Convex and non-convex worlds in machine learning

Anna Choromanska

Courant Institute of Mathematical Sciences
New York University

Convex and non-convex worlds

Machine learning and optimization - many machine learning problems are formulated as minimization of some loss function on a training set of examples. Loss functions expresses the discrepancy between the predictions of the model being trained and the actual problem instances. Optimization algorithms can then minimize this loss. (*Wikipedia*)

Convex and non-convex worlds

Machine learning and optimization - many machine learning problems are formulated as minimization of some loss function on a training set of examples. Loss functions expresses the discrepancy between the predictions of the model being trained and the actual problem instances. Optimization algorithms can then minimize this loss. (*Wikipedia*)

Convex world

local min = global min
strictly convex: unique min
efficient solvers
strong theoretical guarantees

Convex and non-convex worlds

Machine learning and optimization - many machine learning problems are formulated as minimization of some loss function on a training set of examples. Loss functions expresses the discrepancy between the predictions of the model being trained and the actual problem instances. Optimization algorithms can then minimize this loss. (*Wikipedia*)

Convex world

local min = global min
strictly convex: unique min
efficient solvers
strong theoretical guarantees

Non-convex world

multiple local min $\neq$ global min
many solvers come from convex world
weak theoretical guarantees if any

Layout of the talk

Optimization solvers: generic optimization vs bound majorization

partition function-based objectives

Design (convex) solver: quadratic bound majorization

[J**C**12, **C**KJ12, A**C**JK14]

Layout of the talk

Optimization solvers: generic optimization vs bound majorization
partition function-based objectives
Design (convex) solver: quadratic bound majorization
[J**C**12, **C**KJ12, A**C**JK14]

Challenging problems: multi-class classification
Design objective: statistical and computational constraints
online multi-class partition trees for logarithmic time predictions
[**C**L14, **C**AL13, **C**CB15, **C**CJM15, **C**CJM13, **C**JKMM13, B**C**CL15, **C**M12]

Layout of the talk

Optimization solvers: generic optimization vs bound majorization
partition function-based objectives
Design (convex) solver: quadratic bound majorization
[J**C**12, **C**KJ12, A**C**JK14]

Challenging problems: multi-class classification
Design objective: statistical and computational constraints
online multi-class partition trees for logarithmic time predictions
[**C**L14, **C**AL13, **C**CB15, **C**CJM15, **C**CJM13, **C**JKMM13, B**C**CL15, **C**M12]

Non-convex problems: deep learning
highly non-convex objective
Build understanding: new theoretical results
[**C**HMCL15, **C**LB15, Z**C**L15]

How to build good efficient convex solver?

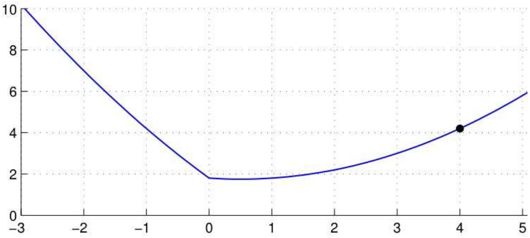
Generic optimization techniques and majorization methods

- Batch
  - steepest descent
  - conjugate gradient
  - Newton
  - (L)BFGS [B70]
- Stochastic
  - SGD [RB51]
  - ASGD [PJ92]
  - SAG [LRSB12]
  - SDCA [SSZ13]
  - SVRG [JZ13]
- Semi-stochastic
  - hybrid deterministic-stochastic methods [FS12]
- Majorization methods
  - MISO [M13]
  - iterative scaling [DR72]
  - EM [DLR77]
  - Quadratic lower bound principle [BL88]

Bound majorization

If cost function $\theta^* = \arg\min_\theta C(\theta)$ has no closed form solution
Majorization uses a surrogate Q with closed form solution
Monotonically improves from initial θ_0

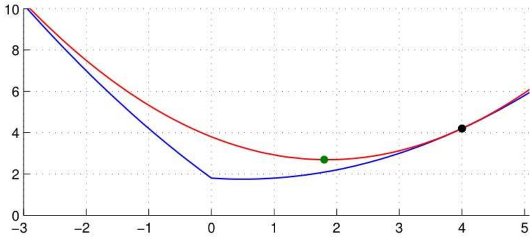
- Find bound $Q(\theta, \theta_i) \geq C(\theta)$ where $Q(\theta_i, \theta_i) = C(\theta_i)$
- Update $\theta_{i+1} = \arg\min_\theta Q(\theta, \theta_i)$
- Repeat until converged



Bound majorization

If cost function $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} C(\boldsymbol{\theta})$ has no closed form solution
Majorization uses a surrogate Q with closed form solution
Monotonically improves from initial $\boldsymbol{\theta}_0$

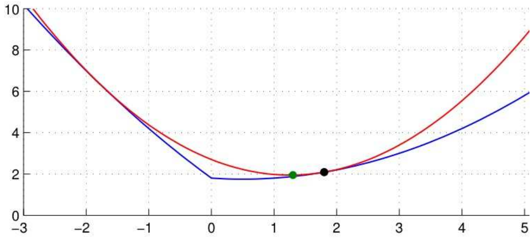
- Find bound $Q(\boldsymbol{\theta}, \boldsymbol{\theta}_i) \geq C(\boldsymbol{\theta})$ where $Q(\boldsymbol{\theta}_i, \boldsymbol{\theta}_i) = C(\boldsymbol{\theta}_i)$
- Update $\boldsymbol{\theta}_{i+1} = \arg\min_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}_i)$
- Repeat until converged



Bound majorization

If cost function $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} C(\boldsymbol{\theta})$ has no closed form solution
Majorization uses a surrogate Q with closed form solution
Monotonically improves from initial $\boldsymbol{\theta}_0$

- Find bound $Q(\boldsymbol{\theta}, \boldsymbol{\theta}_i) \geq C(\boldsymbol{\theta})$ where $Q(\boldsymbol{\theta}_i, \boldsymbol{\theta}_i) = C(\boldsymbol{\theta}_i)$
- Update $\boldsymbol{\theta}_{i+1} = \arg\min_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}_i)$
- Repeat until converged



Generic optimization techniques vs majorization methods

Majorization methods preferred until [W03, AG07].

...Why? Slower than other optimizers, because of loose & complicated bounds.

Let's fix this!!!

Partition Function

Log-linear model partition functions

$$Z(\boldsymbol{\theta}) = \sum_{y} h(y) \exp(\boldsymbol{\theta}^{\top} \mathbf{f}(y))$$

Partition function ensures that $p(y|\boldsymbol{\theta})$ normalizes.

It is a central quantity to optimize in

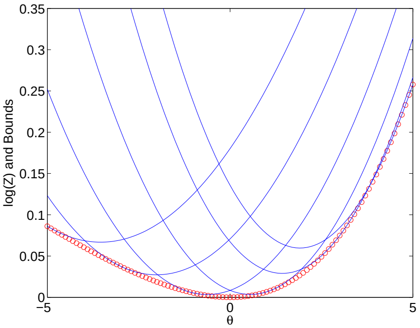
- maximum likelihood and e-family [P36]
- maximum entropy [J57]
- conditional random fields [LMP01]
- log-linear models [DR72]
- graphical models, HMMs [JGJS99].

Problem: it's ugly to minimize, we much prefer quadratics

Partition Function Bound

The bound $\ln Z(\boldsymbol{\theta}) \leq \ln z + \frac{1}{2}(\boldsymbol{\theta}-\tilde{\boldsymbol{\theta}})^\top \boldsymbol{\Sigma}(\boldsymbol{\theta}-\tilde{\boldsymbol{\theta}}) + (\boldsymbol{\theta}-\tilde{\boldsymbol{\theta}})^\top \boldsymbol{\mu}$ is tight at $\tilde{\boldsymbol{\theta}}$ and holds for parameters given by

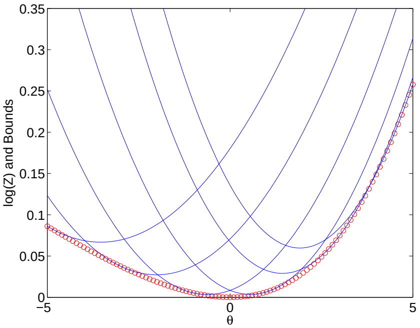
Input $\tilde{\boldsymbol{\theta}}, \mathbf{f}(y), h(y)\ \forall y \in \Omega$
Init $z \to 0^+, \boldsymbol{\mu} = \mathbf{0}, \boldsymbol{\Sigma} = z\mathbf{I}$ For each $y \in \Omega$ { $\alpha = h(y)\exp(\tilde{\boldsymbol{\theta}}^\top \mathbf{f}(y))$ $\mathbf{r} = \mathbf{f}(y) - \boldsymbol{\mu}$ $\boldsymbol{\Sigma} \mathrel{+}= \frac{\tanh(\frac{1}{2}\ln(\alpha/z))}{2\ln(\alpha/z)}\mathbf{r}\mathbf{r}^\top$ $\boldsymbol{\mu} \mathrel{+}= \frac{\alpha}{z+\alpha}\mathbf{r}$ $z \mathrel{+}= \alpha$ }
Output $z, \boldsymbol{\mu}, \boldsymbol{\Sigma}$



Partition Function Bound

The bound $\ln Z(\boldsymbol{\theta}) \;\leq\; \ln z + \frac{1}{2}(\boldsymbol{\theta}-\tilde{\boldsymbol{\theta}})^\top \boldsymbol{\Sigma}(\boldsymbol{\theta}-\tilde{\boldsymbol{\theta}}) + (\boldsymbol{\theta}-\tilde{\boldsymbol{\theta}})^\top \boldsymbol{\mu}$ is tight at $\tilde{\boldsymbol{\theta}}$ and holds for parameters given by

Input $\tilde{\boldsymbol{\theta}}, \mathbf{f}(y), h(y)\ \forall y \in \Omega$
Init $z \to 0^+, \boldsymbol{\mu} = \mathbf{0}, \boldsymbol{\Sigma} = z\mathbf{I}$ For each $y \in \Omega$ { $\alpha \quad = h(y)\exp(\tilde{\boldsymbol{\theta}}^\top \mathbf{f}(y))$ $\mathbf{r} \quad = \mathbf{f}(y) - \boldsymbol{\mu}$ $\boldsymbol{\Sigma} \mathrel{+}= \frac{\tanh(\frac{1}{2}\ln(\alpha/z))}{2\ln(\alpha/z)}\mathbf{r}\mathbf{r}^\top$ $\boldsymbol{\mu} \mathrel{+}= \frac{\alpha}{z+\alpha}\mathbf{r}$ $z \mathrel{+}= \alpha$ }
Output $z, \boldsymbol{\mu}, \boldsymbol{\Sigma}$



$\mathcal{O}(nd^2)$ and update via $\boldsymbol{\theta} \leftarrow \tilde{\boldsymbol{\theta}} - \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ in $\mathcal{O}(d^3)$.

Conditional Random Fields (CRFs)

- Trained on *iid* data $\{(x_1, y_1), \ldots, (x_t, y_t)\}$
- Each CRF is a log-linear model

$$p(y|x_j, \boldsymbol{\theta}) = \frac{1}{Z_{x_j}(\boldsymbol{\theta})} h_{x_j}(y) \exp(\boldsymbol{\theta}^\top \mathbf{f}_{x_j}(y))$$

- Regularized maximum likelihood objective is

$$J(\boldsymbol{\theta}) = \sum_{j=1}^{t} \log \frac{h_{x_j}(y_j)}{Z_{x_j}(\boldsymbol{\theta})} + \boldsymbol{\theta}^\top \mathbf{f}_{x_j}(y_j) - \frac{t\lambda}{2} \|\boldsymbol{\theta}\|^2$$

Maximum Likelihood Algorithm for CRFs

While not converged
For $j = 1, \ldots, t$
Compute bound for $\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j$ from $h_{x_j}, \mathbf{f}_{x_j}, \tilde{\boldsymbol{\theta}}$
Set $\tilde{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Lambda}} \sum_j \frac{1}{2}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})^\top(\boldsymbol{\Sigma}_j + \lambda \mathbf{I})(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}) + \sum_j \boldsymbol{\theta}^\top(\boldsymbol{\mu}_j - \mathbf{f}_{x_j}(y_j) + \lambda \tilde{\boldsymbol{\theta}})$

Maximum Likelihood Algorithm for CRFs

While not converged
For $j = 1, \ldots, t$
Compute bound for $\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j$ from $h_{x_j}, \mathbf{f}_{x_j}, \tilde{\boldsymbol{\theta}}$
Set $\tilde{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Lambda}} \sum_j \frac{1}{2}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})^\top(\boldsymbol{\Sigma}_j + \lambda\mathbf{I})(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})$
$+ \sum_j \boldsymbol{\theta}^\top(\boldsymbol{\mu}_j - \mathbf{f}_{x_j}(y_j) + \lambda\tilde{\boldsymbol{\theta}})$

Theorem

The algorithm outputs a $\hat{\boldsymbol{\theta}}$ such that

$$J(\hat{\boldsymbol{\theta}}) - J(\boldsymbol{\theta}_0) \geq (1-\epsilon) \max_{\boldsymbol{\theta}\in\boldsymbol{\Lambda}} (J(\boldsymbol{\theta}) - J(\boldsymbol{\theta}_0))$$

within $\left\lceil \log\left(\frac{1}{\epsilon}\right) / \log\left(1 + \frac{\lambda \log n}{2r^2 n}\right)\right\rceil$ *steps.*

Experiments - Markov CRFs

- Bound admits low-rank version ($\mathcal{O}(tnd)$)
  - As in LBFGS, use rank-k storage $\mathbf{\Sigma} = \mathbf{VSV}^\top + \mathbf{D}$
  - Absorb residual into diagonal $\mathbf{D} \Rightarrow$ Low-rank is still a bound

Experiments - Markov CRFs

- Bound admits low-rank version ($\mathcal{O}(tnd)$)
  - As in LBFGS, use rank-k storage $\mathbf{\Sigma} = \mathbf{VSV}^\top + \mathbf{D}$
  - Absorb residual into diagonal $\mathbf{D} \Rightarrow$ Low-rank is still a bound
- Graphical models, e.g. Markov CRFs
  - Build junction tree and run a *Collect* algorithm
  - Only needs $\mathcal{O}(td^2 \sum_c |Y_c|)$ rather than $\mathcal{O}(td^2 n)$

Experiments - Markov CRFs

- Bound admits low-rank version ($\mathcal{O}(tnd)$)
  - As in LBFGS, use rank-k storage $\mathbf{\Sigma} = \mathbf{VSV}^\top + \mathbf{D}$
  - Absorb residual into diagonal $\mathbf{D} \Rightarrow$ Low-rank is still a bound
- Graphical models, e.g. Markov CRFs
  - Build junction tree and run a *Collect* algorithm
  - Only needs $\mathcal{O}(td^2 \sum_c |Y_c|)$ rather than $\mathcal{O}(td^2 n)$

CONLL dataset

Algorithm	time	passes
L-BFGS	1.00t	17
CG	3.47t	23
Bound	**0.64t**	**4**

Bound

Algorithm	time	passes
L-BFGS	1.00t	22
CG	5.94t	27
IIS	$\geq$ 6.35t	$\geq$ 150

[W03]

Experiments - Markov CRFs

- Bound admits low-rank version ($\mathcal{O}(tnd)$)
  - As in LBFGS, use rank-k storage $\mathbf{\Sigma} = \mathbf{VSV}^\top + \mathbf{D}$
  - Absorb residual into diagonal $\mathbf{D} \Rightarrow$ Low-rank is still a bound
- Graphical models, e.g. Markov CRFs
  - Build junction tree and run a *Collect* algorithm
  - Only needs $\mathcal{O}(td^2 \sum_c |Y_c|)$ rather than $\mathcal{O}(td^2 n)$

CONLL dataset

Algorithm	time	passes
L-BFGS	1.00t	17
CG	3.47t	23
Bound	**0.64t**	**4**

Bound

Algorithm	time	passes
L-BFGS	1.00t	22
CG	5.94t	27
IIS	$\geq$ 6.35t	$\geq$ 150

[W03]

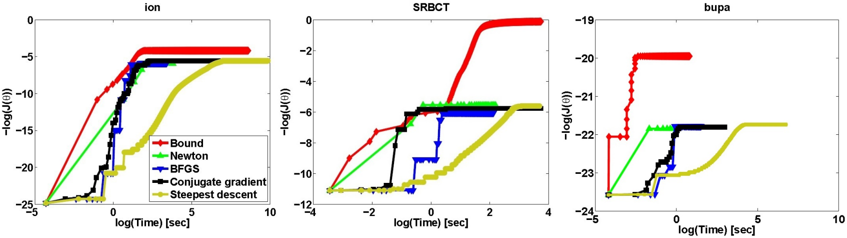
- Latent models
  - Objective function is non-concave: ratio of partition functions
  - Apply Jensen to numerator and our bound to denominator
  - Often better solution than BFGS, Newton, CG, SD, ...

Experiments - Latent models

- Bounding also simplifies mixture models with hidden variables (mixtures of Gaussians, HMMs, latent graphical models)
- Assume exponential family mixture components (Gaussian, multinomial, Poisson, Laplace)
- Latent CRF or log-linear model [Quattoni et al. '07]

$$L(\boldsymbol{\theta}) \;=\; \prod_{j=1}^{t} \frac{\sum_m \exp\left(\boldsymbol{\theta}^\top \mathbf{f}_{j,y_j,m}\right)}{\sum_{y,m} \exp\left(\boldsymbol{\theta}^\top \mathbf{f}_{j,y,m}\right)} \geq Q(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}})$$

- Apply Jensen to numerator and our bound to denominator



Experiments - (Semi-)Stochastic Bound Majorization

- Computing the bound is $\mathcal{O}(t) \rightarrow$ intractable for large t
- Semi-stochastic: compute bound on data mini-batches
  - convergence to a stationary point under weak assumptions (in particular convexity is not required)
  - **linear** convergence rate for logistic regression problem when batch size grows sufficiently fast

Experiments - (Semi-)Stochastic Bound Majorization

- Computing the bound is $\mathcal{O}(t) \rightarrow$ intractable for large t
- Semi-stochastic: compute bound on data mini-batches
  - convergence to a stationary point under weak assumptions (in particular convexity is not required)
  - **linear** convergence rate for logistic regression problem when batch size grows sufficiently fast

Theorem

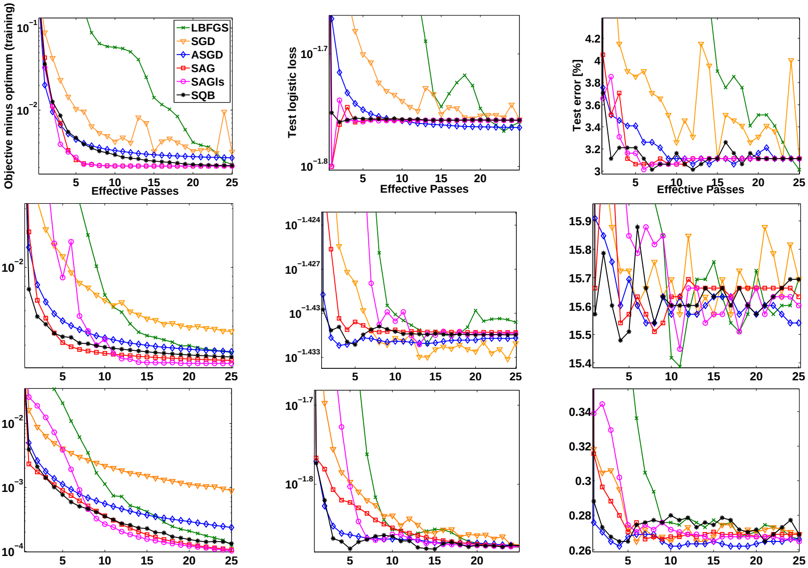
For each iteration we have (for any $\epsilon > 0$)

$$J(\boldsymbol{\theta}_k) - J(\boldsymbol{\theta}^*) \leq \left(1 - \frac{\rho}{L}\right)^k [J(\boldsymbol{\theta}_0) - J(\boldsymbol{\theta}^*)] + \mathcal{O}(C_k) \ ,$$

with $C_k = \max\{B_k, (1 - \frac{\rho}{L} + \epsilon)^k\}$ *and* $B_k = \|\nabla J(\boldsymbol{\theta})_{|\boldsymbol{\theta}=\boldsymbol{\theta}_k} - \mathbf{g}_{\mathcal{T}}^k\|^2$, *where* $\mathcal{T}$ *is the mini-batch.*

Experiments - (Semi-)Stochastic Bound Majorization

Datasets: rcv1, adult, protein



How to design good objective function?

eXtreme multi-class classification problem

Problem setting

- classification with large number of classes
- data is accessed online

Goal:

- good predictor with **logarithmic** training **and** testing **time**
- reduction to tree-structured binary classification
- top-down approach for class partitioning allowing gradient descent style optimization

What was already done...

- Intractable
  - one-against-all [RK04]
  - variants of ECOC [DB95], e.g. PECOC [LB05]
  - clustering-based approaches [BWG10, WMY13]
- Choice of partition not addressed
  - Filter Tree and error-correcting tournaments [BLR09]
- Choice of partition addressed, but dedicated to conditional probability estimation
  - conditional probability tree [BLLSS09]
- Splitting criteria not well-suited to large class setting
  - decision trees [KM95]
- ...

How do you learn the structure?

- Not all partitions are equally difficult, e.g.
  if you do $\{1,7\}$ vs $\{3,8\}$, the next problem is hard;
  if you do $\{1,8\}$ vs $\{3,7\}$, the next problem is easy;
  if you do $\{1,3\}$ vs $\{7,8\}$, the next problem is easy.

How do you learn the structure?

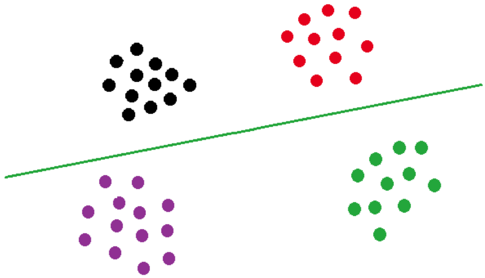
- Not all partitions are equally difficult, e.g.
  if you do $\{1,7\}$ vs $\{3,8\}$, the next problem is hard;
  if you do $\{1,8\}$ vs $\{3,7\}$, the next problem is easy;
  if you do $\{1,3\}$ vs $\{7,8\}$, the next problem is easy.
- [BWG10]: Better to confuse near leaves than near root.
  Intuition: The root predictor tends to be overconstrained while the leafwards predictors are less constrained.

How do you learn the structure?

Our approach:

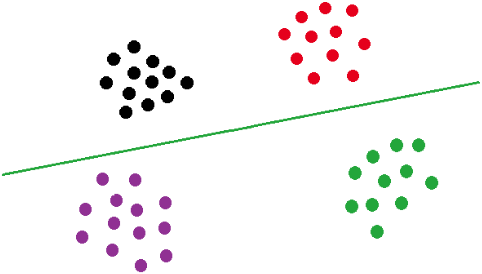
- **top-down** approach for class partitioning
- splitting criterion guaranteeing
  balanced tree $\Rightarrow$ **logarithmic training and testing time**
  <u>and</u>
  small classification error

Pure split and balanced split



- $k_r(x)$: number of data points in the same class as x on the right side of the partitioning
- $k(x)$: total number of data points in the same class as x
- n_r: number of data points on the right side of the partitioning
- n: total number of data points

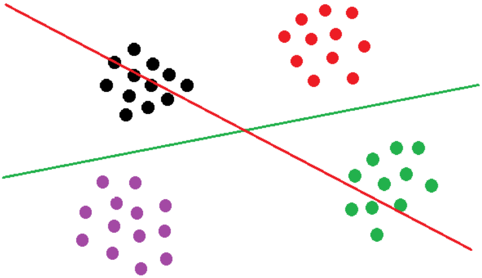
Pure split and balanced split



- $k_r(x)$: number of data points in the same class as x on the right side of the partitioning
- $k(x)$: total number of data points in the same class as x
- n_r: number of data points on the right side of the partitioning
- n: total number of data points

Measure of balanceness: $\frac{\mathbf{n_r}}{\mathbf{n}}$

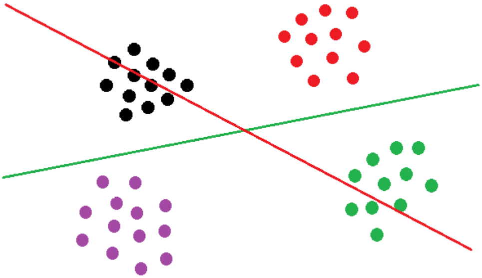
Pure split and balanced split



- $k_r(x)$: number of data points in the same class as x on the right side of the partitioning
- $k(x)$: total number of data points in the same class as x
- n_r: number of data points on the right side of the partitioning
- n: total number of data points

Measure of balanceness: $\frac{\mathbf{n_r}}{\mathbf{n}}$

Pure split and balanced split



- $k_r(x)$: number of data points in the same class as x on the right side of the partitioning
- $k(x)$: total number of data points in the same class as x
- n_r: number of data points on the right side of the partitioning
- n: total number of data points

Measure of balanceness: $\frac{n_r}{n}$ **Measure of purity:** $\frac{k_r(x)}{k(x)}$

Pure split and balanced split

- k: number of classes
- $\mathcal{H}$: hypothesis class (typically: linear classifiers)
- $\pi_y = \frac{|\mathcal{X}_y|}{n}$
- balance $= Pr(h(x) > 0)$
- purity $= \sum_{y=1}^{k} \pi_y \min(Pr(h(x) > 0|y), Pr(h(x) < 0|y))$

Pure split and balanced split

- k: number of classes
- $\mathcal{H}$: hypothesis class (typically: linear classifiers)
- $\pi_y = \frac{|\mathcal{X}_y|}{n}$
- balance $= Pr(h(x) > 0)$
- purity $= \sum_{y=1}^{k} \pi_y \min(Pr(h(x) > 0|y), Pr(h(x) < 0|y))$

Definition (Balanced split)

The hypothesis $h \in \mathcal{H}$ induces a balanced split iff

$$\exists_{c \in (0,0.5]} c \leq \text{balance} \leq 1 - c.$$

Pure split and balanced split

- k: number of classes
- $\mathcal{H}$: hypothesis class (typically: linear classifiers)
- $\pi_y = \frac{|\mathcal{X}_y|}{n}$
- balance $= Pr(h(x) > 0)$
- purity $= \sum_{y=1}^{k} \pi_y \min(Pr(h(x) > 0|y), Pr(h(x) < 0|y))$

Definition (Balanced split)

The hypothesis $h \in \mathcal{H}$ induces a balanced split iff

$$\exists_{c \in (0,0.5]} c \leq \text{balance} \leq 1 - c.$$

Definition (Pure split)

The hypothesis $h \in \mathcal{H}$ induces a pure split iff

$$\exists_{\delta \in [0,0.5)} \text{purity} \leq \delta.$$

Objective function

$$J(h) = 2\sum_{y=1}^{k} \pi_y \left| P(h(x) > 0) - P(h(x) > 0|y) \right|$$

$$= 2\mathbb{E}_{x,y}\left[\left| P(h(x) > 0) - P(h(x) > 0|y) \right| \right]$$

J(h) ⇒ Splitting criterion (objective function)

Given a set of n examples each with one of k labels, find a **partitioner** h that maximizes the objective.

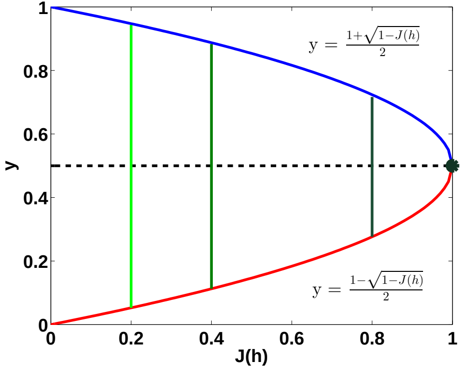
Lemma

For any hypothesis $h : \mathcal{X} \mapsto \{-1, 1\}$, the objective $J(h)$ satisfies $J(h) \in [0, 1]$. Furthermore, h induces a maximally pure and balanced partition iff $J(h) = 1$.

Balancing and purity factors

- **Balacing factor**

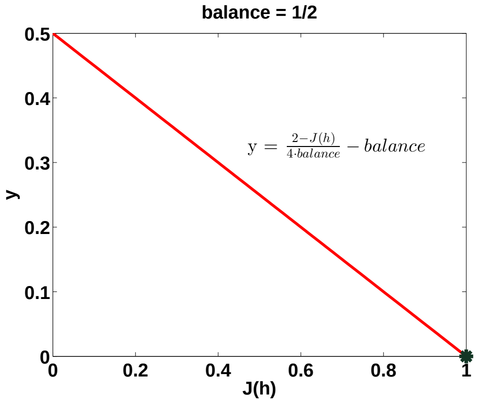
$$\text{balance} \in \left[\frac{1-\sqrt{1-{\color{red}J(h)}}}{2}, \frac{1+\sqrt{1-{\color{red}J(h)}}}{2}\right]$$



Balancing and purity factors

- **Purity factor**

$$\text{purity} \leq \frac{2 - J(h)}{4 \cdot \text{balance}} - \text{balance}$$



What is the quality of obtained tree?

- In each node of the tree $\mathcal{T}$ optimize the splitting criterion
- Apply recursively to construct a tree structure
- Measure the quality of the tree using entropy

$$G_{\mathcal{T}} = \sum_{l \in \text{leafs of } \mathcal{T}} w_l \sum_{y=1}^{k} \pi_{l,y} \ln \left(\frac{1}{\pi_{l,y}} \right)$$

Why?

Small entropy of leafs $\Rightarrow$ pure leafs

Goal: maximizing the objective reduces the entropy

What is the quality of obtained tree?

Definition (Weak Hypothesis Assumption)

Let m denotes any node of the tree $\mathcal{T}$, and let $\beta_m = P(h_m(x) > 0)$ and $P_{m,i} = P(h_m(x) > 0|i)$. Furthermore, let $\gamma \in \mathbb{R}^+$ be such that for all m, $\gamma \in (0, \min(\beta_m, 1 - \beta_m)]$. We say that the *weak hypothesis assumption* is satisfied when for any distribution $\mathcal{P}$ over $\mathcal{X}$ at each node m of the tree $\mathcal{T}$ there exists a hypothesis $h_m \in \mathcal{H}$ such that $J(h_m)/2 = \sum_{i=1}^{k} \pi_{m,i}|P_{m,i} - \beta_m| \geq \gamma$.

Theorem

Under the Weak Hypothesis Assumption, for any $\epsilon \in [0, 1]$, *to obtain* $G_{\mathcal{T}} \leq \epsilon$ *it suffices to make* $\left(\frac{1}{\epsilon}\right)^{\frac{4(1-\gamma)^2 \ln k}{\gamma^2}}$ *splits.*

What is the quality of obtained tree?

Definition (Weak Hypothesis Assumption)

Let m denotes any node of the tree $\mathcal{T}$, and let $\beta_m = P(h_m(x) > 0)$ and $P_{m,i} = P(h_m(x) > 0|i)$. Furthermore, let $\gamma \in \mathbb{R}^+$ be such that for all m, $\gamma \in (0, \min(\beta_m, 1 - \beta_m)]$. We say that the *weak hypothesis assumption* is satisfied when for any distribution $\mathcal{P}$ over $\mathcal{X}$ at each node m of the tree $\mathcal{T}$ there exists a hypothesis $h_m \in \mathcal{H}$ such that $J(h_m)/2 = \sum_{i=1}^{k} \pi_{m,i}|P_{m,i} - \beta_m| \geq \gamma$.

Theorem

Under the Weak Hypothesis Assumption, for any $\epsilon \in [0, 1]$, to obtain $G_{\mathcal{T}} \leq \epsilon$ it suffices to make $\left(\frac{1}{\epsilon}\right)^{\frac{4(1-\gamma)^2 \ln k}{\gamma^2}}$ splits.

- Tree depth $\approx \log \left[\left(\frac{1}{\epsilon}\right)^{\frac{4(1-\gamma)^2 \ln k}{\gamma^2}}\right] = \mathcal{O}(\ln k) \Rightarrow$

$\Rightarrow$ **logarithmic training and testing time**

LOMtree algorithm

- Recall the objective function we consider at every tree node:

$$J(h) = 2\mathbb{E}_y[|\mathbb{E}_x[\mathbb{1}(h(x) > 0)] - \mathbb{E}_x[\mathbb{1}(h(x) > 0|y)]|].$$

Problem: discrete optimization
Relaxation: drop the indicator operator and look at margins

LOMtree algorithm

- Recall the objective function we consider at every tree node:

$$J(h) = 2\mathbb{E}_y[|\mathbb{E}_x[\mathbb{1}(h(x) > 0)] - \mathbb{E}_x[\mathbb{1}(h(x) > 0|y)]|].$$

Problem: discrete optimization
Relaxation: drop the indicator operator and look at margins

LOMtree algorithm

- Recall the objective function we consider at every tree node:

$$J(h) = 2\mathbb{E}_y[|\mathbb{E}_x[\mathbb{1}(h(x) > 0)] - \mathbb{E}_x[\mathbb{1}(h(x) > 0|y)]|].$$

<u>Problem:</u> discrete optimization
<u>Relaxation:</u> drop the indicator operator and look at margins

- The objective function becomes

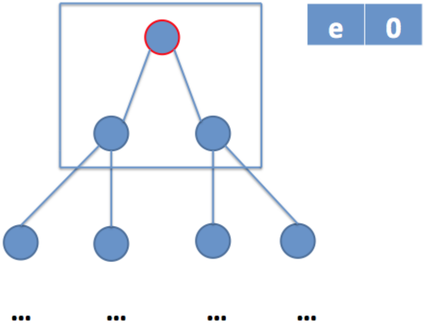
$$J(h) = 2\mathbb{E}_y[|\mathbb{E}_x[h(x)] - \mathbb{E}_x[h(x)|y]|]$$

  - Keep the online empirical estimates of these expectations.
  - The sign of the difference of two expectations decides whether to send an example to the left or right child node.

LOMtree algorithm

Let $e = 0$ and for all y, $e_y = 0$, $n_y = 0$
For each example (x, y)

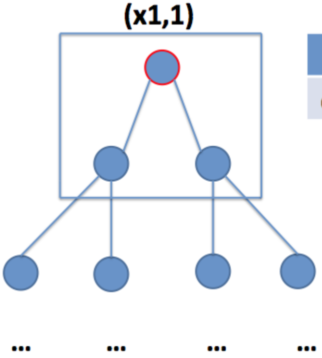
- if $e_y < e$ then $b = -1$ else $b = 1$
- Update w using (x, b)
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$



LOMtree algorithm

Let $e = 0$ and for all y, $e_y = 0$, $n_y = 0$
For each example (x, y)

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update w using (x, b)
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

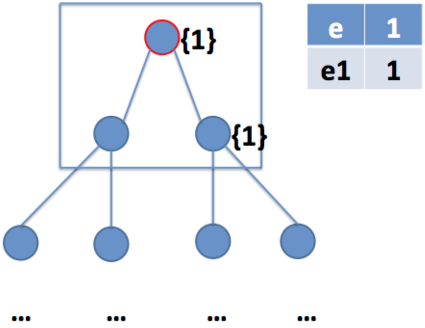


e	0
e1	0

LOMtree algorithm

Let $e = 0$ and for all y, $e_y = 0$, $n_y = 0$
For each example (x, y)

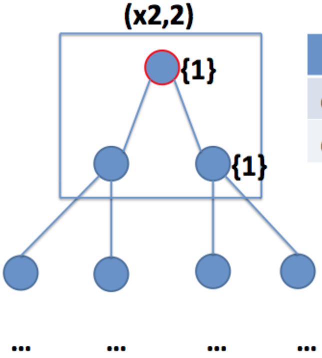
- if $e_y < e$ then $b = -1$ else $b = 1$
- Update w using (x, b)
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y-1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$



LOMtree algorithm

Let $e = 0$ and for all y, $e_y = 0$, $n_y = 0$
For each example (x, y)

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update w using (x, b)
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

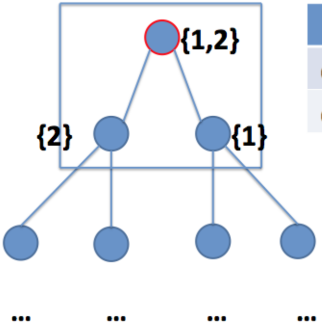


e	1
e1	1
e2	0

LOMtree algorithm

Let $e = 0$ and for all y, $e_y = 0$, $n_y = 0$
For each example (x, y)

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update w using (x, b)
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

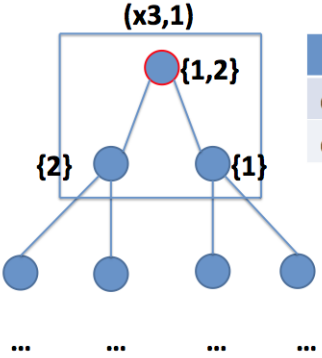


e	0
e1	1
e2	-1

LOMtree algorithm

Let $e = 0$ and for all y, $e_y = 0$, $n_y = 0$
For each example (x, y)

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update w using (x, b)
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

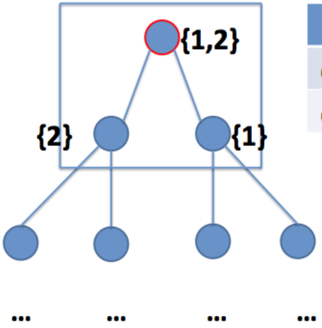


e	0
e1	1
e2	-1

LOMtree algorithm

Let $e = 0$ and for all y, $e_y = 0$, $n_y = 0$
For each example (x, y)

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update w using (x, b)
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

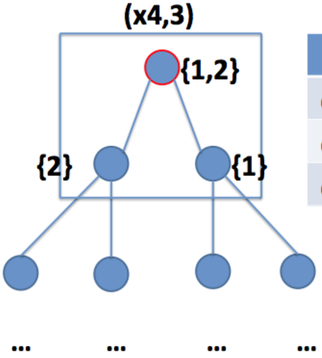


e	1/3
e1	1
e2	-1

LOMtree algorithm

Let $e = 0$ and for all y, $e_y = 0$, $n_y = 0$
For each example (x, y)

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update w using (x, b)
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

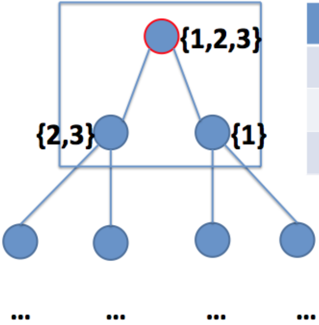


e	1/3
e1	1
e2	-1
e3	0

LOMtree algorithm

Let $e = 0$ and for all y, $e_y = 0$, $n_y = 0$
For each example (x, y)

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update w using (x, b)
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$



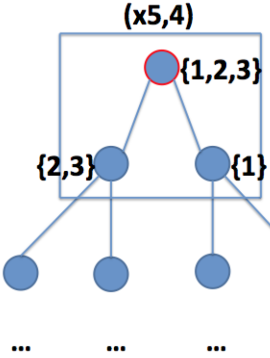
e	0
e1	1
e2	-1
e3	-1

LOMtree algorithm

Let $e = 0$ and for all y, $e_y = 0$, $n_y = 0$

For each example (x, y)

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update w using (x, b)
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$



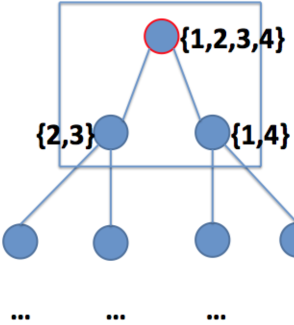
e	0
e1	1
e2	-1
e3	-1
e4	0

LOMtree algorithm

Let $e = 0$ and for all y, $e_y = 0$, $n_y = 0$

For each example (x, y)

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update w using (x, b)
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$



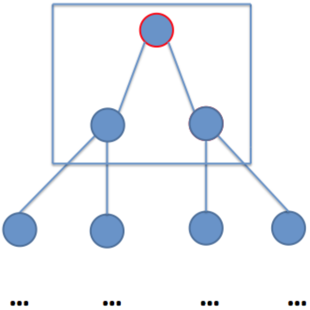
e	3/5
e1	1
e2	-1
e3	-1
e4	1

LOMtree algorithm

Let $e = 0$ and for all y, $e_y = 0$, $n_y = 0$
For each example (x, y)

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update w using (x, b)
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

Apply recursively to
construct a tree structure.



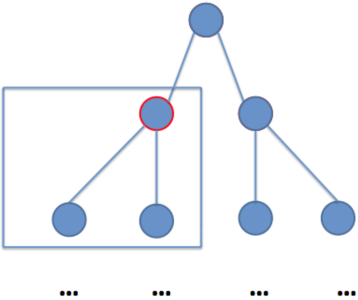
LOMtree algorithm

Let $e = 0$ and for all y, $e_y = 0$, $n_y = 0$

For each example (x, y)

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update w using (x, b)
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

Apply recursively to construct a tree structure.



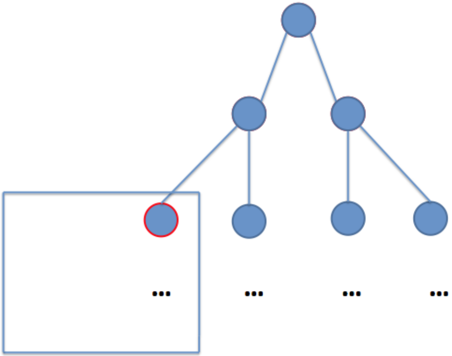
LOMtree algorithm

Let $e = 0$ and for all y, $e_y = 0$, $n_y = 0$

For each example (x, y)

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update w using (x, b)
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

Apply recursively to construct a tree structure.



Experiments

Table : Training time on selected problems.

	Isolet	Sector	Aloi
LOMtree	**16.27s**	**12.77s**	**51.86s**
OAA	19.58s	18.37s	11m2.43s

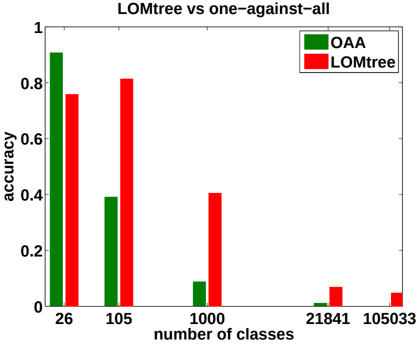
Table : Per-example test time on all problems.

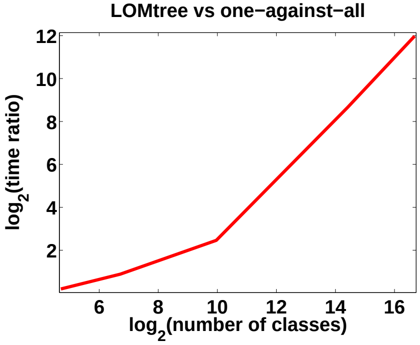
	Isolet	Sector	Aloi	ImNet	ODP
LOMtree	**0.14ms**	**0.13ms**	**0.06ms**	**0.52ms**	**0.26ms**
OAA	0.16 ms	0.24ms	0.33ms	0.21s	1.05s

Table : Test error (%) and confidence interval on all problems.

	LOMtree	Rtree	Filter tree
Isolet (26)	**6.36**$\pm$1.71	16.92$\pm$2.63	15.10$\pm$2.51
Sector (105)	16.19$\pm$2.33	**15.77**$\pm$2.30	17.70$\pm$2.41
Aloi (1000)	**16.50**$\pm$0.70	83.74$\pm$0.70	80.50$\pm$0.75
ImNet (22K)	**90.17**$\pm$0.05	96.99$\pm$0.03	92.12$\pm$0.04
ODP (105K)	**93.46**$\pm$0.12	93.85$\pm$0.12	93.76$\pm$0.12

Experiments





How to understand non-convex optimization?

Why non-convex optimization and deep learning?

State-of-the art results on number of problems:

- image recognition [KSH12, CMGS10]
- speech recognition [HDYDMJSVNSK12, GMH13]
- natural language processing [WCA14]
- video recognition [KTSLSF-F14, SZ14]

Why non-convex optimization and deep learning?

ImageNet 2014 Challenge: mostly convolutional networks

Classification+localization with provided training data: Ordered by localization error

Team name	Entry description	**Localization error**	Classification error
VGG	a combination of multiple ConvNets (by averaging)	0.253231	0.07405
VGG	a combination of multiple ConvNets (fusion weights learnt on the		
VGG			

Classification+localization with provided training data: Ordered by classification error

Team name	Entry description	**Classification error**	Localization error
GoogLeNet	No localization. Top5 val score is 6.66% error.	0.06656	0.606257
VGG	a combination of multiple ConvNets, including a net trained on		
VGG			

Object detection with additional training data: Ordered by number of categories won

Team name	Entry description	Description of outside data used	**Number of object categories won**	mean AP
GoogLeNet	Ensemble of detection models. Validation is 44.5% mAP	Pretraining on ILSVRC12 classification data.	142	0.439329
CUHK DeepID-	Co			

Ordered by mean average precision

Team name	Entry description	Description of outside data used	**mean AP**	Number of object categories won
GoogLeNet	Ensemble of detection models. Validation is 44.5% mAP	Pretraining on ILSVRC12 classification data.	0.439329	142
CUHK DeepID	*Combine multiple models described in the abstract without contextual modeling. The*	*ImageNet classification and*	0.406998	

Machine Learning Competitions Won (Yoshua Bengio)

- Winning the ICMI 2013 Grand Challenge on Emotion Recognition in the Wild! The challenge baseline accuracy was 27.5% - our approach yielded 41.0%) Kahou, S. E., Pal, C., Bouthillier, X., Froumenty, P., Gulcehre, C., *, Memisevic, R., Vincent, P., Courville, A. and Bengio, Y. (2013) Combining Modality Specific Deep Neural Networks for Emotion Recognition in Video. *(ICMI '13)*
- Unsupervised and Transfer Learning Challenge, presented at an ICML 2011 and IJCNN 2011 workshops of the same name, was won by LISA members using unsupervised layer-wise pre-training
- We also won the Transfer Learning Challenge at NIPS 2011's Challenges in Learning Hierarchical Models Workshop, using *spike-and-slab sparse coding* (ICML 2012 paper)

Challenge

Goal: Understanding loss function in deep learning.

Recent related works: Choromanska et al., 2015, Goodfellow et al., 2015, Dauphin et al., 2014, Saxe et al., 2014.

Challenge

Goal: Understanding loss function in deep learning.

Recent related works: Choromanska et al., 2015, Goodfellow et al., 2015, Dauphin et al., 2014, Saxe et al., 2014.

Questions:

- Why the result of multiple experiments with multilayer networks consistently give very similar performance despite the presence of many local minima?

Challenge

Goal: Understanding loss function in deep learning.

Recent related works: Choromanska et al., 2015, Goodfellow et al., 2015, Dauphin et al., 2014, Saxe et al., 2014.

Questions:

- Why the result of multiple experiments with multilayer networks consistently give very similar performance despite the presence of many local minima?
- What is the role of saddle points in the optimization problem?

Challenge

Goal: Understanding loss function in deep learning.

Recent related works: Choromanska et al., 2015, Goodfellow et al., 2015, Dauphin et al., 2014, Saxe et al., 2014.

Questions:

- Why the result of multiple experiments with multilayer networks consistently give very similar performance despite the presence of many local minima?
- What is the role of saddle points in the optimization problem?
- Is the surface of the loss function of multilayer networks structured?

Multilayer network and spin-glass model

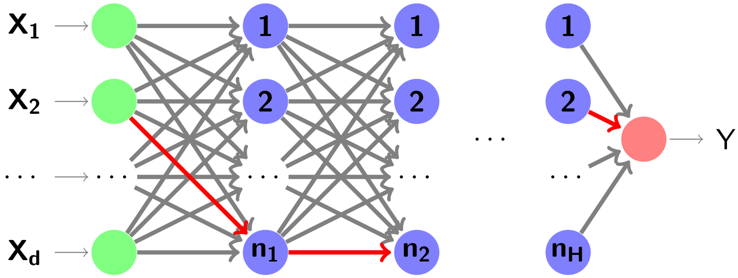
Can we use the spin-glass theory to explain the optimization paradigm with large multilayer networks?

Multilayer network and spin-glass model

Can we use the spin-glass theory to explain the optimization paradigm with large multilayer networks?

What assumptions need to be made?

Loss function in deep learning and assumptions



$$Y = \frac{1}{\Lambda^{(H-1)/2}} \sum_{i=1}^{\Psi} X_i A_i \prod_{k=1}^{H} w_i^{(k)},$$

- Ψ - number of input-output paths, $\Lambda = \sqrt[H]{\Psi}$ (assume $\Lambda \in \mathbb{Z}^+$)
- $H - 1$ - number of hidden layers
- $w_i^{(k)}$ - the weight of the k^{th} segment of the i^{th} path
- A_i - Bernoulli r.v. denoting path activation (0/1)

Loss function in deep learning and assumptions

Consider hinge loss

$$L(\mathbf{w}) = \max(0, 1 - Y_t Y),$$

where Y_t corresponds to the true data labeling $(1/-1)$, and $\mathbf{w}$ denotes all network weights.

- max operator is often modeled as Bernoulli r.v. $(0/1)$. Denote it as M and its expectation as $\rho^{'}$. Therefore

$$L(\mathbf{w}) = M(1 - Y_t Y) = M + \frac{1}{\Lambda^{(H-1)/2}} \sum_{i=1}^{\Psi} Z_i I_i \prod_{k=1}^{H} w_i^{(k)}, \quad (1)$$

  where $Z_i = -Y_t X_i$, and $I_i = MA_i$ is a Bernoulli r.v. $(0/1)$.
- assume $I_1, I_2, \ldots, I_\Psi$ are identically distributed (*A1p*)
- assume each X_i is a standard Gaussian r.v. (*A2p*)

Loss function in deep learning and assumptions

- assume network parametrization is redundant (*A3p*)
- assume unique parameters are uniformly distributed on the graph of connections of the network (*A4p*), i.e. every H-length product of unique weights appears in Equation 1 (the set of all products is $\{w_{i_1} w_{i_2} \dots w_{i_H}\}_{i_1,i_2,\dots,i_H=1}^{\Lambda}$).

$$L(\mathbf{w}) = M + \frac{1}{\Lambda^{(H-1)/2}} \sum_{i_1,i_2,\dots,i_H=1}^{\Lambda} Z_{i_1,i_2,\dots,i_H} I_{i_1,i_2,\dots,i_H} w_{i_1} w_{i_2} \dots w_{i_H}.$$

Loss function in deep learning and assumptions

Definition

A network $\mathcal{M}$ which has the same graph of connections as network $\mathcal{N}$, whose size is N, and s unique weights satisfying $s \leq N$ is called a (s, ϵ)-*reduction image* of $\mathcal{N}$ for some $\epsilon \in [0, 1]$ if the prediction accuracy of $\mathcal{N}$ and $\mathcal{M}$ differ by no more than ϵ (thus they classify at most ϵ fraction of data points differently).

Theorem

Let $\mathcal{N}$ be a neural network giving the output whose expectation wrt. A's is Y_N. Let $\mathcal{M}$ be its (s, ϵ)-reduction image for some $s \leq N$ and $\epsilon \in [0, 0.5]$. By analogy, let Y_s be the expected output of network $\mathcal{M}$. Then the following holds

$$corr(sign(Y_s), sign(Y_N)) \geq \frac{1 - 2\epsilon}{1 + 2\epsilon},$$

where $corr(A, B) = \frac{\mathbb{E}[(A - \mathbb{E}[A]])(B - \mathbb{E}[B]])}{std(A)std(B)}$, std is the standard deviation and $sign(\cdot)$ denotes the sign of prediction.

Loss function in deep learning and assumptions

- assume the independence of $Z_{i_1,i_2,\ldots,i_H}$ and $I_{i_1,i_2,\ldots,i_H}$ $(A5u)$

$$\mathbb{E}_{M,I_1,I_2,\ldots,I_\Psi}[L(\mathbf{w})] = \rho^{'} + \rho \underbrace{\frac{1}{\Lambda^{(H-1)/2}} \sum_{i_1,i_2,\ldots,i_H=1}^{\Lambda} Z_{i_1,i_2,\ldots,i_H} w_{i_1} w_{i_2} \ldots w_{i_H}}_{\mathcal{L}_{\Lambda,H}}.$$

- assume that Z's are independent $(A6u)$
- impose spherical constraint $(A7p)$

$$\frac{1}{\Lambda}\sum_{i=1}^{\Lambda} w_i^2 = 1.$$

We obtain the Hamiltonian of the spherical spin-glass model!!!

Question: What happens when $\Lambda \to \infty$?

Important quantities

Definition

Let the following quantity be called an **energy barrier**

$$E_\infty = E_\infty(H) = 2\sqrt{\frac{H-1}{H}}.$$

Definition

Let the normalized minimum of the Hamiltonian $\mathcal{L}_{\Lambda,H}$ be called a **ground state** and be defined as

$$E_0 = \frac{1}{\Lambda} \inf_{\boldsymbol{\sigma} \in S^{N-1}(\sqrt{\Lambda})} \mathcal{L}_{\Lambda,H}(\boldsymbol{\sigma}).$$

Let $(E_k(H))_{k\in\mathbb{N}}$ be a strictly decreasing sequence, that is converging to E_∞ as $k \to \infty$.

Hamiltonian of the spherical spin-glass model: properties

- All critical values of the Hamiltonian of fixed index[1] (non diverging with Λ) must lie in the band $(-\Lambda E_0(H), -\Lambda E_\infty(H))$.
- Finding a critical value with index larger or equal to k (for any fixed integer k) below energy level $-\Lambda E_k(H)$ is improbable.
- With overwhelming probability the critical values just above the global minimum (ground state) are local minima exclusively. Above the band $(-\Lambda E_0(H), -\Lambda E_1(H))$ containing only local minima (critical points of index 0), there is another one, $(-\Lambda E_1(H), -\Lambda E_2(H))$, where one can only find local minima and saddle points of index 1, and above this band there exists another one, $(-\Lambda E_2(H), -\Lambda E_3(H))$, where one can only find local minima and saddle points of index 1 and 2, and so on.

[1] Index of $\nabla^2\mathcal{L}$ at $\mathbf{w}$ is the number of negative eigenvalues of the Hessian $\nabla^2\mathcal{L}$ at $\mathbf{w}$. Local minima have index 0.

Hamiltonian of the spherical spin-glass model: properties

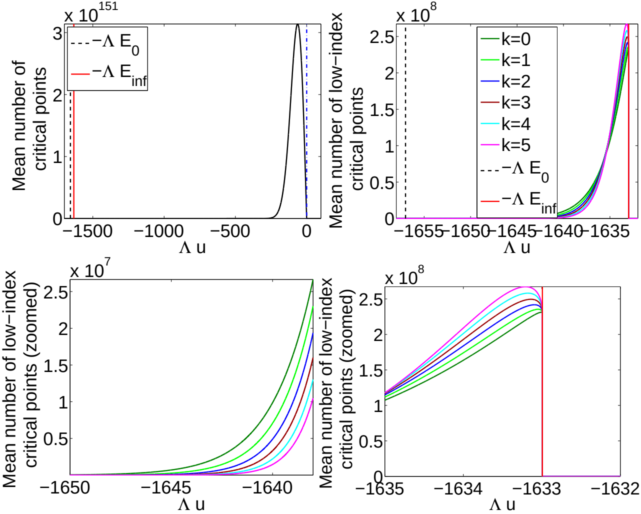
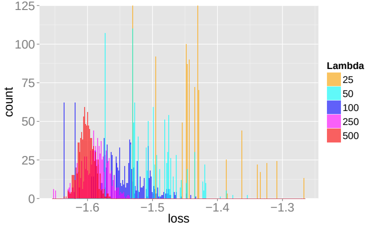
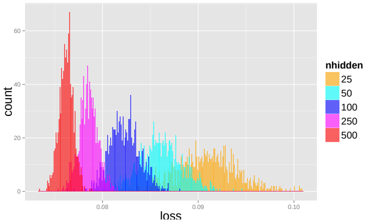
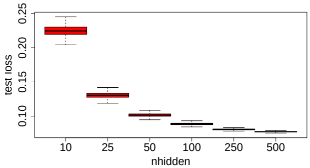


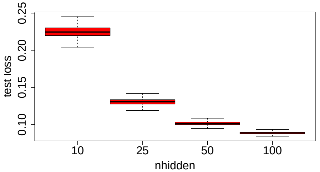
Figure : $H = 3$ and $\Lambda = 1000$. Black line: $u = -\Lambda E_0(H)$, red line: $u = -\Lambda E_\infty(H)$.

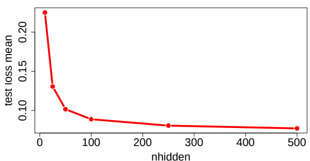
Comparison of deep network and spherical spin-glass model

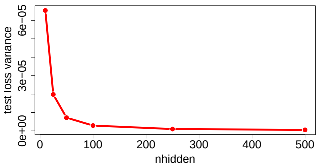




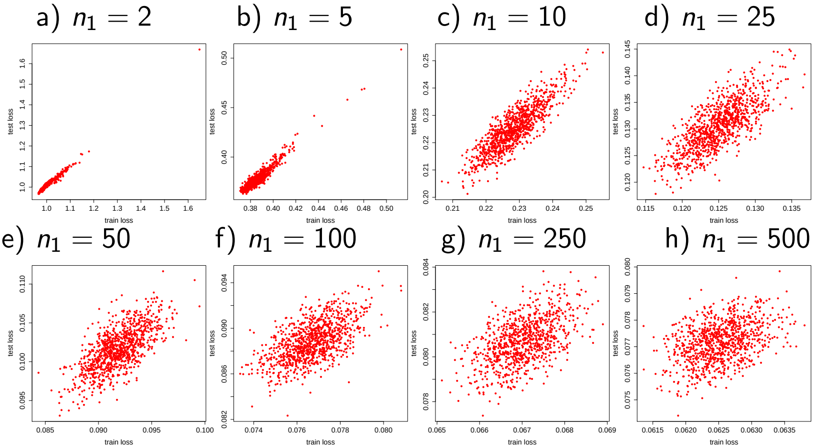








Deep network: correlation between train and test loss



n_1	25	50	100	250	500
ρ	0.7616	0.6861	0.5983	0.5302	0.4081

Table : Pearson correlation between training and test loss.

Deep network: index of recovered solutions

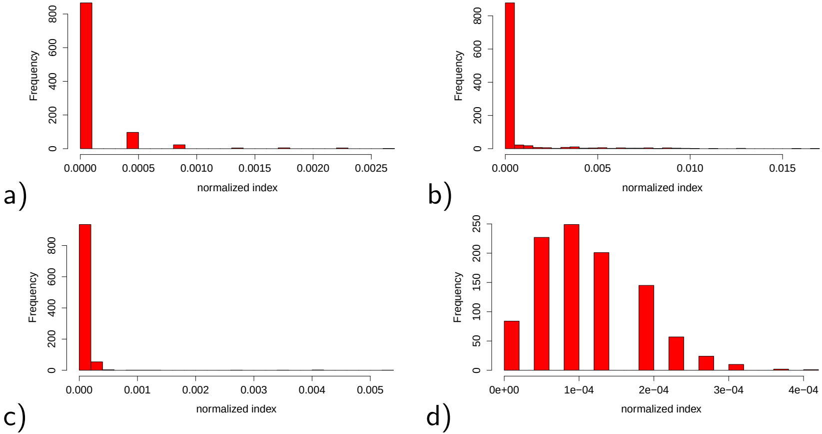


Figure : Distribution of normalized index of solutions for $n_1 = \{a)10, b)25, c)50, d)100\}$ hidden units.

Spherical spin-glass versus deep network

Conjecture (Deep learning)

For large-size networks, most local minima are equivalent and yield similar performance on a test set.

Spherical spin-glass

Critical points form an ordered structure such that there exists an *energy barrier* $\Lambda E_{-\infty}$ (a certain value of the Hamiltonian) below which with overwhelming probability one can find only low-index critical points, most of which are concentrated close to the barrier.

Spherical spin-glass versus deep network

Conjecture (Deep learning)

The probability of finding a "bad" (high value) local minimum is non-zero for small-size networks and decreases quickly with network size.

Spherical spin-glass

Low-index critical points are 'geometrically' lying closer to the ground state than high-index critical points.

Spherical spin-glass versus deep network

Conjecture (Deep learning)

Saddle points play a key-role in the optimization problem in deep learning.

Spherical spin-glass

With overwhelming probability one can find only high-index saddle points above energy $\Lambda E_{-\infty}$ and there are exponentially many of those.

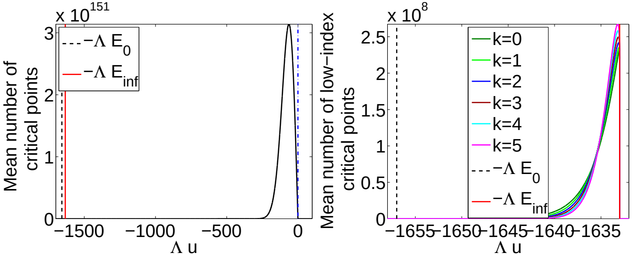


Figure : $H = 3$ and $\Lambda = 1000$. Black line: $u = -\Lambda E_0(H)$ (ground state), red line: $u = -\Lambda E_\infty(H)$ (energy barrier).

Spherical spin-glass versus deep network

Conjecture (Deep learning)

Struggling to find the global minimum on the training set (as opposed to one of the many good local ones) is not useful in practice and may lead to overfitting.

n_1	25	50	100	250	500
ρ	0.7616	0.6861	0.5983	0.5302	0.4081

Table : Pearson correlation between training and test loss for different numbers of hidden units of a network with one hidden layer. *MNIST* dataset.

Spherical spin-glass

Recovering the ground state, i.e. global minimum, takes exponentially long time.

Take-home message

- For large-size networks, most local minima are equivalent and yield similar performance on a test set.
- The probability of finding a “bad” (high value) local minimum is non-zero for small-size networks and decreases quickly with network size.
- Struggling to find the global minimum on the training set (as opposed to one of the many good local ones) is not useful in practice and may lead to overfitting.

Open problem

Can we establish a stronger connection between the loss function of the deep model and the spherical spin-glass model by dropping the unrealistic assumptions?

Convexity and non-convexity: challenges

Convex and non-convex world

- Building solvers, i.e. bound majorization
  - New and tight **quadratic bound on the partition function**
  - **Linear convergence** of the batch/semi-stochastic variants
  - **Competetive/better** than **state-of-the-art** methods
  - Admits multiple extensions
- Designing problem-specific, i.e. multi-classification, objectives
  - **Logarithmic training and testing time**
  - **Reduction** from multi-class to binary classification
  - New **splitting criterion** with desirable properties
    - allows gradient descent style optimization
    - makes decision trees applicable to multi-class classification

Non-convex world

- Understading why non-convex approaches work
  - **Deep learning: state-of-the-art** in numerous problems
  - Possible connection between **spin-glass theory** and **deep learning**
  - Landscape is **highly non-convex** but most likely **structured**

Acknowledgments

Courant Institute of Mathematical Sciences: Yann LeCun, Gérard Ben Arous
Columbia University: Tony Jebara, Shih-Fu Chang
George Washington University: Claire Monteleoni
NYU Polytechnic School of Engineering: Mariusz Bojarski
Microsoft Research: John Langford, Alekh Agarwal
Google Research: Krzysztof Choromanski, Dimitri Kanevsky
IBM T. J.Watson Research Center: Aleksandr Aravkin
ATT Shannon Research Laboratories: Phyllis Weiss, Alice Chen
LEAR team of Inria: Zaid Harchaoui

PostDocs: Pablo Sprechmann
PhD students: Michael Mathieu, Mikael Bruce Henaff, Sixin Zhang, Ross Goroshin, Rahul Krishnan, Wojciech Zaremba, Hyungtae Kim