Adaptive Sensing and Active Learning

Information Processing Lab

SSP Workshop August 7, 2012

Rob Nowak www.ece.wisc.edu/~nowak
BIGDATA: An Interactive Approach

$\mathcal{X}$: models/hypotheses under consideration

$\mathcal{Y}$: possible measurements/experiments

$y_1(x), y_2(x), \ldots$: information/data

model space

sensing space

data space
Sparse Signals

Humans as Sensors

Humans are much more reliable and consistent at making comparative judgements, than in giving numerical ratings or evaluations

Bijmolt and Wedel (1995)
Stewart, Brown, and Chater (2005)
Machine Learning from Human Judgements

Recommendation Systems

Document Classification

Optimizing Experimentation

Challenge:
Computing is cheap, but human assistance/guidance is expensive

Goal:
Optimize such systems with as little human involvement as possible
Machine Learning (Passive)

Raw unlabeled data

$X_1, X_2, X_3, \ldots$

Labeled data

$(X_1, Y_1), (X_2, Y_2), (X_3, Y_3), \ldots$

passive learner

rule for predicting $Y$ from $X$

expert/oracle provides labels
Active Learning

Raw unlabeled data

\[ X_1, X_2, X_3, \ldots \]

machine requests labels for **selected** data

\[ (X_1, ?) \]
\[ (X_1, Y_1) \]

\[ X_2 \text{ identical or very similar to } X_1 \ldots \]
\[ \text{no need to ask for label} \]

\[ (X_3, ?) \]
\[ (X_3, Y_3) \]

rule for predicting \( Y \) from \( X \)

active learner

expert/oracle provides labels
Outline

1. Derivative Free Optimization using Human Subjects

2. Binary Classification via Active Learning

3. Ranking from Pairwise Comparisons

minimizing a convex function

best linear classifier

ranking or embedding objects in a low-dimensional space
Human oracles can provide function values or comparisons, but not function gradients. Convex functions to be minimized are often used in optimization problems. Methods that don’t use gradients are called Derivative Free Optimization (DFO).
A Familiar Application

better

worse

spherical correction
cylindrical correction

optimal prescription
In the Future... Custom Frame Optimization

optimization dimensions: frame size, material, shape, color, lens tint
Assume that the (unknown) function $f$ to be optimized is strongly convex with Lipschitz gradients.

The function will be minimized by asking pairwise comparisons of the form:

Is $f(x) > f(y)$ ?

Assume that the answers are probably correct: for some $\delta > 0$

$$\mathbb{P}(\text{answer} = \text{sign}(f(x) - f(y))) \geq \frac{1}{2} + \delta$$
Optimization with Pairwise Comparisons

initialize: \( x_0 = \) random point
for \( n = 0, 1, 2, \ldots \)
1) select one of \( d \) coordinates uniformly at random
and consider line along coordinate that passes \( x_n \)
2) minimize along coordinate using pairwise
comparisons and binary search
3) \( x_{n+1} = \) approximate minimizer

begin with large interval \([y_0^-, y_0^+]\);
midpoint \( y_0 \) is estimate of minimizer

line search iteratively reduces interval containing minimum
Optimization based on Pairwise Comparisons

**Optimization with Pairwise Comparisons**

initialize: \( x_0 = \text{random point} \)

for \( n = 0, 1, 2, \ldots \)

1) select one of \( d \) coordinates uniformly at random and consider line along coordinate that passes \( x_n \)

2) minimize along coordinate using pairwise comparisons and binary search

3) \( x_{n+1} = \text{approximate minimizer} \)

split intervals \([y_0^-, y_0]\) and \([y_0, y_0^+]\) and compare function values at these points with \( f(y_0) \)
Optimization based on Pairwise Comparisons

**Optimization with Pairwise Comparisons**

initialize: $x_0 =$ random point
for $n = 0, 1, 2, \ldots$

1) select one of $d$ coordinates uniformly at random and consider line along coordinate that passes $x_n$
2) minimize along coordinate using pairwise comparisons and binary search
3) $x_{n+1} =$ approximate minimizer

line search iteratively reduces interval containing minimum
reduce to smallest interval containing minimum of these points
Optimization based on Pairwise Comparisons

Optimization with Pairwise Comparisons
initialize: $x_0 =$ random point
for $n = 0, 1, 2, \ldots$
1) select one of $d$ coordinates uniformly at random and consider line along coordinate that passes $x_n$
2) minimize along coordinate using pairwise comparisons and binary search
3) $x_{n+1}$ = approximate minimizer

line search iteratively reduces interval containing minimum

repeat...
Optimization based on Pairwise Comparisons

Optimization with Pairwise Comparisons
initialize: $x_0 =$ random point
for $n = 0, 1, 2, \ldots$
1) select one of $d$ coordinates uniformly at random and consider line along coordinate that passes $x_n$
2) minimize along coordinate using pairwise comparisons and binary search
3) $x_{n+1} =$ approximate minimizer

line search iteratively reduces interval containing minimum

repeat...
Convergence Analysis

If we want \( \text{error} := \mathbb{E}[f(x_k) - f(x^*)] \leq \epsilon \), we must solve \( k \approx d \log \frac{1}{\epsilon} \) line searches (standard coordinate descent bound) and each must be at least \( \sqrt{\frac{\epsilon}{d}} \) accurate

**Noiseless Case:**

- each line search requires \( \frac{1}{2} \log \left( \frac{d}{\epsilon} \right) \) comparisons
- \( \Rightarrow \) total of \( n \approx d \log \frac{1}{\epsilon} \log \frac{d}{\epsilon} \) comparisons
- \( \Rightarrow \epsilon \approx \exp \left( -\sqrt{\frac{n}{d}} \right) \)

**Noisy Case:** probably correct answers to comparisons:

\[
P(\text{answer} = \text{sign}(f(x) - f(y))) \geq \frac{1}{2} + \delta
\]

\( \Rightarrow \) take majority vote of repeated comparisons to mitigate noise

**Bounded Noise** (\( \delta \geq \delta_0 > 0 \)):

- line searches require \( C \log \frac{d}{\epsilon} \) comparisons,
- where \( C > 1/2 \) depends on \( \delta_0 \) \( \Rightarrow \) \( \epsilon \approx \exp \left( -\sqrt{\frac{n}{dC}} \right) \)

**Unbounded Noise** (\( \delta \propto |f(x) - f(y)| \)):

- line searches require \( \left( \frac{d}{\epsilon} \right)^2 \) comparisons \( \Rightarrow \epsilon \approx \sqrt{\frac{d^3}{n}} \)
Lower Bounds

\[ f_0(x) = |x + \epsilon|^2 \]

\[ f_1(x) = |x - \epsilon|^2 \]

For unbounded noise, \( \delta \propto |f(x) - f(y)| \), Kullback-Leibler Divergence between response to \( f_0(x) > f_0(y) \) vs. \( f_1(x) > f_1(y) \) is \( O(\epsilon^4) \), and KL Divergence between \( n \) responses is \( O(n\epsilon^4) \)

with \( \epsilon \sim n^{-1/4} \)

- KL Divergence = constant
- squared distance between minima \( \sim n^{-1/2} \)

\[ \Rightarrow \quad \mathbb{P} \left( f(x_n) - f(x^*) \geq n^{-1/2} \right) \geq \text{constant} \]

matches \( O(n^{-1/2}) \) upper bound of algorithm

Jamieson, Recht, RN (2012)
A Surprise

Could we do better with function evaluations (e.g., ratings instead of comparisons)?

suppose we can obtain noisy function evaluations of the form: \( f(x) + \text{noise} \)

\[
\begin{align*}
  f(x) &= 10 \\
  f(y) &= 9 \\
  f(z) &= 1
\end{align*}
\]

function values seem to provide much more information than comparisons alone

\[
\begin{align*}
  f(y) &< f(x) \\
  f(z) &< f(x)
\end{align*}
\]

\[ f(z) = 1 \]

lower bound on optimization error with noisy function evaluations: \( \sqrt{\frac{d}{n}} \)

upper bound on optimization error with noisy pairwise comparisons: \( \sqrt{\frac{d^3}{n}} \)

evaluations give at best a small improvement over comparisons

Jamieson, Recht, RN (2012)

see Agrawal, Dekel, Xiao (2010)

for similar upper bounds for function evals

if we could measure noisy gradients (and function is strongly convex), then \( O\left(\frac{d}{n}\right) \) convergence rate is possible

Nemirovski et al 2009
A Runoff Is Down to the Wire in Texas

July 30, 2012

ERIK ECKHOLM

In Texas, it was the_registered_days of summer, and Raum to ranges was the closest match. The months were under a heat wave that the state had anticipated for weeks. The\n
unlabeled documents

expert/oracle

provides labels to machine learner

Tong and Koller (2001)
A Possible Application

submitted manuscripts

to improve notoriously long review process, we will use Dave’s expertise to train a computer to automatically accept/reject submissions

features used by computer

= \{ \# of equations, length of proofs, \# of mentions of Shannon, etc \}
Active Learning

**Learning Problem:** Consider a binary prediction problem involving a collection of “classifiers.” Each classifier maps points in the “feature-space” (e.g., $\mathbb{R}^d$) to binary labels. The features and labels are governed by an *unknown* distribution $P$. The goal is to select the classifier that minimizes the probability of misclassification using as few training examples as possible.

Standard approaches assume training data are obtained prior to learning. However, some examples are more informative than others, so sequential selection of data can dramatically accelerate learning.
1D Classification - Classic Binary Search

-active learning: sequentially select points for labeling

\[ \frac{1}{3} = 0101 \ldots \]
\[ \epsilon \text{-accuracy requires } \log_2 \frac{1}{\epsilon} \text{ queries} \]

-passive learning: query points uniformly (possibly random)

\[ \epsilon \text{-accuracy requires } O \left( \frac{1}{\epsilon} \right) \text{ queries} \]
Dealing with Noise (Horstein’s Algorithm)

see Burnashev & Zigangirov ’74 for rigorous analysis; also independently proposed by Karp & Kleinberg ’07

requires $C_b \log \frac{1}{\epsilon}$ samples for $\epsilon$-accuracy

$P(Y = +1 | X = x)$

Update ‘posterior’ density based on noise bound $b$
**Noisy Generalized Binary Search**

initialize: $p_0$ uniform over $\mathcal{H}$ and $\alpha < \beta < 1/2$.

for $n = 0, 1, 2, \ldots$

1) $x_n = \arg \min_{x \in \mathcal{X}} |\sum_{h \in \mathcal{H}} p_n(h) h(x)|$

2) Obtain noisy response $y_n$

3) Bayes update: $\forall h$

$$p_{n+1}(h) \propto p_n(h) \times \left\{ \begin{array}{ll} 1 - \beta & , h(x_n) = y_n \\ \beta & , h(x_n) \neq y_n \end{array} \right.$$

hypothesis selected at each step:

$$\hat{h}_n := \arg \max_{h \in \mathcal{H}} p_n(h)$$

also requires as few as $\log \frac{1}{\epsilon}$ samples for $\epsilon$-accuracy

... but more in general, depending on complexity of optimal decision boundary and noise characteristics

“generalized” binary search is similar to classic binary search
Nonparametric Binary Classification

\[
X := \text{feature space, typically } \mathbb{R}^d \quad \mathcal{Y} := \{-1, +1\}, \text{ labels}
\]

Key Questions:

1. When can active learning provide reductions in sample complexity?
2. What active learning strategies/policies are optimal?

\[
\mathbb{P}(Y = 1|X = x)
\]

unknown

1/2-level set is optimal decision boundary

optimal decision set
allowable questions: is \(x\) in the set?

Castro and N (2008), Raginsky and Rahklin (2011), Hanneke (2011)
Bounds on Sample Complexity

**Key complexity parameters**

- $\mathbb{P}(Y = 1 | X = x)$
- Smoothness of conditional probability function at the boundary, $\kappa$
- Holder regularity of the decision boundary, $\alpha$

**Training examples:** $\{(x_i, y_i)\}_{i=1}^n$ selected sequentially and adaptively (active learning) or at random (passive learning)

**Minimax rate of convergence to Bayes error:**

- **Active:** $n^{-\frac{\kappa}{2\kappa + \rho - 2}}$
- **Passive:** $n^{-\frac{\kappa}{2\kappa + \rho - 1}}$

$\rho := \frac{d-1}{\alpha}$

As $\rho \to 0$ and $\kappa \to 1$

Active learning yields exponential improvement!
Implications in Practice

simple decision set

\[ \exp(-cn) \]

complex decision set

\[ \frac{1}{n} \]

learning curves

Passive Learning

Active Learning

Learning curves

Active Learning

Passive Learning

\[ \frac{1}{n} \]
Bartender: “What beer would you like?”
Jeff: “Hmm... I usually drink Duff”
Bartender: “Try these two samples. Do you prefer A or B?”
Jeff: “B”
Bartender: “Ok try these two: C or D?” ....
Ranking Based on Pairwise Comparisons

Consider 10 beers ranked from best to worst: $D < G < I < C < J < E < A < H < B < F$

Which questions should we ask? How many are needed?

Does adaptively help?
Randomly Selected Pairwise Comparisons

Consider 10 beers ranked from best to worst:

\[ D < G < I < C < J < E < A < H < B < F \]

almost all pairs must be compared, i.e., about \( n(n-1)/2 \) comparisons

That’s a lot of beer!
Ranking with Adaptively Selected Queries

Insert H into: D < G < C < E < A < B < F

{ }

{H < E}

{H < E}, {G < H}

{H < E}, {G < H}, {H < C}

D < G < H < C < E < A < B < F

to correctly place an object into an ordered list of \( k \) objects requires \( \log_2 k \) comparisons
Adaptively Selected Pairwise Comparisons

Consider 10 beers ranked from best to worst:

\[ D < G < I < C < J < E < A < H < B < F \]

Binary insertion sort:

Select \( m \) pairwise comparisons according to binary sort

Perfect recovery if

\[ \log_2 k \] comparisons to insert an item into a list of \( k \) objects

\[ \Rightarrow \quad n \log_2 n \] comparisons to rank \( n \) objects

That's still a lot of beer!
Beer Space

Suppose beers can be embedded (according to characteristics) into a low-dimensional Euclidean space.

Jeff’s latent preferences in “beer space” (e.g., bitterness, color, maltiness,...)

\[ \| x_i - W \| < \| x_j - W \| \iff x_i < x_j \]
Ranking According to Distance

C < A < B < E < G < D < F
Ranking According to Distance

E < B < F < G < C < A < D
Goal: Determine ranking by asking comparisons like “Do you prefer A or B?”

... now there are at most $n^{2d}$ rankings (instead of $n!$), and so in principle no more than $2d \log n$ bits of information are needed.
Lazy Binary Search

Consider $n$ objects $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$. Many comparisons are redundant because the objects embed in $\mathbb{R}^d$, and therefore it may be possible to correctly rank based on a small subset.

binary information we can gather: $q_{i,j} \equiv \text{do you prefer } x_i \text{ or } x_j$

Optimal selection of a sequence of $q_{i,j}$ requires a computationally difficult search, involving a combinatorial optimization.

**Lazy Binary Search**

input: $x_1, \ldots, x_n \in \mathbb{R}^d$
initialize: $x_1, \ldots, x_n$ in uniformly random order

for $k=2, \ldots, n$
    for $i=1, \ldots, k-1$
        if $q_{i,k}$ is *ambiguous* given $\{q_{i,j}\}_{i,j<k}$,
            then ask for pairwise comparison,
        else impute $q_{i,j}$ from $\{q_{i,j}\}_{i,j<k}$

output: ranking of $x_1, \ldots, x_n$ consistent with *all* pairwise comparisons
suppose we have ranked 4 beers
ranking implies that Jeff’s optimal preferences are in shaded region
suppose we have ranked 4 beers
ranking implies that Jeff’s optimal preferences are in shaded region

Answers to queries that intersect shaded region are ambiguous, otherwise they are not.

**Key Observation:** most queries will not be ambiguous, therefore the expected total number of queries made by lazy binary search is about $d \log n$

K. Jamieson and RN (2011)
BeerMapper app learns a person's ranking of beers by selecting pairwise comparisons using lazy binary search and a low-dimensional embedding based on key beer features.
BeerMapper - Under the Hood

Algorithm requires feature representations of the beers \( \{ x_1, \ldots, x_n \} \subset \mathbb{R}^d \)

Reviews for each beer

Bag of Words weighted by TF*IDF

Get 15 nearest neighbors using cosine distance

Non-metric multidimensional scaling

Embedding in 3 dimensions

Two Hearted Ale - Input ~2500 natural language reviews

http://www.ratebeer.com/beer/two-hearted-ale/1502/2/1/

Bottle 355ml.
Clear light to medium yellow orange color with a average, frothy, good lacing, fully lasting, off-white head. Aroma is moderate to heavy malty, moderate to heavy hoppy, perfume, grapefruit, orange shell, soap. Flavor is moderate to heavy sweet and bitter with a average to long duration. Body is medium, texture is oily, carbonation is soft. [250908]

An orange beer with a huge off-white head. The aroma is sweet and very freshly hoppy with notes of hop oils - very powerful aroma. The flavor is sweet and quite hoppy, that gives flavors of oranges, flowers as well as hints of grapefruit. Very refreshing yet with a powerful body.
BeerMapper - Under the Hood

Algorithm requires feature representations of the beers \( \{x_1, \ldots, x_n\} \subset \mathbb{R}^d \)

**Two Hearted Ale - Weighted Bag of Words (sorted by weights):**
ipa hops citrus floral orange pine grapefruit head hoppy aroma white pours bitter golden piney hazy balanced cloudy malt amber sweet lacing bells strong light favorite gold off medium perfect hearted nose thick smooth excellent huge smell wonderful crisp poured fresh beautiful lots bell’s creamy body copper flavors smells slightly fruity love yellow ever there amazing notes fluffy clean frothy sweetness brew long awesome ale caramel aromas flowers lemon palate malts over down get after tastes mouthfeel your backbone dry other leaves centennial top slight bite solid again batch right nicely through clear it’s extremely foamy aftertaste still
BeerMapper - Under the Hood

Algorithm requires feature representations of the beers \( \{x_1, \ldots, x_n\} \subset \mathbb{R}^d \)

Weighted count vector for the \( i \)th beer:

\[ z_i \in \mathbb{R}^{400,000} \]

Cosine distance:

\[ d(z_i, z_j) = 1 - \frac{z_i^T z_j}{||z_i|| ||z_j||} \]

Two Hearted Ale - Nearest Neighbors:
Bear Republic Racer 5
Avery IPA
Stone India Pale Ale & IPA
Founders Centennial IPA
Smuttynose IPA
Anderson Valley Hop Ottin IPA
AleSmith IPA
BridgePort IPA
Boulder Beer Mojo IPA
Goose Island India Pale Ale
Great Divide Titan IPA
New Holland Mad Hatter Ale
Lagunitas India Pale Ale
Heavy Seas Loose Cannon Hop3
Sweetwater IPA

Reviews for each beer
Bag of Words weighted by TF*IDF
Get 15 nearest neighbors using cosine distance
Non-metric multidimensional scaling
Embedding in 3 dimensions
BeerMapper - Under the Hood

Algorithm requires feature representations of the beers \( \{x_1, \ldots, x_n\} \subset \mathbb{R}^d \)

Weighted count vector for the \( i \)th beer:

\[ z_i \in \mathbb{R}^{400,000} \]

Cosine distance:

\[ d(z_i, z_j) = 1 - \frac{z_i^T z_j}{||z_i|| ||z_j||} \]

Two Hearted Ale - Nearest Neighbors:
- Bear Republic Racer 5
- Avery IPA
- Stone India Pale Ale & #40;IPA& #41;
- Founders Centennial IPA
- Smuttynose IPA
- Anderson Valley Hop Ottin IPA
- AleSmith IPA
- BridgePort IPA
- Boulder Beer Mojo IPA
- Goose Island India Pale Ale
- Great Divide Titan IPA
- New Holland Mad Hatter Ale
- Lagunitas India Pale Ale
- Heavy Seas Loose Cannon Hop3
- Sweetwater IPA

Reviews for each beer

Bag of Words weighted by TF*IDF

Get 15 nearest neighbors using cosine distance

Non-metric multidimensional scaling

Embedding in 3 dimensions
BeerMapper - Under the Hood

Algorithm requires feature representations of the beers \( \{x_1, \ldots, x_n\} \subset \mathbb{R}^d \)

- Reviews for each beer
- Bag of Words weighted by TF*IDF
- Get 15 nearest neighbors using cosine distance
- Non-metric multidimensional scaling
- Embedding in 3 dimensions

Sanity check: styles should cluster together and similar styles should be close.

- Red = IPA
- Green = Pale Ale
- Magenta = Amber Ale
- Cyan = Lager + Pilsener
- Yellow = Belgians (light + dark)
- Black = Stout + Porter
- Blue = Everything else
Derivative Free Optimization using Human Subjects

Binary Classification via Active Learning

Ranking from Pairwise Comparisons

**Challenge:**
Computing is cheap, but human assistance/guidance is expensive

**Goal:**
Optimize such systems with as little human involvement as possible

Humans are much more reliable and consistent at making comparative judgements, than in giving numerical ratings or evaluations

“Binary search” procedures can play a role in *active learning*
References


J. Haupt, R. Castro, R. Baraniuk, and R. Nowak, “Sequentially designed compressed sensing,” SSP 2012


A. Agraval, O. Dekel and L. Xiao, “Optimal algorithms for online convex optimization with multi-point bandit feedback,” COLT 2010


S. Tong and D. Koller, “Support vector machine active learning with applications,” JMLR 2001


