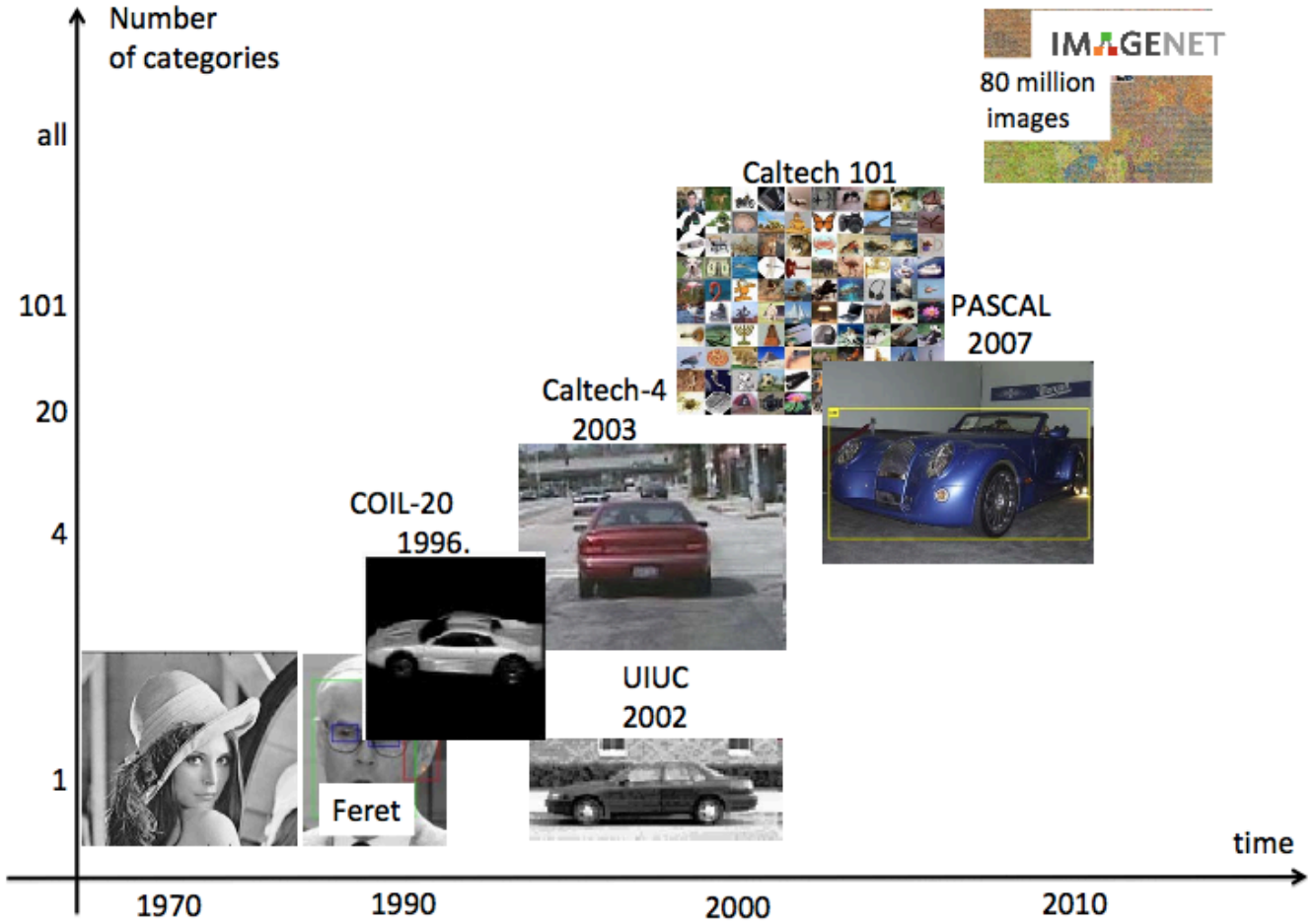
Large-scale learning for image classification

Zaid Harchaoui

CVPR Tutorial, June 2013

Large-scale image datasets



From "The Promise and Perils of Benchmark Datasets and Challenges", D. Forsyth, A. Efros, F.-F. Li, A. Torralba and A. Zisserman, Talk at "Frontiers of Computer Vision"

Large-scale supervised learning

Large-scale image classification

Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \{1, \ldots, k\}$ be labelled training images

$$\underset{\mathbf{W} \in \mathbb{R}^{d \times k}}{\text{Minimize}} \quad \lambda \, \Omega(\mathbf{W}) + \frac{1}{n} \sum_{i=1}^{n} L\big(y_i, \mathbf{W}^T \mathbf{x}_i\big)$$

Problem : minimizing such objectives in the large-scale setting

$$n \gg 1, \quad d \gg 1, \quad k \gg 1$$

Large-scale supervised learning

Large-scale image classification

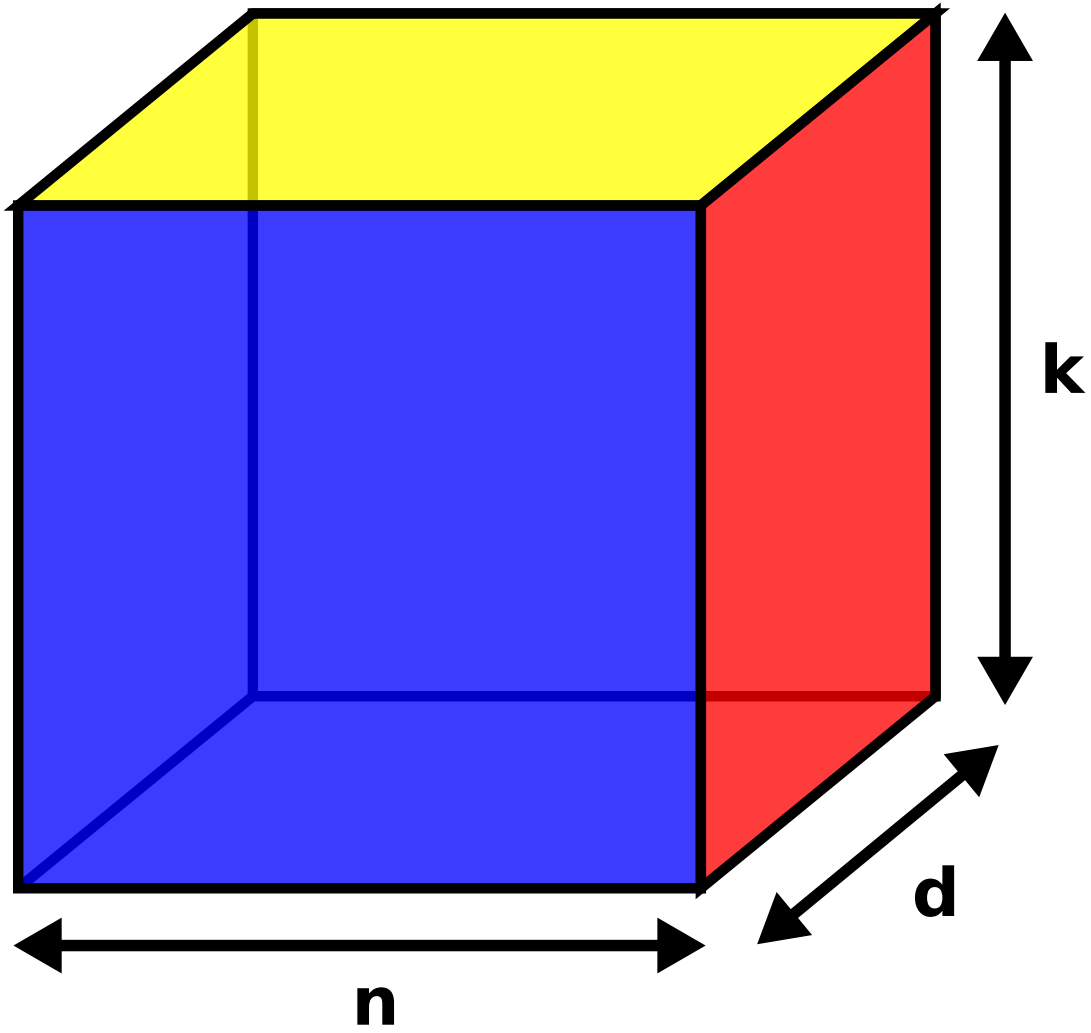
Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \{1, \ldots, k\}$ be labelled training images

$$\underset{\mathbf{W} \in \mathbb{R}^{d \times k}}{\text{Minimize}} \quad \lambda \, \Omega(\mathbf{W}) + \frac{1}{n} \sum_{i=1}^{n} L\big(y_i, \mathbf{W}^T \mathbf{x}_i\big)$$

Problem : minimizing such objectives in the large-scale setting

$$n \gg 1, \quad d \gg 1, \quad k \gg 1$$

Machine learning cuboid



Working example : ImageNet dataset

ImageNet dataset

- Large number of examples : $n = 17$ millions
- Large feature size : $d = 4.10^3, \ldots, 2.10^5$
- Large number of categories : $k = 10,000$

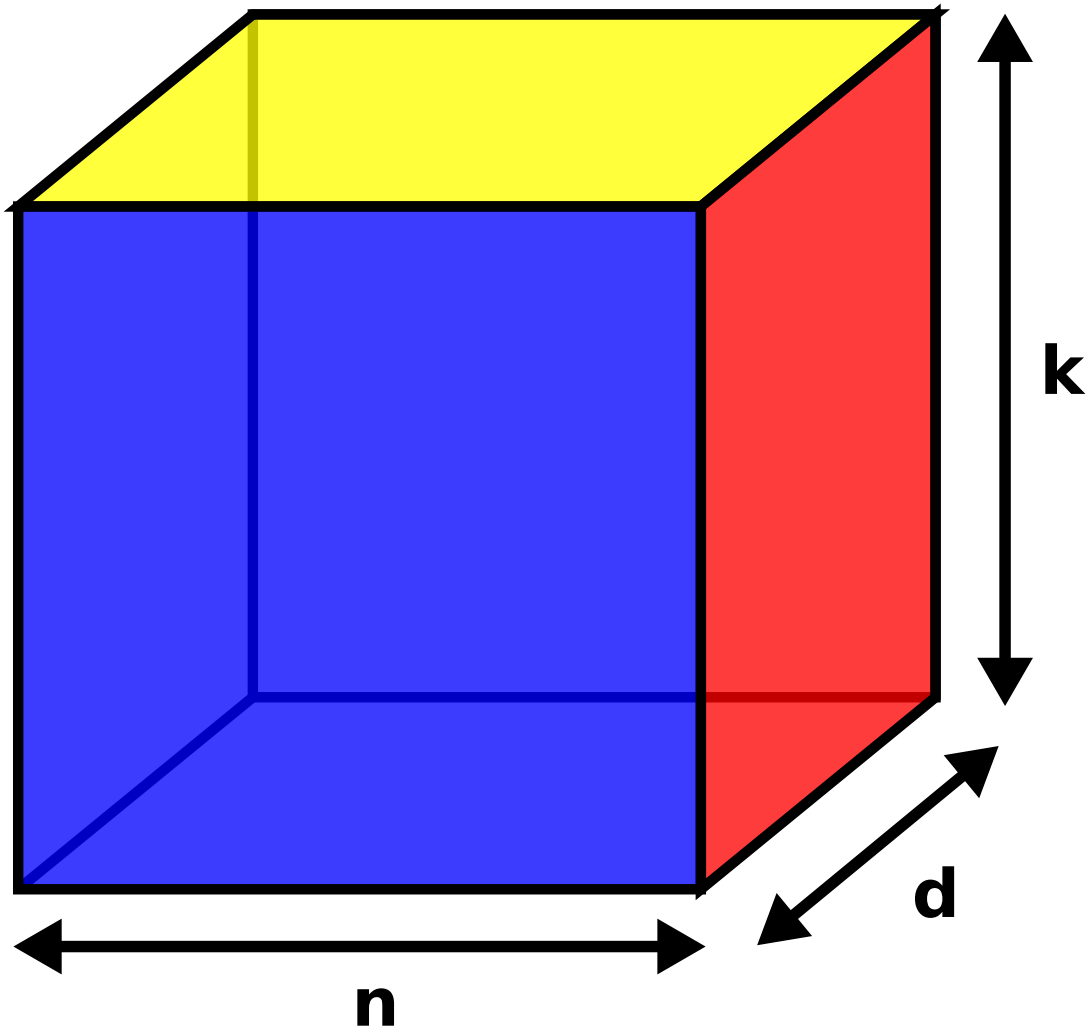
General strategy for large-scale problems

Strategy

Most approaches boil down to a general "divide-and-conquer" strategy

Break the large learning problem into small and easy pieces

Machine learning cuboid



Decomposition principle

Decomposition principle

- Decomposition over examples : stochastic/incremental gradient descent
- Decomposition over features : (primal) regular coordinate descent
- Decomposition over categories : one-versus-rest strategy
- Decomposition over latent structure : atomic decomposition

Decomposition principle

Decomposition principle

- Decomposition over examples : stochastic/incremental gradient descent
- Decomposition over features : (primal) coordinate descent
- Decomposition over categories : one-versus-rest strategy
- Decomposition over latent structure : atomic decomposition

Decomposition over examples

Stochastic/incremental gradient descent

- Bru, 1890 : algorithm to adjust a slant θ of cannon in order to obtain a specified range r by trial and error, firing one shell after another

$$\theta_t = \theta_{t-1} - \frac{\gamma_0}{t}(r - r_t)$$

- Perceptron, Rosenblatt, 1957

$$\begin{aligned} \mathbf{w}_t &= \mathbf{w}_{t-1} - \gamma_t(y_t \phi(\mathbf{x}_t)) && \text{if } y_t \phi(\mathbf{x}_t) \leq 0 \\ &= \mathbf{w}_{t-1} && \text{otherwise} \end{aligned}$$

Decomposition over examples

Stochastic/incremental gradient descent

- Bru, 1890 : algorithm to adjust a slant θ of cannon in order to obtain a specified range r by trial and error
- Perceptron, Rosenblatt, 1957
- 60s-70s : extensions in learning, optimal control, and adaptive signal processing
- 80s-90s : extensions to non-convex learning problems
- see "Efficient backprop" in *Neural networks : Tricks of the trade*, LeCun et al., 1998, for wise advice and overview on sgd algorithms

Decomposition over examples

Stochastic/incremental gradient descent

- **Initialize** : $\mathbf{W} = 0$
- **Iterate** : pick an example $(\mathbf{x}_t, y_t)$

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \gamma_t \underbrace{\nabla_{\mathbf{W}} Q(\mathbf{W}; \mathbf{x}_t, y_t)}_{\text{one example at a time}}$$

Why ?

Where does these update rules come from ?

Plain gradient descent

Plain gradient descent versus stochastic/incremental gradient descent

Grouping the regularization penalty and the empirical risk

$$\nabla_{\mathbf{W}} J(\mathbf{W}) = \frac{1}{n} \sum_{i=1}^{n} \left\{ n\lambda \, \Omega(\mathbf{W}) + L\left(y_i, \mathbf{W}^T \mathbf{x}_i\right) \right\}$$

Plain gradient descent

Plain gradient descent versus stochastic/incremental gradient descent

Grouping the regularization penalty and the empirical risk, and expanding the sum onto the examples

$$\begin{aligned}\nabla_{\mathbf{W}} J(\mathbf{W}) &= \frac{1}{n} \sum_{i=1}^{n} \left\{ n\lambda\, \Omega(\mathbf{W}) + L\left(y_i, \mathbf{W}^T \mathbf{x}_i\right) \right\} \\ &= \nabla_{\mathbf{W}} \left\{ \frac{1}{n} \sum_{i=1}^{n} Q(\mathbf{W}; \mathbf{x}_i, y_i) \right\}\end{aligned}$$

Plain gradient descent

Plain gradient descent

- **Initialize** : $\mathbf{W} = 0$
- **Iterate** :

$$\begin{aligned}\mathbf{W}_{t+1} &= \mathbf{W}_t - \gamma_t \nabla J(\mathbf{W}) \\ &= \mathbf{W}_t - \gamma_t \nabla_{\mathbf{W}} \left\{ \frac{1}{n} \sum_{i=1}^{n} Q(\mathbf{W}; \mathbf{x}_i, y_i) \right\}\end{aligned}$$

Plain gradient descent

Plain gradient descent

- **Initialize** : $\mathbf{W} = 0$
- **Iterate** :

$$\begin{aligned}\mathbf{W}_{t+1} &= \mathbf{W}_t - \gamma_t \nabla_{\mathbf{W}} J(\mathbf{W}) \\ &= \mathbf{W}_t - \gamma_t \underbrace{\nabla_{\mathbf{W}} \left\{ \frac{1}{n} \sum_{i=1}^{n} Q(\mathbf{W}; \mathbf{x}_i, y_i) \right\}}_{\text{sum over all examples !}}\end{aligned}$$

Strengths and weaknesses

- Strength : robust to setting of step-size sequence (line-search)
- Weakness : demanding disk/memory requirements

Stochastic/incremental gradient descent

Stochastic/incremental gradient descent

Leveraging the decomposable structure over examples

$$\nabla_{\mathbf{W}} J(\mathbf{W}) = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\mathbf{W}} Q(\mathbf{W}; \mathbf{x}_i, y_i)$$

$$= \frac{1}{n} \left\{ \nabla_{\mathbf{W}} Q(\mathbf{W}; \mathbf{x}_1, y_1) + \cdots + \frac{1}{n} (\nabla_{\mathbf{W}} Q(\mathbf{W}; \mathbf{x}_n, y_n) \right\}$$

Decomposition over examples

Stochastic/incremental gradient descent

- Leveraging the decomposable structure over examples

$$\nabla_{\mathbf{W}} J(\mathbf{W}) = \frac{1}{n} \left\{ \underbrace{\nabla_{\mathbf{W}} Q(\mathbf{W}; \mathbf{x}_1, y_1)}_{\text{cheap to compute}} + \cdots + \underbrace{\nabla_{\mathbf{W}} Q(\mathbf{W}; \mathbf{x}_n, y_n)}_{\text{cheap to compute}} \right\}$$

- Make incremental gradient steps along $Q(\mathbf{W}; \mathbf{x}_t, y_t)$ at each iteration t, instead of full gradient steps along $\nabla J(\mathbf{W})$ at each iteration

Stochastic/incremental gradient descent

Stochastic/incremental gradient descent

- **Initialize** : $\mathbf{W} = 0$
- **Iterate** : pick an example $(\mathbf{x}_t, y_t)$

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \gamma_t \nabla_{\mathbf{W}} Q(\mathbf{W}; \mathbf{x}_t, y_t)$$

Stochastic/incremental gradient descent

Stochastic/incremental gradient descent

- **Initialize** : $\mathbf{W} = 0$
- **Iterate** : pick an example $(\mathbf{x}_t, y_t)$

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \gamma_t \underbrace{\nabla_{\mathbf{W}} Q(\mathbf{W}; \mathbf{x}_t, y_t)}_{\text{one example at a time}}$$

Strengths and weaknesses

- Strength : little disk requirements
- Weakness : may be sensitive to setting of step-size sequence

Stochastic/incremental gradient descent

What's "stochastic" in this algorithm ?

Looking at the objective as a stochastic approximation of the expected training error

$$\begin{aligned}\nabla_{\mathbf{W}} J(\mathbf{W}) &= \frac{1}{n}\sum_{i=1}^{n} \nabla_{\mathbf{W}} Q(\mathbf{W}; \mathbf{x}_i, y_i) \\ &= \frac{1}{n}\left\{ \nabla_{\mathbf{W}} Q(\mathbf{W}; \mathbf{x}_1, y_1) + \cdots + \frac{1}{n}(\nabla_{\mathbf{W}} Q(\mathbf{W}; \mathbf{x}_n, y_n) \right\}\end{aligned}$$

Stochastic/incremental gradient descent

What's "stochastic" in this algorithm ?

$$\begin{aligned}\nabla_{\mathbf{W}} J(\mathbf{W}) &= \frac{1}{n} \sum_{i=1}^{n} \nabla_{\mathbf{W}} Q(\mathbf{W}; \mathbf{x}_i, y_i) \\ &\approx \mathbf{E}_{\mathbf{x},y}[\nabla_{\mathbf{W}} Q(\mathbf{W}; \mathbf{x}, y)]\end{aligned}$$

Practical consequences

- Shuffle the examples before launching the algorithm, in case they form a correlated sequence
- Perform several passes/epochs over the training data, shuffling the examples before each pass/epoch

Mini-batch extensions

Mini-batch extensions

- Regular stochastic gradient descent : *extreme* decomposition strategy picking *one example* at a time
- Mini-batch extensions : decomposition onto *mini-batches* of size B_t at iteration t

When to choose one or the other ?

- Regular stochastic gradient descent converges for simple objectives with "moderate non-smoothness"
- For more sophisticated objectives, SGD does not converge, and mini-batch SGD is a must

Theory digest

Theory digest

- Fixed stepsize $\gamma_t \equiv \gamma \longrightarrow$ stable convergence
- Decreasing stepsize $\gamma_t = \frac{\gamma_0}{t+t_0} \longrightarrow$ faster local convergence, with γ_0 and t_0 properly set
- Note : stochastic gradient descent is an *extreme* decomposition strategy picking *one example* at a time

In practice

- Pick a random batch of reasonable size, and find best pair (γ_0, t_0) through cross-validation
- Run stochastic gradient descent with sequence of decreasing stepsize $\gamma_t = \frac{\gamma_0}{t+t_0}$

Tricks of the trade : life is simpler in large-scale settings

Life is simpler in large-scale settings

- Shuffle the examples before launching the algorithm, and process the examples in a balanced manner w.r.t the categories
- Regularization through early stopping : perform only a few several passes/epochs over the training data, and stop when the accuracy on a held-out validation set does not increase anymore
- Fixed step-size works fine : find best γ through cross-validation on a small batch

Stochastic/incremental gradient descent

Put the shoulder to the wheel

Let's try it out !

Ridge regression

Ridge regression

Training data : $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$

$$\underset{\mathbf{w} \in \mathbb{R}^d}{\text{Minimize}} \quad \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} L\big(y_i, \mathbf{w}^T\mathbf{x}_i\big)$$

Key calculations

$$Q(\mathbf{w}; \mathbf{x}_i, y_i) = \frac{n\lambda}{2}\|\mathbf{w}\|_2^2 + (y_i - \mathbf{w}^T\mathbf{x}_i)^2$$
$$\nabla Q(\mathbf{w}; \mathbf{x}_i, y_i) = n\lambda\mathbf{w} + (y_i - \mathbf{w}^T\mathbf{x}_i)\mathbf{x}$$

Logistic regression

Logistic regression

Training data : $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$

$$\underset{\mathbf{w} \in \mathbb{R}^d}{\text{Minimize}} \quad \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} L\big(y_i, \mathbf{w}^T\mathbf{x}_i\big)$$

Key calculations

$$Q(\mathbf{w}; \mathbf{x}_i, y_i) = \frac{n\lambda}{2}\|\mathbf{w}\|_2^2 + \log\left(1 + \exp(-y_i\mathbf{w}^T\mathbf{x}_i)\right)$$

$$\nabla Q(\mathbf{w}; \mathbf{x}_i, y_i) = n\lambda\mathbf{w} + -\frac{1}{1 + \exp(y_i\mathbf{w}^T\mathbf{x}_i)}y_i\mathbf{x}_i$$

Linear SVM with linear hinge loss

Linear SVM with linear hinge loss

Training data : $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$

$$\underset{\mathbf{w} \in \mathbb{R}^d}{\text{Minimize}} \quad \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} L\big(y_i, \mathbf{w}^T\mathbf{x}_i\big)$$

Key calculations

$$Q(\mathbf{w}; \mathbf{x}_i, y_i) = \frac{n\lambda}{2}\|\mathbf{w}\|_2^2 + \max\left(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right)$$

$$\nabla Q(\mathbf{w}; \mathbf{x}_i, y_i) = \begin{cases} n\lambda\mathbf{w} - y_i\mathbf{x}_i & \text{if } 1 - y_i\mathbf{x}_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

Linear SVM with linear hinge loss

Linear SVM with linear hinge loss

Training data : $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$

$$\underset{\mathbf{w}\in\mathbb{R}^d}{\text{Minimize}} \quad \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} L\big(y_i, \mathbf{w}^T\mathbf{x}_i\big)$$

Non-differentiable loss

- Rule : if $Q(\mathbf{w}; \mathbf{x}, y)$ has a finite number of a non-differentiable points, then just make no update, and pick another example.
- Theoretical justification : the set of a non-differentiable points will have measure zero, and convergence guarantee is still valid

$$\nabla \mathbf{E}_{\mathbf{x},y}[Q(\mathbf{W}; \mathbf{x}, y)] = \mathbf{E}_{\mathbf{x},y}[\nabla Q(\mathbf{W}; \mathbf{x}, y)]\,.$$

A quick overview

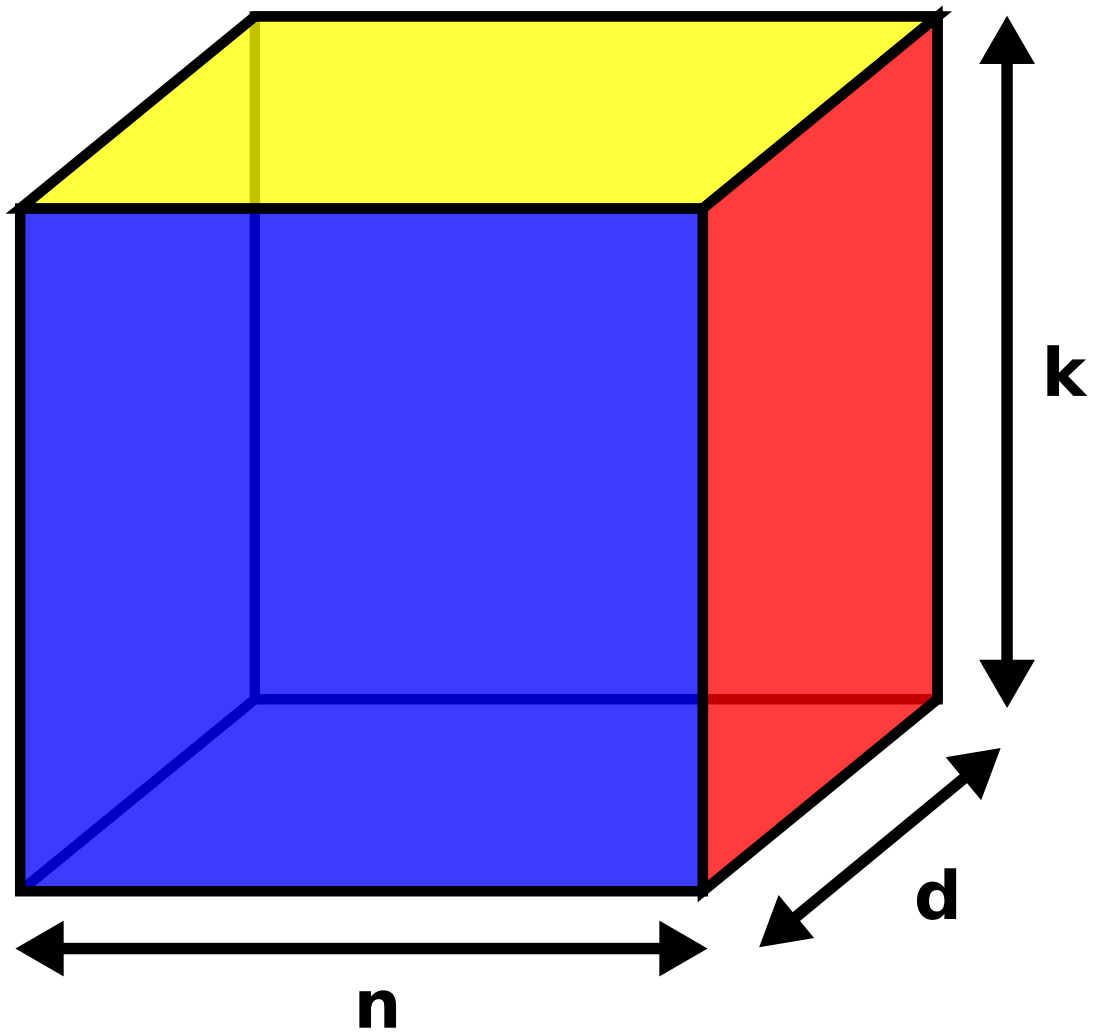
Convergence guarantees

- Least-square loss : smooth $\rightarrow$ fast and stable convergence
- Logistic loss : smooth $\rightarrow$ fast and stable convergence
- Linear hinge loss : non-smooth $\rightarrow$ slower convergence

Convergence guarantees

Take-home message : smooth loss is nicer

Machine learning cuboid



Decomposition principle

Decomposition principle

- Decomposition over examples : stochastic/incremental gradient descent
- Decomposition over categories : one-versus-rest strategy
- Decomposition over latent structure : atomic decomposition

Multi-class linear SVM with regular linear hinge loss

Multi-class linear SVM with regular linear hinge loss

Training data : $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \{0,1\}^k$

$$\min_{\mathbf{W} \in \mathbb{R}^{d \times k}} \quad \lambda \left\| \mathbf{w} \right\|_2^2 + \frac{1}{n} \sum_{i=1}^{n} \text{BinaryHingeLoss}_i$$

One-versus-rest reduction

- Turn original label $y_i \in \{0,1\}^k$ into binary label $\tilde{y}_i \in \{-1,+1\}$

$$\text{BinaryHingeLoss}_i = \max(0, 1 - \tilde{y}_i \mathbf{w}^T \mathbf{x}_i)$$

- Note : any loss could do, i.e. also the logistic loss

Multi-class linear SVM with regular linear hinge loss

Multi-class linear SVM with regular linear hinge loss

Training data : $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \{0,1\}^k$

$$\min_{\mathbf{w} \in \mathbb{R}^{d \times k}} \quad \sum_{\ell=1}^{k} \lambda_\ell \left\| \mathbf{w}_\ell \right\|_2^2 + \frac{1}{n} \sum_{\ell=1}^{k} \sum_{\substack{i=1 \\ y_i \equiv \text{class } \ell}}^{n} \text{BinaryHingeLoss}_i$$

Decomposition over categories

Leverage decomposable structure over categories

Multi-class linear SVM with regular linear hinge loss

Multi-class linear SVM with regular linear hinge loss

Training data : $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \{0,1\}^k$

$$\begin{cases} \min\limits_{\mathbf{w}_1 \in \mathbb{R}^d} & \lambda_1 \left\|\mathbf{w}_1\right\|_2^2 + \dfrac{1}{n} \sum\limits_{\substack{i=1 \\ y_i \equiv \text{class } 1}}^{n} \text{BinaryHingeLoss}_i \\ \ldots & \\ \min\limits_{\mathbf{w}_\ell \in \mathbb{R}^d} & \lambda_\ell \left\|\mathbf{w}_\ell\right\|_2^2 + \dfrac{1}{n} \sum\limits_{\substack{i=1 \\ y_i \equiv \text{class } \ell}}^{n} \text{BinaryHingeLoss}_i \\ \ldots & \\ \min\limits_{\mathbf{w}_k \in \mathbb{R}^d} & \lambda_k \left\|\mathbf{w}_k\right\|_2^2 + \dfrac{1}{n} \sum\limits_{\substack{i=1 \\ y_i \equiv \text{class } k}}^{n} \text{BinaryHingeLoss}_i \end{cases}$$

Multi-class through one-vs-rest

Multi-class through one-vs-rest

- Overall : simplest multi-class classification algorithm
- Computational strength : easy to optimize by decomposition over classes
- Statistical weakness : no universally consistent loss can be decomposable over classes (do we really care ? we'll see)

Multi-class through one-vs-rest

In practice

State-of-the-art performance using a balanced version of the binary loss, and learning the optimal imbalance β through cross-validation

$$\text{Empirical risk} = \frac{\beta}{n^+} \sum_{i \in \text{positive examples}} \text{BinaryHingeLoss}_i + \frac{1-\beta}{n^-} \sum_{i \in \text{negative examples}} \text{BinaryHingeLoss}_i$$

Multi-class with non-decomposable loss functions

Other multi-class loss functions

- Multinomial logistic loss
- Crammer & Singer multi-class loss

$$R_{\text{MUL}} = \frac{1}{n} \sum_{i=1}^{n} \left\{ \max_{y} (\Delta(\mathbf{y}_i, y) + \mathbf{w}_y^T \mathbf{x}_i) - \mathbf{w}_{\mathbf{y}_i}^T \mathbf{x}_i \right\}$$

The multi-class binary hinge loss is the only decomposable loss

Multi-class with non-decomposable loss functions

More sophisticated losses

Loss functions tailored to optimize a convex surrogate of top-k accuracy

$$\text{Accuracy}_{\text{top}-k} = \frac{\#\text{ images whose correct label lies in top-k scores}}{\text{Total number of images}}$$

- Ranking losses

$$R_{\text{RNK}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{y=1}^{k}\max_{y}\left(0, \Delta(\mathbf{y}_i, y) - (\mathbf{w}_{\mathbf{y}_i}^T - \mathbf{w}_y)^T\mathbf{x}_i\right)$$

- Weighted ranking losses, and other variations

Yet to prove themselves compared to one-vs-rest with binary loss on real-world datasets

Multi-class with non-decomposable loss functions

	Sampling	Update
R_{OVR}	Draw $(\mathbf{x}_i, y_i)$ from S	$\delta_i = 1$ if $L_{\mathsf{OVR}}(\mathbf{x}_i, y_i; \mathbf{w}) > 0, 0$ otherwise. $\mathbf{w}^{(t)} = (1 - \eta_t)\mathbf{w}^{(t-1)} + \eta_t \delta_i \mathbf{x}_i y_i$
R_{MUL}	Draw $(\mathbf{x}_i, y_i)$ from S	$\bar{y} = \arg\max_y \mathbf{D}(y_i, y) + \mathbf{w}'_y \mathbf{x}_i$ and $\delta_i = \begin{cases} 1 & \text{if } \bar{y} \neq y_i \\ 0 & \text{otherwise.} \end{cases}$ $\mathbf{w}_y^{(t)} = \begin{cases} \mathbf{w}_y^{(t-1)}(1 - \eta_t) + \delta_i \eta_t \mathbf{x}_i & \text{if } y = y_i \\ \mathbf{w}_y^{(t-1)}(1 - \eta_t) - \delta_i \eta_t \mathbf{x}_i & \text{if } y = \bar{y} \\ \mathbf{w}_y^{(t-1)}(1 - \eta_t) & \text{otherwise.} \end{cases}$
R_{RNK}	Draw $(\mathbf{x}_i, y_i)$ from S Draw $\bar{y} \neq y_i$ from $\mathcal{Y}$	$\delta_i = 1$ if $L_{\mathsf{tri}}(\mathbf{x}_i, y_i, \bar{y}; \mathbf{w}) > 0, 0$ otherwise. $\mathbf{w}_y^{(t)} = \begin{cases} \mathbf{w}_y^{(t-1)}(1 - \eta_t) + \delta_i \eta_t \mathbf{x}_i & \text{if } y = y_i \\ \mathbf{w}_y^{(t-1)}(1 - \eta_t) - \delta_i \eta_t \mathbf{x}_i & \text{if } y = \bar{y} \\ \mathbf{w}_y^{(t-1)}(1 - \eta_t) & \text{otherwise.} \end{cases}$

Experimental results

Datasets

	Total # of		Partition		
	images	classes	train	val	test
Fungus	88K	134	44K	5K	39K
Ungulate	183K	183	91.5K	5K	86.5K
Vehicle	226K	262	113K	5K	108K
ILSVRC10	1.4M	1,000	1.2M	50K	150K
ImageNet10K	9M	10,184	4.5M	50K	4.45M

Table : Datasets considered

Stochastic gradient descent is competitive with batch solvers

Stochastic gradient descent is competitive with batch solvers

Average training time (in CPU seconds) on 3 fine-grained datasets

(a) w-OVR SVM : LibSVM vs SGD

	LibSVM (batch) / SGD (online)		
	Fungus	Ungulate	Vehicle
10	12 / 7	31 / 18	107 / 39
25	95 / 16	175 / 36	835 / 119
50	441 / 38	909 / 67	3,223 / 271
100	1,346 / 71	3,677 / 133	11,679 / 314

(b) MUL SVM : SVM-light vs SGD

	SVM-light (batch) / SGD (online)		
	Fungus	Ungulate	Vehicle
10	45 / 36	324 / 81	557 / 209
25	99 / 72	441 / 198	723 / 369
50	198 / 261	855 / 420	1,265 / 747
100	972 / 522	1,674 / 765	3,752 / 1,503

Superiority of one-vs-rest with *weighted* binary loss

Superiority of one-vs-rest with *weighted* binary loss over unweighted one

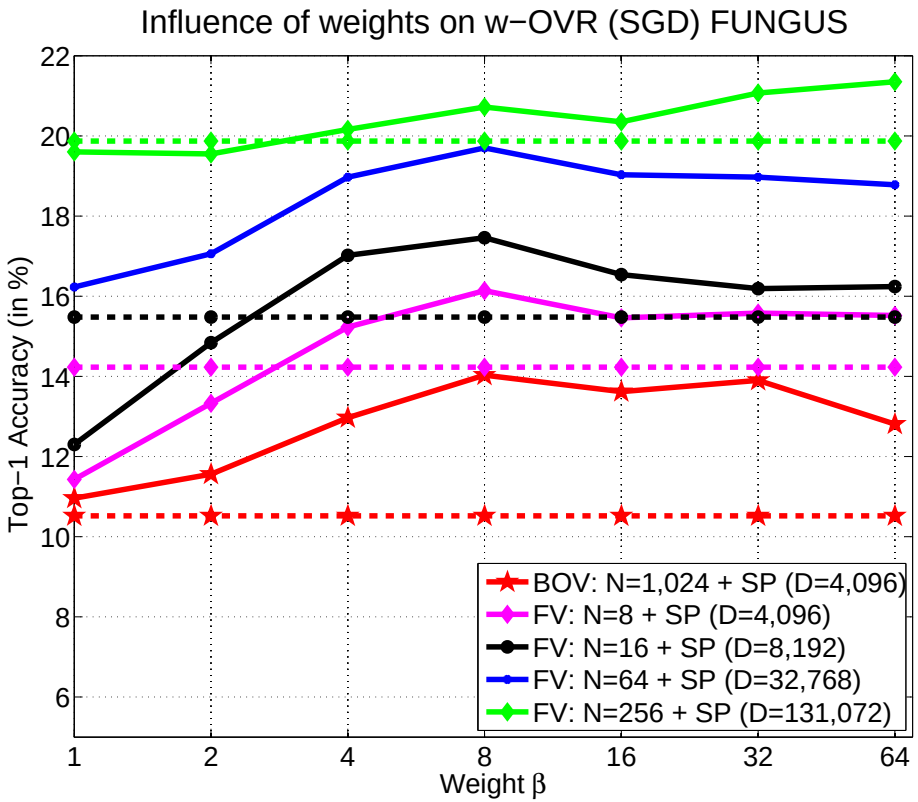
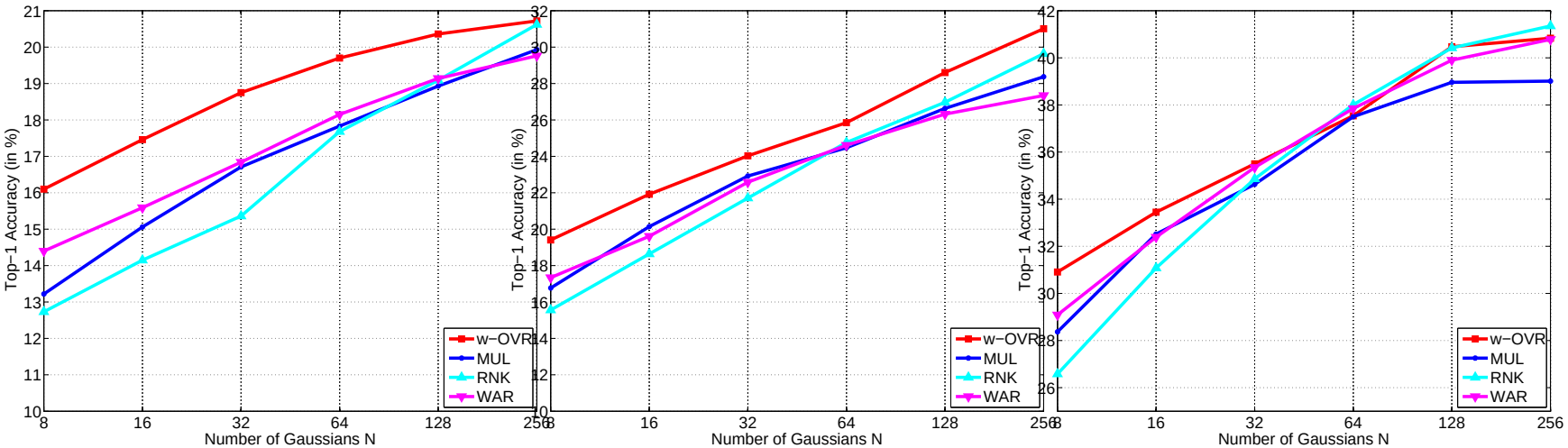


Figure : Influence of data rebalancing in weighted one-vs-rest (w-OVR) vs unweighted one-vs-rest (u-OVR) on Fungus (134 classes).

One-vs-rest with binary hinge-loss works fine for expressive features

One-vs-rest with binary hinge-loss works fine for expressive features



(a) Fungus (b) Ungulate (c) Vehicle

Figure : Comparison of Top-1 Accuracy between the w-OVR, MUL, RNK and WAR SVMs as a function of the number of Gaussians used to compute the FV (*i.e.* as a function of the FV dimensionality). No spatial pyramids were used to speed-up these experiments.

Beyond one-vs-rest strategies

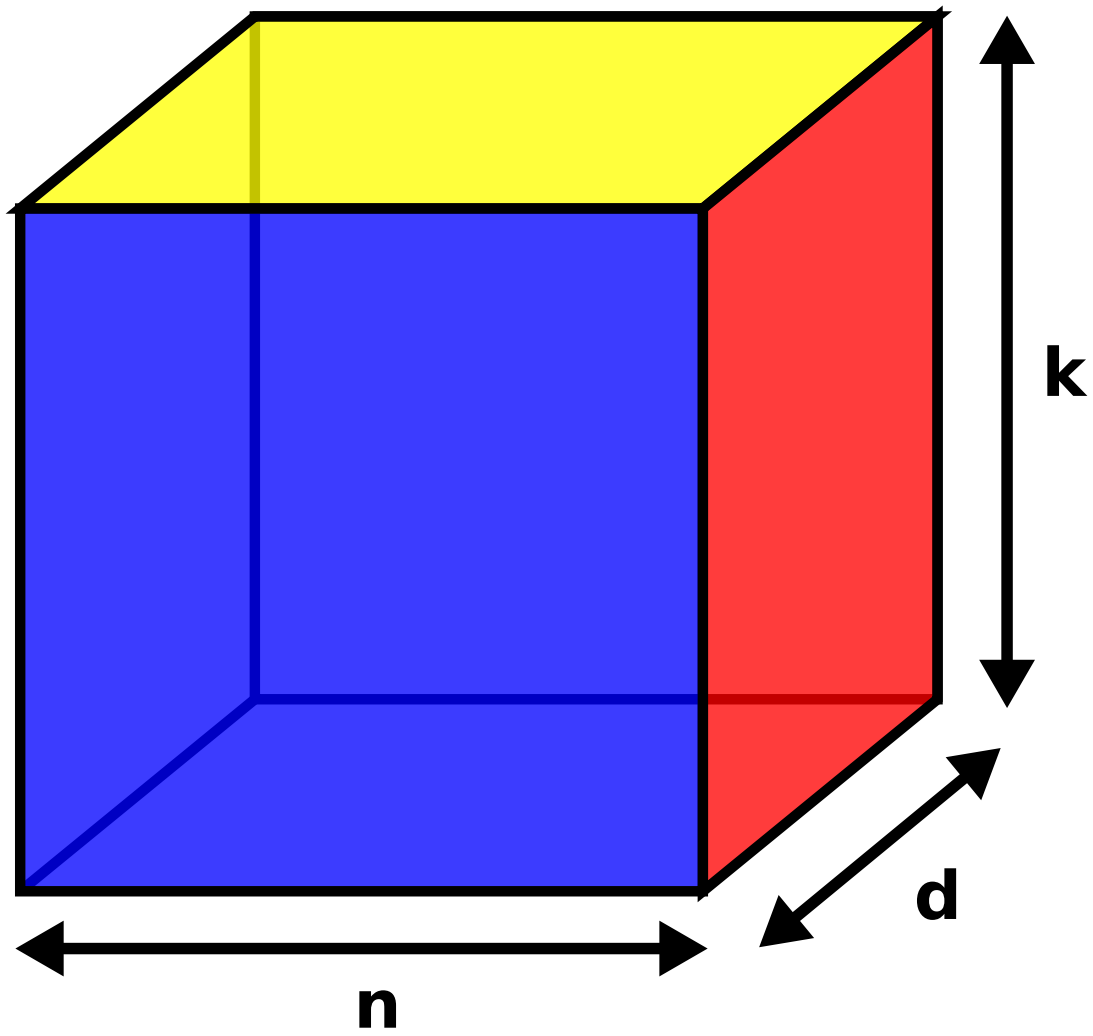
Large-scale learning

Training data : $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\mathbb{R}^d \times \mathcal{Y} = \{0,1\}^k$

$$\underset{\mathbf{W}\in\mathbb{R}^{d\times k}}{\text{Minimize}} \quad \lambda\Omega(\mathbf{W}) + \frac{1}{n}\sum_{i=1}^{n} \text{Loss}_i$$

- Discover latent structure of the classes
- And keep scalability and efficiency of one-versus-rest strategies

Machine learning cuboid



Decomposition principle

Decomposition principle

- Decomposition over examples : stochastic/incremental gradient descent
- Decomposition over latent structure : atomic decomposition

Learning with atom. penalty

Learning with low-rank regularization penalty

Training data : $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\mathbb{R}^d \times \mathcal{Y} = \{0,1\}^k$

$$\underset{\mathbf{W}\in\mathbb{R}^{d\times k}}{\text{Minimize}} \quad \lambda \text{Rank}(\mathbf{W}) + \frac{1}{n}\sum_{i=1}^{n} \text{Loss}_i \ ?$$

- Embedding motivation : classes may embedded in a low-dimensional subspace of the feature space
- Computational motivation : algorithm scales with the number of latent classes r, assuming that $r \ll k$
- Extension of Reduced-Rank Regression (see e.g. Velu, Reinsel, 1998) $\rightarrow$ non-smooth, non-convex optimization problem

Learning with atom. penalty

Learning with low-rank regularization penalty

Training data : $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \mathbb{R}^d \times \mathcal{Y} = \{0, 1\}^k$

$$\underset{\mathbf{W} \in \mathbb{R}^{d \times k}}{\text{Minimize}} \quad \underbrace{\lambda \|\sigma(\mathbf{W})\|_1 + \frac{1}{n} \sum_{i=1}^{n} \text{Loss}_i}_{\text{convex}}$$

- Embedding motivation : classes may embedded in a low-dimensional subspace of the feature space
- Computational motivation : algorithm scales with the number of latent classes r, assuming that $r \ll k$
- Extension of Reduced-Rank Regression (see e.g. Velu, Reinsel, 1998)
  $\rightarrow$ non-smooth, non-convex optimization problem

Learning with atom. penalty

Learning with low-rank regularization penalty

Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \{1, \ldots, k\}$ be labelled training images

$$\underset{\mathbf{W} \in \mathbb{R}^{d \times k}}{\text{Minimize}} \quad \underbrace{\lambda \|\sigma(\mathbf{W})\|_1 + \frac{1}{n} \sum_{i=1}^{n} \text{Loss}_i}_{\text{convex}}$$

- Tight convex relaxation (Amit et al., 2007 ; Argyriou et al., 2007)
- Enforces a low-rank structure of $\mathbf{W}$ (sparsity of spectrum $\sigma(\mathbf{W})$)
- Convex, but non-differentiable

Learning with atom. penalty

Learning with low-rank regularization penalty

Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \{1, \ldots, k\}$ be labelled training images

$$\underset{\mathbf{W} \in \mathbb{R}^{d \times k}}{\text{Minimize}} \quad \underbrace{\lambda \left\| \sigma(\mathbf{W}) \right\|_1}_{non-smooth} + \underbrace{R_n(\mathbf{W})}_{smooth}$$

where $R_n(\mathbf{W})$ is the empirical risk with the multinomial logistic loss

$$R_n(\mathbf{W}) = \frac{1}{n} \sum_{i=1}^{n} \log \left(1 + \sum_{\ell \in \mathcal{Y} \setminus \{y_i\}} \exp \left\{ \mathbf{w}_\ell^T \mathbf{x}_i - \mathbf{w}_{y_i}^T \mathbf{x}_i \right\} \right)$$

Learning with atom. penalty

Learning with low-rank regularization penalty

Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \{1, \ldots, k\}$ be labelled training images

$$\underset{\mathbf{W} \in \mathbb{R}^{d \times k}}{\text{Minimize}} \quad \underbrace{\lambda \left\| \sigma(\mathbf{W}) \right\|_1}_{decomposable?} + \underbrace{R_n(\mathbf{W})}_{smooth}$$

where $R_n(\mathbf{W})$ is the empirical risk with the multinomial logistic loss

$$R_n(\mathbf{W}) = \frac{1}{n} \sum_{i=1}^{n} \log \left(1 + \sum_{\ell \in \mathcal{Y} \setminus \{y_i\}} \exp \left\{ \mathbf{w}_\ell^T \mathbf{x}_i - \mathbf{w}_{y_i}^T \mathbf{x}_i \right\} \right)$$

Stochastic atom descent

We want an efficient and scalable algorithm
Let's get inspiration from ℓ_1 case...

Atom-descent algorithms

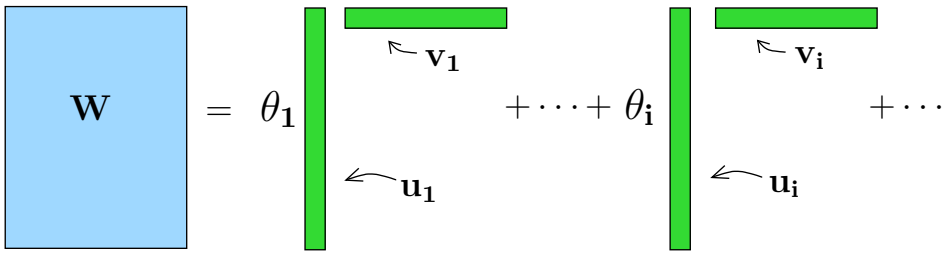
- Leverage a decomposable structure of regularization : atomic decomposition
- Perform a stochastic version of coordinate descent on this representation
- Efficient and scalable algorithms

Atom-descent for trace-norm regularization

- Leverage a non-flat decomposable structure, in contrast to one-vs-rest
- Learn a latent embedding of the classes

Lifting to an infinite-dimensional space

The trace-norm is the smallest ℓ_1-norm of the weight vector associated with an atomic decomposition onto rank-one subspaces



$$\|\sigma(\mathbf{W})\|_1 = \min_{\theta} \left\{ \|\theta\|_1 , \quad \exists N,\ \mathbf{M}_i \in \mathcal{M},\ \text{with } \mathbf{W} = \sum_{i=1}^{N} \theta_i \mathbf{M}_i \right\}$$

$$\mathcal{M} = \{\mathbf{u}\mathbf{v}^T \mid \mathbf{u} \in \mathbb{R}^d,\ \mathbf{v} \in \mathbb{R}^k,\ \|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1\}$$

Lifted objective

Lifting

- **Original objective** :

$$J(\mathbf{W}) := \lambda \left\| \sigma(\mathbf{W}) \right\|_1 + R_n(\mathbf{W})$$

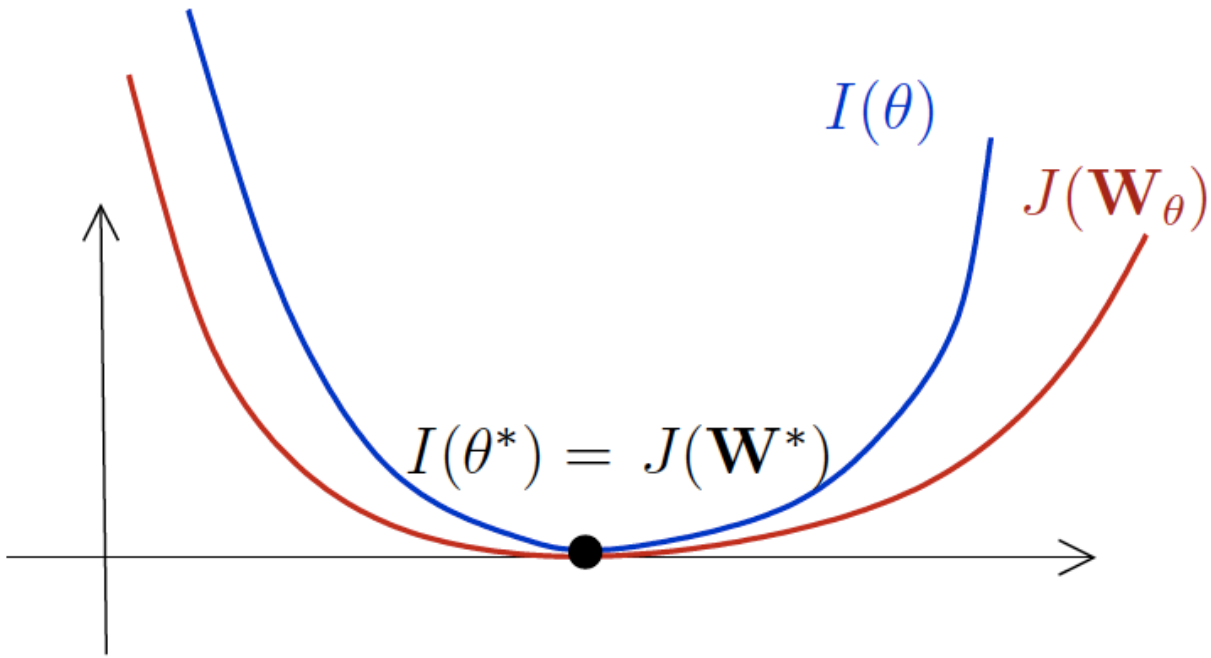
- **Lifted objective** :

$$I(\boldsymbol{\theta}) := \lambda \sum_{j \in \mathsf{supp}(\theta)} \theta_j + R_n(\mathbf{W}_{\boldsymbol{\theta}})$$

Equivalence

Equivalence

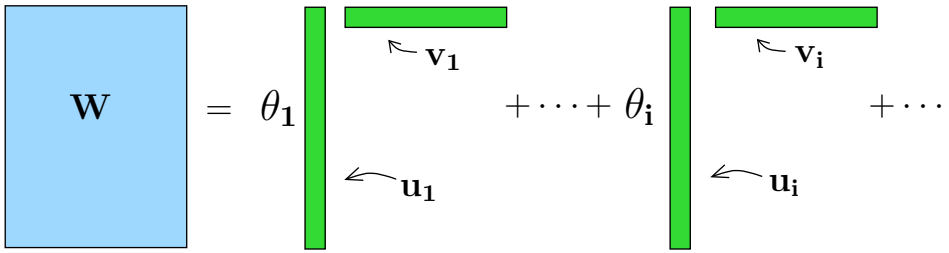
Assume that the loss function $L(y, \cdot)$ is convex and smooth.
Then the two problems are equivalent



Stochastic atom-descent descent : high-level idea

Sketch

- At each iteration, pick a random mini-batch B_t, then pick the rank-1 subspace yielding the steepest descent, and perform descent along that direction
- Periodically perform second-order minimization on current subspace



Stochastic atom descent

Algorithm

- **Initialize** : $\boldsymbol{\theta} = 0$
- **Iterate** : pick a random mini-batch B_t, find coordinate θ_i of steepest descent

$$
\begin{aligned}
i^\star &= \operatorname*{Arg\,max}_i \frac{\partial I_{B_t}(\boldsymbol{\theta})}{\partial \theta_i} \\
&= \operatorname*{Arg\,max}_i \langle \mathbf{u}_i \mathbf{v}_i^T, -\nabla R_{B_t}(\mathbf{W}_{\boldsymbol{\theta}}) \rangle \\
&= \operatorname*{Arg\,max}_{\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1} \mathbf{u}^T \left(-\nabla R_{B_t}(\mathbf{W}_{\boldsymbol{\theta}})\right) \mathbf{v} \ .
\end{aligned}
$$

then descend along $\theta_{i^\star}$

Periodically minimize $I(\boldsymbol{\theta})$ over $\operatorname{supp}(\boldsymbol{\theta})$ up to optimality.

Stochastic atom descent

Algorithm

- **Initialize** : $\boldsymbol{\theta} = 0$
- **Iterate** : pick a random mini-batch B_t, find coordinate θ_i of steepest descent

  Finding $i^\star$ corresponds to
  finding top singular vectors $\mathbf{u}_1$ and $\mathbf{v}_1$ of $-\nabla R_{B_t}(\mathbf{W}_{\boldsymbol{\theta}})$
  Runtime : $O(dk)$ (few Power/Lanczos iter.)
  Descend along $\theta_{i^\star}$

  Periodically minimize $I(\boldsymbol{\theta})$ over supp$(\boldsymbol{\theta})$ up to optimality.

Stochastic atom descent

Algorithm

- **Initialize** : $\boldsymbol{\theta} = 0$
- **Iterate** : pick a random mini-batch B_t, find coordinate θ_i of steepest descent

  Find $i^\star$ corresponds to find top singular vectors $\mathbf{u}_1$ and $\mathbf{v}_1$ of $R_{B_t}(\mathbf{W}_{\boldsymbol{\theta}})$
  then descend along $\theta_{i^\star}$

  Periodically minimize $I(\boldsymbol{\theta})$ over supp$(\boldsymbol{\theta})$ up to optimality.
  (quasi-)Newton method with box constraints

Experimental results

Benchmark

- ImageNet dataset
- Subset of classes "Vehicles262", "Fungus134', and "Ungulate183"

Fisher vector image representation (Perronnin & Dance, 2007)

1. Extracted SIFT and local color descriptors reduced to 128 D
2. Train a Gaussian mixture model of 16 centroids $\rightarrow$ Fisher vectors of dim. 4096
3. Explicit embedding (Perronnin et al., 2010 ; Vedaldi & Zisserman, 2010)

Experimental results

Computations on each mini-batch

1. parallelized objective evaluation and gradient evaluation
2. efficient matrix computations for high-dimensional features

Efficient strategy : training with compression and testing without compression

1. product quantization of visual descriptors (Jegou et al., 2011)
2. during training, all matrix computations on features are performed in the compressed domain
3. for testing, all matrix computations are performed on uncompressed features
4. Note : compared to train/test without compression, average loss of performance only of 0.9% on a subset of Vehicles with 10 categories.

Experimental results

Classification accuracy comparison

- Classification accuracy : top-k accuracy, i.e.

$$\text{Accuracy}_{\text{top}-k} = \frac{\#\text{ images whose correct label lies in top-k scores}}{\text{Total number of images}}$$

- Competitors : our approach (TR-Multiclass) and k independently trained one-vs-rest classifiers (OVR)

Experimental results

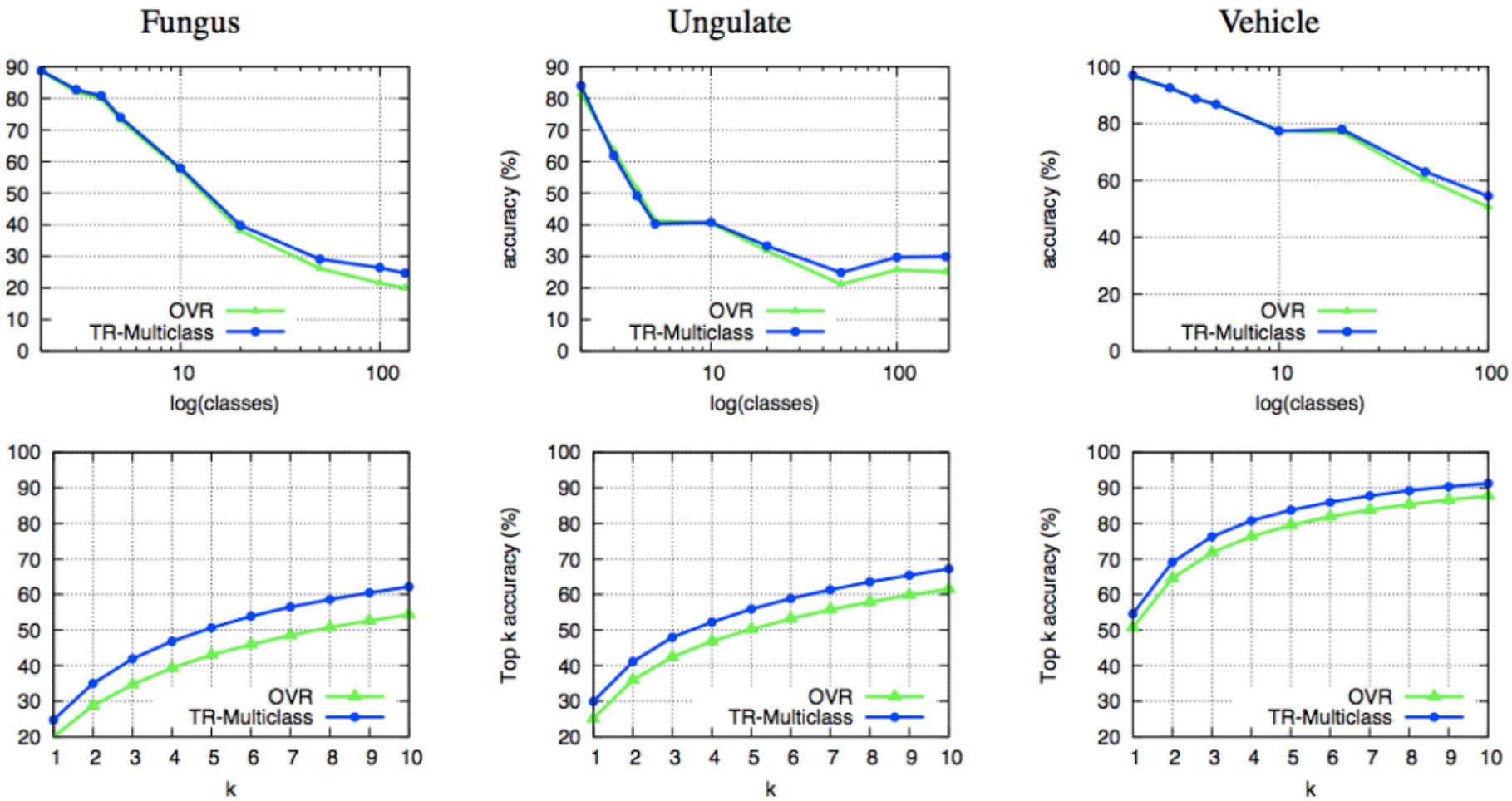
Cheatsheet

$$\text{OVR} \quad \underset{\mathbf{W}\in\mathbb{R}^{d\times k}}{\text{Minimize}} \quad \sum_{\ell=1}^{k} \lambda_\ell \left\|\mathbf{w}_\ell\right\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} \text{BinaryHingeLoss}_i$$

$$\text{L2-Multiclass} \quad \underset{\mathbf{W}\in\mathbb{R}^{d\times k}}{\text{Minimize}} \quad \lambda\|\mathbf{W}\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} \text{MultinomialLogisticLoss}_i$$

$$\text{TR-Multiclass} \quad \underset{\mathbf{W}\in\mathbb{R}^{d\times k}}{\text{Minimize}} \quad \lambda\|\sigma(\mathbf{W})\|_1 + \frac{1}{n}\sum_{i=1}^{n} \text{MultinomialLogisticLoss}_i$$

Experimental results



A posteriori low-dimensional embedding

Conclusion

Large-scale learning through decomposition

- Stochastic gradient descent is a decomposition over examples
- One-vs-rest is a decomposition over categories
- Stochastic atom descent is a decomposition over latent structure

Conclusion

Be your own cook

- Mix these decompositions and come up with your own algorithms for new problems
- Implement your own codes and master the algorithm before using an off-the-shelf implementation

Public code

- Download it and start running your own large-scale experiments
- Joust SGD : `lear.inrialpes.fr/software`

References

Recent references

- *Good practices in large-scale learning for image classification*, Z. Akata, F. Perronnin, Z. Harchaoui, C. Schmid, TPAMI 2013
- *Large-scale classification with trace-norm regularization*, Z. Harchaoui, M. Douze, M. Paulin, J. Malick, CVPR 2012
- *Learning with matrix gauge regularizers*, M. Dudik, Z. Harchaoui, J. Malick, NIPS Opt. 2011
- *Image Classification with the Fisher Vector : Theory and Practice*, J. Sanchez, F. Perronnin, T. Mensink, J. Verbeek, IJCV 2013
- *Product quantization for nearest-neighbor search*, H. Jegou, M. Douze, C. Schmid, IEEE Trans. PAMI, 2011
- *Efficient additive kernels via explicit feature maps*, A. Vedaldi, A. Zisserman, CVPR 2010