

# Compact Representations for “Large-scale” Computation

Vidit Jain

University of Massachusetts Amherst

April 17, 2009

Compression

Conditional probability trees

Hashing

Feature hashing

# Linear compression

- ▶ Johnson-Lindenstrauss Lemma [JL84]
  - ▶ choose a projection plane “at random”
  - ▶ the distances are “approximately” preserved with “high probability”
- ▶ Optimal recovery theorem [CRT06]
  - ▶ Assume that we are given  $K (\geq M \log N)$  random measurements. Then, the  $\ell_1$  minimizer to the convex problem is unique, and with high probability, is a *good* approximation.

# Linear compression

- ▶ Representation discovery
- ▶ Devil's advocate: space requirements

# An interesting problem setup

- ▶ Compute conditional probability
- ▶ Applications:  $\Pr[\text{ad-word} \mid \text{doc}]$ ,  $\Pr[\text{identity} \mid \text{face image}]$
- ▶ # classes,  $n \geq 1,000,000$
- ▶ One-vs-all classification, training:  $O(n)$
- ▶  $O(\log n)$  solution ?

# Use a conditional probability tree [BLL<sup>+</sup>09]

- ▶ Binary linear regressors at all internal nodes,  $f_i = Q_i(1|x)$
- ▶ Estimate  $Q(y|x) = \prod_{i \in T(y)} Q_i(\text{right}_i(y)|x)$
- ▶ **Theorem 1:** Any tree-based approach has squared loss bounded by the *tree-depth squared* times the average squared loss of the node regressors used.  $\Omega(\log n)$  computation.
- ▶ Geometric proof of Lemma 2 – *Beautiful!!*
- ▶ PECOC approach: squared loss bounded by *four* times the average squared loss.  $\Omega(n)$  computation.
- ▶ Accuracy-time tradeoff.  $4(\log_k n)^2 \binom{k-1}{k}^2$  loss factor.  $O(k \log_k n)$  computation.

# Online tree construction

- ▶ Algorithm 3
- ▶  $obj(p, L, R, \alpha) = (1 - \alpha)2(p - \frac{1}{2}) + \alpha \log_2 \frac{L+1}{R+1}$
- ▶ Free parameter  $\alpha$  determines the tradeoff between maintaining a balance tree and consistency with the existing regressor.

# Experiments

- ▶ Evaluation: empirical squared loss
- ▶ Progressive validation (online)
- ▶ Reuters RCV1
- ▶ Web advertising
- ▶ Same performance with much less computation.

# Internal nodes of this tree

- ▶ # internal nodes = # regressors to be learned =  $n$
- ▶ compact representation
  - ▶ computational and statistical efficiency
  - ▶ regression coefficients (model parameters)
  - ▶ random projections (parameters) ?
- ▶ *hash functions*
  - ▶ put similar data in the same bin(s).
  - ▶ do we need a projection matrix?

# Approximate neighbor search in high dimensions

- ▶ c-approx. r-near neighbor
- ▶ locality-sensitive hashing (LSH) [IM98]
  - ▶ if  $\|p - q\| \leq r$  then  $Pr[h(p) = h(q)]$  is “not-so-small”
  - ▶ if  $\|p - q\| > cr$  then  $Pr[h(p) = h(q)]$  is “small”
- ▶ Examples:  $h(x) = x_i$ , for hamming distance

# LSH algorithm

- ▶ Hash the  $d$ -dimensional points using  $L$  hash functions so as to ensure that, for each function, the probability of collision is much higher for neighboring points and smaller for others.
- ▶ Retrieve points from the buckets the query is hashed to.
- ▶ Query time:  $O(dL)$

# Compact regressors

- ▶ Near-neighbor  $\rightarrow$  probability estimate
- ▶ Hash functions  $\rightarrow$  regressors (internal nodes of that tree)
- ▶ Hashing vs. random projections
  - ▶ sparsity
  - ▶ projection matrix

# Feature hashing for multi-task learning [WDA<sup>+</sup>09]

- ▶ Kernel-trick vs. hashing
- ▶ With hashing, sparsity is preserved.
- ▶ Extension to a multi-task setting
  - ▶ different hash functions for each task
- ▶ Parameter vectors could be compressed

# Kernel approximation

- ▶ Generic randomization

$$\begin{aligned}k(x, x') &= \mathbf{E}_{c \sim P(c)} [\phi_c(x) \phi_c(x')] \\ \bar{k}(x, x') &= \frac{1}{n} \sum_{i=1}^n \phi_c(x) \phi_c(x') // \textit{sampling} \\ &= \langle \bar{\phi}(x), \bar{\phi}(x') \rangle \\ \bar{\phi} &= M\phi\end{aligned}$$

- ▶  $\bar{\phi}$  has more desirable computational properties than  $\phi$

# Hash kernel [SPD<sup>+</sup>09]

$$\bar{k}(x, x') = \langle \bar{\phi}(x), \bar{\phi}(x') \rangle$$

$$\bar{\phi}_i(x) = \sum_{j:h(j)=i} \phi_j(x)$$

- ▶ accumulate all coordinates  $j$  of  $x$  into coordinates of  $\phi$  for which the hash function  $h$  generates the same value  $i$
- ▶ Strings:  $\phi_j(x) := \lambda_j \#_j(x)$ .  $O(|x| + |x'|)$
- ▶ Multi-class:  $\bar{\phi}_i(x, y) = \sum_{j:h(i,y)=j} \phi_j(x)$

# Unbiased hash kernel [WDA<sup>+</sup>09]

- ▶ *signed* sum of hashed features.
- ▶  $\phi_i^{(h,\xi)}(x) = \sum_{j:h(j)=i} \xi(j)x_j$ 
  - ▶  $\xi : \mathbb{N} \rightarrow \{\pm 1\}$
- ▶ Proves exponential tail-bounds for efficient multi-task learning through hashing.
  - ▶ approximate orthogonality: little interaction between hash-feature space for multi-task hashing.
- ▶ Collaborative spam-filtering.

# References



A. Beygelzimer, J. Langford, Y. Lifshits, G. Sorkin, and A. Strehl.  
Conditional probability tree estimation analysis and algorithms.  
*ArXiv e-prints*, March 2009.



E.J. Candes, J. Romberg, and T. Tao.  
Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information.  
*IEEE Transactions on Information Theory*, 52(2):489–509, Feb. 2006.



Piotr Indyk and Rajeev Motwani.  
Approximate nearest neighbors: towards removing the curse of dimensionality.  
*In ACM symposium on Theory of computing*, 1998.



W. Johnson and J. Lindenstrauss.  
Extensions of lipschitz maps into a hilbert space.  
*Contemporary Mathematics*, 26:189–206, 1984.



Q. Shi, J. Petterson, G. Dror, J. Langford, A. Smola, A. Strehl, and V. Vishwanathan.  
Hash kernels.  
*In International Conference on Artificial Intelligence and Statistics*, 2009.



K. Weinberger, A. Dasgupta, J. Attenberg, J. Langford, and A. Smola.  
Feature hashing for large scale multitask learning.  
*ArXiv e-prints*, February 2009.