My Latex Beamer Template

Hidemitsu Asano

Kyoto Univ./RIKEN

June 11, 2015

Introduction

Terminology

- S : A market string (wti x 100 p vs .48 1.21@1.24)

Terminology

- S : A market string (wti x 100 p vs .48 1.21@1.24)
- M : A market
  - *product*: a financial instrument ("wti, "brent", "goog")
  - *month*: the month for which the financial contract expires ("jan", "x", "march")
  - *strike*1..*N*: represents the strike price(s) of the financial contract
  - *strategy*: represents the strategy type of the financial contract ("put", "call", "strad")
  - *cross*: a hedge price for the financial contract
  - *bid*: a bid price for the financial contract
  - *offer*: an offer price for the financial contract

Terminology

- S :
  0: "wti", **1**: "x", **2**: "100", **3**: "p", **4**: "vs", **5**: ".48", **6**: "1.21", **7**: "1.24"
- M :
  - *product*: 0, "wti"
  - *month*: 1, "x"
  - *strike*1: 2, "100"
  - *strategy*: 3, "p"
  - *cross*: 5, ".48"
  - *bid*: 6, "1.21"
  - *offer*: 7, "1.24"

Domain Complexity

- Could just map all pairs $(s, m) \in (S \times M)$ to explicitly model $P(M|S)$, but...

Domain Complexity

- Could just map all pairs $(s, m) \in (S \times M)$ to explicitly model $P(M|S)$, but...
- $|S|$ is large (2+ million distinct messages for crude traders alone)

Domain Complexity

- Could just map all pairs $(s, m) \in (S \times M)$ to explicitly model $P(M|S)$, but...
- $|S|$ is large (2+ million distinct messages for crude traders alone)
- $|M|$ is also large, albeit less than $|S|$
    - only by a couple orders of magnitude
    - example: "z 150 call" $\equiv$ "dec 150 call"

Domain Complexity

- Could just map all pairs $(s, m) \in (S \times M)$ to explicitly model $P(M|S)$, but...
- $|S|$ is large (2+ million distinct messages for crude traders alone)
- $|M|$ is also large, albeit less than $|S|$
  - only by a couple orders of magnitude
  - example: "z 150 call" $\equiv$ "dec 150 call"
- $P(M|S)$ is still desired, but with a more efficient representation than $O(|M||S|)$

Semantic Labeling (Intuition)

Use domain knowledge to label each token of the string

Semantic Labeling (Intuition)

Use domain knowledge to label each token of the string

- Provide $X = L(S)$ where $L(S)$ *labelizes* each token
- Design $L(S)$ such that $|X| << |S|$

Semantic Labeling (Intuition)

Use domain knowledge to label each token of the string

- Provide $X = L(S)$ where $L(S)$ *labelizes* each token
- Design $L(S)$ such that $|X| << |S|$

- We hope that $P(M|X)$ is distributed similarly to $P(M|S)$, but in practice one instance of X fans out to more possible M's than S does

Semantic Labeling (Examples)

- wti x 100 c

  becomes

  PRODUCT MONTH NUMBER PRODUCT|STRATEGY

Semantic Labeling (Examples)

- wti x 100 c

  becomes

  PRODUCT MONTH NUMBER PRODUCT|STRATEGY

- brent z 50/60 ps vs .43

  becomes

  PRODUCT MONTH NUMBER OTHER NUMBER STRATEGY OTHER NUMBER

Generalization By Labeling

PRODUCT MONTH NUMBER OTHER NUMBER OTHER NUMBER

Generalization By Labeling

PRODUCT MONTH NUMBER OTHER NUMBER OTHER NUMBER

- brent z 50/60 ps vs .43
- wti x 55/60 cs vs 1.23
- go jan 120,125 fnc cross 2.78

Generalization By Labeling

PRODUCT MONTH NUMBER OTHER NUMBER OTHER NUMBER

- brent z 50/60 ps vs .43
- wti x 55/60 cs vs 1.23
- go jan 120,125 fnc cross 2.78

No algorithms necessary to generalize, just need some data!

Model Details

- **Current Model**:
  1. Retain a multinomial distribution over M conditioned on each observed, labelized sequence $x = L(s)$
  2. When several markets are possible given x, use analytics (eg. implied premiums) to filter out unlikely markets
  3. If analytics aren't available then we can maximize the posterior distribution $P(M|X = x)$ instead

Model Details

- **Current Model**:
  1. Retain a multinomial distribution over M conditioned on each observed, labelized sequence $x = L(s)$
  2. When several markets are possible given x, use analytics (eg. implied premiums) to filter out unlikely markets
  3. If analytics aren't available then we can maximize the posterior distribution $P(M|X = x)$ instead
- **Cons**:
  - Does not learn relationships between similar sequences. "x 10 c" and "hello x 10 c" are distinct sequences and thus create independent multinomial distributions over M
  - Fails to incorporate analytical features into the input vector- can't directly query the probability model with analytical random variables

Model Alternatives

Vectorizing the input:

- Treat each token of the sequence $x_0, x_1, ..., x_n$ as a discrete input vector of size n.
- Outputs are also a vector, one column for each atrribute of market, each value being a position from the sequence.
  - *product*: 0
  - *month*: 3
  - *strike*1: 1
  - *strike*2: 2
  - *strategy*: 3
  - *cross*: 4
  - *bid*: 5
  - *offer*: 6
- Now we can use any machine learning technique that can tolerate discrete input / output vectors

Conclusions

Use domain knowledge to simplify the learning problem

- Most algorithms don't work "out of the box" with traditional machine learning techniques

Conclusions

Use domain knowledge to simplify the learning problem

- Most algorithms don't work "out of the box" with traditional machine learning techniques
- But A good abstraction can make machine learning practically unnecessary

Future Work

- Consider sequence learning approaches, like hidden markov models or dynamic bayesian networks
- Incorporate analytical features directly into the probability model
- Unsupervised learning (use analytics to discover reasonable markets)