Fundamentals

Seminar Data Analytics I
International Master's Program in Data Analytics
University of Hildesheim
Summer Semester 2018

Daniel Obando Montero
Shabanaz Chamurally
Daniela Thyssens

Motivation

1. **Stochastic Gradient Descent Training for L1-regularized Log-linear Models with Cumulative Penalty**
   - Stochastic gradient descent (SGD) is attractive because it often requires much less training time in practice than batch training algorithms.
   - However, L1-regularization cannot be efficiently applied in SGD training.
   - Paper present a simple method for solving two problems in SGD learning.

2. **Curriculum Learning**
   - Humans and animals learn much better when the examples are presented in order of increasing difficulty and organized in a meaningful order.
   - The paper formalizes how a curriculum can be used in the context of machine learning.

3. **Combined Regression & Ranking**
   - Many real-world data mining tasks require the achievement of two distinct goals when applied to unseen data: first, to induce an accurate preference ranking, and second to give good regression performance.
   - The paper present an efficient and effective Combined Regression and Ranking method (CRR) that optimizes regression and ranking objectives simultaneously.

Stochastic Gradient Descent Training for L1-regularized Log-linear Models with Cumulative Penalty

Yoshimasa Tsuruoka, Jun'ichiTsujii and Sophia Ananiadou

ACL '09 Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1 Pages 477-485
Suntec, Singapore — August 02 - 07, 2009

Seminar Data Analytics I
International Master's Program in Data Analytics
University of Hildesheim
Summer Semester 2018

Daniel Obando Montero

Outline

1. Introduction
2. Log-Linear Models in NLP and L1
3. SGD - L1
4. OWL-QN
5. Experiments
6. Criticism
7. Conclusions

Outline

1. Introduction
2. Log-Linear Models in NLP and L1
3. SGD - L1
4. OWL-QN
5. Experiments
6. Criticism
7. Conclusions

Introduction

- SGD is a very attractive learning framework.
- Not efficient with L1.
- 2 problems:
  - Inefficiency of applying the L1 penalty to the weights of the features.
  - Doesn’t always lead to compact models.

Introduction

- Objective: present a simple method for solving the two problems in SGD learning.

Introduction

- How?
    - keep track of the total penalty and the penalty applied on each weight
    - The penalty is applied based on the difference between those values.

Introduction

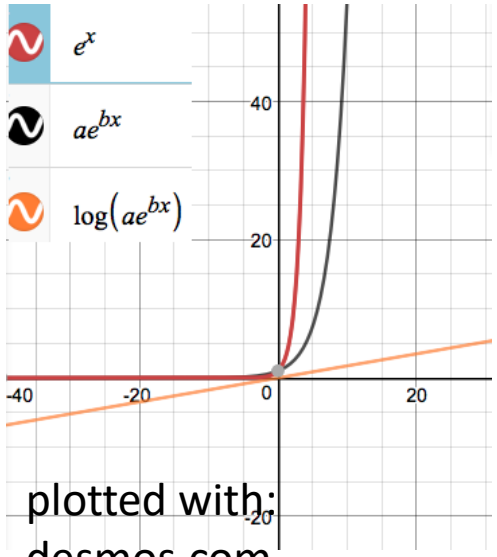
- How?
  - vs “state-of-the-art” OWL-QN
  - experiments with:
    - text chunking, named entity recognition and POS tagging

Outline

1. Introduction
2. Log-Linear Models in NLP and L1
3. SGD - L1
4. OWL-QN
5. Experiments
6. Criticism
7. Conclusions

Log-Linear Models in NLP and L1

- Where f_i indicates the occurrence of feature i and w_i is the weight of the feature.
- Z(x) is a partition function that allows a well formed normal distribution.



plotted with: desmos.com

$$p(\mathbf{y}|\mathbf{x}) = \frac{e^{\sum_i w_i f_i(\mathbf{y},\mathbf{x})}}{Z(\mathbf{x})}$$

$$Z(\mathbf{x}) = \sum_y e^{\sum_i w_i f_i(\mathbf{y},\boldsymbol{x})}$$

(Tsuruoka, 2009)

Log-Linear Models in NLP and L1

- Maximize the conditional likelihood of the data.
- N = # of training samples.
- y_j is the correct output of input x_j
- R is the L1-regularization term known as LASSO regression (Least Absolute Shrinkage and Selection Operator).
- C is the metaparameter that controls the degree of regularization.

$$\mathcal{L}_{\mathbf{w}} = \sum_{j=1}^{N} \log p(\mathbf{y}_j|\mathbf{x}_j;\mathbf{w}) - R(\mathbf{w})$$

$$R(\mathbf{w}) = C\sum_i |w_i|$$

$$L(j,\mathbf{w}) \ \forall \ \text{sample} \in \log \ p(\mathbf{y}_j|\mathbf{x}_j;\mathbf{w})$$

$$\mathcal{L}_{\mathbf{w}} = \sum_{j=1}^{N} L(j,\mathbf{w}) - C\sum_i |w_i|$$

(Tsuruoka, 2009)

Outline

1. Introduction
2. Log-Linear Models in NLP and L1
3. SGD - L1
4. OWL-QN
5. Experiments
6. Criticism
7. Conclusions

Stochastic Gradient Descent(SGD)

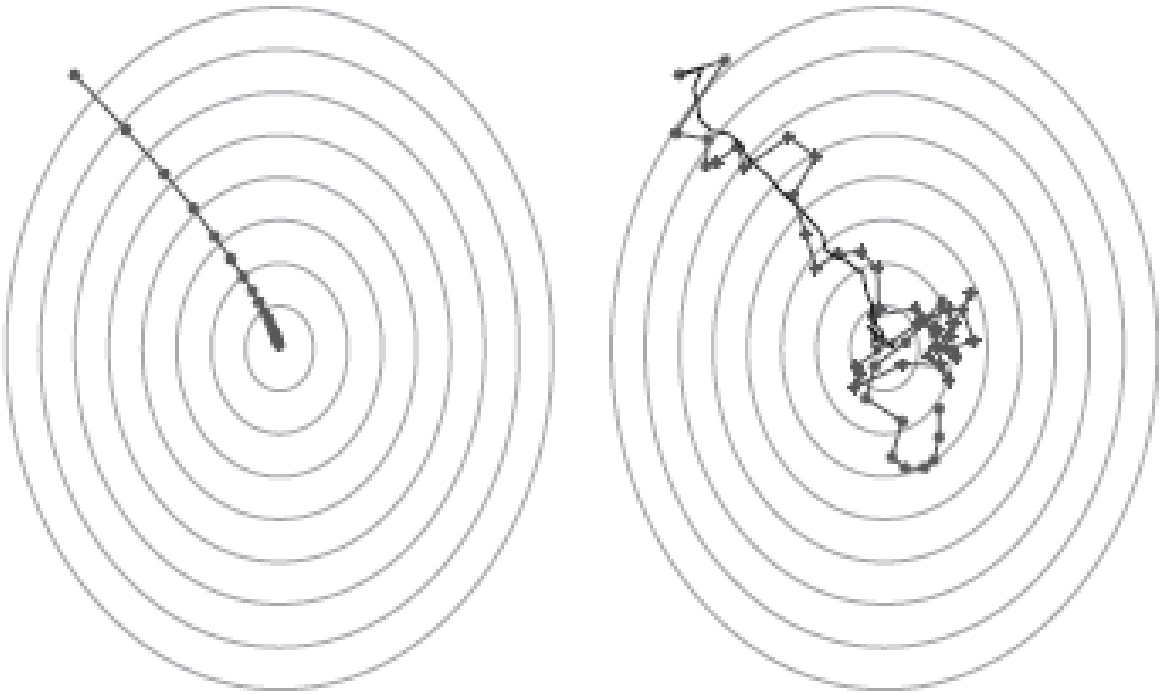
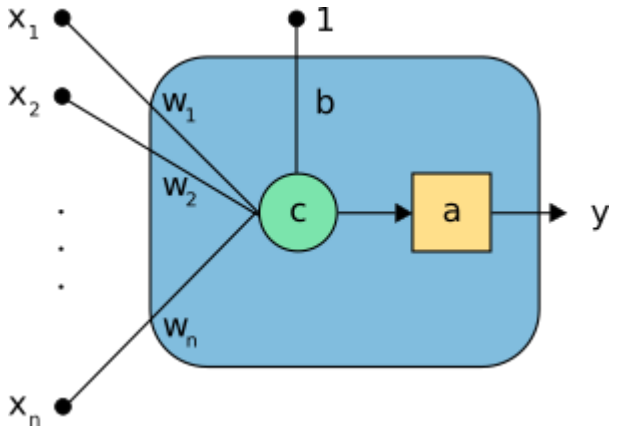
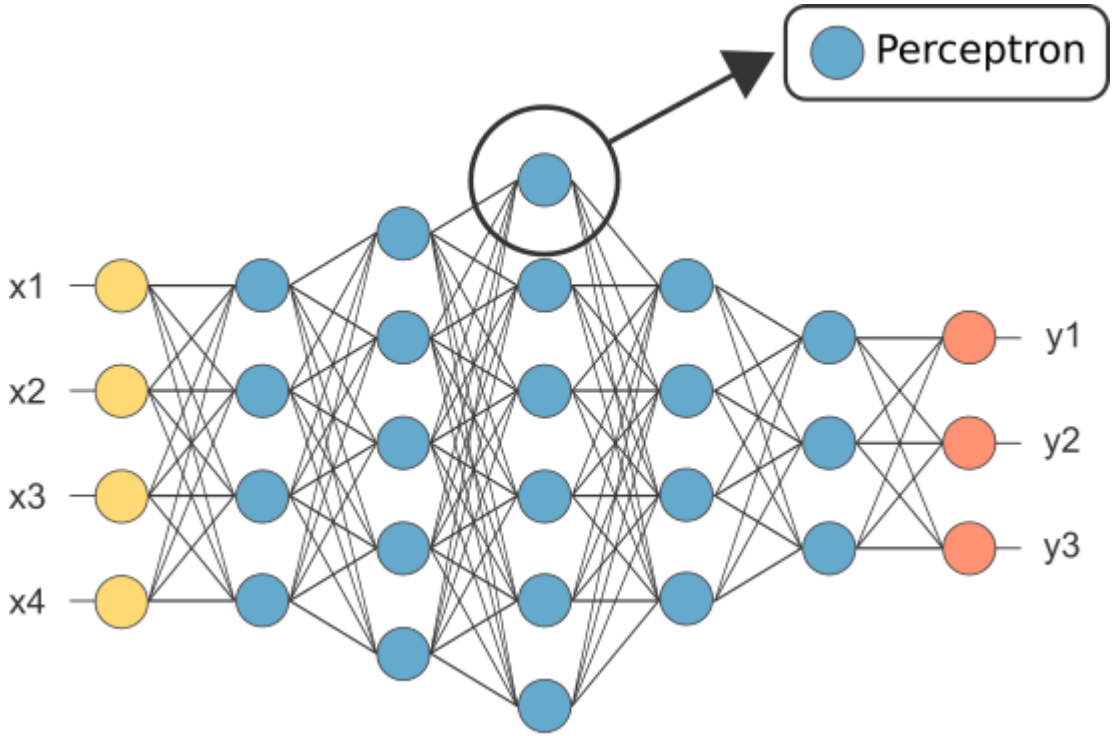


figure source: researchgate.net

Perceptron





$h(x) = W\mathbf{x} + b$

$y = a(h)$

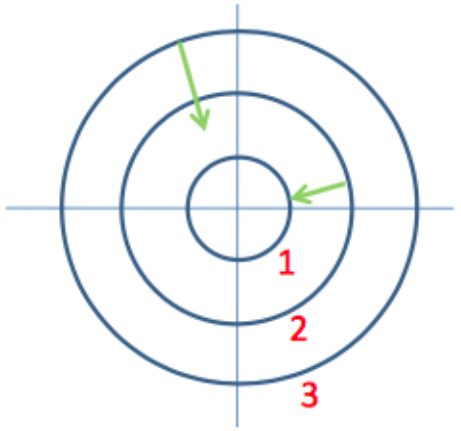
figures source: neuraldesigner.com

SGD - L1

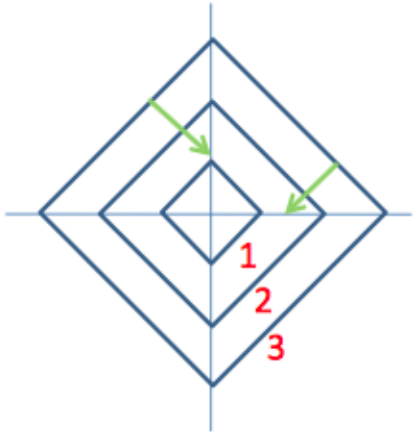
- SGD-L1 (Naive).
- SGD-L1 (Clipping + Lazy Update).
- SGD-L1 (Cumulative).
- SGD-L1 (Cumulative + Exponential Decay).

SGD - L1

- Why L1?
- Is capable of learning good models when most features are irrelevant.
- Many of the parameters are exactly zero.
- L2 “pushes” to 0 but it never reaches it



Negative gradient of L_2-norm always points directly toward 0



“Negative gradient” of L_1-norm (direction of steepest descent) points toward coordinate axes

(Andrew, 2007)

SGD and L1

The updates of the weights of the features is given by the following formula at the given j sample.

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \eta k \frac{\partial}{\partial \mathbf{w}} (L(j, \mathbf{w}) - \frac{C}{N} \sum_i |w_i|)$$

Then, the update for the weight of each feature i is:

$$w_i^{k+1} = w_i^k + \eta k \frac{\partial}{\partial w_i} (L(j, \mathbf{w}) - \frac{C}{N} |w_i|)$$

(Tsuruoka, 2009)

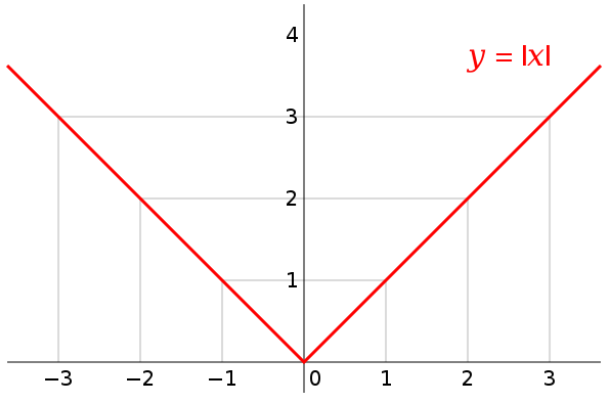


figure source: wikipedia.org

SGD and L1 (naive)

$$w_i^{k+1} = w_i^k + \eta^k \frac{\partial L(j, \mathbf{w})}{\partial w_i} - \frac{C}{N} \eta_k \text{sign}(w_i)$$

$$\text{sign}(x) = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ 0 & x = 0 \end{cases}$$

(Tsuruoka, 2009)

SGD and L1 (naive)

- Two problems:
  - at each update, the penalty must be applied to all features.
  - it does not produce a compact model.
    - i.e. most of the weights of the features do not become zero as a result of training.

SGD-L1 (Clipping + Lazy Update)

$$w_i^{k+\frac{1}{2}} = w_i^k + \eta_k \frac{\partial L(j,\mathbf{w})}{\partial w_i}\Bigg|_{w=w^k}$$

if $w_i^{k+\frac{1}{2}} > 0$ **then**

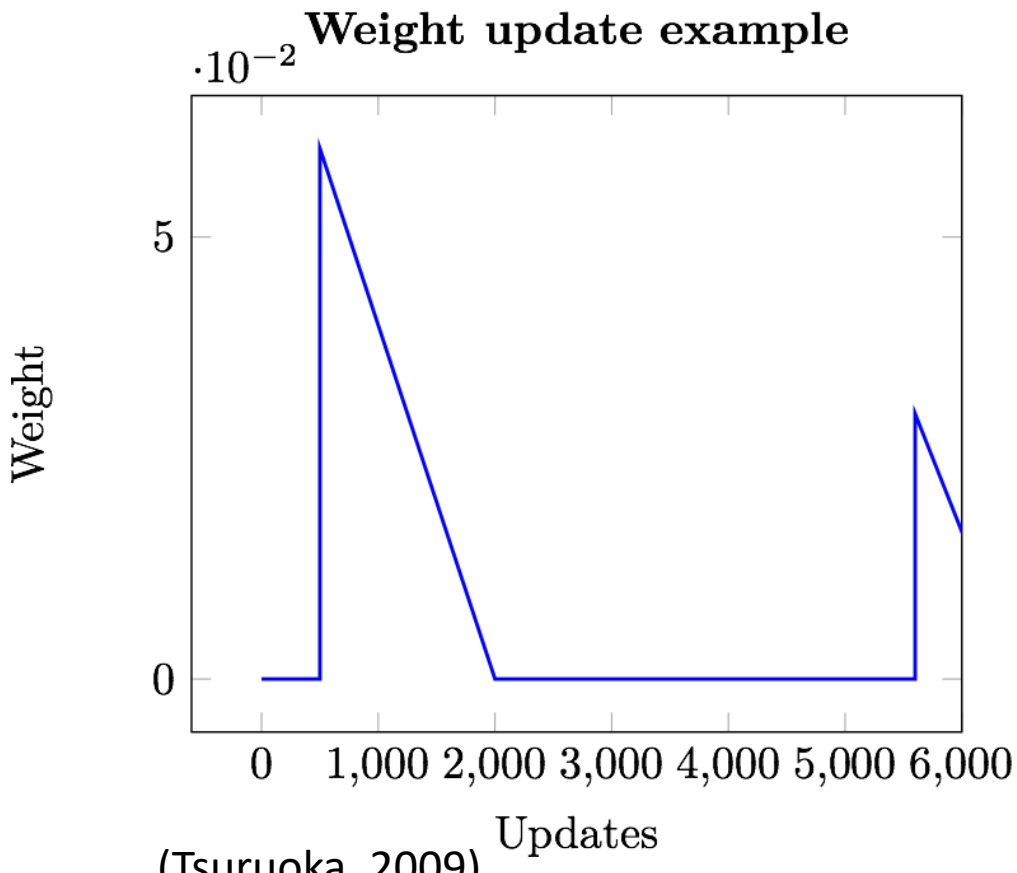
$$w_i^{k+1} = \max(0, w_i^{k+\frac{1}{2}} - \frac{C}{N}\eta_k)$$

else if $w_i^{k+\frac{1}{2}} < 0$ **then**

$$w_i^{k+1} = \min(0, w_i^{k+\frac{1}{2}} + \frac{C}{N}\eta_k)$$

(Carpenter, 2008)

SGD-L1 (Clipping + Lazy Update)



(Tsuruoka, 2009)

SGD-L1 (Cumulative)

- u_k is the value of the total penalty each weight could have received.
- q_i^k is the total L1 penalty that w_i has actually received.

$$u_k = \frac{C}{N}\sum_{t=1}^{k}\eta_t \quad q_i^k = \sum_{t=1}^{k}(w_i^{t+1} - w_i^{t+\frac{1}{2}})$$

$$w_i^{k+\frac{1}{2}} = w_i^k + \eta_k \left.\frac{\partial L(j,\mathbf{w})}{\partial w_i}\right|_{w=w^k}$$

$$\textbf{if } w_i^{k+\frac{1}{2}} > 0 \textbf{ then}$$
$$\quad w_i^{k+1} = \max(0, w_i^{k+\frac{1}{2}} - (u_k + q_i^{k-1}))$$
$$\textbf{else if } w_i^{k+\frac{1}{2}} < 0 \textbf{ then}$$
$$\quad w_i^{k+1} = \min(0, w_i^{k+\frac{1}{2}} + (u_k - q_i^{k-1}))$$

(Tsuruoka, 2009)

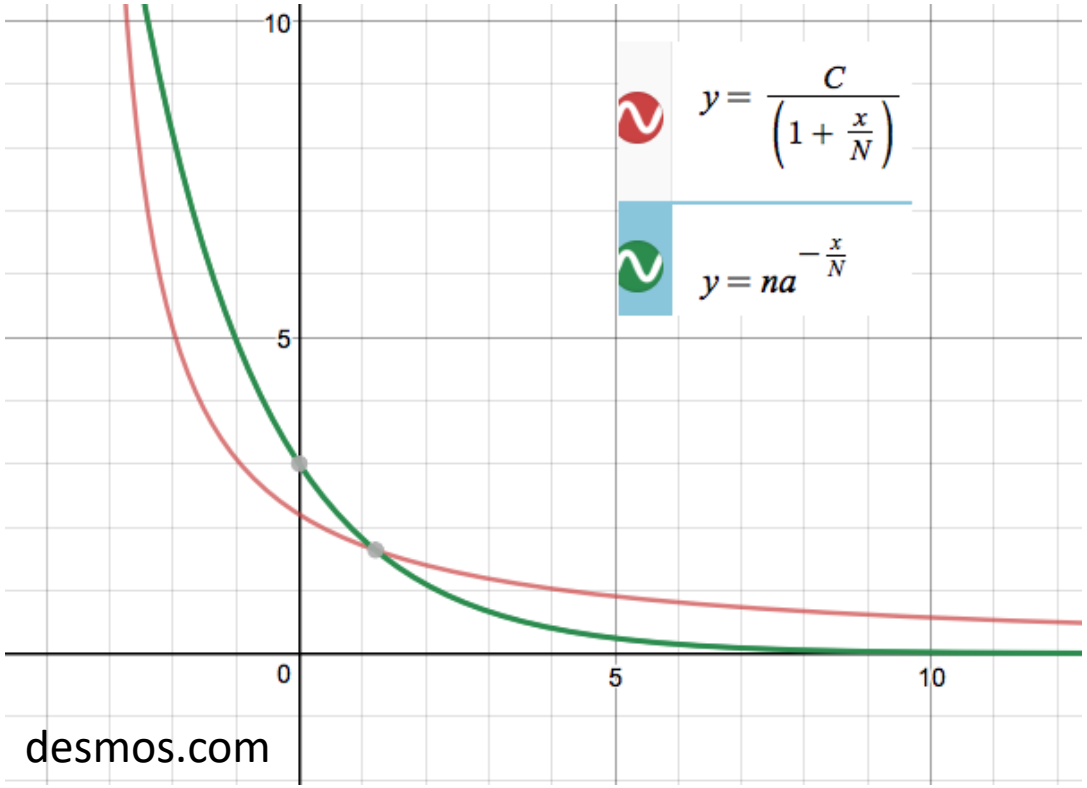
SGD-L1 (Cumulative + Exponential Decay)

n_0 and alpha are meta-parameters

$$\eta_k = \frac{\eta_0}{1 + \frac{k}{N}}$$

$$\eta_k = \eta_0 \alpha^{-\frac{k}{N}}$$

(Tsuruoka, 2009)



```
1  train (C):
2      u ← 0
3      Initialize w_i and q_i with zero for all i
4      for k = 0 to maxIterations do
5          η ← learningRate(k)
6          u ← u + η C/N
7          Select sample j randomly
8          updateWeights(j)
9      end
10
11 updateWeights (j):
12     for i ∈ features used in sample j do
13         w_i ← w_i + η ∂L(j,w)/∂w_i
14         applyPenalty(i)
15     end
16
17 applyPenalty (i):
18     z ← w_i
19     if w_i > 0 then
20         w_i = max(0, w_i − (u + q_i))
21     else if w_i < 0 then
22         w_i = min(0, w_i + (u − q_i))
23     q_i ← q_i + (w_i − z)
```

(Tsuruoka, 2009)

Outline

1. Introduction
2. Log-Linear Models in NLP and L1
3. SGD - L1
4. OWL-QN
5. Experiments
6. Criticism
7. Conclusions

OWL-QN

- Orthant-Wise Limited-memory Quasi Newton
- Scalable training of L1-regularized log-linear models
- Galen Andrew and Jianfeng Gao. 2007.
- Proposal:
  - An algorithm based on L-BFGS for training large-scale log-linear models using L1-regularization.

OWL-QN

Newton

Gradient Descent

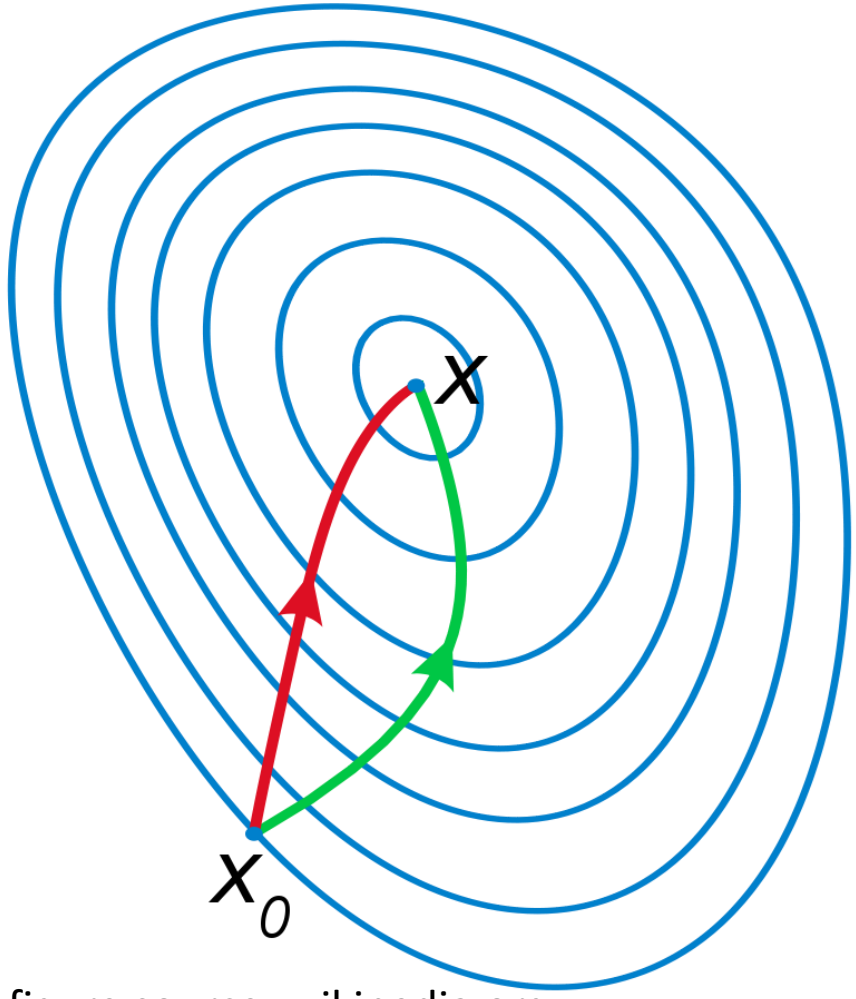


figure source: wikipedia.org

OWL-QN

- Newton’s method: Jacobian/Hessian.
- Quasi Newton uses approximations.
- Broyden-Fletcher-Goldfarb-Shanno (BFGS).
- Convergence and quadratic Taylor Series.
- Limited-memory BFGS (L-BFGS).

OWL-QN

- H_k: L-BFGS approximation to the inverse Hessian of the loss.

$$f(x) = \ell(x) + r(x)$$

$$r(x) = C \sum_i |x_i|, \ C > 0$$

$$\ell(x) = -\sum_{j=1}^{M} \log P_w(y_j | x_j)$$

$$x^{k+1} = x^k - \alpha \mathbf{H_k} g^k, \ \text{for } \alpha \ \in (0, \infty)$$

(Andrew, 2007)

OWL-QN

choose initial point x^0
$displacement \Leftarrow \{\}, gradientChange \Leftarrow \{\}$
for $k = 0$ ***to*** $maxIterations$ **do**
  Compute $orthant^k = -\diamond f(x^k)$
  Compute $descentDirection^k \Leftarrow$
    $\mathbf{H_k} orthant^k$ using $displacement$ and $gradientChange$
  $projection^k \Leftarrow \pi(descentDirection^k; orthant^k)$
  Find x^{k+1} with constrained line search
  if termination condition satisfied **then**
    Stop and return x^{k+1}
  Update $displacement$ with $s^k = x^{k+1} - x^k$
  Update $gradientChange$ with $y^k = \nabla\ell(x^{k+1}) - \nabla\ell(x^k)$
end

(Andrew, 2007)

- Variant of L-BFGS
- pseudo gradients.
- projection of descent onto the defined orthant.
- to remain in the valid region were the projection is valid.
- discarding the vectors from last iteration.

Outline

1. Introduction
2. Log-Linear Models in NLP and L1
3. SGD - L1
4. OWL-QN
5. Experiments
6. Criticism
7. Conclusions

Experiments

- Conditional Random Fields (CRF):
  - Text Chunking
  - Named Entity Recognition
  - Part-Of-Speech Tagging (POS-Tagging)

Experiments

- Processor Xeon 2.13GHz
- CRF++ version 0.50 developed by Taku Kudo

Experiments - Text Chunking

- Training data: 8936 sentences.
- Held-out data: 1000
- C was tuned to maximize the likelihood of the held-out data.
- The learning rate parameters were tuned to maximize the value of the objective function in 30 passes.

(Tsuruoka, 2009)

Experiments - Text Chunking

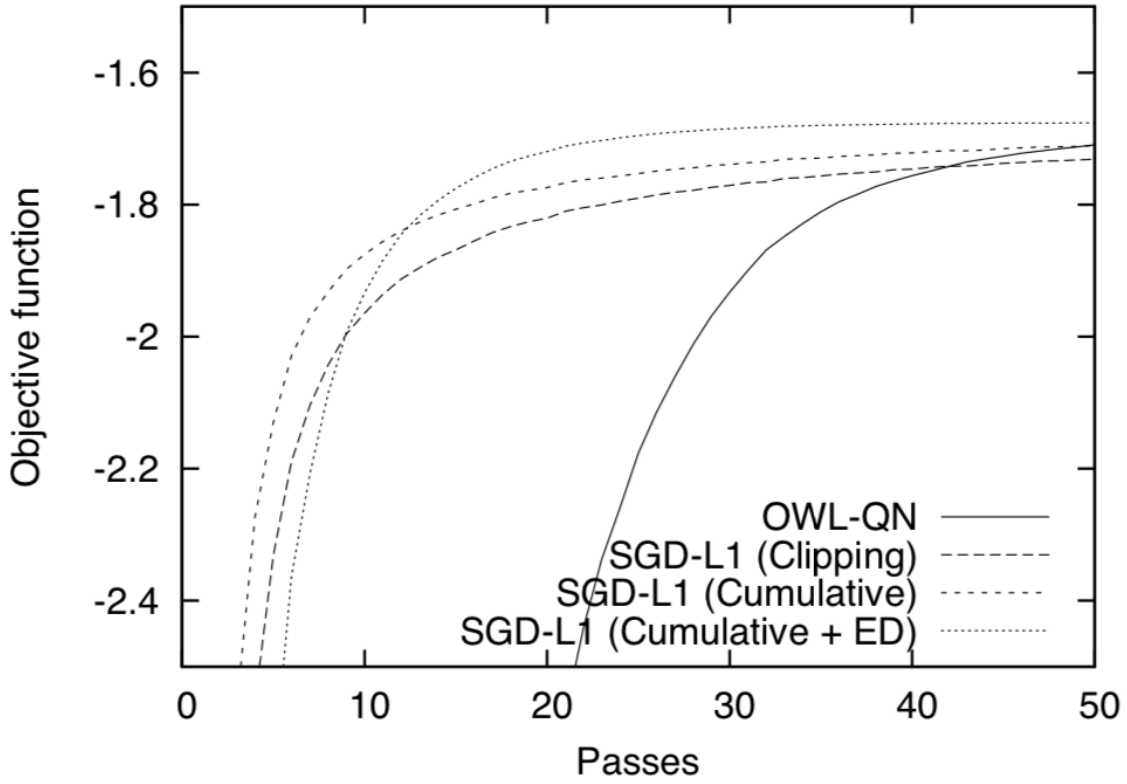


Figure 3: CoNLL 2000 chunking task: Objective

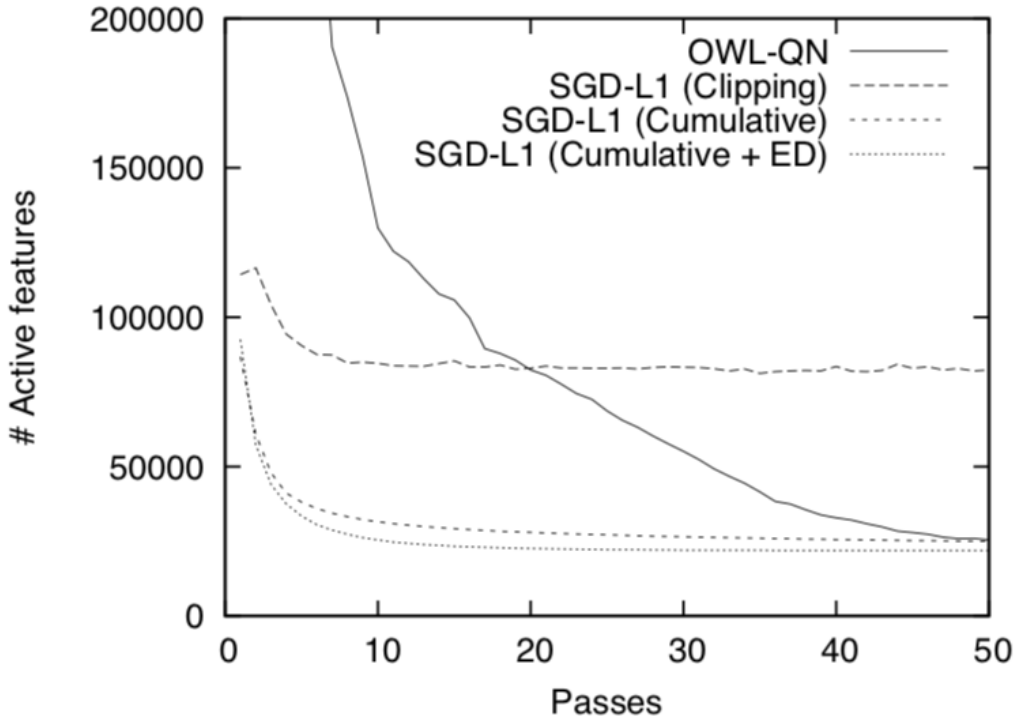


Figure 4: CoNLL 2000 chunking task: Number of active features.

(Tsuruoka, 2009)

Experiments - Text Chunking

	Passes	$\mathcal{L}_{\mathbf{w}}/N$	# Features	Time (sec)	F-score
OWL-QN	160	-1.583	18,109	598	93.62
SGD-L1 (Naive)	30	-1.671	455,651	1,117	93.64
SGD-L1 (Clipping + Lazy-Update)	30	-1.671	87,792	144	93.65
SGD-L1 (Cumulative)	30	-1.653	28,189	149	93.68
SGD-L1 (Cumulative + Exponential-Decay)	30	-1.622	23,584	148	93.66

Table 1: CoNLL-2000 Chunking task. Training time and accuracy of the trained model on the test data.

(Tsuruoka, 2009)

Experiments - Named Entity Recognition

- 18,546 sentences

	Passes	$\mathcal{L}_{\mathbf{w}}/N$	# Features	Time (sec)	F-score
OWL-QN	161	-2.448	30,710	2,253	71.76
SGD-L1 (Naive)	30	-2.537	1,032,962	4,528	71.20
SGD-L1 (Clipping + Lazy-Update)	30	-2.538	279,886	585	71.20
SGD-L1 (Cumulative)	30	-2.479	31,986	631	71.40
SGD-L1 (Cumulative + Exponential-Decay)	30	-2.443	25,965	631	71.63

Table 2: NLPBA 2004 Named entity recognition task. Training time and accuracy of the trained model on the test data.

(Tsuruoka, 2009)

Experiments - Named Entity Recognition

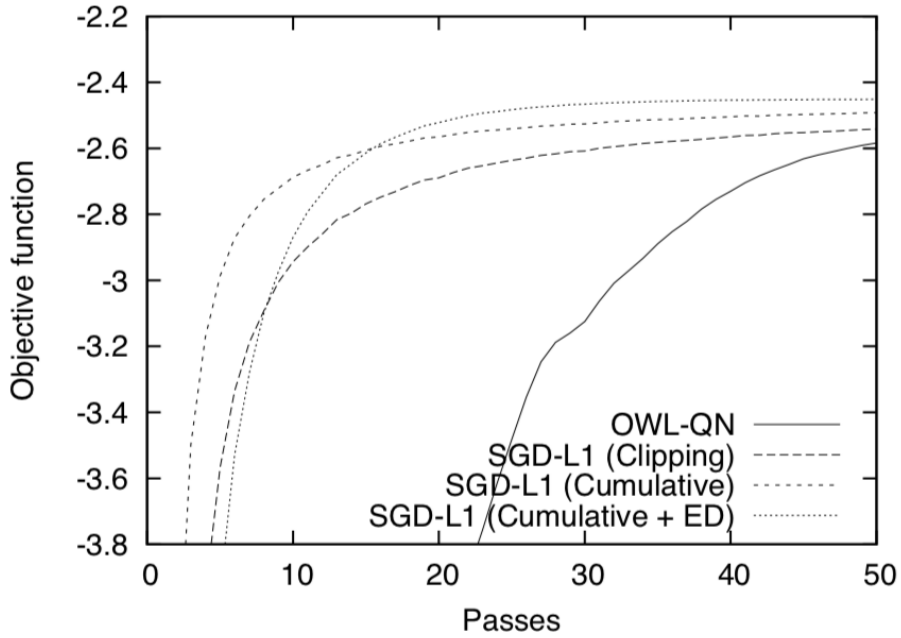


Figure 5: NLPBA 2004 named entity recognition task: Objective.

(Tsuruoka, 2009)

Experiments - POS-Tagging

	Passes	$\mathcal{L}_{\mathbf{w}}/N$	# Features	Time (sec)	Accuracy
OWL-QN	124	-1.941	50,870	5,623	97.16%
SGD-L1 (Naive)	30	-2.013	2,142,130	18,471	97.18%
SGD-L1 (Clipping + Lazy-Update)	30	-2.013	323,199	1,680	97.18%
SGD-L1 (Cumulative)	30	-1.987	62,043	1,777	97.19%
SGD-L1 (Cumulative + Exponential-Decay)	30	-1.954	51,857	1,774	97.17%

Table 3: POS tagging on the WSJ corpus. Training time and accuracy of the trained model on the test data.

(Tsuruoka, 2009)

Experiments - POS-Tagging

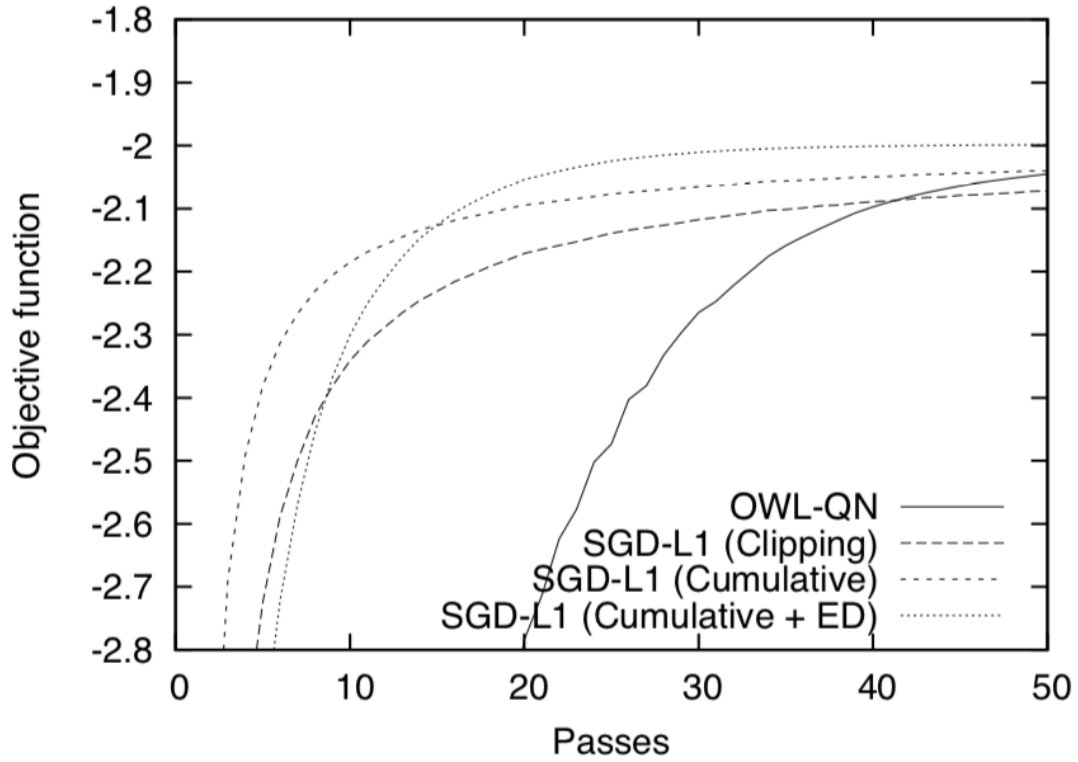


Figure 6: POS tagging task: Objective.

(Tsuruoka, 2009)

Outline

1. Introduction
2. Log-Linear Models in NLP and L1
3. SGD - L1
4. OWL-QN
5. Experiments
6. Criticism
7. Conclusions

Criticism

- Google Scholar references since 2009: 161
- “State-of-the-art” Google Scholar references since 2007: 459

Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty

Y Tsuruoka, J Tsujii, S Ananiadou - … of the Joint Conference of the 47th …, 2009 - dl.acm.org

Stochastic gradient descent (SGD) uses approximate gradients estimated from subsets of the training data and updates the parameters in an online fashion. This learning framework is attractive because it often requires much less training time in practice than batch training algorithms. However, L1-regularization, which is becoming popular in natural language processing because of its ability to produce compact models, cannot be efficiently applied in SGD training, due to the large dimensions of feature vectors and the fluctuations of …

☆ 99 Cited by 164 Related articles All 19 versions

Scalable training of L 1-regularized log-linear models

G Andrew, J Gao - Proceedings of the 24th international conference on …, 2007 - dl.acm.org

The L-BFGS limited-memory quasi-Newton method is the algorithm of choice for optimizing the parameters of large-scale log-linear models with L 2 regularization, but it cannot be used for an L 1-regularized loss due to its non-differentiability whenever some parameter is zero. Efficient algorithms have been proposed for this task, but they are impractical when the number of parameters is very large. We present an algorithm Orthant-Wise Limited-memory Quasi-Newton (OWL-QN), based on L-BFGS, that can efficiently optimize the L 1-regularized …

☆ 99 Cited by 459 Related articles All 9 versions

Criticism

- Experiments section is not detailed:
  - Software and hardware description is shallow.
  - Why did they use the held-out data in the training?
  - Why did they not use the same amount of features for each variant during the experiment?

Outline

1. Introduction
2. Log-Linear Models in NLP and L1
3. SGD - L1
4. OWL-QN
5. Experiments
6. Criticism
7. Conclusions

Conclusions

- As future work, they discussed that it would be interesting to tune the metaparameters of the learning rate in an automatic way. Existing algorithms are too expensive.
- They presented a variant of SGD that can efficiently train L1-regularized log-linear models.
- The authors demonstrated empirically that their training algorithm can produce compact and accurate models much more quickly than OWL-QN.
- Trying to find the shortest/straight path to a minimum is expensive and complex.

Annex 1

$$f(x)=\sum_{n=0}^{\infty}\frac{f^{(n)}(a)}{n!}(x-a)^n$$

$$=f(a)+f'(a)(x-a)+\frac{f''(a)}{2!}(x-a)^2+\frac{f'''(a)}{3!}(x-a)^3+\cdots$$

image source: Stewart, 2011

$$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

$$f'(a)=\lim_{x\to a}\frac{f(x)-f(a)}{x-a}$$

image source: Stewart, 2011

$$\mathbf{H}=\begin{bmatrix}\frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n}\\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2}\end{bmatrix}.\quad J(x_1,x_2,x_3,\cdots x_n)=\begin{bmatrix}\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \cdots & \frac{\partial f_1}{\partial x_n}\\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \cdots & \frac{\partial f_2}{\partial x_n}\\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \cdots & \frac{\partial f_3}{\partial x_n}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \frac{\partial f_n}{\partial x_3} & \cdots & \frac{\partial f_n}{\partial x_n}\end{bmatrix}$$

image source: brilliant.org

image source: chegg.com

Annex 2

A vector d is referred to as a descent direction at x if $f'(x;\mathbf{d}) < 0$

$$f'(x;\mathbf{d}) = \lim_{\alpha \downarrow 0} \frac{f(x+\alpha\mathbf{d} - f(x))}{\alpha}$$

Sign function σ

$$\sigma(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$$

The projection of $x$onto an orthant defined by y

$$\pi(x;y) = \begin{cases} x_i & \text{if } \sigma(x_i) = \sigma(y_i) \\ 0 & \text{otherwise} \end{cases}$$

if $\mathbf{B_k}$is the approximated Hessian matrix of a smooth function f at x^k, and $g^k$is the gradient of f at x^k:

$$Q(x) = f(x^k) + (x - x^k)^T g^k + \frac{1}{2}(x - x^k)^T \mathbf{B_k}(x - x^k)$$

the value that minimizes Q is computed according to $x^* = x^k - \mathbf{H_k} g^k$ where $\mathbf{H_k} = \mathbf{B_k^{-1}}$

a Quasi Newton Method explores

$$x^{k+1} = x^k - \alpha \mathbf{H_k} g^k, \text{ for } \alpha \in (0, \infty)$$

(Andrew, 2007)

and maintains only the curvature information from the most recent points

$$s^k = x^k - x^{k-1}$$

$$y^k = g^k - g^{k-1}$$

Choosing an orthant, pseudo-gradient

$$\diamond_i f(x) = \begin{cases} \partial_i^- f(x) & \text{if } \partial_i^- f(x) > 0 \\ \partial_i^+ f(x) & \text{if } \partial_i^+ f(x) < 0 \\ 0 & \text{otherwise} \end{cases}$$

and

$$\partial_i^{\pm} f(x) = \frac{\partial}{\partial x_i}\ell(x) + \begin{cases} C\sigma(x_i) & \text{if } x_i \neq 0 \\ \pm C & \text{if } x_i = 0 \end{cases}$$

The orthant to explore is the one that contains x^k into which $-\diamond f(x^k)$ leads:

$$\xi_i^k = \begin{cases} \sigma(x_i^k) & \text{if } x_i^k \neq 0 \\ \sigma(-\diamond f(x^k)) & \text{if } x_i^k = 0 \end{cases}$$

Constraint linear search

$$x^{k+1} = \pi(x^k + \alpha p^k; \xi^k)$$

For any sign vector $\xi \in \{-1, 0, 1\}^n$, let us define

$$\Omega_\xi = \{x \in \mathbb{R}^n : \pi(x;\xi) = x\},$$

which is the intersection of an orthant and a plane constraining some coordinates to be zero. Clearly, for all x in Ω_ξ,

Curriculum Learning

Seminar Data Analytics 1

International Master's Program in Data Analytics
University of Hildesheim
Summer Semester 2018

Shabanaz Chamurally

Outline

1. General Information about the paper
2. What is the paper about?
   - Motivation
   - Objectives
3. Contributions
4. A curriculum as a continuation method
   - Continuation method
   - How a curriculum can act as a continuation method
5. Methodology
6. Experiments
7. Potential advantages of Curriculum Learning
8. Relation to other machine learning approaches
9. Conclusion

1. General Information about the paper

- **Authors:** Yoshua Bengio, Jérôme Louradour, Ronan Collobert, Jason Weston
- **The paper:** Published in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.
- **Citation:** Cited 748 times (as of 4/5/2018)
- **Available at:** https://qmro.qmul.ac.uk/xmlui/bitstream/handle/123456789/15972/Bengio%2C%202009%20Curriculum%20Learning.pdf?sequence=1

2. What is the paper about?
- *Motivation*

"Humans and animals learn much better when the examples are not randomly presented but organized in a meaningful order ... Training is highly organized based on an education system and a curriculum which introduces different concepts at different times ... to ease the learning of new abstractions."

From previous research (Elmann, 1993; Rohde & Plaut, 1999; Krueger & Dayan, 2009) which have raised the question:

Can machine learning algorithms benefit from similar training strategy?

2. What is the paper about? - *Objectives*

- Formalize curriculum learning strategies in the context of machine learning
- Explore curriculum learning in various setups
- Show that significant improvements in generalization can be achieved
- Hypothesis that curriculum learning has an effect on the speed of convergence
- In the case of non-criteria, curriculum learning can affect the quality of the local minima
- How a curriculum can be seen as a continuation method

3. Contributions

- Explore cases involving vision and language tasks that show that a curriculum learning can benefits machine learning
- Show how curriculum strategies improved generalization and give rise to faster convergence
- Introduce a hypothesis which may help to explain why curriculum learning can help find a better local minima of a non-convex training criterion
- Explore the relation of curriculum learning with other strategies proposed

4. A curriculum as a continuation method - Continuation Methods

- Continuation methods are optimization strategies for dealing with minimizing non-convex criteria.

- Involves a sequence of training
  1. Starting from one that is easier to optimize
  2. Increasing difficulty in optimization
  3. Ending with training criterion of interest

4. A curriculum as a continuation method - Continuation Methods

- E.g.

Cost Function: $C_\lambda(\theta)$	
Objective	Minimize $\mathbf{C_1}$
Criteria easily optimized	$\mathbf{C_0}$

- Using continuation methods:
  1. Minimize $\mathbf{C_0}$
  2. Gradually increase $\boldsymbol{\lambda}$ while keeping $\boldsymbol{\theta}$ at a local minimum
  3. Till reach the criterion of interest $\mathbf{C_1}$

4. A curriculum as a continuation method - How a curriculum can be a continuous method

- Introduce the idea that a curriculum can also be a sequence of training criteria.

- Each training criteria is associated with a **different set of weights** on the training examples.
  1. Initially, the **weights** favour "**easier**" examples or examples illustrating the **simplest** concepts that can be learned most easily.
  2. The next training criterion is a **slight change** in the **weighting** of example.
  3. The **probability** of sampling **slightly more difficult** examples **increases**.
  4. At the **end** of the sequence, the **reweighting** of the examples is **uniform.**

5. Methodology
- *Notations*

Notations	Description
z	Random variable representing an example for the learner
$P(z)$	Target training distribution from which the learner should learn a function
$0 \leq \lambda \leq 1$	Training step
$0 \leq W_\lambda \leq 1$	Weight applied to example z at step λ
$W_1(z) = 1$	Weight applied to z at the end ($\lambda = 1$)
$Q_\lambda(z)$	Training Distribution at step λ
$H(Q_\lambda)$	Entropy of training distribution at step λ

5. Methodology
- *Description*

- The training distribution at step λ is:

$$\boldsymbol{Q_\lambda(z)} \propto \boldsymbol{W_\lambda(z)}\, \boldsymbol{P(z)} \quad \forall \boldsymbol{z}$$

- Such that $\int \boldsymbol{Q_\lambda(z)dz} = \mathbf{1} \quad \forall \boldsymbol{z}$
- By monotonically increasing $\boldsymbol{\lambda}$, from $\boldsymbol{\lambda} = 0$ to $\boldsymbol{\lambda} = 1$
  - $\boldsymbol{Q_\lambda}$ is called a curriculum if the *entropy of these distributions increases*

$$\boldsymbol{H(Q_\lambda)} < \boldsymbol{H(Q_{\lambda+\epsilon})} \qquad \forall \boldsymbol{\epsilon > 0}$$

  - $\boldsymbol{W_\lambda(z)}$ is also monotonically increasing

$$\boldsymbol{W_{\lambda+\epsilon}(z)} \geq \boldsymbol{W_\lambda(z)} \qquad \forall \boldsymbol{z}, \forall \boldsymbol{\epsilon > 0}$$

5. Methodology

- *Illustration*

1. Q_λ *is a* ***finite set*** *of examples*
2. Increasing λ *means adding* ***new*** *examples to that set*
3. Support of Q_λ ***increases*** *with* λ
   - The **entropy increase** so as to **increase** the **diversity** of training examples

4. The training distributions:
   1. **Start** with a small set of **easy** examples
   2. **Slight** change in $\boldsymbol{W_\lambda(z)}$ that **increase** the probably of more **difficult** examples as they get added.
   3. At the **end** all examples will have an **equal** weight of 1.
   4. **Train** on the target training set.

6. Experiments: *Toy Experiments with a convex Criterion*

- Introducing Gradually More Difficult Examples speeds-up Online Training

Train a Perceptron from artificially generated data by introducing gradually more difficult examples.

Notations:

Target: $y = sign(w'x_{relevant})$	
w	Weight vector of a Bayes classifier
(x, y)	Training pairs
$x_{relevant}$	Relevant input sampled from a Uniform(0,1)
$x_{irrelevant}$	Irrelevant input, not predictive of the target class. Either set to 0 or sampled from a Uniform (0,1)

6. Experiments: *Toy Experiments with a convex Criterion*

- Introducing Gradually More Difficult Examples speeds-up Online Training

Definition of Easy and Difficult examples

- Train the perceptron from **easy** examples to most **difficult** ones.
  - Easy examples: with **all** irrelevant inputs **zeroed out**
  - Difficult examples: with **none** of the irrelevant inputs **zeroed out**.

- Another way to differentiate between easy and difficult
  - By the margin $yw'x$.
  - Easy examples: **larger** margin
  - Difficult examples: **smaller** margin values

- Increase the difficulty levels in each step by increasing the weights of the examples.

6. Experiments: *Toy Experiments with a convex Criterion*

- Introducing Gradually More Difficult Examples speeds-up Online Training

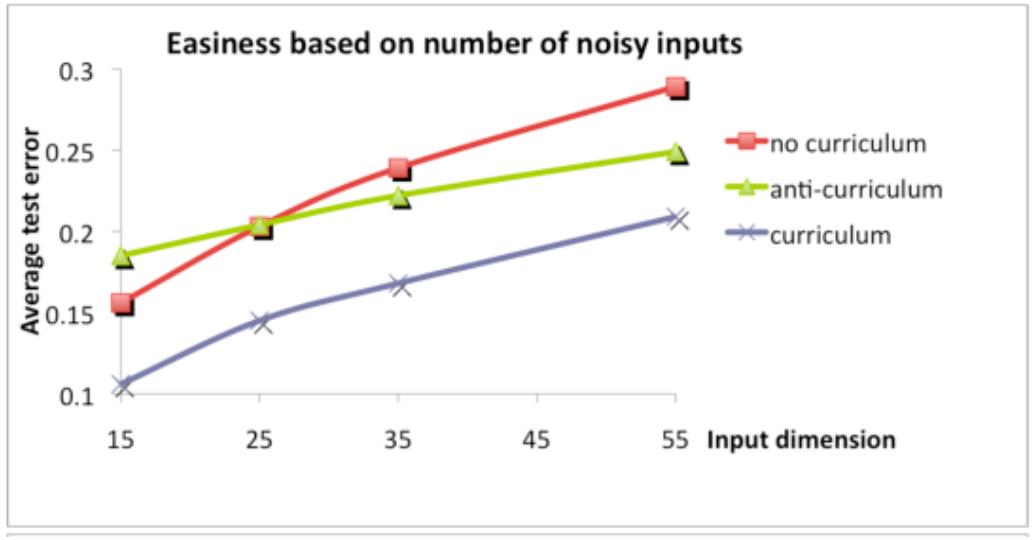
The Experiment

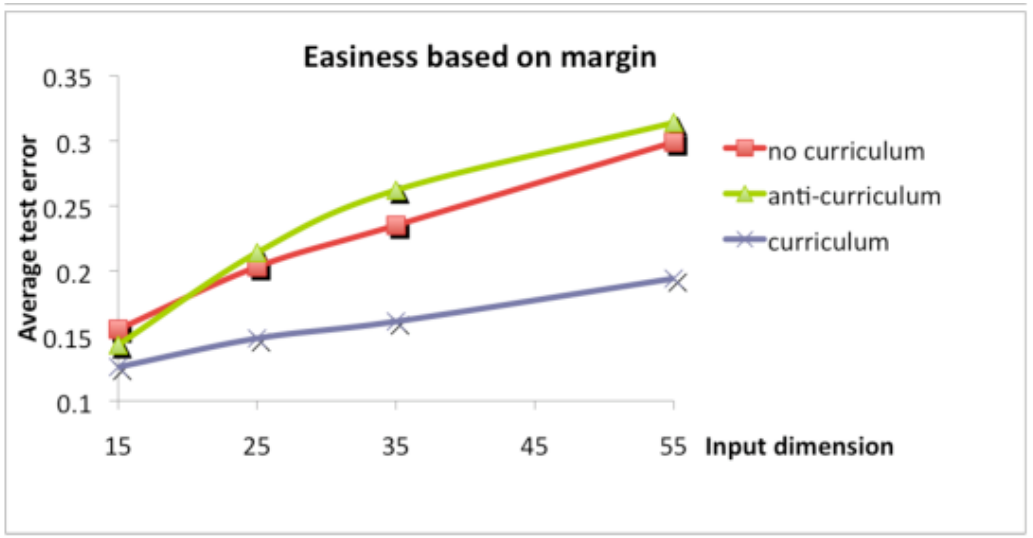
- Train the perceptron with 200 examples
  - i.e. 200 perceptron updates
- Mesure generalization error at the end of training
- Averaged error across 500 repetitions
  - from different initial conditions
  - and different random sampling of training examples

6. Experiments: *Toy Experiments with a convex Criterion*

- Introducing Gradually More Difficult Examples speeds-up Online Training

Results





- Error rate between the curriculum strategy and the no-curriculum strategy are statistically significant.

6. Experiments: On Shape Recognition

Objective

- Classify geometrical shapes into 3 classes:
  - Rectangle
  - Ellipse
  - Triangle

Input

- 32 x 32 grey-scale image

Datasets

- BasicShapes
- GeomShapes

6. Experiments: On Shape Recognition

Definition of Easy examples

- BasicShapes:
  - Squares
  - Circles
  - Equilateral triangles

6. Experiments: On Shape Recognition

Definition of Difficult examples

- GeomShapes:
  - Rectangles
  - Ellipses
  - Triangles

6. Experiments: On Shape Recognition

Difference between BasicShapes and Geomshapes:

- BasicShapes exhibit less variability in shape

Similarities between BasicShapes and Geomshapes:

- Variabilities in both:
  1. Object position
  2. Size
  3. Orientation
  4. Grey levels of the foreground and background

6. Experiments: On Shape Recognition

The Experiment

- Carried out on a multi-layer neural network with 3 hidden layers

- The curriculum consists in a 2-step schedule:
  1. Perform gradient descent on the BasiShapes training set, until “switch epoch” is reached
  2. Then perform gradient descent on the GeomShapes training set.

- An epoch is a stochastic gradient descent pass through a training set of 10000 examples.

6. Experiments: On Shape Recognition

The Experiment

- After switching to the target training distribution, training continues until:
  1. 256 epochs
  2. Or validation set error reaches a minimum (early stopping)

- Used 10000 examples in both sets, 5000 examples for validation and 5000 for testing.

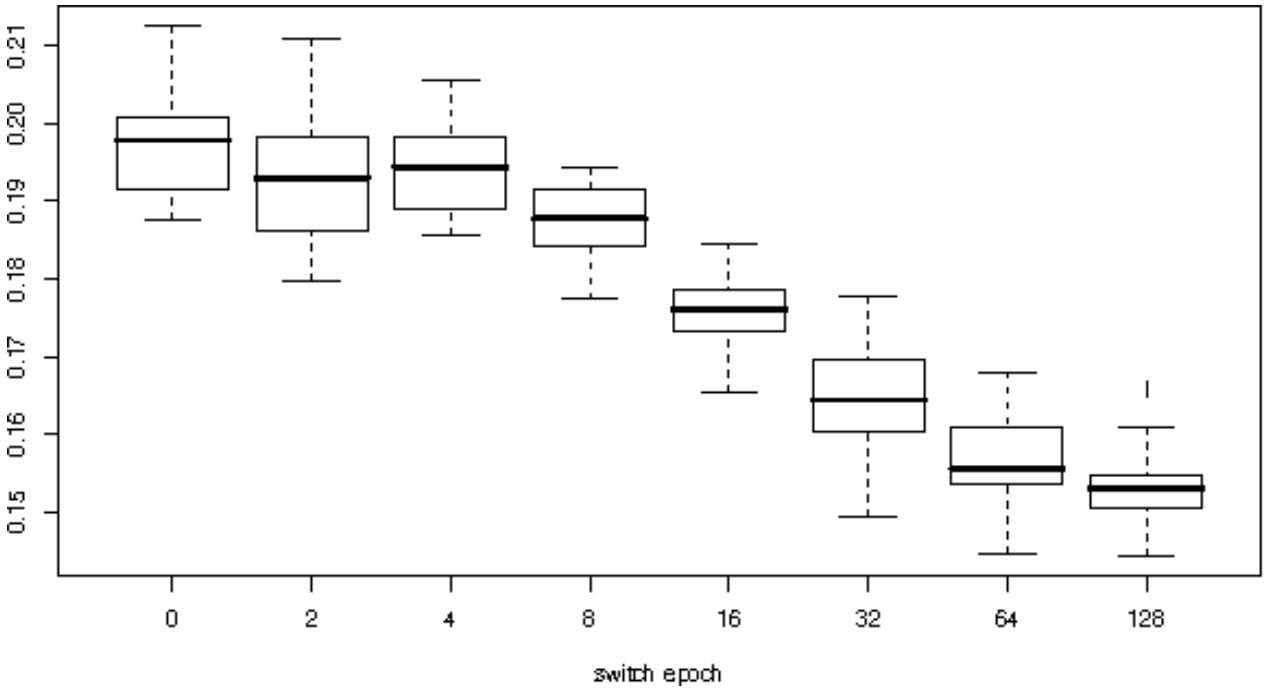
6. Experiments: On Shape Recognition

Results

- Generalization error is evaluated on GeomShapes test set
- **Baseline**: Training only on GeomShapes training set (for the same number of training epochs).
  Switch epochs = 0
- Distribution test errors:
  1. Over 20 different random seeds
  2. Different values of switch epochs
     - 0: The baseline with no curriculum
     - Powers of 2 until 128

6. Experiments: On Shape Recognition

Results



- Figure shows the distribution test error after early stopping.
- Each box corresponds to 20 seeds for initializing the parameters.
- Horizontal line: Median (50th percentile)
- Borders: 25th and 75th percentile
- End of bars: 5th and 95th percentile

Observation:

- Best generalization is obtained by doing a 2 stage curriculum.

6. Experiments: On Shape Recognition

Potential issue

Curriculum-trained model overall saw more examples than the no-curriculum examples.

To eliminate the explanation that better results are obtained with the curriculum-trained model:

- Trained a no-curriculum with union of BasicShapes and GeomShapes:
  - Final test error significantly worse than with the curriculum
- Trained only with BasicShapes:
  - Yielded poor results

6. Experiments: On Language Modeling

Objective

- Train a language model for predicting the best word which can follow a given context of words in a correct English sentence.

- Compute a score for the next word that will have a large rank compared to the scores of other words
  - Given any fixed size window of text **s**, a language model $f(s)$ produces a score for these windows of text.
  - The score of a correct windows **s** should be larger, by a margin of 1, than any other word sequence s^w where the last word has been replaced by another word **w**.

6. Experiments: On Language Modeling

Ranking Loss Function

The ranking loss over sequences **s** sampled from a dataset $\boldsymbol{S}$ of valid English text windows:

$$C_s = \sum_{w \in \mathcal{D}} \frac{1}{|\mathcal{D}|} C_{s,w} = \sum_{w \in \mathcal{D}} \frac{1}{|\mathcal{D}|} \max\left(0, 1 - f(s) + f(s^w)\right)$$

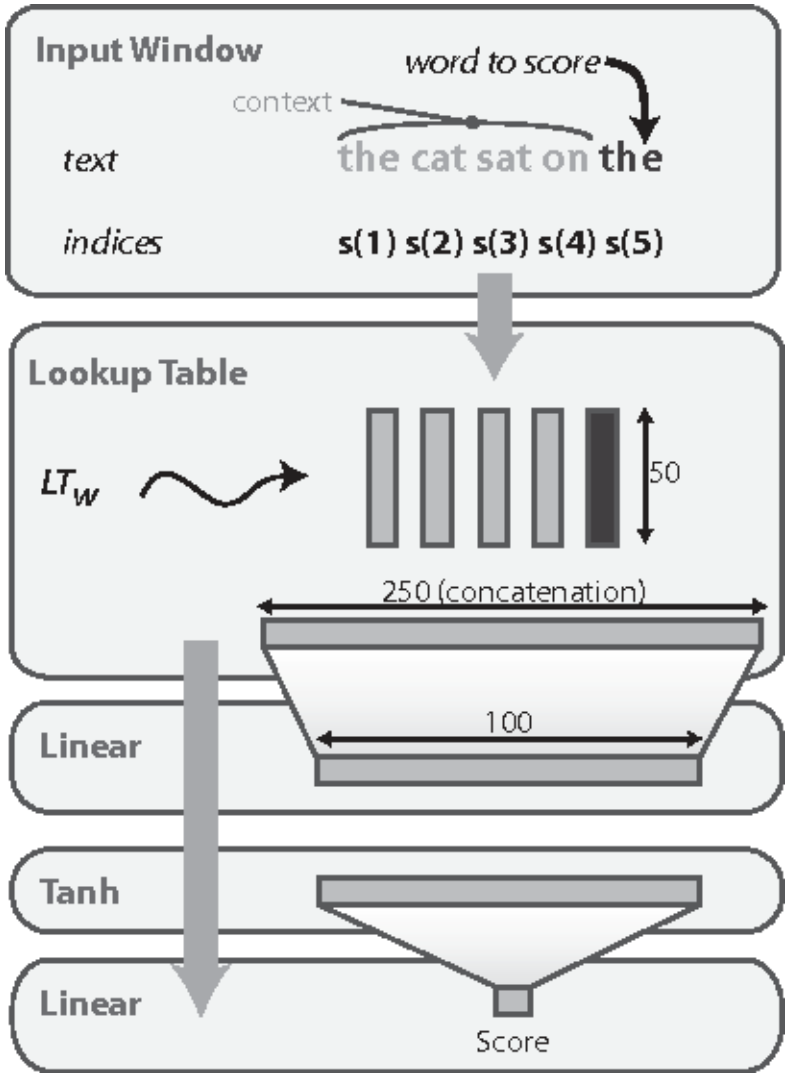
Where $\mathcal{D}$ is the word vocabulary.

For each word sequence **s**

- $f(s)$ and $f(s^w)$ is computed
- And the gradient of $\max\left(0, 1 - f(s) + f(s^w)\right)$

6. Experiments: On Language Modeling

Architecture of language model



- The first layer extracts features for each word in a sentence.

Map words into real-valued vectors

- The second layer extracts features from the sentence treating it as a sequence with local and global structure (i.e., it is not treated like a bag of words).
  - Each word i is embedded into a d-dimensional space using a look-up table LTw (.)
  - An input window $\{s_1, s_2, ... s_n\}$ of n words in D is transformed into a series of vectors $\{W_{s1}, W_{s2}, ... W_{sn}\}$ by applying the look-up table to each of its words

- Vectors Obtained by the look-up table layer are then concatenated and fed to a classical linear layer.
- Following the 3 above layers is a non-linearity, tanh(.)
- The score of the language model is finally obtained after a linear layer.

6. Experiments: On Language Modeling

The Experiment

- Training set S is all the possible windows of text of size = 5 from Wikipedia
- Obtained 631 million of windows
- The curriculum is chosen to grow the vocabulary size:
  1. **First pass** over Wikipedia was performed using the **5000 most frequent** words in the vocabulary
  2. Which is then **increased by 5000 words** at each subsequent pass through Wikipedia
  3. At each pass, any window of text containing a word **not** in the considered dictionary was **discarded**.
- The training set is thus increased after each pass through Wikipedia.

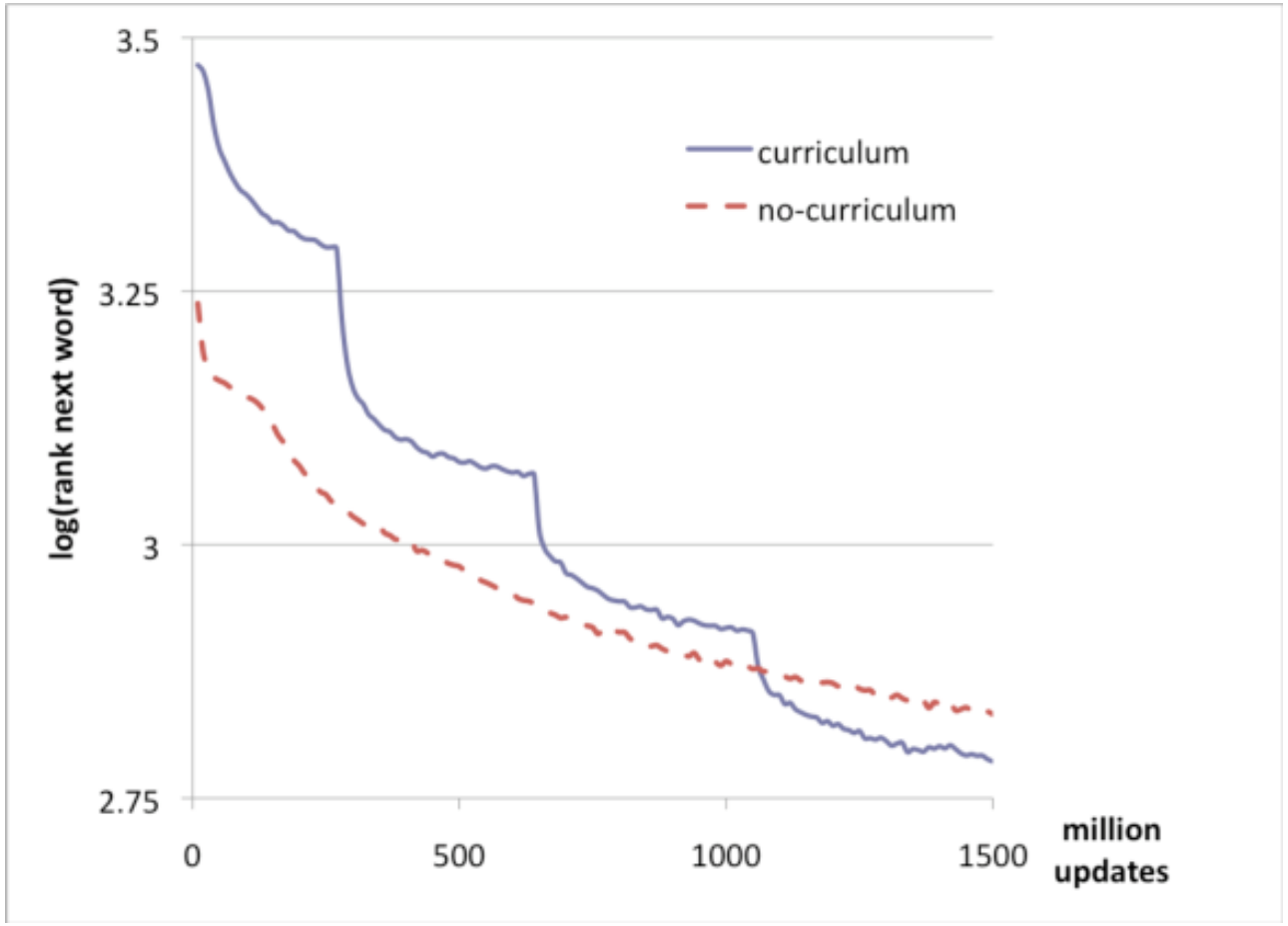
6. Experiments: On Language Modeling

Results

- The ranking loss is **minimized** using stochastic gradient descent.
- Compare against no curriculum
  - Where the network is trained using **final desired vocabulary** size of 20,000.
- Evaluation criterion:
  - Is the average of the **log of the rank** of the last word in each test window
  - Taken in a test set of 10,000 windows not seen during training
  - With words from the most 20,000 frequent ones.

6. Experiments: On Language Modeling

Results



- The log rank on the target distribution with the curriculum strategy **crosses the error** of the no-curriculum strategy after about 1 billion updates.
  - Shortly after switching to the target vocabulary size of 20 000 words and the difference keeps increasing afterwards.
- The final test set average log-ranks are 2.78 and 2.83 respectively and the difference is **statistically significant**.

7. Potential advantages of a curriculum strategy

- **Faster training time** in the online setting because the learner **wastes less time** with noisy or harder to predict examples, when the learner is not ready yet.

- Guiding training towards **better local minima** in parameter space, specifically useful for non-convex methods.

- A curriculum can be seen as a **continuation method**

8. Relation to other machine learning approaches

- **Unsupervised pre-training procedures**: An appropriate curriculum strategy acts to **help** the training process
  - **faster convergence** to better solutions and both methods have a **regularizing effect** and **lower** the generalization error for the same training error
- **Active Learning**: The learner **benefits most** from focussing on “interesting examples” which are close to the learner's frontier of knowledge instead of focussing on examples that are “too easy” or “too hard”. The learner could succeed to capture and gradually expand its border

8. Relation to other machine learning approaches

- **Boosting algorithm**: **Difficult** algorithm are **gradually** emphasized.
  - However, a curriculum starts by focussing on the easier examples rather than a uniform distribution over trained set
- **Transfer and lifelong learning**: Initial tasks are used to **guide** the learner so that it will **perform better** on the final task learning to improve generalization.
  - Curriculum learning **adds** the notion of **guiding** the optimisation process to **converge better** or towards better local minima

9. Conclusion

- Started with the question "**Can machine learning algorithm benefit from a curriculum strategy?**"
- From the experiments, it is **plausible** that some curriculum strategies work better than others
- Better results could be obtained with more appropriate curriculum strategies
- It would be useful to understand general principles that makes some curriculum work better than others and why.

Criticism

- In the experiments concerning the convex criterion:
  - Anti-curriculum performs better than no-curriculum.
  - Given that curriculum learning is modelled on the idea that learning benefits when examples are presented in order of increasing difficulty, one would expect anti-curriculum to perform worse.

- The results of the experiments were all compared to the results of a no-curriculum model:
  - The paper could have explored how a curriculum strategy can perform better than existing machine learning approaches.

Combined Regression & Ranking

Presentation on the paper by D. Sculley (2010)

Daniela Thyssens

Outline

1. General information about the paper
2. What is the paper about?
3. Methodology
   - Learning in general & Notation
   - Minimizing Loss & Loss Functions
   - Overall Optimization Problem & Algorithm
   - Learning with Stochastic Gradient Descent
4. Experiments
   - Setup and Results for each experiment
5. Future work
6. Conclusion and own thoughts

   Appendix

1. General information about the paper

- **The author**: D. Sculley is a Manager, Tech Lead and Staff Software Engineer at Google, Inc. (Research at Google)

- **The paper**: Published in the Proceedings of the 16th ACM SIGKDD international conference
  - The conference is hosted by SIGKDD (specialised interest group on knowledge discovery and data mining)

What is the paper about?

- The paper argues that combining regression and ranking methods is beneficial for many real world tasks.
  - General throwback: Doesn’t good regressing performance already include accurate ranking?
  - On the other hand one could ask why not (combining both)?

- What is fundamental about it? (for text analysis and application)
  - Typical supervised machine learning problem
  - Easy to implement algorithm – stochastic gradient descent (popular method)

What is the paper about?
- *Goal of the paper*

- Claim:

  The combined method achieves the same performance as (& sometimes outperforms) regression- and ranking-only methods for large scale experiments

- The combined regression and ranking (CRR) algorithm gives optimized regression and ranking results simultaneously

- CRR as an “easy to implement” – optimization algorithm
  - Through the use of stochastic gradient descent

- Efficient for the use of large-scale data sets

What is the paper about?
- *Motivation: Ranking vs. Regression*

- Case: extreme minority classes
  - example, the query (=target) for a search on the web is very specific.
  - Disproportionately high amount of target values y=0 → easy to predict
  - But almost useless trying to optimize ranking
    - Because features of minority class (y=1) are treated as noise and hence mainly ignored, there's a high probability of them being ranked "not relevant"

  → we still need good regression **and** good ranking for web-search results

- Case: order-preserving transformation in ranking
  - Invariance of pairwise loss

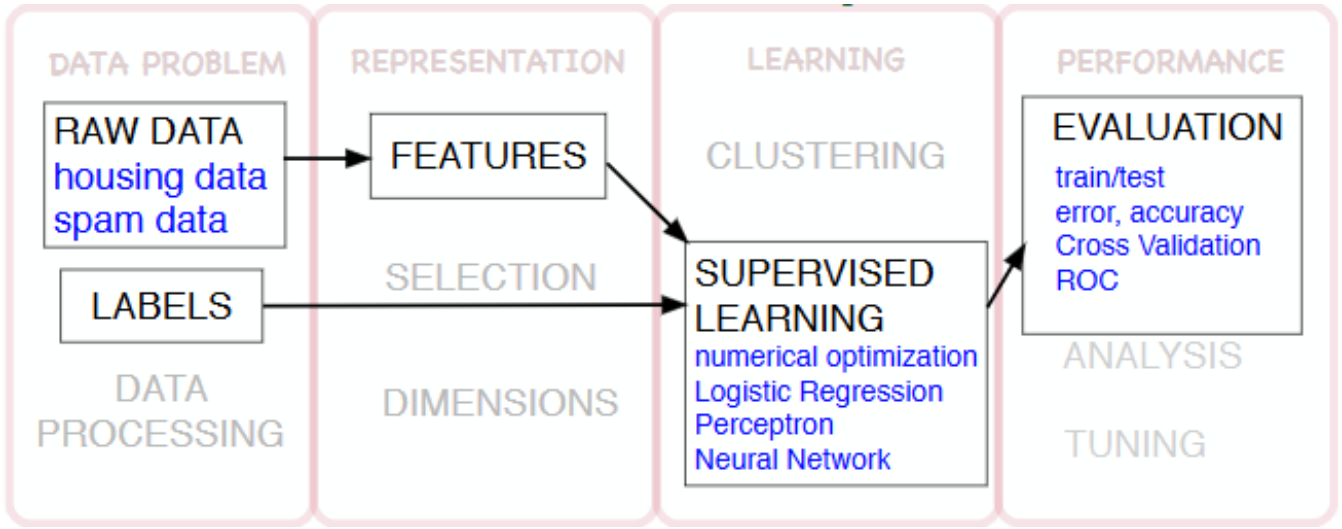
What is the paper about?
- *Motivation: Why combining?*

- "Many real world tasks *require* the achievement (…)" of both
  - (D. Sculley, Combined Regression and Ranking, 2010, p.1)

- Arguments:
  - Perfect regression results do not imply good ranking (extreme minority class)
  - Don't loose out on performance for both goals
  - Overall enhancement of the regression result (rare events)

- Examples of where both ranking and regression are useful:
  - Text classification (Experiment 1)
  - Ranking concerning Web-search results (Experiment 2)
  - Click-Prediction for sponsored ads – commercial application (Experiment 3)

Methodology
-*Learning and testing*



- Here: Labels → y
  Features → $\boldsymbol{x}$
  Superv. Learning → optimization – minimizing loss functions (through stochastic gradient descent)

Source: https://www.semanticscholar.org

Methodology
-Notation & Preliminaries

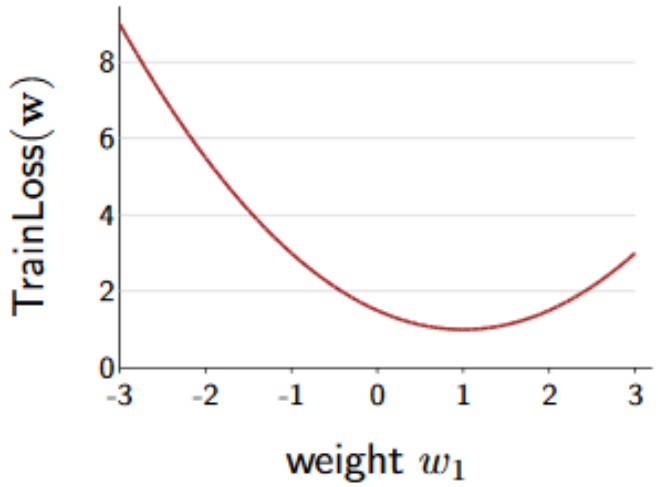
- "Input" space: $\boldsymbol{X}$ and "Output" space: $\boldsymbol{Y}$
- Training set **D** drawn i.i.d. from $\mu(z)$ on $Z = X \times Y$

	Dataset	Variables	Training Examples	Model **w**
Regression	D	**x**: Feature Vector y: Label q: Identifier	Tuples (x,y,q)	predicts target value y given **x** through $f(\boldsymbol{w},\boldsymbol{x})$ → *supervised regression*
	expanded version ↓			
Ranking	D (expanded by P candidate pairs)	**a,b**: Candidate Pairs y: Label q: Identifier	Tuple Pairs (a,y_a,q_a) & (b,y_b,q_b)	optimizes the pairwise objective function $f(\boldsymbol{w}, \boldsymbol{a}-\boldsymbol{b})$ → *supervised ranking*

- $y_a > y_b$ implies **a** is preferred to/ranked better than **b**
- q is the identifier for the query shard – if there is no query shard : $q = 1$

Methodology
-minimizing Train-Loss

$\mathbf{w} \in \mathbb{R}$



$\mathbf{w} \in \mathbb{R}^2$

[gradient plot]

- Find **w** such that TrainLoss(w) is minimized
- Note: the plot is one dimensional, but **w** is actually R-dimensional
  → in general, not possible to set w to a single value that makes every example have low loss.

Source: https://web.stanford.edu/class/cs221/lectures/learning1.pdf

Methodology
-Loss functions measuring TrainLoss

- Supervised regression:

$$L(w, D) = \frac{1}{|D|} \sum_{(x,y,q) \in D)} l(y, f(w, x)) \longrightarrow \text{(aggregated loss)}$$

- Supervised ranking:

$$L(w, P) = \frac{1}{|P|} \sum_{((a,y_a,q_a),(b,y_b,q_b) \in P)} l(t(y_a - y_b), f(w, a - b))$$

- Function type: standard convex
  - Squared loss → classical LMS **regression** method

$$l(y, y') = (y - y')^2$$

  - Logistic loss → predicting real valued probability scores (**ranking**)

$$l(y, y') = y \log(y') + (1 - y) \log(1 - y')$$

Methodology
-Loss function Example

- Squared loss Example :
  - $l\big(y, f(w,x)\big) = (y - f(w,x))^2$ Where $f(w,x) = \boldsymbol{w} \cdot \boldsymbol{x}$
- Example: $\boldsymbol{w}$=[2,-1], $\boldsymbol{x}$=[2,0] and y= -1
- Then the loss is 25
  - $[(-1) - ((2 \times 2) + ((-1) \times 0))]^2 = 25$

Methodology
-*Optimization Problem*

- Linearly combining aggregate losses into one min. problem:

$$\min_{w \in \mathbb{R}} \alpha L(\boldsymbol{w}, D) + (1-\alpha) L(\boldsymbol{w}, P) + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 \quad (\alpha \in [0,1])$$

- (combined) **Empirical Risk Minimization (ERM)** with **L2-regularization**:
  - Linear model represented by $\boldsymbol{w}$: convex data fitting term + “regularizer”

Methodology
-*Optimization Problem*

- Linearly combining aggregate losses into one min. problem:

$$\min_{w \in \mathbb{R}} \alpha L(\boldsymbol{w}, D) + (1-\alpha) L(\boldsymbol{w}, P) + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 \quad (\alpha \in [0,1])$$

- (combined) **Empirical Risk Minimization (ERM)** with L2-regularization
  - Solution should:
    - **Generalize (convergence with increasing number of examples - predictive)**

Methodology
-*Optimization Problem*

- Linearly combining aggregate losses into one min. problem:

$$\min_{w \in \mathbb{R}} \alpha L(\boldsymbol{w}, D) + (1-\alpha) L(\boldsymbol{w}, P) + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 \quad (\alpha \in [0,1])$$

- (combined) **Empirical Risk Minimization (ERM)** with L2-regularization
  - Solution should:
    - **Generalize (convergence with increasing number of examples - predictive)**
    - **Be Stable (low complexity)**

Methodology
-*Optimization Problem*

- Linearly combining aggregate losses into one min. problem:

$$\min_{w \in \mathbb{R}} \alpha L(\boldsymbol{w}, D) + (1-\alpha) L(\boldsymbol{w}, P) + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 \quad (\alpha \in [0,1])$$

- (combined) **Empirical Risk Minimization (ERM)** with L2-regularization
  - Solution should:
    - **Generalize (convergence with increasing number of examples - predictive)**
    - **Be Stable (low complexity)**
      - λ is the regularization parameter
        - prevents the model from predicting noise
        - penalising additional information that doesn't reduce the error of the model significantly

Methodology
-*CRR Algorithm*

- Outputs **w** where the objective achieves min. value
- Sampling from P
- **Stochastic gradient descent** - efficiently reduces training times
  - Stochastic Gradient Step for squared loss:

$$(1 - \eta_i \lambda)\mathbf{w}_{i-1} + \eta_i \, \boldsymbol{x}(y - \langle \mathbf{w}_{i-1}, \boldsymbol{x} \rangle)$$

  - Stochastic Gradient Step for logistic loss :

$$(1 - \eta_i \lambda)\mathbf{w}_{i-1} + \eta_i \, \boldsymbol{x}(y - \frac{1}{1+e^{-\langle \mathbf{w}_{i-1}, \boldsymbol{x} \rangle}})$$

Algorithm 1 Combined Regression and Ranking.

Given: tradeoff parameter α, regularization parameter λ, training data D, iterations t

$\mathbf{w_0} \leftarrow \oslash$
for $i = 1$ to t **do**
  pick z uniformly at random from $[0, 1]$
  if $z < \alpha$ **then**
    $(\mathbf{x}, y, q) \leftarrow \text{RandomExample}(D)$
  else
    $((\mathbf{a}, y_a, q), (\mathbf{b}, y_b, q)) \leftarrow \text{RandomCandidatePair}(P)$
    $\mathbf{x} \leftarrow (\mathbf{a} - \mathbf{b})$
    $y \leftarrow \text{t}(y_a - y_b)$
  end if
  $\eta_i \leftarrow \frac{1}{i\lambda}$
  $\mathbf{w}_i \leftarrow \text{StochasticGradientStep}(\mathbf{w}_{i-1}, \mathbf{x}, y, \lambda, \eta_i)$
end for
return $\mathbf{w}_t$

Source: D. Sculley (2010)

Methodology
-Learning with Stochastic Gradient Descent

Stochastic Gradient Descent

- initialize $w = 0$
- repeat
  - randomly choose training point $(x_i, y_i) \in \mathcal{D}_n$
  - $w \leftarrow w - \eta \underbrace{\nabla_w \ell(f_w(x_i), y_i)}_{\text{Grad(Loss on i'th example)}}$
- until stopping criteria met

Source: https://www.semanticscholar.org

Methodology
Stochastic Gradient Step

- Deriving the gradient step for squared loss: $(1-\eta_i\lambda)\mathbf{w}_{i-1} + \eta_i\, \boldsymbol{x}(y - \langle \mathbf{w}_{i-1}, \boldsymbol{x}\rangle)$

- The prediction function: $f(x, w) = \langle \mathbf{w}_{i-1}, \boldsymbol{x}\rangle$

  - $\mathbf{w} \leftarrow \mathbf{w}_{i-1} - \eta_i \{-\lambda \mathbf{w}_{i-1} + \nabla_w [(y - f(\boldsymbol{x}, \boldsymbol{w}))^2]\}$
  - $\mathbf{w} \leftarrow \mathbf{w}_{i-1} - \eta_i \left\{-\lambda \mathbf{w}_{i-1} + \frac{\partial}{\partial w} [(y - f(\boldsymbol{x}, \boldsymbol{w}))^2]\right\}$
  - $\mathbf{w} \leftarrow \mathbf{w}_{i-1} - \eta_i \{-\lambda \mathbf{w}_{i-1} + \boldsymbol{x}[y - \langle \mathbf{w}_{i-1}, \boldsymbol{x}\rangle] \}$
  - $\mathbf{w} \leftarrow (1 - \lambda\boldsymbol{\eta})\mathbf{w}_{i-1} - \eta_i \{\boldsymbol{x}[y - \langle \mathbf{w}_{i-1}, \boldsymbol{x}\rangle] \}$
  - $\mathbf{w} \leftarrow (1-\eta_i\lambda)\mathbf{w}_{i-1} + \eta_i\, \boldsymbol{x}(y - \langle \mathbf{w}_{i-1}, \boldsymbol{x}\rangle)$

- i denotes the iteration step hence, for each iteration SGD picks **random** training data $(\boldsymbol{x}, y)$ and updates $\boldsymbol{w}$ <u>on this example only</u>.

Methodology
-Evaluation / Performance Metrics

- Regression metric:

$$\boldsymbol{MSE} = \frac{1}{|D|}\sum_{(x,y,q)\in D_t)}(y - f(\boldsymbol{w}, \boldsymbol{x}))^2$$

- Ranking metrics:

$$\mathbf{1} - \boldsymbol{AUC\ Loss}$$

Probability that the model ranks a randomly chosen observation with $y = 0$ (by mistake) higher than a randomly chosen observation with $y = 1$

Methodology
-Performance Metrics continued

- Ranking metrics:
  - $\boldsymbol{MAP} = arithmetic\ mean\ of\ \boldsymbol{AP}(\boldsymbol{q})$

$$\boldsymbol{AP}(\boldsymbol{q}) = \frac{1}{r_q}\sum_{n=1}^{|Q|} P@n * r(n) \qquad where \quad P@n = \frac{r_n}{n}$$

    - Relevant vs. irrelevant examples across query shard set
    - $\boldsymbol{r}(\boldsymbol{n})$ returns 1 if example n is relevant and 0 otherwise.

  - $\boldsymbol{NDCG} = Z_n \sum_{j=1}^{n} \frac{2^{\boldsymbol{y_j}} - 1}{\log(1+j)}$

    - quality of ranking for multiple levels of relevance

Experiments

- Benchmark: Checking effectiveness of CRR against natural competitors: regression- and ranking-only.
  - On one hand, a good strategy to prove the “best of both” argument.
  - On the other hand, fairly easy to show that ranking-only method is outperformed by CRR when it comes to tasks where regression takes a considerable role and vice versa for regression-only tasks.
- 2/3 experiments use publicly available benchmark datasets for text-mining (RCV1) and document ranking (LETOR).
- The third experiment, regarding click prediction uses a proprietary dataset.

Experiments
-RCV1 Experiment

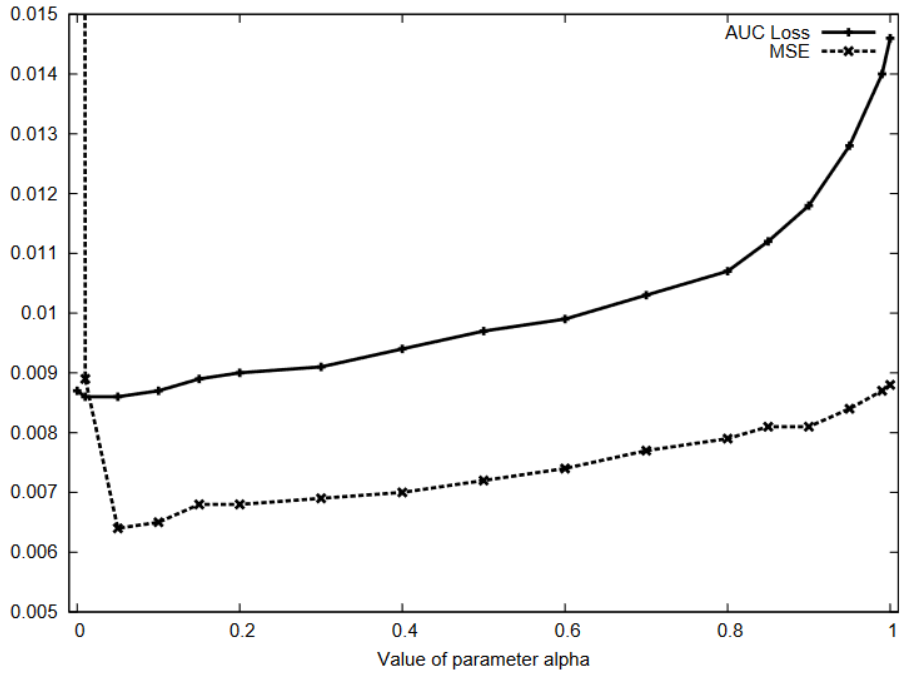
- RCV1 (or RCV1-v2/LYRL2004) is a text categorization test collection distributed as on-line appendices.
  - Large dataset, sparsity, different class distribution levels
  - Use larger split (781,265 examples) of dataset for training, and the smaller (23,149) for testing.
  - Logistic loss
  - 1,000,000 stochastic gradient descent steps
  - 40 topics chosen out of 103
- Goal:
  - predict excellent **ranking** (relevant vs. irrelevant topic identification)
  - and get good overall topic-based relevance scores (**regression** part)

Experiments
-RCV1 Experiment - Results

- CRR gives desired result: Best of both performance
  - **Best MSE and best AUC Loss** in **16/40** topic-identifications (*)
  - In 19/40 cases CRR achieves best performance in one metric and in range of 0.001 of the best performance of the other metric.
  - For all **40/40** cases **CRR achieves best performance in at least one metric** (considering that α is set to 0.5)
  - **CRR performs notably good** on **the MSE** metric **concerning minority class distributions** (<10%) and outperforms regression-only in 20/33 cases.
    → adding ranking information to minority class regression problems enhances regression performance.
  - CRR benefits diminish as class-distribution → more evenly balanced

Experiments

-RCV1 Experiment -Results



Task	% positive	Regression AUC Loss	Regression MSE	Ranking AUC Loss	Ranking MSE	CRR AUC Loss	CRR MSE	
E141	0.05%	**0.000**	0.001	**0.000**	0.293	**0.000**	**0.000**	*
GOBIT	0.06%	0.002	**0.001**	**0.001**	0.162	0.002	**0.001**	
E61	0.06%	0.002	**0.001**	**0.001**	0.320	**0.001**	**0.001**	*
GTOUR	0.10%	0.030	**0.001**	**0.005**	0.245	**0.005**	**0.001**	*
C331	0.13%	0.003	**0.001**	**0.001**	0.205	**0.001**	**0.001**	*
E143	0.15%	**0.001**	**0.001**	**0.001**	0.296	**0.001**	**0.001**	*
G152	0.15%	0.005	**0.001**	**0.003**	0.239	**0.003**	**0.001**	*
G155	0.16%	0.007	0.002	**0.004**	0.223	**0.004**	**0.001**	*
E411	0.17%	**0.002**	0.002	**0.002**	0.289	**0.002**	**0.001**	*
C313	0.18%	0.047	**0.002**	**0.014**	0.281	0.016	**0.002**	
E311	0.19%	**0.001**	0.002	**0.001**	0.311	**0.001**	**0.001**	*
C32	0.19%	0.019	**0.002**	**0.012**	0.180	0.013	**0.002**	
G157	0.19%	**0.001**	0.002	**0.001**	0.254	**0.001**	**0.001**	*
C16	0.21%	0.022	**0.002**	**0.012**	0.234	0.013	**0.002**	
GWELF	0.22%	0.010	**0.002**	**0.005**	0.236	0.006	**0.002**	
E513	0.23%	0.004	0.002	**0.003**	0.300	**0.003**	**0.001**	*
E14	0.28%	0.008	0.003	**0.003**	0.281	0.004	**0.002**	
C173	0.33%	0.005	0.003	**0.004**	0.237	**0.004**	**0.002**	*
E121	0.41%	0.007	0.004	**0.004**	0.261	0.005	**0.003**	
GENT	0.46%	0.014	**0.004**	**0.008**	0.126	**0.008**	**0.004**	*
C34	0.52%	0.018	0.005	**0.011**	0.231	0.012	**0.004**	
GHEA	0.85%	0.007	0.008	**0.005**	0.140	0.006	**0.006**	
C183	0.87%	0.013	0.008	**0.009**	0.275	0.010	**0.006**	
GDEF	1.01%	0.015	0.009	**0.009**	0.208	**0.009**	**0.007**	*
C42	1.48%	0.009	0.010	**0.006**	0.242	0.007	**0.008**	
E211	1.76%	0.013	0.011	**0.010**	0.245	**0.010**	**0.009**	*
E51	2.77%	0.025	0.019	**0.019**	0.280	0.021	**0.016**	
M12	3.16%	0.010	0.015	**0.008**	0.288	0.009	**0.014**	
C24	3.98%	0.031	0.027	**0.025**	0.157	0.026	**0.024**	
GDIP	4.34%	0.019	0.023	**0.017**	0.188	0.018	**0.022**	
M13	6.89%	**0.007**	**0.018**	**0.007**	0.221	**0.007**	**0.018**	*
GPOL	7.11%	0.021	**0.031**	**0.020**	0.175	0.021	**0.031**	
C152	8.34%	0.026	0.036	**0.023**	0.178	0.024	**0.035**	
C151	10.22%	0.010	**0.024**	**0.009**	0.188	**0.009**	0.025	
M14	10.98%	0.005	0.021	**0.004**	**0.115**	**0.004**	0.022	
ECAT	14.90%	0.033	0.054	**0.030**	0.188	0.031	**0.053**	
C15	18.05%	**0.013**	**0.036**	**0.013**	0.132	**0.013**	0.037	
MCAT	25.41%	0.011	**0.039**	**0.010**	0.113	**0.010**	0.043	
GCAT	30.11%	**0.012**	**0.043**	**0.012**	0.062	**0.012**	0.046	
CCAT	46.59%	**0.022**	**0.067**	**0.022**	0.073	**0.022**	0.070	

Source: D. Sculley (2010)

Experiments
-LETOR Experiment

- Public benchmark data: LETOR 4.0 learning to rank data set
  - 2 tasks in the data set (MQ2007,MQ2008) drawn from TREC challenge data sets in information retrieval
  - Relevance labels (such as star indication): {0,1,2}
  - Querysharded + dense information retrieval features (TF-IDF similarities, PageRank,…)
  - Commonly used for ranking-only purposes.
- Goal: propose relevant documents to the end user
  - Obtain **accurate ranking**
  - **Predicting actual relevance** value of a document (**regression**)
    - Why?
      – Search engines could annotate (query-independent) relevance scores and quality thresholds for documents.

Experiments
-LETOR Experiment - Results

- CRR does not give the desired result: **Not** Best of Both performance.
  - Performs **as good as ranking-only**
  - **MSE** metric performs **better for CRR compared to ranking-only**, but only approaches the one for regression-only.
- Overall, the performance is not too bad, considering that class distribution is fairly balanced.

Experiments
-LETOR Experiment - Results

Method	MQ2007 MAP	MQ2007 NDCG	MQ2007 MSE	MQ2008 MAP	MQ2008 NDGC	MQ2008 MSE
Regression	0.456	0.492	**0.182**	0.464	0.476	**0.144**
RankSVM (baseline [18])	0.464	0.497	-	0.470	0.483	-
Ranking	**0.466**	**0.500**	0.498	**0.479**	**0.490**	0.850
CRR	0.465	0.498	0.214	**0.479**	0.489	0.229

Source: D. Sculley (2010)

Experiments (commercial application) *-Click Prediction Experiment*

- Advertisement Impression - Data from live traffic on Google Search
  - Several million advertisement impressions
  - Each example: high-dim. **Representation of ad** together **with label 1 (clicked) or 0 (not clicked)**

- Goal: which advertisement to show when user enters a search query?
  - **Usefulness of CRR:**
    - Real-time auctions for ad-placements are ranked based on: bid*(price(CTR)) (**ranking**)
    - ad-pricing is governed by “next price auction” (need to predict the price at the next lower rank) (**regression**)

Experiments
- *Click Prediction Experiment - Results*

- CRR gives desired result: **Best of both performance**
- AUC Loss is **equal to that in ranking-only** and MSE **performs as well as in regression-only**
- Improvement in AUC Loss: “small”, but statistically significant

Method	AdSet1 AUC Loss	MSE
Regression	0.133	**0.084**
Ranking	**0.132**	0.094
CRR	**0.132**	**0.084**

Source: D. Sculley (2010)

Future Work

- Could extend the CRR algorithm to non-linear functions
  - Instead of predictions depending on dot product ($\langle \boldsymbol{w}, \boldsymbol{x} \rangle$), one could use similarity measure and map features ($\boldsymbol{x}$) to new features ($\boldsymbol{x}'$).
- Applying L1 regularization (sparse models)
- Increase sophistication of ranking functions
  - from pairwise approach to direct optimization of MAP/NDCG
- Verifying CRR performance for predicting bounce rates

Conclusion & own Thoughts (Results)

- From the experiments CRR gives **2/3** times the **desired result**:
  "Best of Both Performance" (text-mining tasks and click-prediction)

- Concerning text mining, combining regression and ranking enhances **mainly** the results for **minority class distributions**
  - In this case, regression results benefit more from the fact that ranking is added as another information constrain to the optimization problem in rare events.

- Ranking tasks itself are not necessarily improved by regression and hence **CRR does not necessarily outperform ranking-only** (2^{nd} experiment)
  - On the other hand, the distribution was predominantly evenly distributed.

Conclusion & own Thoughts (Methodology)

- The choice of the loss function: is squared loss appropriate?
  - We could also use the absolute value, or L1 loss:

$$V(f(x), y) = |f(x) - y|$$

    which is less prone to outliers.

- Gradient Descent is usually slow
  - each iteration needs to go over the whole training example set (expensive)

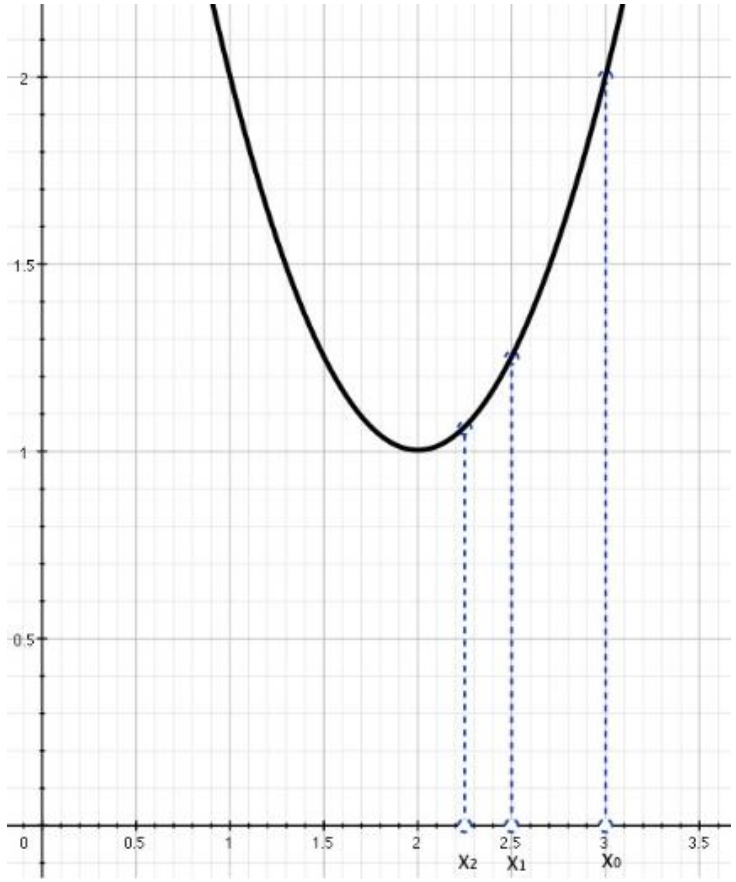
Appendix-SGD

- Numerical example – Gradient Step
- Suppose: $J(w) = (w-2)^2 + 1$, initial guess for a min: $w_0 = 3$

  - Iteration 1: $\boldsymbol{w_1} = w_0 - \lambda \frac{\partial J(w_0)}{\partial w}$
    $= 3 - 0.25(2 * 3 - 4)$
    $= \mathbf{2.5}$

  - Iteration 2: $\boldsymbol{w_2} = w_1 - \lambda \frac{\partial J(w_1)}{\partial w}$
    $= 2.5 - 0.25(2 * 2.5 - 4)$
    $= \mathbf{2.25}$

  - Iteration 3: $\boldsymbol{w_3} = w_2 - \lambda \frac{\partial J(w_2)}{\partial w}$
    $= 2.25 - 0.25(2 * 2.25 - 4)$
    $= \mathbf{2.125}$



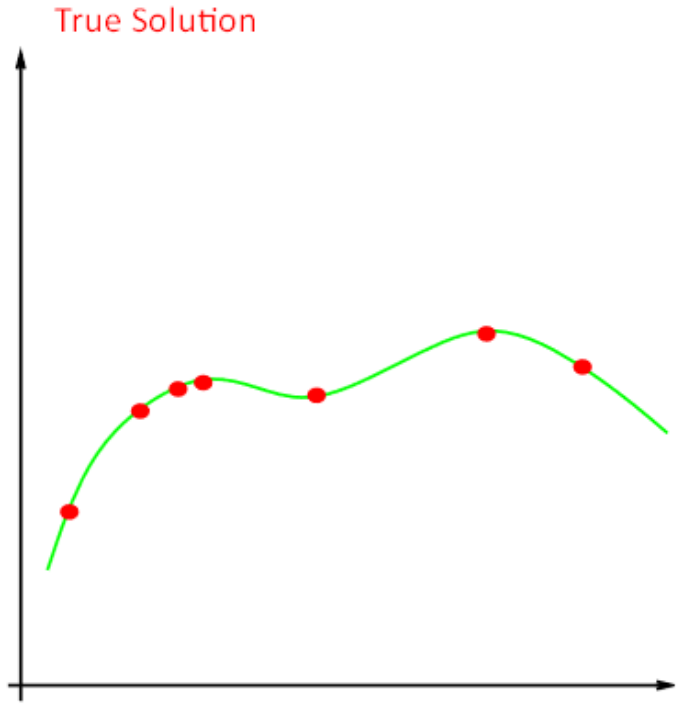
Source: https://pdfs.semanticscholar.org/presentation/c133/a0f119b917a91ad7cdf30bd26ae6e31c51b2.pdf

Appendix-SGD

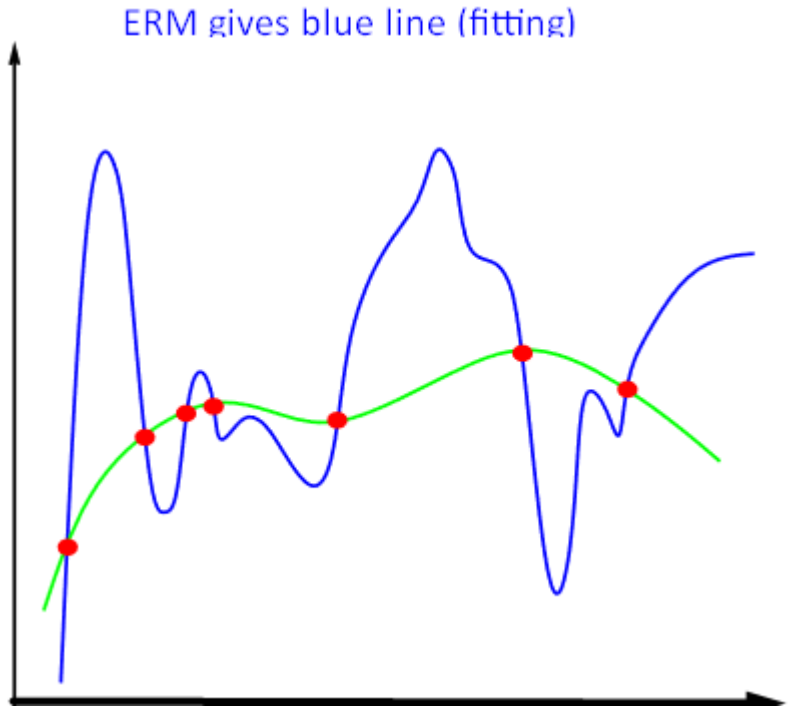
- SGD in terms of expectations
  - Since $l(\boldsymbol{w}) = E_{\boldsymbol{x}}(l(\boldsymbol{w}, \boldsymbol{x})) = \int p(\boldsymbol{x}) l(\boldsymbol{w}, \boldsymbol{x}) d\boldsymbol{x}$
  - $\nabla_w l(\boldsymbol{w}) = \nabla_w [E_{\boldsymbol{x}}(l(\boldsymbol{w}, \boldsymbol{x}))] = E_{\boldsymbol{x}}[\nabla_w l(\boldsymbol{w}, \boldsymbol{x})]$
- Then
  - $\mathbf{w} \leftarrow \mathbf{w}_{i-1} - \eta_i \, E_{\boldsymbol{x}}[\nabla_w l(\boldsymbol{w}, \boldsymbol{x})]$ → we can't compute this
- Need to approximate with sample data:
  - $\nabla_w l(\boldsymbol{w}) \approx \sum_{\boldsymbol{i=1}}^{\boldsymbol{D}} \nabla_w l\left(\boldsymbol{w}, \boldsymbol{x}^i\right)$
  - **So** if we have just one sample (as is the case for SGD- we're updating for each drawn sample), this approximation is very noisy
    - Need to penalize for this noise → $-\lambda \mathbf{w}_{i-1}$

Appendix-ERM

- ERM and generalization

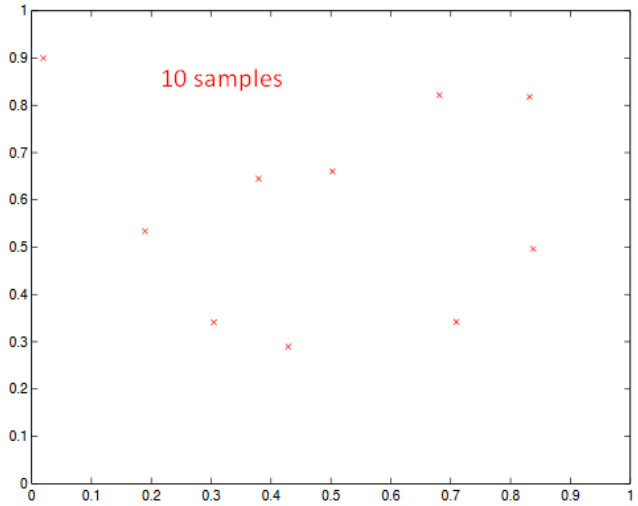


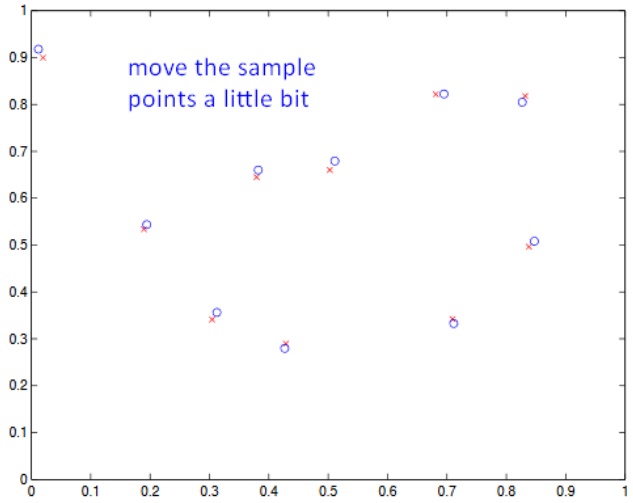
Under which conditions does the ERM line converge with increasing numbers of examples.



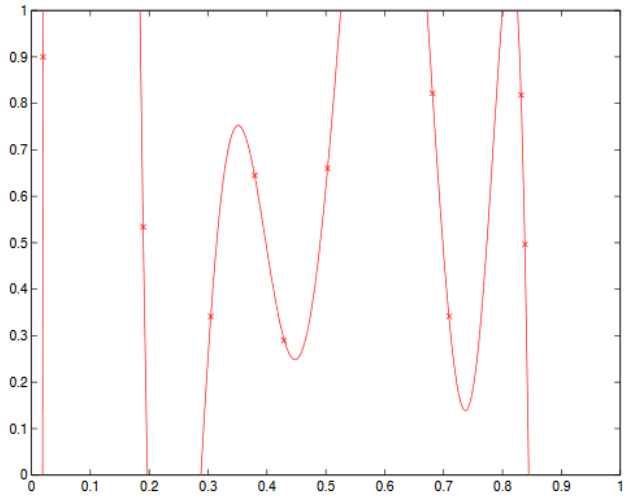
Source: http://www.mit.edu/~9.520/spring11/slides/class02.pdf

Appendix-ERM and complexity/stability

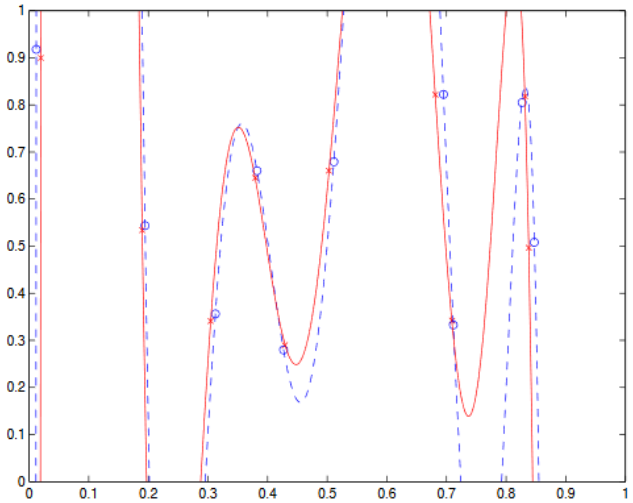




➔
fit smoothest
polynomial
(high complexity)



➔
model predicts
relatively bad
now with this
fitting
(the solution
changes a lot)



Source: http://www.mit.edu/~9.520/spring11/slides/class02.pdf

Comparison among the three papers

- Similarities:
  - Training a model
  - Minimizing loss
  - Stochastic gradient descent methods as a fundamental/easy to implement and efficient algorithm step to optimize training time. (practical for large datasets)
- Differences:
  - L1 regularization vs. L2 regularization

References

- Bengio, Y. (2009). Learning deep architectures for AI. *Foundations & Trends in Mach. Learn., to appear*
- Bengio, Y., Ducharme, R., & Vincent, P. (2001). A neural probabilistic language model. Adv. Neural Inf. Proc. Sys. 13 (pp. 932{938)
- Bottou, L., Bousquet, O. (2008). The Tradeoffs of Large Scale Learning. *In proceedings of the conference "Neural Information Processing Systems 2007". Neural Information Processing Systems 20, 161-168. NIPS.*
- Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. *Int. Conf. Mach.Learn. 2008* (pp. 160{167).
- Elman, J. L. (1993). Learning and development inneural networks: The importance of starting small. *Cognition*, *48*, 781{799.
- Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In Pr ceedings of ICML, pages 33–40.
- Goodfellow, Ian, et al. "Deep Learning." Deep Learning, MIT Press, 2016, www.deeplearningbook.org/.
- Guestrin, C. (2014). CSE546: *Stochastic Gradient Descent*, [Lecture Notes]. Retrieved from https://pdfs.semanticscholar.org/presentation/767a/4171f9f5c5108caac49400e5871a21b96039.pdf
- Hsieh, C. (2018). ECS171: *Optimization*, week 4 notes [Lecture Notes]. Retrieved from http://www.stat.ucdavis.edu/~chohsieh/teaching/ECS171_Winter2018/lecture4.pdf
- Krueger, K. A., & Dayan, P. (2009). Flexible shaping: how learning in small steps helps. Cognition, 110,380{394.
- Lewis, D. D.; Yang, Y.; Rose, T.; and Li, F. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research, 5:361-397, 2004. http://www.jmlr.org/papers/volume5/lewis04a/lewis04a.pdf.*

References

- Poggio, T. (2011). 9.520: *The Learning Problem and Regularization*, week 2 notes [Lecture Notes]. Retrieved from http://www.mit.edu/~9.520/spring11/slides/class02.pdf
- Rosenberg, D. S. (2015). ML1003: *Gradient and Stochastic Gradient Descent* [Lecture Notes]. Retrieved from https://pdfs.semanticscholar.org/presentation/2c4a/20b112b4160fea7330795853437ba3621733.pdf
- Rohde, D., & Plaut, D. (1999). Language acquisitionin the absence of explicit negative evidence: How important is starting small? Cognition, 72, 67{109
- Schwenk, H., & Gauvain, J.-L. (2002). Connectionist language modeling for large vocabulary continuous speech recognition. International Conference on Acoustics, Speech and Signal Processing (pp. 765{768). Orlando, Florida
- Sadigh, D. (2018). CS221: *Machine Learning 1*, week 2 notes [Lecture Notes]. Retrieved from https://web.stanford.edu/class/cs221/lectures/learning1.pdf
- Sculley, D. (2010, July). Combined regression and ranking. *In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM.*
- Shalev-Shwartz, S., Singer, Y., Srebro, N. (2007). Pegasos: primal estimated sub-gradient solver for SVM. In *ICML '07: Proceedings of the 24th international conference on Machine learning*. *ICML.*
- Stewart, J., 2011. Calculus: Early Transcendentals. Cengage Learning.
- Tsuruoka, Y., Tsujii, J. I., & Ananiadou, S. (2009). Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP
- Yoonkyung, L. (2014). *A statistical View of Ranking: Midway between Classification and Regression* [Lecture Notes].Retrieved from http://www.mit.edu/~9.520/spring11/slides/class02.pdf