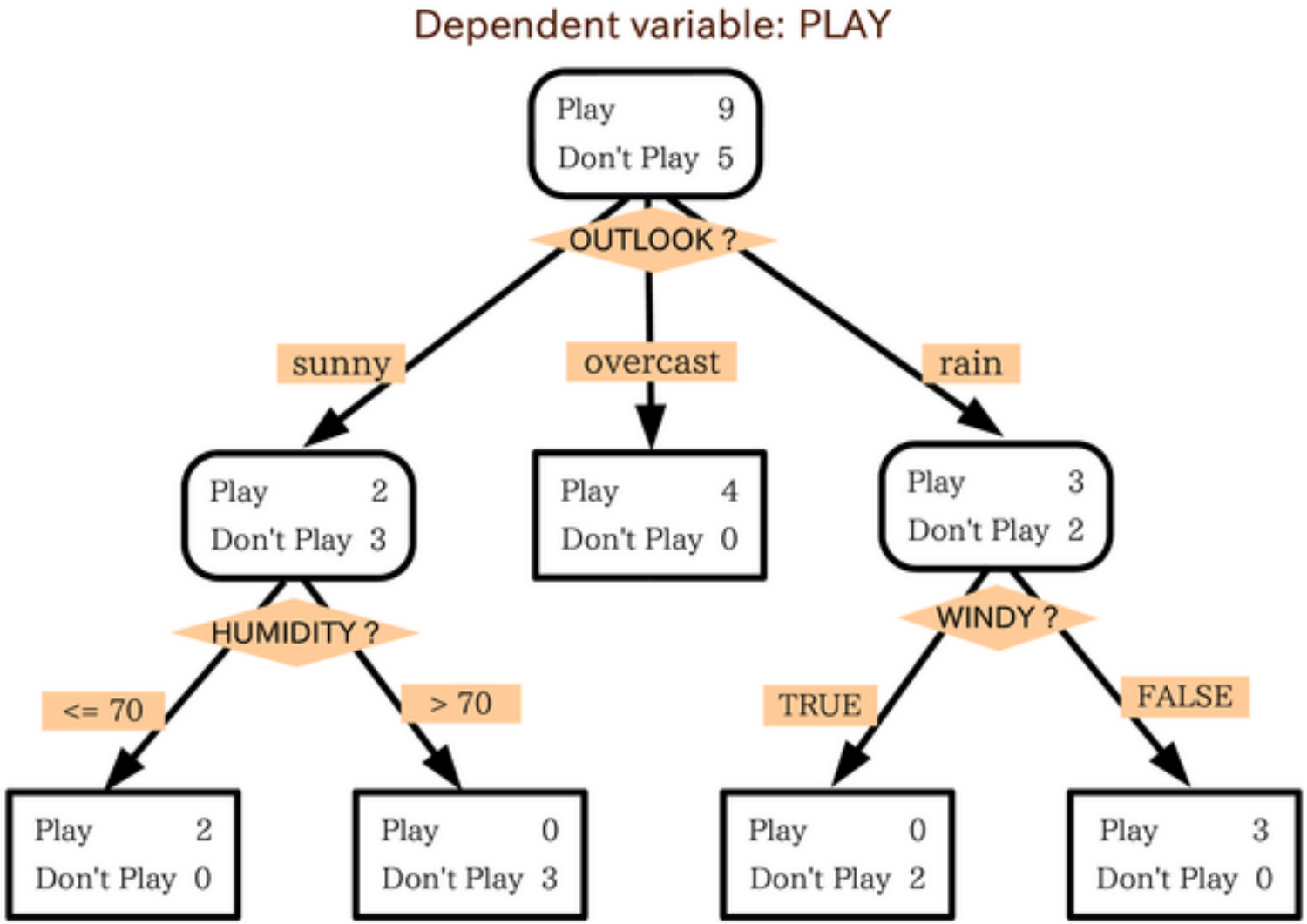

Figure: Decision Tree Example

Random Forests: Design considerations

- How many features per tree? Typically $m \approx \sqrt{(M)}$
- How many trees?
- How to split the nodes? Information Gain / Gini Impurity
- Gini: $\sum_{j=1}^{m} f_j(1 - f_j)$
- Info Gain: $-\sum_{j=1}^{m} f_j \log_2 f_j$

Random Forests

- They are very competitive amongst current algorithms
- Runs efficiently on large data sets
- Thousands of input variables can be handled
- As a by-product, we find out what variables are important for classification
- Offers an approach to detect and handle with variable interactions
- They tend not to overfit

Gradient Boosting Machines

- Assume you have a regression problem: $y = f(\mathbf{x})$
- We minimize the mean squared prediction error: $(\hat{y} - y)^2$ to learn a model
- Assume you have a model $f_1(\mathbf{x})$, can you improve it?
- Try and build $f_2(\mathbf{x}) = f_1(\mathbf{x}) + h(\mathbf{x}) = y$
- Or, $h(\mathbf{x}) = y - f_1(\mathbf{x})$. In general, we learn an $f_{m+1}(\mathbf{x}) = f_m(\mathbf{x}) + h(\mathbf{x})$

Gradient Boosting Machines

- Observe that the residual $y - f_m(\mathbf{x})$ is the negative gradient of $\frac{1}{2}(y - f_m(\mathbf{x}))^2$
- So, gradient boosting machines are simply gradient descent algorithms
- We generalize this idea to other loss functions and we get GBMs
- $f_0(\mathbf{x}) = argmin_\gamma \sum_i L(y_i, \gamma)$. Essentially learn a constant that minimizes the loss
- $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + argmin_{h \in H} \sum_i L(y_i, f_{m-1}(\mathbf{x}) + h(\mathbf{x}))$
- That is a hard problem. So, we simplify by taking a step in the direction of the **gradient**
- $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) - \gamma_m \sum_i \nabla_h L(y_i, f_m(\mathbf{x_i}))$

Overview

1. Machine Learning
2. Regression
   - kNN
   - Linear Regression
   - Kernel Regression
3. Classification
   - Naive Bayes
   - Logistic Regression
   - Support Vector Machines
   - Random Forests
   - Gradient Boosting Machines
4. Assignments, Weekly Reading

Weekly Reading

- http://www.deeplearningbook.org/contents/ml.html
- http://www.ats.ucla.edu/stat/mult_pkg/faq/general/odds_ratio.htm
- https://en.wikipedia.org/wiki/Odds_ratio
- https://en.wikipedia.org/wiki/Support_vector_machine
- https://en.wikipedia.org/wiki/Random_forest
- http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf


thank you
danke
謝謝
ngiyabonga
teşekkür ederim
gracias
спасибо
Баярлалаа
рахмат
bedankt
dziękuję
obrigado
hvala
mauruuru
dank je
tapadh leat
mochchakkeram
go raibh maith agat
sukriya
kop khun krap
terima kasih
sagolun
taiku
grazie
arigatô
takk
dakujem
мерси
merci
ευχαριστώ
감사합니다
ありがとう
mersi
barka
welalin
tack
spas
kia ora
vinaka
blagodaram
misaotra
matondo
paldies
grazzi
mahalo
хвала
asante
manana
obrigada
murakoze
tenki
faafetai lava
kiitos
dankie
dhanyavad
nanni
nandri
bayarlalaa
gracie
köszönöm
dankon
aciú
akun
enkosi
sobodi
dêkuji
chnorakaloutioun
gratias ago
gràcies
sulpây
djiere dieuf
tau
дякую
mamnun
chokrane
trugarez
mèsi
didi madloba
najis tuke
kam sah hamnida
rahmat
tanemirt
rahmet
diolch
dhanyavadagalu
shukriya
merce
xièxie