The Hashing Trick and other Related Methods for Speeding-Up Machine Learning

BIL 2014

Ricardo Ñanculef Alegría
jnancu@inf.utfsm.cl

Informatics Department
Federico Santa María University - Chile



Departamento de Informática
Universidad Técnica Federico Santa María

Context of this Presentation

Large-Scale Machine Learning, Data Mining and Optimization

- Machine Learning (ML): Extracting a predictor $f(x)$ from empirical data $x \in \mathbb{R}^d$.
- Data Mining (DM): Discovering a pattern $f(x)$ from empirical data $x \in \mathbb{R}^d$.
- Optimization: Usually ML and DM tasks can be formulated as optimization problems on a dataset $D \subset \mathbb{R}^{d \times m}$. For example,

$$\min_{w \in \mathbb{R}^d} \|w\|^2 \quad \text{s.t.} \quad \langle w, x_i \rangle + b \geq 1 \quad \forall i = 1, \ldots, m$$

- Complexity: Determined by m and d
  - **cardinality (m):** number of empirical observations.
  - **dimensionality (d):** number of attributes or dimensions of the feature space.
- Large-scale applications: Current ML applications need to manipulate lots of very high dimensional data (documents, images, microarray data).

Context of this Presentation

Curses of Dimensionality

- Training time increases (linear or super linearly) as the dimensionality increases.
- Space (memory) requirements increase as the dimensionality increases.
- Generalization ability (predictive power) reduces as the dimensionality increases.
- .. or ... required labelled training data increases as the dimensionality increases.

Dimensionality Reduction

- Design a map $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}^k$, with $k \ll d$ such that, the "structure" of data is approximately preserved.
- Most ML and DM algorithms are based on distances, norms or inner products. Thus, it would be nice to design maps $\varphi(\cdot)$ preserving these properties.

Context of this Presentation

- Most used method for dimensionality reduction:

PCA

- Let $X \in \mathcal{M}(m \times d)$ a matrix such that $X_{(k)}$ is the k-th datum.
- Empirical Covariance Matrix: $C = \frac{1}{m} X^T X \in \mathcal{M}(d \times d)$ (assuming centered inputs).
- Eigendecomposition: $C = V \Sigma V^T$.
- PCA: Let $A_k = V_k^T$, where V_k is composed of the first k columns of V. Thus, PCA is the map $\varphi : \mathbb{R}^d \to \mathbb{R}^k$,

$$\varphi(x) = A_k x$$

- **PCA is the optimal orthogonal projection into k dimensions** with respect to the reconstruction error $\sum_i \|x_i - V_k V_k^T x_i\|_{\ell_2}^2$.
- **However, PCA complexity is $\mathcal{O}(d^3)$ or $\mathcal{O}(md^2)$.**
- **In addition, PCA gives no guarantees for the preservation of specific/local norms, distances or inner products.**

The Johnson-Lindenstrauss Lemma (JLL)

Theorem [1]

Let $\epsilon \in (0, \frac{1}{2})$ and let $P = \{p_1, p_2, \ldots, p_m\}$ be a set of m points in $\mathbb{R}^d$. Let k be an integer with $k \geq C\epsilon^{-2} \log m$. Then, there exists a linear mapping $\varphi : \mathbb{R}^d \to \mathbb{R}^k$ such that

$$(1-\epsilon)\|p_i - p_j\| \leq \|\varphi(p_i) - \varphi(p_j)\| \leq (1+\epsilon)\|p_i - p_j\|$$

for all $i, j = 1, 2 \ldots, m$, where $\|\cdot\|$ denotes the ℓ_2 norm.

- The map is linear.
- The dimensionality of the map is independent of d.

The Johnson-Lindenstrauss Lemma (JLL) - variant

Alternative Statement

Let $\epsilon \in (0, \frac{1}{2})$, $\delta \in (0, 1)$ and $k = C\epsilon^{-2} \log \delta/2$. Then, there exists a linear mapping $\varphi : \mathbb{R}^n \to \mathbb{R}^k$ such that, $\forall x \in \mathbb{R}^d$

$$P\left[(1-\epsilon)\|\mathbf{x}\| \leq \|\varphi(\mathbf{x})\| \leq (1+\epsilon)\|\mathbf{x}\|\right] \geq 1-\delta$$

- For instance, $\epsilon = 0.1$ and $\delta = 0.01$ gives $k \approx 10^3$.
- In the text classification problems you can have $d = 10^6, 10^9$.
- The probability of not preserving a norm$\pm\epsilon$ decays exponentially with k.
- Preservation of inner products follows with probability $1 - 3\delta$ because $\langle x, y \rangle = \frac{1}{2}\left(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - \|\mathbf{x} - \mathbf{y}\|^2\right)$.

JLL Implementations

Original Map (by Johnson and Lindenstrauss, 1984) [1]

Orthogonal Projections on a Random Hyperplane:

- Choose k random orthonormal vectors from $\mathbb{R}^d$: $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k$.
- Arrange them as rows in a matrix $A \in \mathcal{M}(k \times d)$.

$$A_{(i)} = \mathbf{u}_i$$

- Define $\varphi(x) = \sqrt{n/k} \cdot A \cdot x$.
- A is orthonormal and has a spherically symmetric distribution.
- Generation time: $\mathcal{O}(dk + dk^2)$.
- Projection time / Memory: $\mathcal{O}(dk)$.
- In DM, the technique is called *random projections*.

JLL Implementations

A simpler map (Indyk and Motwani, 1998) [2]

Independent Gaussian Entries:

- Choose $A_{ij} \sim \mathcal{N}(0,1)$, $\forall i,j$.
- Define $\varphi(x) = \frac{1}{\sqrt{k}} \cdot A \cdot x$.

- A is no longer orthonormal.
- Note if $\|x\|^2 = 1$ then $\|\varphi(x)\|^2$ is chi-squared with k d.o.f.
- Exploit well known tail estimates for the chi-squared distribution.
- All complexities are still $\mathcal{O}(d \times k)$.

JLL Implementations

A sparser map (Achlioptas, 2003) [3]

- Define R to be the probability distribution with density

$$f(x) = \begin{cases} 1/6 & \text{if } x = \pm 1 \\ 2/3 & \text{if } x = 0 \end{cases}.$$

- Choose $A_{ij} \sim S$, $\forall i, j$.
- Define $\varphi(x) = \frac{1}{\sqrt{k}} \cdot A \cdot x$.

- In expectation only $1/3$ of the entries of A are different than 0.
- Projection is 3 times more efficient.
- Storage of A is 3 times cheaper.
- Complexities are still $\mathcal{O}(d \times k)$.

Sparse JLL Implementations

Can we reduce more the size of A?

- Can we reduce the number of reduced dimensions k?
- Optimal projection time: $\mathcal{O}(d)$.

Theorem (Alon, 2003)

The minimum number of required reduced dimensions (k) is

$$\mathcal{O}\left(\frac{\epsilon^{-2}\log\delta/2}{\log(1/\epsilon)}\right) .$$

- Can we improve sparsity in a super-constant way? $\mathcal{O}(d)$ non zero entries?
- Problem: Too sparse matrices can "destroy" sparse vectors, e.g. $x = (1, 0, 0, \ldots, 0)$.

Sparse JLL Implementations

The Fast Johnson Lindenstrauss Transform (Ailon and Chazelle, 2009)

- **Idea:** Precondition the inputs using a randomized Fourier transform P to ensure they are dense enough.
- Preconditioning (computing Px) is $\mathcal{O}(d \log d)$ time.

Lemma (Ailon and Chazelle, 2009)

Let H be a $d \times d$ normalized Walsh-Hadamard matrix and D be a $d \times d$ diagonal matrix with independent Rademacher entries (± 1 with probability $1/2$). Thus, for any x, $\|HDx\|_\infty = \max_i x_i \leq \sqrt{\log(k)/d}\ \|x\|$.

- $P = HD$ densifies x and preserves the euclidean norm (it is orthogonal).
- Computing Px is fast: $\mathcal{O}(d \log d)$ time.

Sparse JLL Implementations

The Fast Johnson Lindenstrauss Transform (Ailon and Chazelle, 2009)

- Combine the densification trick with a sparse projection.

Sparse JLL Matrix (Ailon and Chazelle, 2009)

- Suppose $\|x\|_\infty \leq \sqrt{k/d}\ \|x\|$.
- Let $q = C_0 k^2/d$ (sparseness parameter).
- $\forall i, j$: Set $A_{ij} = 0$ with probability $1 - q$ and $A_{ij} \sim \mathcal{N}(0, 1/q)$ with probability q.
- Thus, with probability at least $1 - \delta$, $\varphi(x) = \frac{1}{k}\ A \cdot x$ preserves the norm of x up to a multiplicative constant $1 \pm \epsilon$.

- Non zero entries are k^3/d in expectation, thus ...
- Projection time: $\mathcal{O}(k^3 + d \log d)$.
- Space: $\mathcal{O}(d + k^3)$.

The Hashing Trick

- Weiberger, Attenberg, Langford and others in 2009: were studying ways to speed up inner product computations $\langle x, y \rangle$, $x, y \in \mathbb{R}^d$.

Idea: Feature Hashing

- Let $h : [d] \to [k]$ be a random hash function mapping the coordinates of $\mathbb{R}^d$ to the coordinates of $\mathbb{R}^k$, e.g., $h : [6] \to [3]$

i	h(i)
1	2
2	1
3	3
4	2
5	3
6	1

The Hashing Trick

Idea: Feature Hashing

- $\langle x, y \rangle$ can be approximated using the following estimator called the hash kernel:

$$\langle x, y \rangle_h = \langle \varphi(x), \varphi(y) \rangle_h \quad ,$$

where the map $\varphi(\cdot)$ is:

$$\varphi(x) := \sum_{i:h(i)=j} \xi_i \, x_i \ .$$

- After computing $\langle x, y \rangle_h$ is $\mathcal{O}(k)$ instead $\mathcal{O}(d)$.
- Application Complexity : $\mathcal{O}(d)$.

The Hashing Trick

- The hashing trick implemented as a matrix multiplication.

i	h(i)
1	2
2	1
3	3
4	2
5	3
6	1

$$Ax = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & +1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \end{pmatrix} x$$

$$= \begin{pmatrix} x_2 - x_6 \\ -x_1 + x_4 \\ x_3 + x_5 \end{pmatrix}$$

The Hashing Trick

Theorem (Weiberger 2009)

$$E_h\left(\langle x, y\rangle_h\right) = \langle x, y\rangle \quad \mathrm{Var}_h\left(\langle x, y\rangle_h\right) = \mathcal{O}\left(\frac{1}{k}\right) \ .$$

Theorem (Dasgupta 2010)

Let $\epsilon \in (0,1)$ and $k > C\epsilon^{-2}\log(2/\delta)$. Suppose $\|x\|_\infty \leq \alpha\|x\|$ with $\alpha \leq \frac{\sqrt{\epsilon}}{16\log\frac{k}{\delta}\sqrt{\log\frac{1}{\delta}}}$.
Then, with probability at least $1-\delta$,

$$(1-\epsilon)\|\mathbf{x}\|^2 \leq \|\varphi(\mathbf{x})\|^2 \leq (1+\epsilon)\|\mathbf{x}\|^2$$

- Preservation of inner products follows with probability $1-3\delta$ because,

$$\langle x, y\rangle = \frac{1}{2}\left(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - \|\mathbf{x}-\mathbf{y}\|^2\right)$$

The Hashing Trick - Complexity

- Computing the feature hashing embedding $\varphi(x)$ is $\mathcal{O}(d)$, the lowest possible complexity! Densification is also efficient.

Complexity Comparison

Method	Generation	Space	Application	Densification
Jonhson&Lindenstrauss, 1984	$\mathcal{O}(kd)$	$\mathcal{O}(kd)$	$\mathcal{O}(kd)$	-
Indyk&Motwani, 1998	$\mathcal{O}(kd)$	$\mathcal{O}(kd)$	$\mathcal{O}(kd)$	-
Achlioptas, 2003	$\mathcal{O}(kd)$	$\mathcal{O}(kd/3)$	$\mathcal{O}(kd/3)$	-
Ailon&Chazelle, 2009	$\mathcal{O}(kd)$	$\mathcal{O}(d+k^3)$	$\mathcal{O}(d+k^3)$	$\mathcal{O}(d \log d)$
Weiberger et al., 2009	$\mathcal{O}(1)$	$\mathcal{O}(d)$	$\mathcal{O}(d)$	?
Dasgupta et al., 2010	✓	✓	✓	$\tilde{\mathcal{O}}(d\epsilon^{-1})$
Dasgupta et al., 2010	✓	✓	✓	$\tilde{\mathcal{O}}(d)$
Ailon&Liberty, 2011	$\mathcal{O}(1)$	$\mathcal{O}(d)$	$\mathcal{O}(d)$	$\mathcal{O}(d)$

- Ailon&Liberty recently embedded the random projection into a new densification process to obtain a method which is $\mathcal{O}(d)$ for any vector.

Main Application

Learning on the Reduced Space

- Suppose you have to learn a model $f(x)$ from a dataset $D \subset \mathbb{R}^{m \times d}$.
  1. Embed the data in $\mathbb{R}^k$ using the hashing trick.
  2. Learn f_φ from the reduced dataset $D_\varphi \subset \mathbb{R}^{m \times k}$.
  3. Embed test data in $\mathbb{R}^k$ and predict using the reduced model f_φ.
  4. Compare the prediction error of the model trained on the compressed data with the model trained in the original space (bounds or empirical comparisons).

$$P(f_\varphi(x) \neq f(x))?$$

- $\delta = P(f_\varphi(x) \neq f(x))$ depends on which and how much information you need to preserve.
  1. For a decision tree you do not get easy guarantees: JLL ensures the preservation of norms, original attributes are lost.
  2. For an SVM you need to preserve essentially the $m^2/2$ inner products among data
  3. For an 1-nn model you essentially need to preserve essentially the m distances

Learning on the Reduced Space

Some Toy Examples - No Big

- SPAM dataset: classification problem, 4601 examples, 57 dimensions.

	5	10	15	20	25	30	full
DTree+PCA	89.1	89.6	89.5	89.4	89.4	89.3	92.3
DTree+HRP	74.7	81.7	83.4	84.4	85.1	85.4	
1NN+PCA	89.2	90.5	90.6	90.6	90.5	90.4	90.5
1NN+HRP	79.0	86.4	88.2	89.1	89.5	89.7	
SVM+PCA	88.0	90.0	90.7	99.0	91.3	91.4	92.3
SVM+HRP	69.8	77.8	81.2	84.3	86.4	88.0	

Learning on the Reduced Space

Some Toy Examples - No Big

- COLON dataset: classification problem, 62 examples, 2000 dimensions.

	10	50	100	200	500	full
DTree+PCA	73.7	72.2	72.2	72.2	72.2	75.5
DTree+HRP	63.8	66.2	66.1	67.1	65.3	
1NN+PCA	68.8	73.3	73.3	73.3	73.3	73.3
1NN+HRP	64.6	70.0	71.0	71.3	72.0	
SVM+PCA	78.8	80.5	80.5	80.5	80.5	80.5
SVM+HRP	67.3	75.8	77.6	78.8	80.1	

Learning on the Reduced Space

Some Toy Examples - No Big

- ADS dataset: classification problem, 3279 examples, 1554 dimensions.

	10	25	50	100	200	500	full
DTree+PCA	95.9	95.9	95.6	95.5	95.5	94.7	96.0
DTree+HRP	89.0	89.6	90.0	90.2	90.5	90.7	
1NN+PCA	95.2	95.2	96.0	95.9	95.9	95.8	95.8
1NN+HRP	92.7	95.1	95.6	95.7	95.9	95.8	
SVM+PCA	94.5	94.5	94.3	94.8	96.1	96.6	96.8
SVM+HRP	86.0	87.6	90.9	93.6	95.4	96.5	

Learning on the Reduced Space

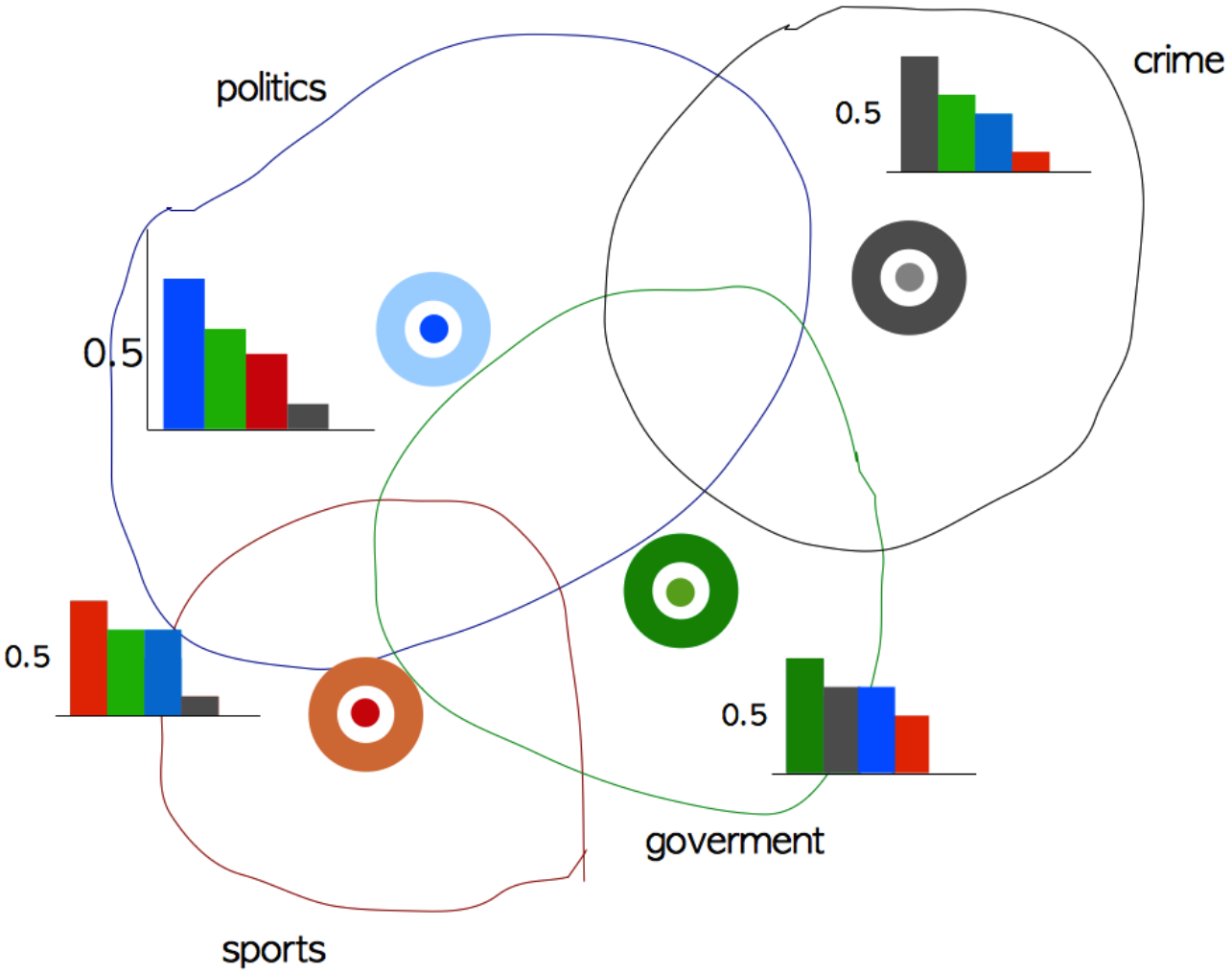
- A more interesting application:

Text Stream Classification (Nanculef, Flaounas, Cristianini 2014)

- A virtually infinite sequence of documents $d_1, d_2, \ldots, d_t$ each one containing some tags (e.g. categories).
- The task is to predict the tags which correspond to a test document d.
- Each document d is represented as a vector $x = x(d)$ where each dimension of x is the frequency of a word from a vocabulary V (TF representation).
- V is formed from all the distinct words seen so far. Thus: very very large, unknown in advance.
- Hashing allows a fixed size vector representation:
  - Instead of hashing $[d]$ to $[k]$, hash directly V to $[k]$.
  - You do not need the original representation, so you do not need to store previous documents.
  - JLL guarantees the geometry of the original space is preserved.

Learning on the Reduced Space

The Classifier



Learning on the Reduced Space

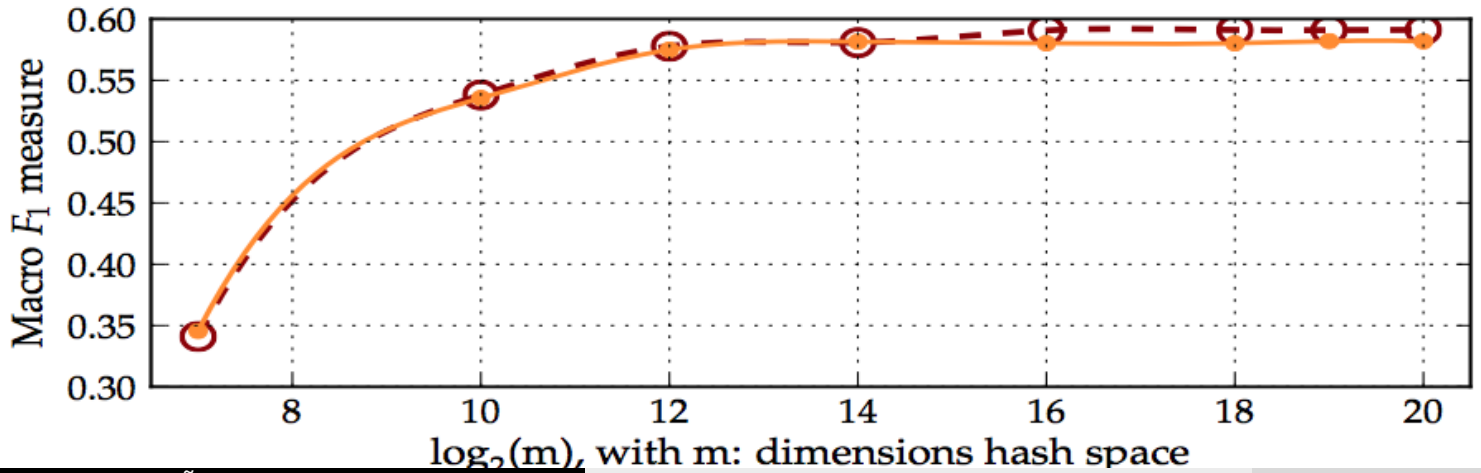
Theorem

Let $f_\varphi(x)$ the classifier computed in the reduced space and $f(x)$ the classifier computed in the original space (never seen). Thus

$$P(f_\varphi(x) \neq f(x)) \leq \exp\left(-\frac{Ck}{\eta^\star n_p}\right) ,$$

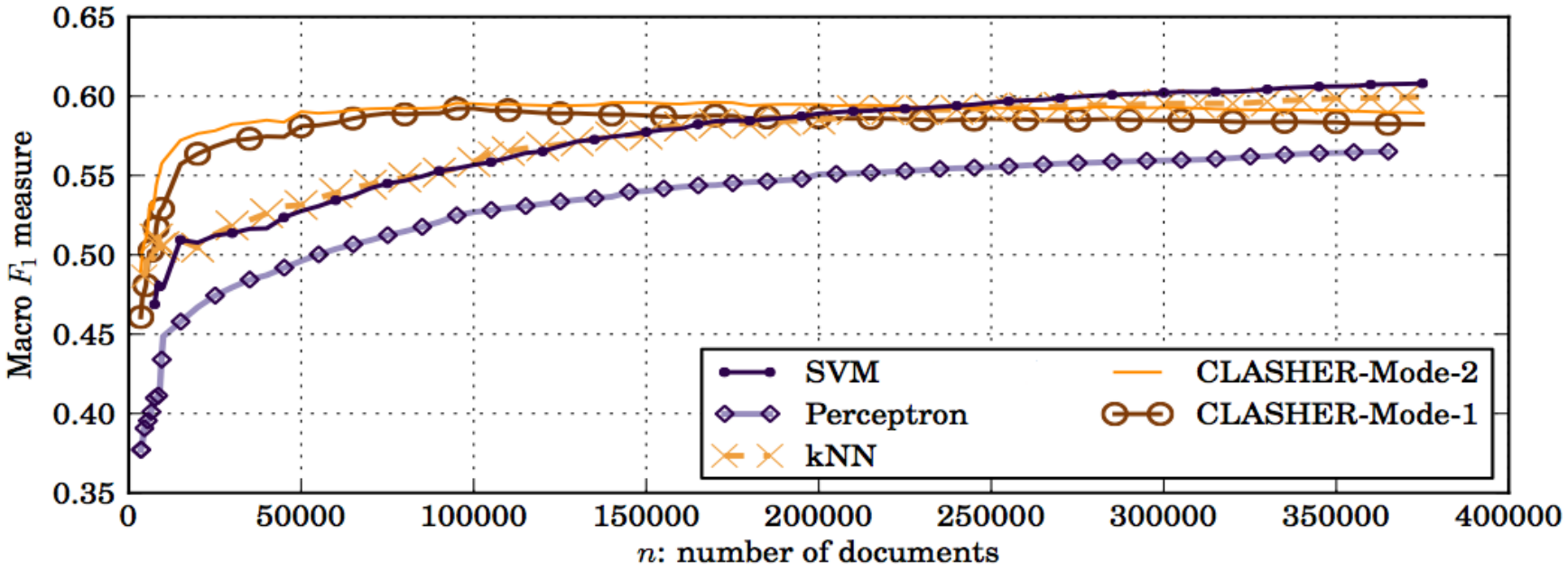
where n_p is the number of prototype in the system and $\eta^\star$ is the margin of separation among classes in the original space.

Reuters Dataset: $\approx 10^6$ documents, $\approx 10^6$ words



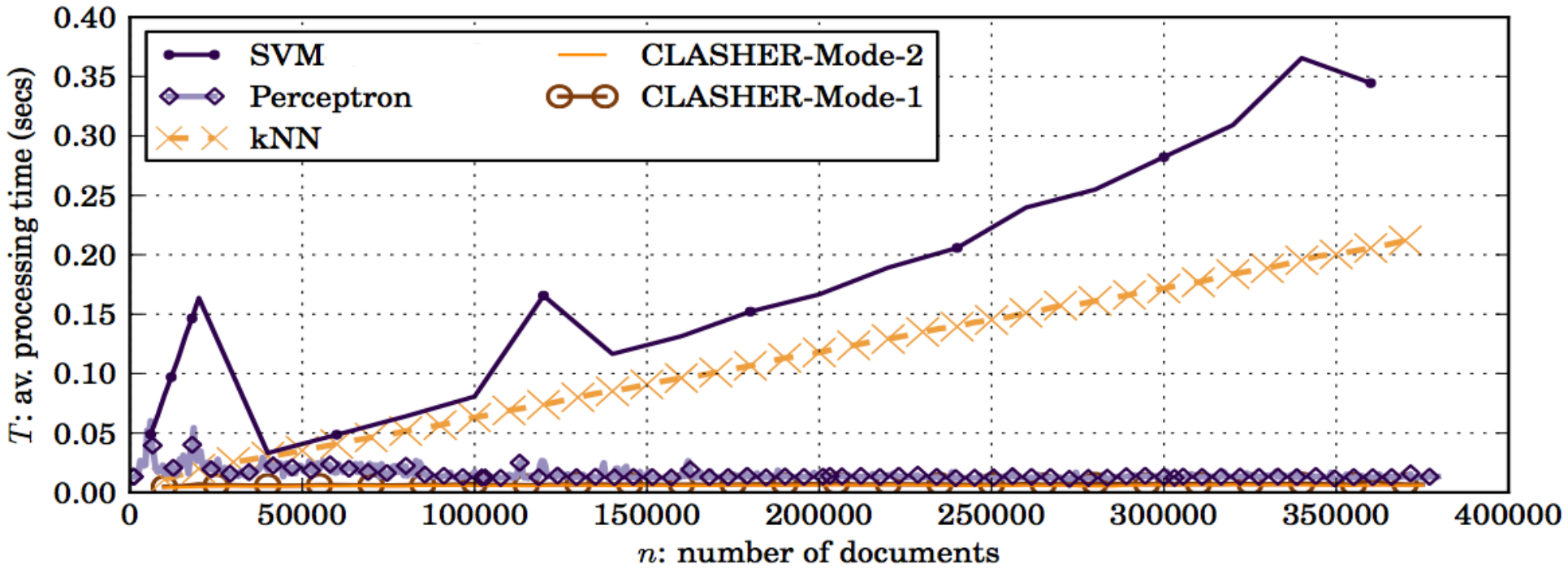
Learning on the Reduced Space

Reuters Dataset: $\approx 10^6$ documents, $\approx 10^6$ words, $k = 4096$



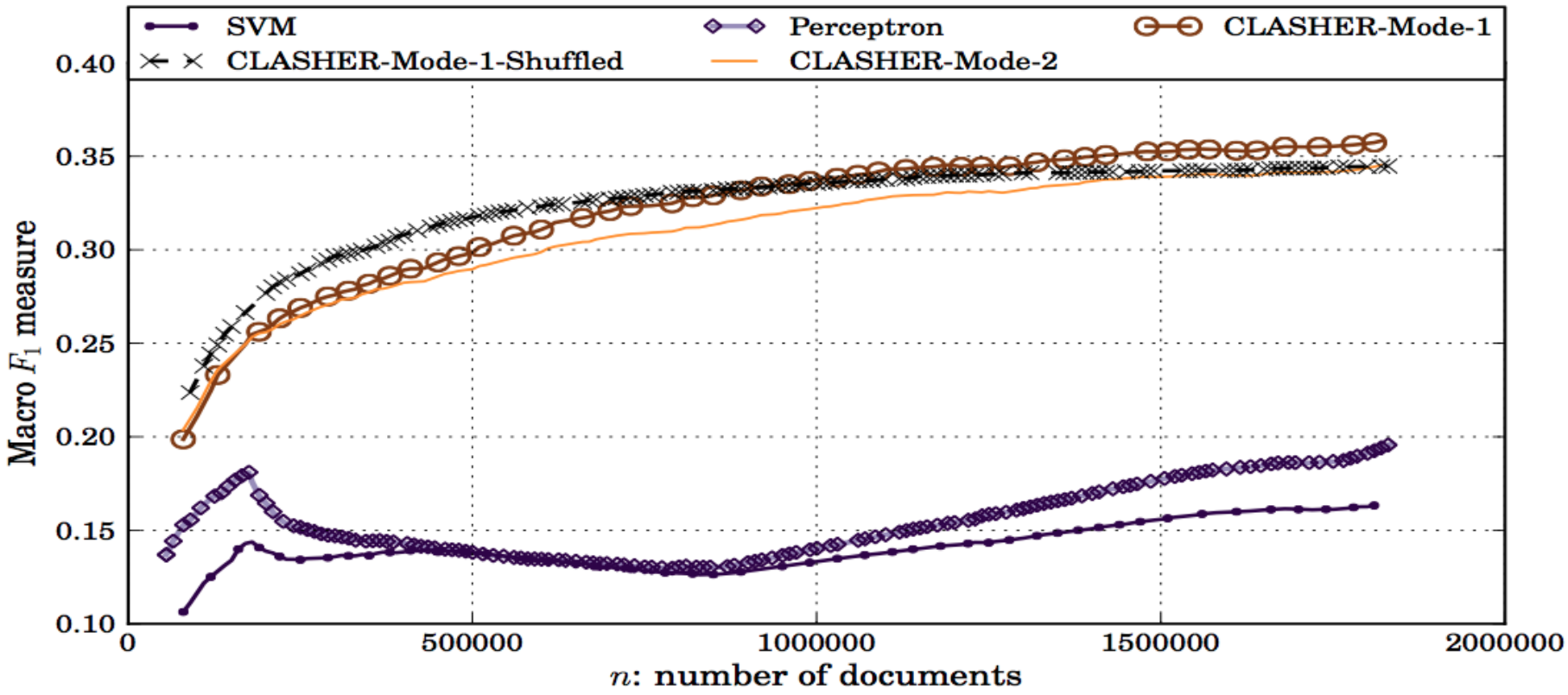
Learning on the Reduced Space

Reuters Dataset: $\approx 10^6$ documents, $\approx 10^6$ words, $k = 4096$



Learning on the Reduced Space

New York Times Dataset: $\approx 2 \times 10^6$ documents, $\approx 10^6$ words, $k = 4096$



Conclusions

- JLL implementations, in particular, the Hashing Trick are very efficient alternatives for dimensionality reduction in large scale applications.
- Machine Learning in the reduced space is feasible, efficient and amenable to analysis.
- Usually you pay a cost in accuracy with respect to more expensive techniques.
- The cost depends on the type of model and the properties you need to preserve.
- Usually, you tradeoff simplicity/power of a model with the guarantees of low corruption.

The End

Many thanks for your attention!

[1] B. Johnson and J. Lindenstrauss, *Extensions of Lipschitz mappings into a Hilbert space*, Contemp. Math. 26 (1984), pp. 189–206.

[2] P. Indyk and R. Motwani, *Approximate nearest neighbors: Towards removing the curse of dimensionality*, Proc. 30th ACM Symp. Theory of Computing (1998), pp. 604–613.

[3] D. Achlioptas, *Database-friendly random projections: Johnson-Lindenstrauss with binary coins*, J. Comput. System Sci. 66 (2003), pp 671–687.

[4] S. Dasgupta and A. Gupta, *An elementary proof of a theorem of Johnson and Lindenstrauss*, Random Structures Algorithms 22 (2003), pp. 60–65.

[5] J. Matousek, *On variants of the Johnson–Lindenstrauss lemma*. Random Structures & Algorithms, 33(2), pp. 142-156.

[6] N. Ailon, B. Chazelle, *Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform*. Proc. 38th ACM Symp Theory of computing (2006) pp. 557-563.