Summary

i. Stochastic gradient descent

ii. Anecdotal evidence

iii. Analysis for a simple case

iv. The tradeoffs of large scale learning

v. Simple benchmarks for SGD.

vi. General convergence results.

vii. Learning with a single epoch.

viii. SGD for Neyman Pearson classification.

I. Learning with Stochastic Gradient Descent

Example

Binary classification

– Patterns x.

– Classes $y = \pm 1$

– Examples $z = (x, y)$

Linear model

– Choose features: $\Phi(x) \in \mathbb{R}^d$

– Linear discriminant function: $f_w(x) = \text{sign}\left(w^\top \Phi(x)\right)$

SVM training

– Choose loss function

$$Q(z,w) = \ell(y, f(x,w)) \quad = \quad \text{(e.g.)} \quad \log\left(1 + e^{-y\, w^\top \Phi(x)}\right)$$

– Cannot minimize the expected risk $E(w) = \int Q(z,w)\, dP(z)$.

– Can compute the empirical risk $E_n(w) = \frac{1}{n}\sum_{i=1}^{n} Q(z_i, w)$.

➡ Minimize L_2 regularized empirical risk

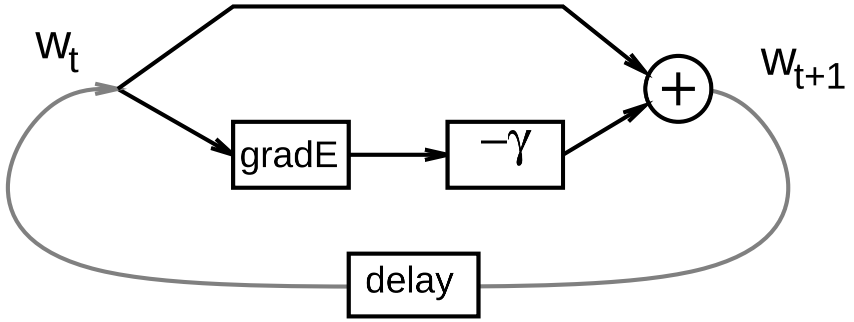
$$\min_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} Q(z_i, w)$$

Choosing λ is the same setting a constraint $\|w\|^2 < B$.

Batch Gradient Descent

Batch: process all examples together

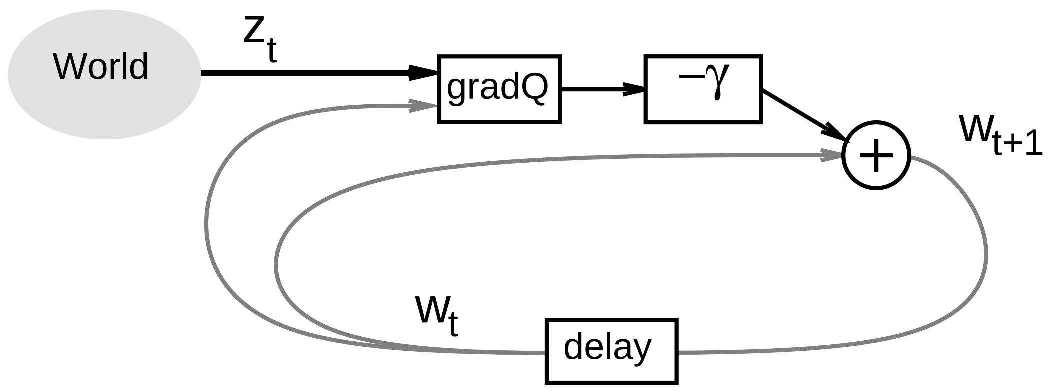
$$\text{Repeat: } w \leftarrow w - \gamma \left(\lambda w + \frac{1}{n} \sum_{i=1}^{n} \frac{\partial Q}{\partial w}(z_i, w) \right)$$



Stochastic Gradient Descent

Online: process examples one by one

$$\text{Repeat: (a) Pick random example } z_t = (x_t, y_t)$$
$$\text{(b) } w \leftarrow w - \gamma_t \left(\lambda w + \frac{\partial Q}{\partial w}(z_t, w) \right)$$



Stochastic versus Online

Stochastic

– Examples drawn randomly from a finite training set.
– In practice one often perform "epochs".

Online

– Examples drawn on-the-fly from the real world.
– Adaptive systems.

In fact the same mathematics apply to both cases.

Stochastic/Online versus Generalization

Stochastic gradient descent optimizes

$$\boldsymbol{E(w) = \mathbb{E}\left[Q(z,w)\right]}$$

for whatever distribution $\boldsymbol{dP(z)}$ the examples are drawn from.

- If the examples are drawn from a finite training set,
  $\rightarrow$ SGD optimizes the empirical error $E_n(w)$.
- If the examples are drawn from Nature,
  $\rightarrow$ SGD optimizes the expected error $E(w)$.

SGD convergence speed statements are generalization results.

SGD Algorithms for everything. . .

Adaline (Widrow and Hoff, 1960)

$$Q_{\text{adaline}} = \tfrac{1}{2}\big(y - w^\top \Phi(x)\big)^2$$
$$\Phi(x) \in \mathbb{R}^d, \;\; y = \pm 1$$

$$w \leftarrow w + \gamma_t \big(y_t - w^\top \Phi(x_t)\big)\, \Phi(x_t)$$

Perceptron (Rosenblatt, 1957)

$$Q_{\text{perceptron}} = \max\{0, -y\, w^\top \Phi(x)\}$$
$$\Phi(x) \in \mathbb{R}^d, \;\; y = \pm 1$$

$$w \leftarrow w + \gamma_t \begin{cases} y_t\, \Phi(x_t) & \text{if } y_t\, w^\top \Phi(x_t) \le 0 \\ 0 & \text{otherwise} \end{cases}$$

Multilayer perceptrons (Rumelhart et al., 1986) . . .

SVM (Cortes and Vapnik, 1995) . . .

Lasso (Tibshirani, 1996)

$$Q_{\text{lasso}} = \lambda |w|_1 + \tfrac{1}{2}\big(y - w^\top \Phi(x)\big)^2$$
$$w = (u_1 - v_1, \ldots, u_d - v_d)$$
$$\Phi(x) \in \mathbb{R}^d, \;\; y \in \mathbb{R}, \;\; \lambda > 0$$

$$u_i \leftarrow \Big[u_i - \gamma_t\big(\lambda - (y_t - w^\top \Phi(x_t))\Phi_i(x_t)\big)\Big]_+$$
$$v_i \leftarrow \Big[v_i - \gamma_t\big(\lambda + (y_t - w_t^\top \Phi(x_t))\Phi_i(x_t)\big)\Big]_+$$

with notation $[x]_+ = \max\{0, x\}$.

K-Means (MacQueen, 1967)

$$Q_{\text{kmeans}} = \min_k \tfrac{1}{2}(z - w_k)^2$$
$$z \in \mathbb{R}^d, \;\; w_1 \ldots w_k \in \mathbb{R}^d$$

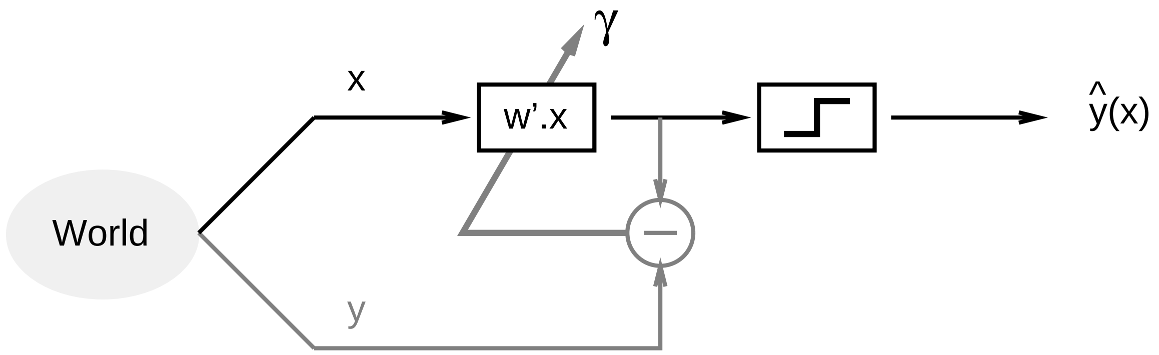
$n_1 \ldots n_k \in \mathbb{N}$, initially 0

$$k^* = \arg\min_k (z_t - w_k)^2$$
$$n_{k^*} \leftarrow n_{k^*} + 1$$
$$w_{k^*} \leftarrow w_{k^*} + \tfrac{1}{n_{k^*}}(z_t - w_{k^*})$$

Adaline

Model: $\boldsymbol{f(x,w) = \mathbf{StepFunction}(\mathbf{w}^\top \Phi(\mathbf{x}))}$

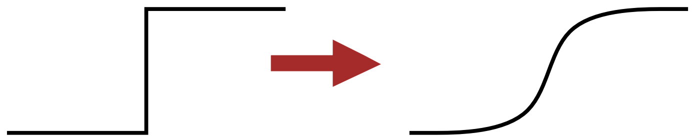
Loss function: $\boldsymbol{Q_{\text{adaline}} = \frac{1}{2}(y - w^\top \Phi(x))^2}$



Update rule: $\boldsymbol{w \leftarrow w + \gamma_t (y_t - w^\top \Phi(x_t))\, \Phi(x_t)}$

(Widrow and Hoff, 1960)

Multilayer Networks



Model:

Acyclic combinations of sigmoid units.

E.g.: $f(x, w) = \sum_j v_j \mathrm{Sigmoid}(u_j{}^\top x)$ and many variations. . .

Loss: $Q_{\mathrm{mlp}} = \frac{1}{2}(y - f(x, w))^2$

Update rule:

- Compute $\partial f(x_t, w)/\partial w$ using the chain rule.
- $w \leftarrow w + \gamma_t (y_t - f(x_t, w)) \frac{\partial f(x_t,w)}{\partial w}$

Non differentiabilities

What if $\boldsymbol{Q(z,w)}$ has a few non differentiable points?

Just pick another example!

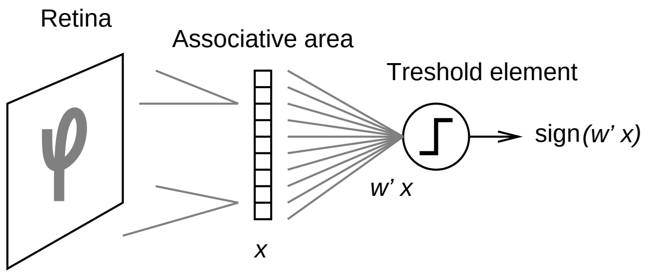
$$\frac{\partial E(w)}{\partial w} = \frac{\partial}{\partial w}\int Q(z,w)\,dP(z) \overset{?}{=} \int \frac{\partial Q(z,w)}{\partial w}\,dP(z)$$

Mild sufficient condition: (bounded convergence theorem)

$$\exists\,\Phi, \forall z,\ \forall\, v \in \vartheta(w),\quad |Q(z,v) - Q(z,w)| \ \le\ |w-v|\,\Phi(z,w)$$

Rosenblatt's Perceptron

Model: $\mathbf{StepFunction}\left(\mathbf{w}^\top \mathbf{\Phi}(\mathbf{x})\right)$

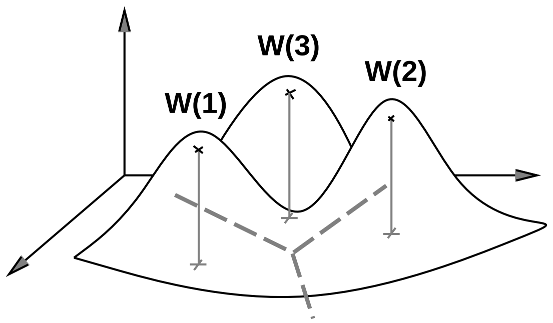


Loss: $Q_{\text{perceptron}} = \max\{0, -y\, w^\top \Phi(x)\}$

Update rule: $w \leftarrow w + \gamma_t \begin{cases} y_t\, \Phi(x_t) & \text{if } y_t\, w^\top \Phi(x_t) \leq 0 \\ 0 & \text{otherwise} \end{cases}$ (Rosenblatt, 1957)

K-Means

Loss: $\boldsymbol{Q_{\text{kmeans}} = \min_k \frac{1}{2}(z - w_k)^2}$



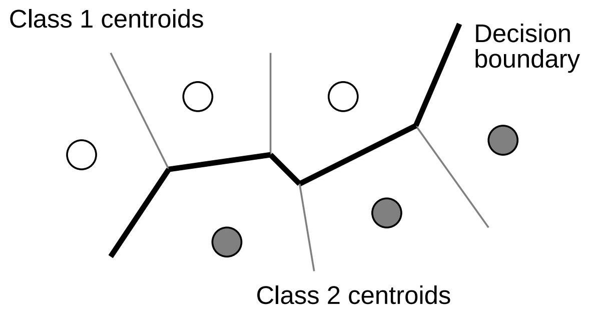
Update rule: $\begin{cases} k^* = \arg\min_k (z_t - w_k)^2 \\ n_{k^*} \leftarrow n_{k^*} + 1 \\ w_{k^*} \leftarrow w_{k^*} + \frac{1}{n_{k^*}}(z_t - w_{k^*}) \end{cases}$ (MacQueen, 1967)

Note the “optimal” grain $\gamma_t = 1/n_{k^*}$.

Learning Vector Quantization

Loss:

$$Q_{\text{lvq}} = \begin{cases} 0 & \text{if correct} \\ \frac{(x-w^+)^2-(x-w^-)^2}{\delta(x-w^-)^2} & \text{if } (x-w^+)^2 < (1+\delta)(x-w^-)^2 \\ 1 & \text{otherwise} \end{cases}$$



w^- : closest centroid.
w^+ : closest centroid w/correct class.

Update:

$$\text{if } \begin{cases} x \text{ is misclassified} \\ \text{and } (x-w^+)^2 < (1+\delta)(x-w^-)^2 \end{cases}$$

$$\text{then } \begin{cases} w^-_{t+1} = w^-_t - \gamma_t k_1 (x - w^-_t) \\ w^+_{t+1} = w^+_t + \gamma_t k_2 (x - w^+_t) \end{cases}$$

(Kohonen et al. 1982, 1988)

Lasso

L1 Regularization:

$$\min_{w} \ \lambda |w|_1 + \frac{1}{n} \sum_{i=1}^{n} \tfrac{1}{2} \big(y_i - w^\top \Phi(x_i)\big)^2$$

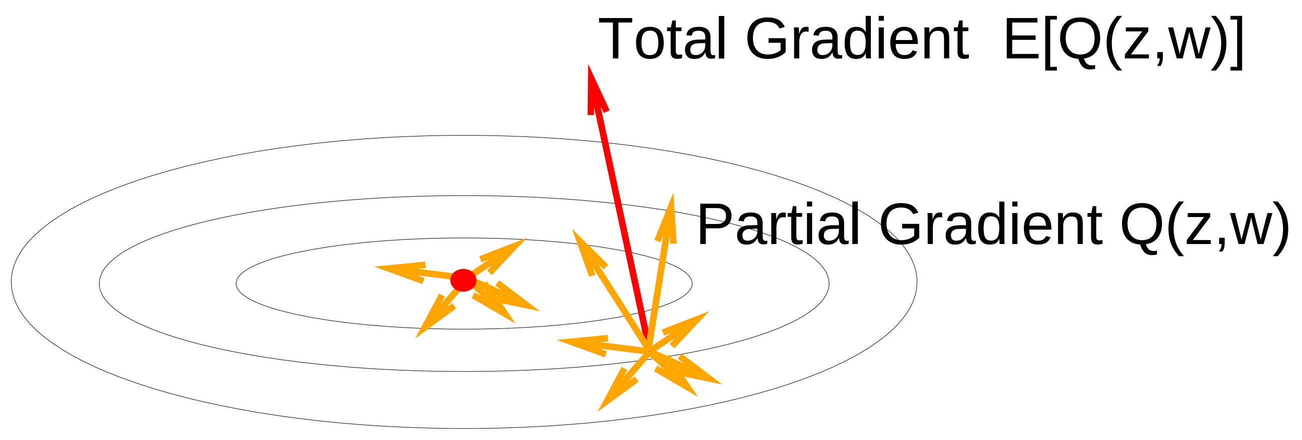
Trick: $w = (u_1 - v_1, \ldots, u_d - v_d) \quad u_i, v_i \in \mathbb{R}^+$

Update Rule: $\begin{cases} u_i \leftarrow \Big[u_i - \gamma_t\big(\lambda - (y_t - w^\top \Phi(x_t))\Phi_i(x_t)\big)\Big]_+ \\ v_i \leftarrow \Big[v_i - \gamma_t\big(\lambda + (y_t - w_t^\top \Phi(x_t))\Phi_i(x_t)\big)\Big]_+ \end{cases}$

using notation $[x]_+ = \max\{0, x\}$.

II. Anecdotal Evidence

Stochastic gradient is slow



Does stochastic gradient converge to the optimum?

Residual noise proportional to learning rate γ_t.
Learning rate cannot decrease too fast.

$$|w_t - w^*|^2 \quad \equiv \quad \frac{1}{t} \qquad (\ldots \text{at best}\ldots)$$

Stochastic gradient is not a good optimization algorithm.

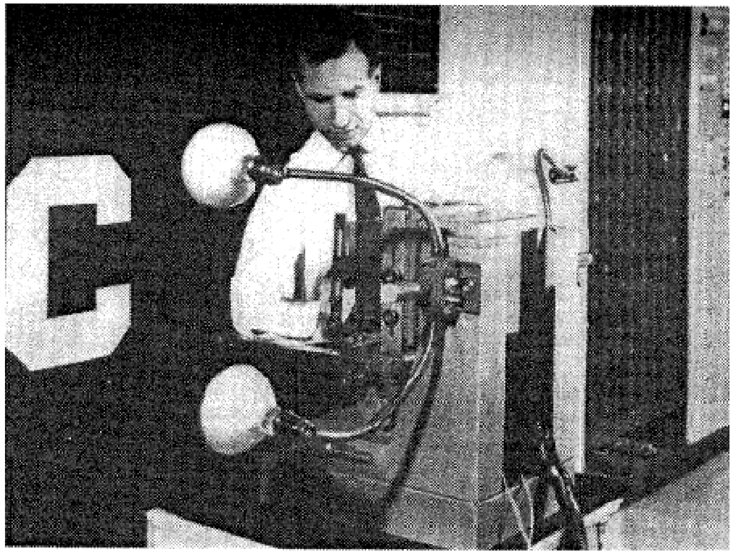
Stochastic gradient is fast

Assume training set contains
10 copies of the 100 same examples.

- Batch — Blindly computes redundant gradients.
  1 epoch on large set $\equiv$ 1 epochs on small set.
- Online — Take advantage of redundancy.
  1 epoch on large set $\equiv$ 10 epochs on small set.

Stochastic gradient **learns** much faster?

1957 - Perceptron

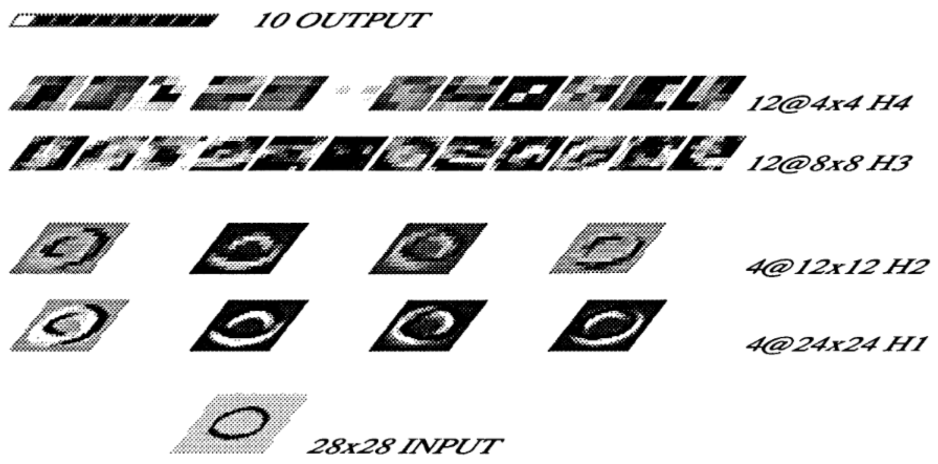


(Frank Rosenblatt, 1957)

Computing
– Relays, potentiometers, electrical motors, . . .

Why did Rosenblatt use a stochastic algorithm?
– Because he did not know better? (unlikely)
– Because he did not have computing resources to do anything else?
– Does this means that a stochastic algorithm does more with less?

1988 - Convolutional networks for OCR



(Le Cun et al., 1989)

Computing

– Sun3 (≈ first generation of Palm Pilot.)

– 9000 training examples, 2000 test examples, three weeks of training.

Why did they use stochastic gradient?

– Because he did not know better? no (Becker & Le Cun, 1989)

– Because he did not have computing resources to do anything else?

– Does this means that a stochastic algorithm does more with less?

1995 - Check reading

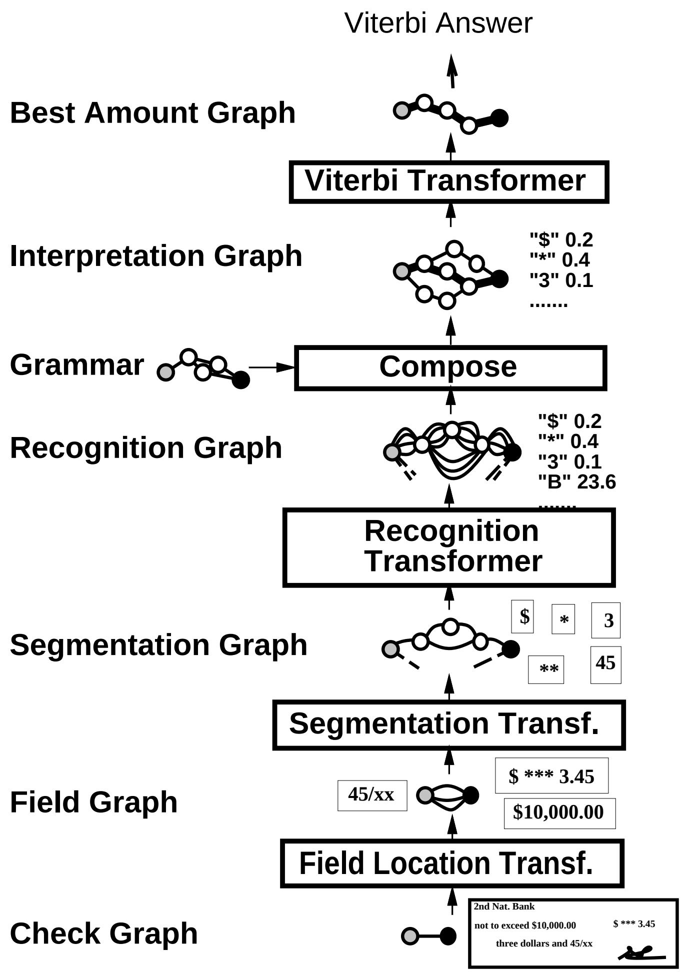
Computing

– Sun4 ($\approx$ your average cell phone.)
– 200K segmented digits.
– 250K unsegmented check images.
– Three weeks of CRF-like training.

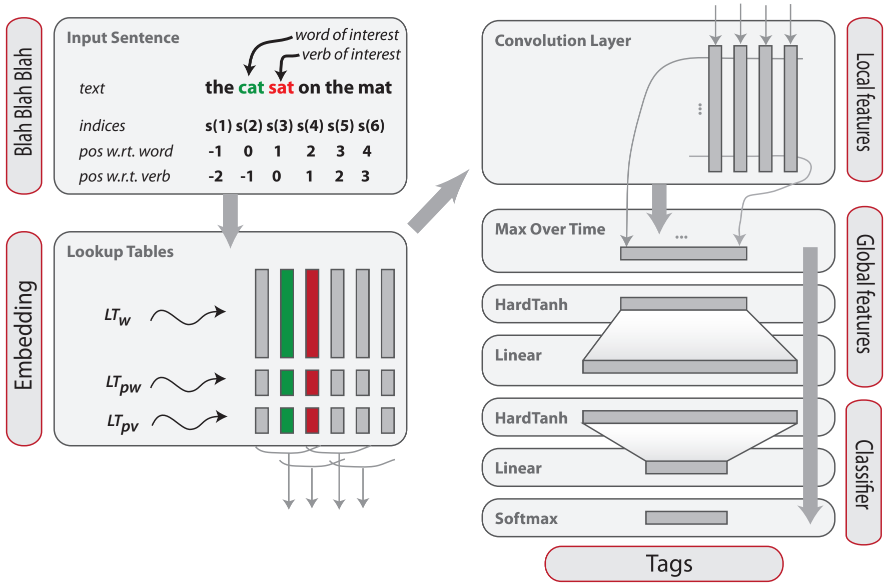
Why did we use stochastic gradient?

– Because we did not know better?
– Because he did not have computing resources to do anything else?
– Does this means that a stochastic algorithm does more with less?

(Bottou, LeCun, et al., CVPR 1997)



2010 - Multiple Natural Language Tasks



Computing

– ≈ 1B words for unsupervised training set. Six weeks of training.

– ≈ 1M words for task dependent training.

Why did they use stochastic gradient?

– Because they did not have computing resources to do anything else?

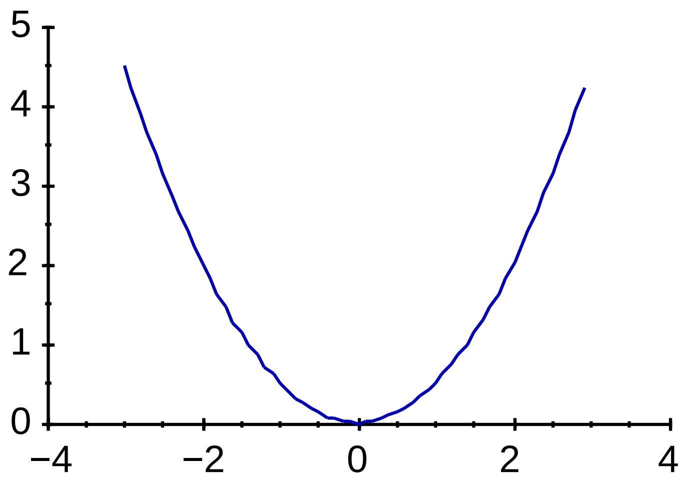
(Collobert, Weston et al., 2009–2011)

III. Analysis for a simple case

One-dimensional optimization

Simple example

- One dimension $x \in \mathbb{R}$
- $f(w) = \frac{1}{2} H w^2 \quad w^* = 0$
- $g(w) = \frac{\partial f}{\partial w} + \xi = Hw + \xi$
- $\mathbb{E}[\xi] = 0 \quad \mathbb{E}[\xi^2] = G$



Without stochastic noise ($\xi = 0$)

Gradient descent with constant gain $\gamma_t = \eta$

$$w_t = w_{t-1} - \eta\, g(w_{t-1})$$

Then

$$\begin{aligned} w_t &= w_{t-1} - \eta\, H\, w_{t-1} \\ &= (1 - \eta H)\, w_{t-1} \\ &= (1 - \eta H)^t\, w_0 \end{aligned}$$

Error w_t^2 decreases exponentially when constant gain $\eta < 2/H$.

Remark

– H is a positive matrix in the multidimensional case.
The convergence condition is then $\eta < 2/\lambda_{\max}$.

With stochastic noise ($\mathbb{E}[\xi] = 0, \quad \mathbb{E}[\xi^2] = G$)

Gradient descent with decreasing gain $\gamma_t = \eta t^{-\alpha}$

$$\begin{aligned} w_t &= w_{t-1} - \eta t^{-\alpha} g(w_{t-1}) \\ &= w_{t-1} - \eta t^{-\alpha} (H w_{t-1} + \xi) \end{aligned}$$

Error recursion

$$\begin{aligned} w_t^2 &= w_{t-1}^2 - 2\eta t^{-\alpha} w_{t-1} (H w_{t-1} + \xi) + \eta^2 t^{-2\alpha} (H w_{t-1} + \xi)^2 \\ \mathbb{E}[w_t^2 | \mathcal{P}_t] &= w_{t-1}^2 - 2\eta t^{-\alpha} w_{t-1} H w_{t-1} + \eta^2 t^{-2\alpha} (H^2 w_{t-1}^2 + G) \\ \mathbb{E}[w_t^2] &= \left(1 - 2\eta H t^{-\alpha} + \eta^2 H^2 t^{-2\alpha}\right) \mathbb{E}[w_{t-1}^2] \; + \; \eta^2 G t^{-2\alpha} \end{aligned}$$

Decomposition

$$
\begin{aligned}
U_t &= \left(1 - 2\eta H t^{-\alpha} + \eta^2 H^2 t^{-2\alpha}\right) U_{t-1} \;+\; \eta^2 G t^{-2\alpha} \\
&= U_S \prod_{j=S+1}^{t} \left(1 - 2\eta H j^{-\alpha} + \eta^2 H^2 j^{-2\alpha}\right) \\
&\quad + \sum_{i=S+1}^{t} \eta^2 G i^{-2\alpha} \prod_{j=i+1}^{t} \left(1 - 2\eta H j^{-\alpha} + \eta^2 H^2 j^{-2\alpha}\right)
\end{aligned}
$$

When $\alpha < 0$

– Everything diverges.

When $\alpha = 0$

– When $2\eta H - \eta^2 H^2 \geq 1$, the error U_t diverges.

– When $2\eta H - \eta^2 H^2 < 1$, the error U_t converges, but not to zero.

Otherwise we pick S large enough to make all the product terms positive.

The Green Term

Case $1/2 < \alpha < 1$

$$
\begin{aligned}
A_{St} &= \sum_{j=S+1}^{t} \log\left(1 - 2\eta H j^{-\alpha} + \eta^2 H^2 j^{-2\alpha}\right) \\
&= \sum_{j=S+1}^{t} -2\eta H j^{-\alpha} + \mathcal{O}\left(j^{-2\alpha}\right) \\
&= -2\eta H \int_S^t x^{-\alpha} dx + \mathcal{O}(1) \\
&= -\kappa\, t^{1-\alpha} + \mathcal{O}(1) \qquad \text{where} \quad \kappa \triangleq \frac{2\eta H}{1-\alpha}
\end{aligned}
$$

Therefore $U_S\, e^{A_{St}} \;\equiv\; e^{-\kappa\, t^{1-\alpha}} \;\longrightarrow 0$
The green term converges exponentially to zero.

The case $0 < \alpha \leq 1/2$ is slightly more complicated and not that interesting.

The Green Term

Case $\alpha = 1$

$$\begin{aligned} A_{St} &= \sum_{j=S+1}^{t} \log\left(1 - 2\eta H j^{-1} + \eta^2 H^2 j^{-2}\right) \\ &= \sum_{j=S+1}^{t} -2\eta H j^{-1} + \mathcal{O}\left(j^{-2}\right) \\ &= -2\eta H \int_S^t x^{-1} dx + \mathcal{O}(1) \\ &= -2\eta H \log(t) + \mathcal{O}(1) \end{aligned}$$

Therefore $U_S\, e^{A_{St}} \equiv t^{-2\eta H} \longrightarrow 0$

The green term converges polynomially to zero.

The choice of η impacts the degree of the polynomial convergence.

The Green Term

Case $\alpha > 1$

$$\begin{aligned} A_{St} &= \sum_{j=S+1}^{t} \log\left(1 - 2\eta H j^{-\alpha} + \eta^2 H^2 j^{-2\alpha}\right) \\ &= \sum_{j=S+1}^{t} -2\eta H j^{-\alpha} + \mathcal{O}\left(j^{-2\alpha}\right) \longrightarrow K > -\infty \end{aligned}$$

Therefore $U_S\, e^{A_{St}} \;\longrightarrow\; U_S\, e^{K} > 0$

The green term does not converges to zero.

Therefore U_t does not converge to zero either...

The Red Term

Case $\alpha = 1$

$$\begin{aligned}
A_{it} &= \sum_{j=i+1}^{t} \log\left(1 - 2\eta H j^{-1} + \eta^2 H^2 j^{-2}\right) \\
&= -2\eta H \sum_{j=i+1}^{t} j^{-1} + \mathcal{O}\left(j^{-2}\right) \;=\; -2\eta H \int_i^t x^{-1} dx + \mathcal{O}(1) \\
&= -2\eta H \left[\log t - \log i\right] + \mathcal{O}(1)
\end{aligned}$$

Therefore we can write the red term as:

$$\begin{aligned}
\sum_{i=S+1}^{t} \eta^2 G i^{-2} e^{A_{it}} &\equiv t^{-2\eta H} \sum_{i=S+1}^{t} i^{-2}\, i^{2\eta H} \\
&\equiv t^{-2\eta H} \left[t^{2\eta H-1} - S^{2\eta H-1}\right] \;\equiv\; t^{-1} \;\longrightarrow 0
\end{aligned}$$

The red term converges like t^{-1}.

The Red Term

Case $1/2 < \alpha < 1$

$$\begin{aligned} A_{it} &= \sum_{j=i+1}^{t} \log\left(1 - 2\eta H j^{-\alpha} + \eta^2 H^2 j^{-2\alpha}\right) \\ &= -2\eta H \sum_{j=i+1}^{t} j^{-\alpha} + \mathcal{O}\left(j^{-2\alpha}\right) \;=\; -2\eta H \int_i^t x^{-\alpha} dx + \mathcal{O}(1) \\ &= -2\eta H \left[\frac{x^{1-\alpha}}{1-\alpha}\right]_i^t + \mathcal{O}(1) \;=\; -\kappa\left(t^{1-\alpha} - i^{1-\alpha}\right) + \mathcal{O}(1) \end{aligned}$$

Then $\displaystyle \sum_{i=S+1}^{t} \eta^2 G i^{-2\alpha} e^{A_{it}} \equiv e^{-\kappa\, t^{1-\alpha}} \sum_{i=S+1}^{t} i^{-2\alpha}\, e^{\kappa\, i^{1-\alpha}}$

We would like to approximate with an integral.

Unfortunately the primitive of $x^{-2\alpha} e^{\kappa x^{1-\alpha}}$ is not obvious.

The Red Term

Case $1/2 < \alpha < 1$ (continued)

Therefore we insert terms that do not affect the asymptotic rate and transform the expression into a known derivative.

$$\begin{aligned}
\cdots &\equiv e^{-\kappa\, t^{1-\alpha}} \sum_{i=S+1}^{t} i^{-2\alpha} e^{\kappa\, i^{1-\alpha}} \\
&\equiv e^{-\kappa\, t^{1-\alpha}} \sum_{i=S+1}^{t} \left(\kappa\,(1-\alpha)\, i^{-2\alpha} - (1+\alpha)\, i^{-\alpha-1} \right) e^{\kappa\, i^{1-\alpha}} \\
&\equiv e^{-\kappa\, t^{1-\alpha}} \left[x^{-\alpha} e^{\kappa\, x^{1-\alpha}} \right]_S^t \\
&\equiv t^{-\alpha}
\end{aligned}$$

The red term therefore converges like $t^{-\alpha}$.

Summary

When $\alpha \leq 0$ or $\alpha > 1$

$\mathbb{E}\left[w_t^2 - w^*\right]$ does not converge to zero.

When $1/2 < \alpha < 1$ (and also when $0 < \alpha \leq 1/2$.)

$$\mathbb{E}\left[w_t^2 - w^*\right] \equiv t^{-\alpha}$$

When $\alpha = 1$

$$\mathbb{E}\left[w_t^2 - w^*\right] \equiv t^{-\min\{1, 2\eta H\}}$$

Conclusions from this simple case

Best convergence speed is $\mathbb{E}\left[(w_t - w^*)^2\right] \equiv t^{-1}$

- This is achieved with gain $\gamma_t = \eta t^{-1}$.
- One should ensure $\eta > 1/2H$ otherwise convergence is much slower.
- Alternatively one can decrease gains a little bit slower,
  i.e., $\gamma_t = \eta t^{-1+\varepsilon} \quad \varepsilon > 0$, and converge almost as fast.

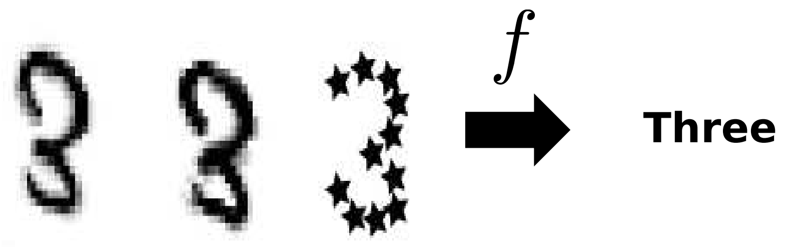
How good or bad is this convergence speed?

- For an offline optimization algorithm, this is very slow.
  Batch gradient descent has $\mathbb{E}\left[(w_t - w^*)^2\right] \equiv e^{-t}$
- For an online optimization algorithm, this is expected.
  How well can we generalize after seeing only t examples?

IV. The tradeoffs of large scale learning

The Machine Learning Problem

Example: Character Recognition



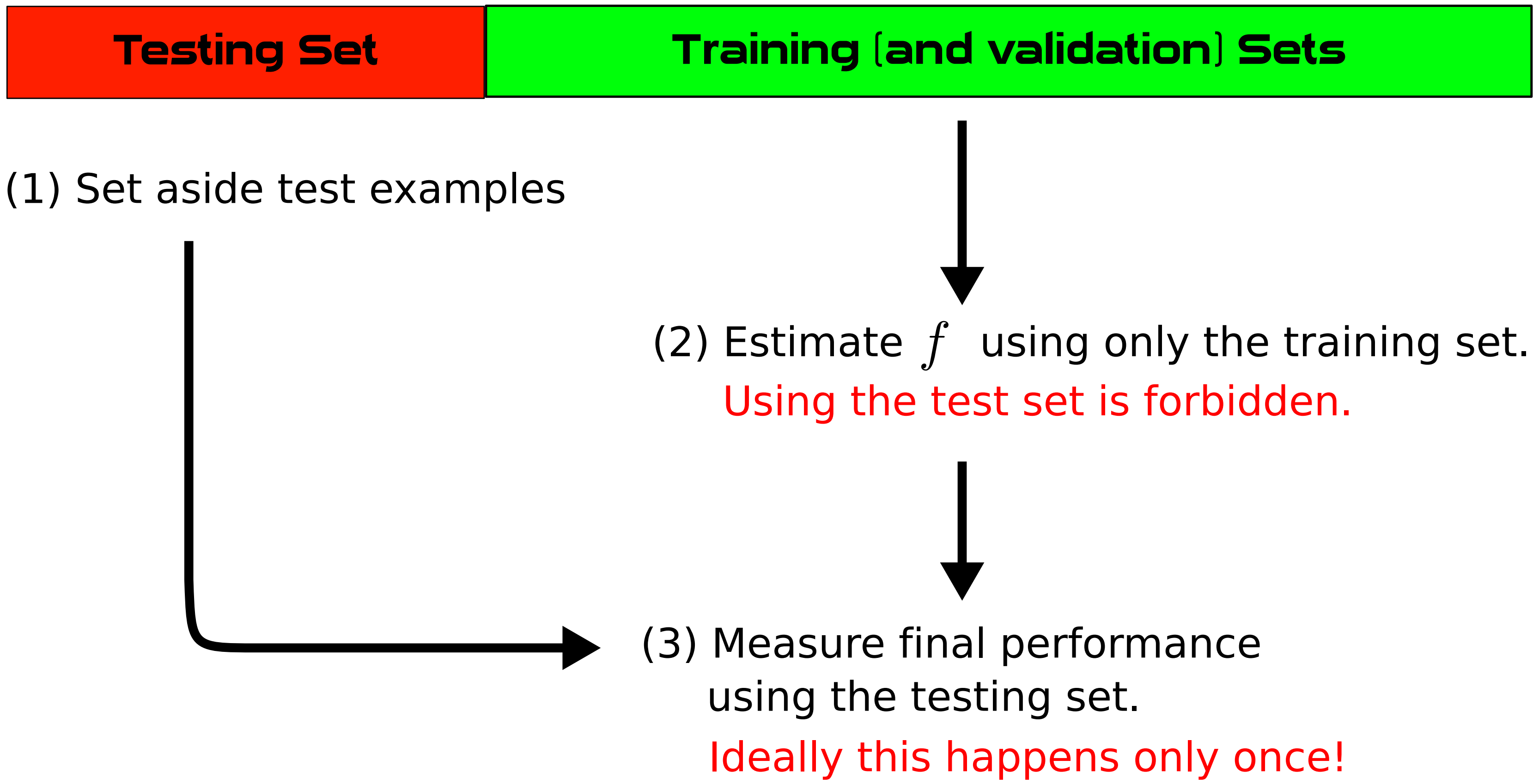
- Find recognition function f
  on the basis of training examples.

- Function f must work
  for all character variants,
  not just the training examples.

Issues

– Approximation: How to represent f?
– Statistics: How many examples do we need to estimate f?
– Statistics: Generalization $\neq$ Learning by rote.
– Computation: How to compute f efficiently?

The Main Experimental Paradigm


Testing Set
Training (and validation) Sets
(1) Set aside test examples
(2) Estimate f using only the training set.
Using the test set is forbidden.
(3) Measure final performance
using the testing set.
Ideally this happens only once!

Variations: k-fold cross-validation, etc.

This is the main driver for progress in machine learning.

Mathematical Statement (i)

- **Assumption**
  Examples are drawn independently from
  an unknown probability distribution $P(x, y)$
  that represents the laws of Nature.

- **Loss Function**
  Function $\ell(\hat{y}, y)$ measures the cost
  of answering $\hat{y}$ when the true answer is y.

- **Expected Risk**
  We seek to find the function f^* that minimizes:

$$\min_f \quad E(f) = \int \ell(\, f(x), y \,)\; dP(x, y)$$

Note: The test set error is an approximation of the expected risk.

Mathematical Statement (ii)

- **Approximation**

  Not feasible to search f^* among all functions.

  Instead, we search $f^*_{\mathcal{F}}$ that minimizes the Expected Risk $E(f)$ within some richly parametrized family of functions $\mathcal{F}$.

- **Estimation**

  Not feasible to minimize the expectation $E(f)$ because $P(x,y)$ is unknown.

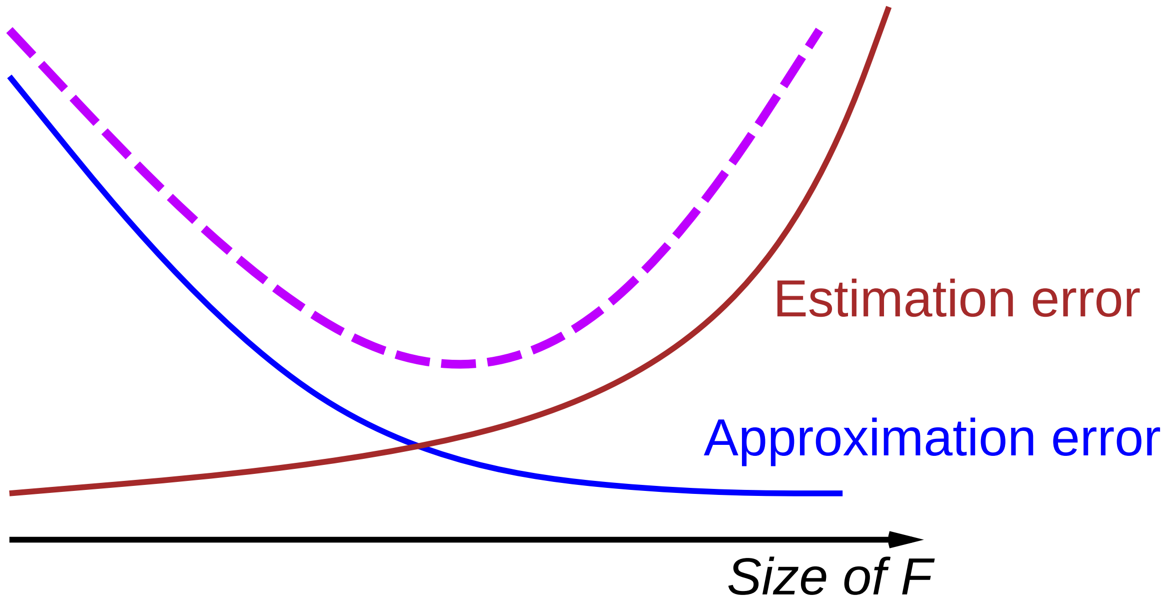
  Instead, we search f_n that minimizes the Empirical Risk $E_n(f)$, that is, the average loss over the training set examples.

$$\min_{f\in\mathcal{F}} \quad E_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(\, f(x_i), y_i \,)$$

In other words, we optimize a surrogate problem!

Approximation-Estimation Tradeoff

$$\begin{aligned} E(f_n) - E(f^*) &= \left(E(f^*_F) - E(f^*)\right) && \text{Approximation Error} \\ &+ \left(E(f_n) - E(f^*_{\mathcal{F}})\right) && \text{Estimation Error} \end{aligned}$$



(e.g. Vapnik, *Statistical Learning Theory*, 1998).

Penalized Empirical Risk

Alternate Formulation

Minimize the Penalized Empirical Risk

$$\min_{f \in \mathcal{F}} \quad \lambda\,\Omega(f) + E_n(f) \;=\; \lambda\,\Omega(f) + \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i)$$

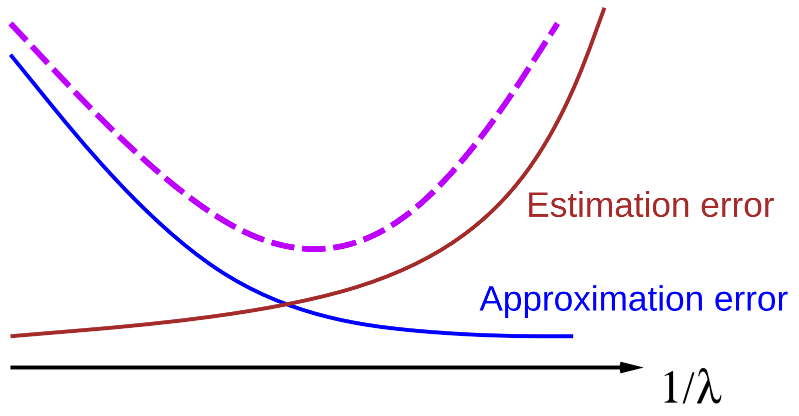
This can be viewed as minimizing f within

$$\mathcal{F}_C \;=\; \{\, f \in \mathcal{F} \mid \Omega(f) \leq C \,\}$$

where C is determined by the choice of the Lagrange coefficient λ.

Typical Example

Support Vector Machine search f with a L_2 penalty term inside a Hilbert space represented using kernel functions.



The Computational Problem

- **Statistical Perspective:**
“It is good to optimize an objective function than ensures a fast estimation rate when the number of examples increases.”

- **Computer Science Perspective:**
“To efficiently solve large problems, it is preferable to choose an optimization algorithm with strong asymptotic properties, e.g. superlinear.”

- **Incorrect Conclusion:**
“To address large-scale learning problems, use a superlinear algorithm to optimize an objective function with fast estimation rate.

- **A finer analysis leads to a dramatically different conclusion.**

The Computational Problem

- Baseline large-scale learning algorithm

Randomly discarding data is the simplest way to handle large datasets.

  - What is the statistical benefit of processing more data?
  - What is the computational cost of processing more data?

- We need a theory that links Statistics and Computation!

– 1967: Vapnik's theory does not discuss computation.
– 1981: Valiant's learnability excludes exponential time algorithms,
  but (i) polynomial time already too slow, (ii) few actual results.

Learning with Approximate Optimization

Computing $f_n = \arg\min_{f \in \mathcal{F}} E_n(f)$ is often costly.

Since we already optimize a **surrogate** function

why should we compute its optimum f_n exactly?

Let's assume our optimizer returns $\tilde{f}_n$

such that $E_n(\tilde{f}_n) < E_n(f_n) + \rho$.

For instance, one could stop an iterative
optimization algorithm long before its convergence.

Decomposition of the Error

$$\begin{aligned} E(\tilde{f}_n) - E(f^*) &= E(f^*_{\mathcal{F}}) - E(f^*) && \text{Approximation error } (\mathcal{E}_{\text{app}}) \\ &+ E(f_n) - E(f^*_{\mathcal{F}}) && \text{Estimation error } (\mathcal{E}_{\text{est}}) \\ &+ E(\tilde{f}_n) - E(f_n) && \text{Optimization error } (\mathcal{E}_{\text{opt}}) \end{aligned}$$

Problem:

Choose $\mathcal{F}$, n, and ρ to make this as small as possible,

subject to budget constraints $\left\{ \begin{array}{l} \text{max number of examples } n \\ \text{max computing time } T \end{array} \right.$

Note: choosing λ is the same as choosing $\mathcal{F}$.

Small-scale vs. Large-scale Learning

We can give *formal definitions*.

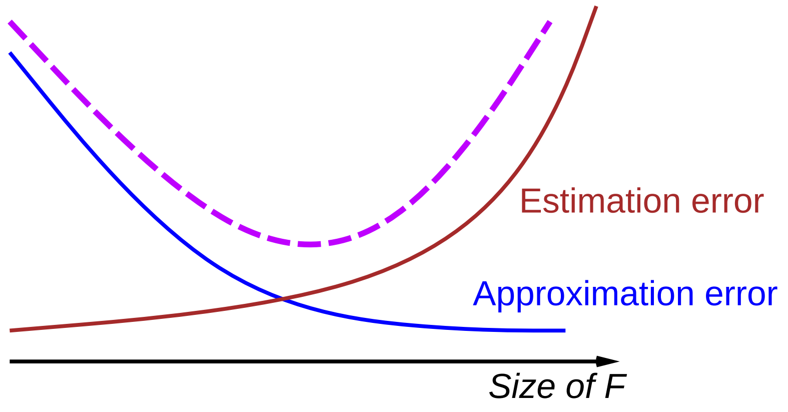
- **Definition 1:**
  We have a **small-scale learning** problem when the **active budget constraint is the number of examples n.**

- **Definition 2:**
  We have a **large-scale learning** problem when the **active budget constraint is the computing time T.**

Small-scale Learning

“The active budget constraint is the number of examples.”

- To reduce the estimation error, take n as large as the budget allows.
- To reduce the optimization error to zero, take $\rho = 0$.
- We need to adjust the size of $\mathcal{F}$.



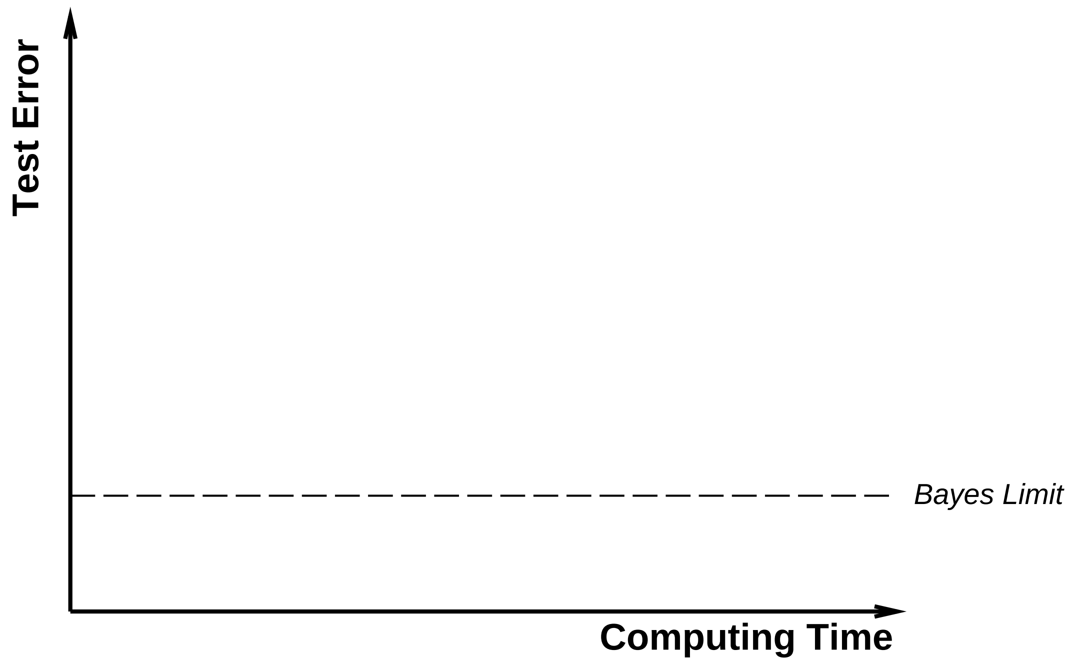
See Structural Risk Minimization (Vapnik 74) and later works.

Large-scale Learning

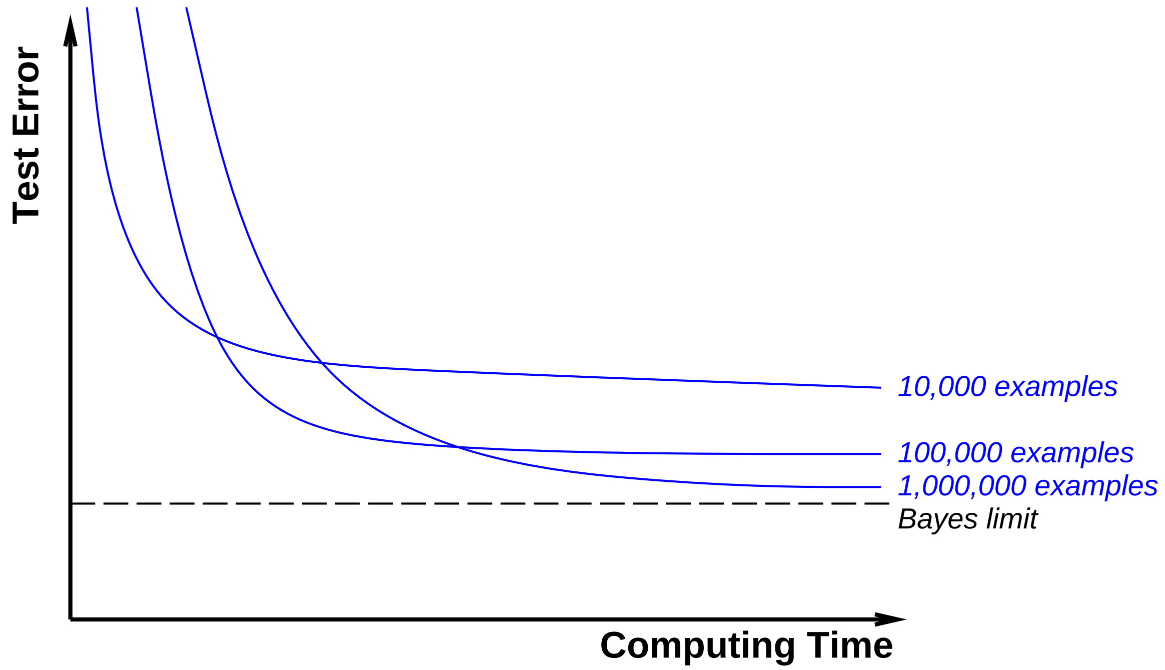
“The active budget constraint is the computing time.”

- More complicated tradeoffs.
  The computing time depends on the three variables: $\mathcal{F}$, n, and ρ.
- Example.
  If we choose ρ small, we decrease the optimization error. But we must also decrease $\mathcal{F}$ and/or n with adverse effects on the estimation and approximation errors.
- The exact tradeoff depends on the optimization algorithm.
- We can compare optimization algorithms rigorously.

Test Error versus Learning Time

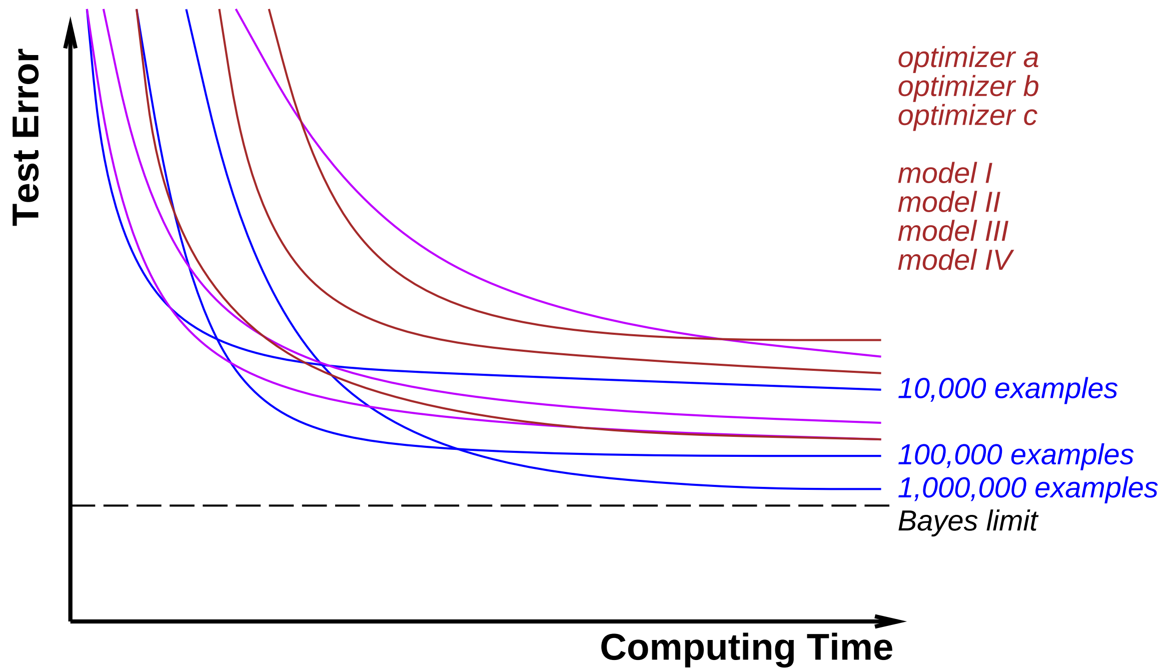


Test Error versus Learning Time



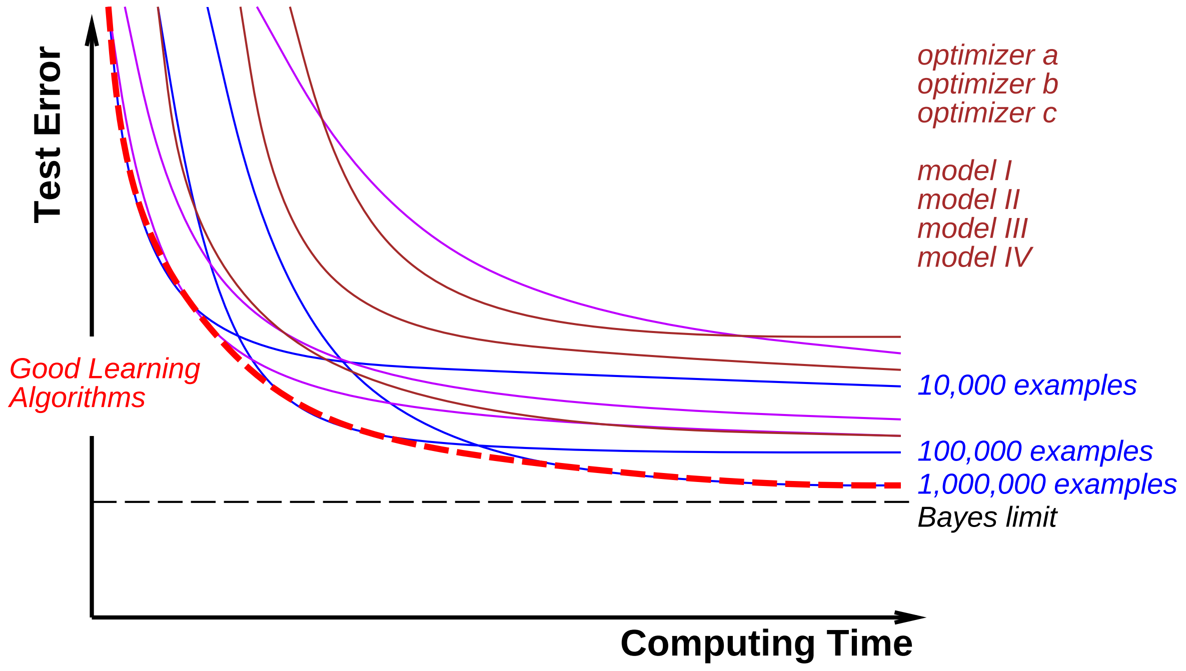
Vary the number of examples. . .

Test Error versus Learning Time



Vary the number of examples, the statistical models, the algorithms,. . .

Test Error versus Learning Time



Not all combinations are equal.

Let's compare the red curve for different optimization algorithms.

Asymptotic Analysis

$$E(\tilde{f}_n) - E(f^*) = \mathcal{E} = \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}}$$

Asymptotic Analysis

All three errors must decrease with comparable rates.

Forcing one of the errors to decrease much faster
- would require additional computing efforts,
- but would not significantly improve the test error.

Statistics

Asymptotics of the statistical components of the error

– Thanks to refined uniform convergence arguments

$$\mathcal{E} \;=\; \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}} \;\sim\; \mathcal{E}_{\text{app}} + \left(\frac{\log n}{n}\right)^{\alpha} + \rho$$

with exponent $\frac{1}{2} \leq \alpha \leq 1$.

There are in fact three (four?) types of bounds to consider:

- Classical V-C bounds (pessimistic): $\mathcal{O}\left(\sqrt{\frac{h}{n}}\right)$
- Relative V-C bounds in the realizable case: $\mathcal{O}\left(\frac{h}{n}\log\frac{n}{h}\right)$
- Localized bounds (variance, Tsybakov): $\mathcal{O}\left(\left[\frac{h}{n}\log\frac{n}{h}\right]^{\alpha}\right)$

Value h describes the *capacity* of our system.

The simplest capacity measure is the *Vapnik-Chervonenkis* dimension of $\mathcal{F}$.

(Bousquet, 2002; Tsybakov, 2004; Bartlett et al., 2005; . . .)

Statistics

Asymptotics of the statistical components of the error

– Thanks to refined uniform convergence arguments

$$\mathcal{E} \;=\; \mathcal{E}_{\rm app} + \mathcal{E}_{\rm est} + \mathcal{E}_{\rm opt} \;\sim\; \mathcal{E}_{\rm app} + \left(\frac{\log n}{n}\right)^{\alpha} + \rho$$

with exponent $\frac{1}{2} \leq \alpha \leq 1$.

Asymptotically effective large scale learning

– Must choose $\mathcal{F}$, n, and ρ such that

$$\mathcal{E} \;\sim\; \mathcal{E}_{\rm app} \;\sim\; \mathcal{E}_{\rm est} \;\sim\; \mathcal{E}_{\rm opt} \;\sim\; \left(\frac{\log n}{n}\right)^{\alpha} \;\sim\; \rho\ .$$

What about optimization times?

First order algorithms

Batch: process all examples together (GD)

– Example: minimization by gradient descent

$$\text{Repeat: } w \leftarrow w - \gamma \left(\lambda w + \frac{1}{n} \sum_{i=1}^{n} \frac{\partial Q}{\partial w}(x_i, y_i, w) \right)$$

Stochastic: process examples one by one (SGD)

– Example: minimization by stochastic gradient descent

Repeat: (a) Pick random example x_t, y_t

(b) $w \leftarrow w - \gamma_t \left(\lambda w + \frac{\partial Q}{\partial w}(x_t, y_t, w) \right)$

Second order algorithms

Batch: (2GD)

– Example: Newton's algorithm

$$\text{Repeat: } w \leftarrow w - H^{-1}\left(\lambda w + \frac{1}{n}\sum_{i=1}^{n}\frac{\partial Q}{\partial w}(x_i, y_i, w)\right)$$

Stochastic: (2SGD)

– Example: Second order stochastic gradient descent

Repeat: (a) Pick random example x_t, y_t

(b) $w \leftarrow w - \gamma_t H^{-1}\left(\lambda w + \frac{\partial Q}{\partial w}(x_t, y_t, w)\right)$

Statistics and Computation

	GD	2GD	SGD	2SGD
Time per iteration :	n	n	1	1
Iters to accuracy ρ :	$\log \frac{1}{\rho}$	$\log\log \frac{1}{\rho}$	$\frac{1}{\rho}$	$\frac{1}{\rho}$
Time to accuracy ρ :	$n \log \frac{1}{\rho}$	$n \log\log \frac{1}{\rho}$	$\frac{1}{\rho}$	$\frac{1}{\rho}$
Time to error ε :	$\frac{1}{\varepsilon^{1/\alpha}} \log^2 \frac{1}{\varepsilon}$	$\frac{1}{\varepsilon^{1/\alpha}} \log \frac{1}{\varepsilon} \log\log \frac{1}{\varepsilon}$	$\frac{1}{\varepsilon}$	$\frac{1}{\varepsilon}$

– 2GD optimizes much faster than GD.
– SGD optimization speed is catastrophic.
– SGD learns faster than both GD and 2GD.
– 2SGD only changes the constants.

V. Experiments with SGD

Benchmarking SGD

Many people associate SGD with trouble

– Historically associated with back-propagation.

– Multilayer networks are very hard problems (nonlinear, nonconvex)

– Notoriously hard to debug (always check the gradients!)

– What is difficult, SGD or MLP?

- **Try PLAIN SGD on a simple learning problem.**

Download from `http://leon.bottou.org/projects/sgd`.
These simple programs are very short.

Text Categorization with SVMs

- **Dataset**
  - Reuters RCV1 document corpus.
  - 781,265 training examples, 23,149 testing examples.
  - 47,152 TF-IDF features.

- **Task**
  - Recognizing documents of category `CCAT`.
  - Minimize $\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\lambda}{2} w^2 + \ell(w\, x_i + b,\, y_i) \right)$.
  - Update $w \leftarrow w - \eta_t \nabla(w_t, x_t, y_t) = w - \eta_t \left(\lambda w + \frac{\partial \ell(w\, x_t + b,\, y_t)}{\partial w}\right)$

Same setup as (Shalev-Schwartz et al., 2007) but plain SGD.

Text Categorization with SVMs

- **Results: Linear SVM**

  $\ell(\hat{y}, y) = \max\{0, 1 - y\hat{y}\} \quad \lambda = 0.0001$

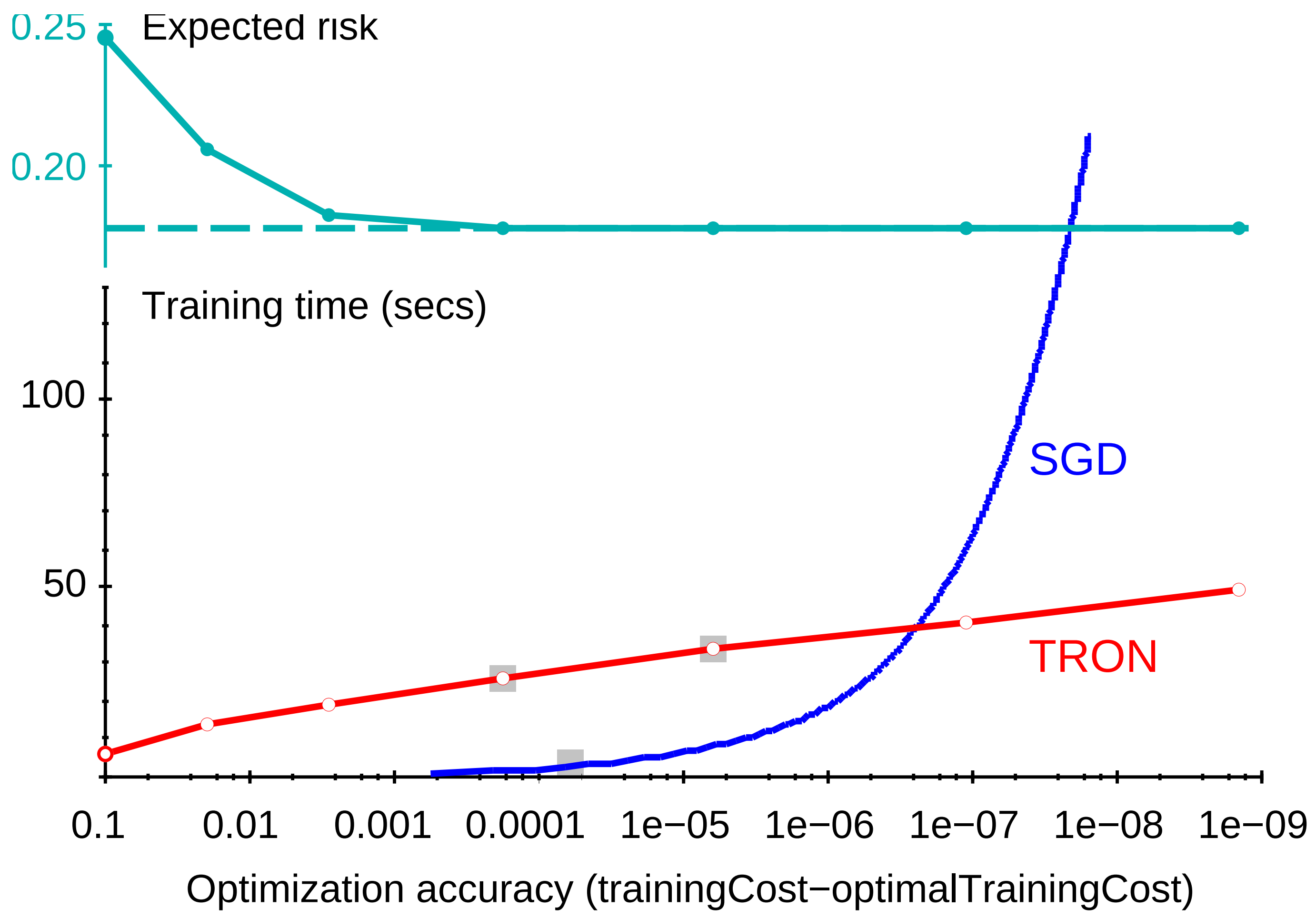
	Training Time	**Primal cost**	**Test Error**
SVMLight	23,642 secs	*0.2275*	*6.02%*
SVMPerf	66 secs	0.2278	6.03%
SGD	1.4 secs	*0.2275*	*6.02%*

- **Results: Log-Loss Classifier**

  $\ell(\hat{y}, y) = log(1 + exp(-y\hat{y})) \quad \lambda = 0.00001$

	Training Time	**Primal cost**	**Test Error**
TRON(LibLinear, $\varepsilon = 0.01$)	30 secs	0.18907	*5.68%*
TRON(LibLinear, $\varepsilon = 0.001$)	44 secs	*0.18890*	5.70%
SGD	2.3 secs	0.18893	*5.66%*

The Wall



0.25
Expected risk
0.20
Training time (secs)
100
50
SGD
TRON
0.1
0.01
0.001
0.0001
1e−05
1e−06
1e−07
1e−08
1e−09
Optimization accuracy (trainingCost−optimalTrainingCost)

More SVM Experiments

From: Patrick Haffner

Date: Wednesday 2007-09-05 14:28:50

. . . I have tried on some of our main datasets. . .

I can send you the example, it is so striking!

– Patrick

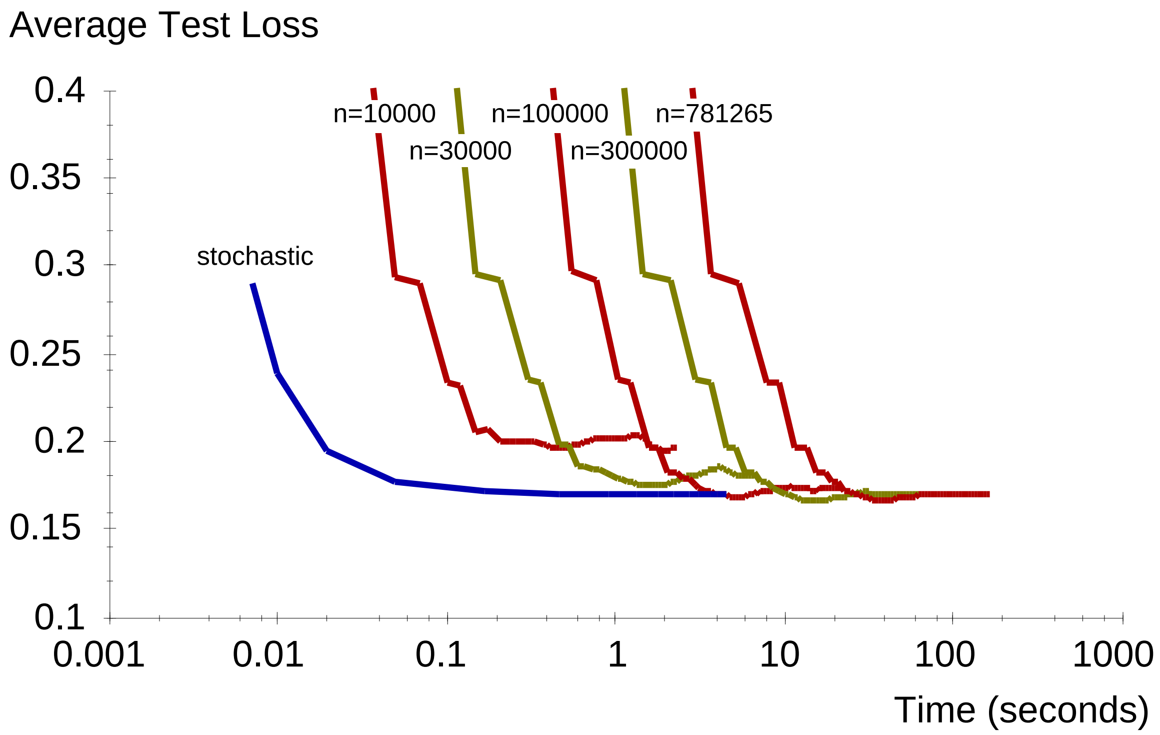
Dataset	**Train size**	**Number of features**	**% non-0 features**	**LIBSVM (SDot)**	**LLAMA SVM**	**LLAMA MAXENT**	**SGDSVM**
Reuters	781K	47K	0.1%	210,000	3930	153	7
Translation	1000K	274K	0.0033%	days	47,700	1,105	7
SuperTag	950K	46K	0.0066%	31,650	905	210	1
Voicetone	579K	88K	0.019%	39,100	197	51	1

More SVM Experiments

From: Olivier Chapelle

Date: Sunday 2007-10-28 22:26:44

. . . you should really run batch with various training set sizes . . .



Log-loss problem

Batch Conjugate Gradient on various training set sizes

Stochastic Gradient on the full set

Why is SGD near the enveloppe?

Text Chunking with CRFs

- **Dataset**
  - CONLL 2000 Chunking Task:
    Segment sentences in syntactically correlated chunks
    (e.g., noun phrases, verb phrases.)
  - 106,978 training segments in 8936 sentences.
  - 23,852 testing segments in 2012 sentences.
- **Model**
  - Conditional Random Field (all linear, log-loss.)
  - Features are n-grams of words and part-of-speech tags.
  - 1,679,700 parameters.

Same setup as (Vishwanathan et al., 2006) but plain SGD.

Text Chunking with CRFs

- **Results**

	Training Time	**Primal cost**	**Test F1 score**
L-BFGS	4335 secs	*9042*	93.74%
SGD	568 secs	9098	*93.75%*

- **Notes**
  - Computing the gradients with the chain rule runs faster than computing them with the usual forward-backward algorithm.

Choosing the Gain Schedule

Decreasing gains: $$w_{t+1} \leftarrow w_t - \gamma_0(1+\gamma_0\lambda t)^{-1}\frac{\partial Q}{\partial w}(w_t, x_t, y_t)$$

Rationale

– Gain $\gamma_t \equiv \eta t^{-1}$ leads to worst-case rate $t^{-\min\{1, 2\eta\lambda_{\min}\}}$.
We want to avoid the slow convergence case $2\eta\lambda_{\min} < 1$.

– Choosing $\eta = 1/\lambda$ works because $\lambda \geq \lambda_{\min}$.

– We are then left to choose an initial gain γ_0.

Example: the SVM benchmark

– Choose initial gain γ_0 to make sure that the expected initial updates are comparable with the expected size of the weights. When $\|x_t\| = 1$, choosing $\gamma_0 = 0.1$ works nicely.

Example: the CRF benchmark

– Choose γ_0 with the secret recipe.

The Secret Recipe

The sample size n does not change the SGD maths!

Constant gain $$w_{t+1} \leftarrow w_t - \gamma \frac{\partial Q}{\partial w}(w_t, x_t, y_t)$$

At any moment during training, we can:
- Pick a random subset of examples with moderate size.
- Try various gains γ on the subsample.
- Pick the gain γ that most reduces the cost.
- Use it for the next 100000 iterations on the full dataset.

Examples

- The CRF benchmark code does this to choose γ_0 before training.
- We could also perform such cheap measurements every so often. The selected gains would then decrease automatically.
- Do not forget to check the gradients.

Getting the Engineering Right

The very simple SGD update offers lots of engineering opportunities.

Example: Sparse Linear SVM

The update $w \leftarrow w - \eta(\lambda w + \nabla\ell(wx_i, y_i))$

can be performed in two steps:

i) $w \leftarrow w - \eta\nabla\ell(wx_i, y_i)$ (sparse, cheap)

ii) $w \leftarrow w\,(1 - \eta\lambda)$ (not sparse, costly)

- **Solution 1**

Represent vector w as the product of a scalar s and a vector v.
Perform (i) by updating v and (ii) by updating s.

- **Solution 2**

Perform only step (i) for each training example.
Perform step (ii) with lower frequency and higher gain.

VI. General Convergence Results

SGD Algorithms for everything...

Adaline (Widrow and Hoff, 1960)

$Q_{\text{adaline}} = \frac{1}{2}\big(y - w^\top \Phi(x)\big)^2$
$\Phi(x) \in \mathbb{R}^d, \;\; y = \pm 1$

$$w \leftarrow w + \gamma_t \big(y_t - w^\top \Phi(x_t)\big)\, \Phi(x_t)$$

Perceptron (Rosenblatt, 1957)

$Q_{\text{perceptron}} = \max\{0, -y\, w^\top \Phi(x)\}$
$\Phi(x) \in \mathbb{R}^d, \;\; y = \pm 1$

$$w \leftarrow w + \gamma_t \begin{cases} y_t\, \Phi(x_t) & \text{if } y_t\, w^\top \Phi(x_t) \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

Multilayer perceptrons (Rumelhart et al., 1986) ...

SVM (Cortes and Vapnik, 1995) ...

Lasso (Tibshirani, 1996)

$Q_{\text{lasso}} = \lambda |w|_1 + \frac{1}{2}\big(y - w^\top \Phi(x)\big)^2$
$w = (u_1 - v_1, \ldots, u_d - v_d)$
$\Phi(x) \in \mathbb{R}^d, \;\; y \in \mathbb{R}, \;\; \lambda > 0$

$$u_i \leftarrow \Big[u_i - \gamma_t\big(\lambda - (y_t - w^\top \Phi(x_t))\Phi_i(x_t)\big)\Big]_+$$
$$v_i \leftarrow \Big[v_i - \gamma_t\big(\lambda + (y_t - w_t^\top \Phi(x_t))\Phi_i(x_t)\big)\Big]_+$$
with notation $[x]_+ = \max\{0, x\}$.

K-Means (MacQueen, 1967)

$Q_{\text{kmeans}} = \min_k \frac{1}{2}(z - w_k)^2$
$z \in \mathbb{R}^d, \;\; w_1 \ldots w_k \in \mathbb{R}^d$
$n_1 \ldots n_k \in \mathbb{N}$, initially 0

$$k^* = \arg\min_k (z_t - w_k)^2$$
$$n_{k^*} \leftarrow n_{k^*} + 1$$
$$w_{k^*} \leftarrow w_{k^*} + \frac{1}{n_{k^*}}(z_t - w_{k^*})$$

Convergence theory summary

1. Convex case
   - General convexity
   - Three proofs
   - Convergence speed
2. Nonconvex case
   - Global confinement
   - Convergence to extremal points
3. Second order stochastic gradient descent
   - Conditions on scaling matrices

Notations

Expected Risk

$$C(w) \triangleq \mathbb{E}_z\, L(z,w) \triangleq \int L(z,w)\; dP(z)$$

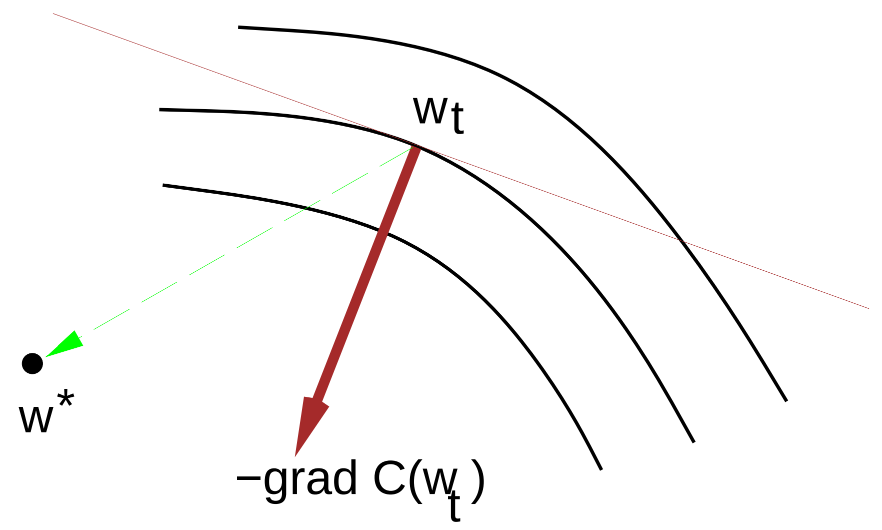
Stochastic Gradient Update

$$\begin{aligned} w_{t+1} &= w_t - \gamma_t J(\mathbf{z}_t, w_t) \\ \mathbb{E}_{\mathbf{z}}\, J(\mathbf{z}, w) &= \nabla_w\, C(w) \end{aligned}$$

Hessian

$$H(w) = \frac{\partial^2}{\partial w^2} C(w)$$

General Convexity



General convexity assumption.

$$\forall \varepsilon > 0, \qquad \inf_{(w-w^*)^2 > \varepsilon} (w - w^*)\, \nabla_w C(w) > 0$$

- Gradient point towards the right direction.
- Gradient does not vanish (no plateaus).

Learning rates

Requirement:

$$\gamma_t \, J(\mathbf{z}_t, w_t) \longrightarrow 0$$

Two possibilities:
- Decreasing learning rates: $\gamma_t \to 0$
- Decreasing gradients: $J(\mathbf{z}_t, w_t) \to 0$.

Note:

$$\mathbb{E}_z \left(J(\mathbf{z}_t, w_t)^2 \right) \approx \mathbb{E}_z \left(J(\mathbf{z}_t, w^*)^2 \right) + (w_t - w^*)' H(w^*)(w_t - w^*)$$
$$\mathbb{E}_z \left(J(\mathbf{z}_t, w_t)^2 \right) \leq A + B(w_t - w^*)^2$$

In general $A \neq 0$. Therefore learning rates must decrease.

Three convergence proofs

I will present three proofs for the following cases.
- Continuous gradient,
- Batch gradient,
- Stochastic gradient

All share the same three-step structure,
but use increasingly sophisticated tools.

Reference: Metivier (1981).

Continuous Gradient: step A.

Differential equation defines $w(t)$:

$$\frac{\mathrm{d}w}{\mathrm{d}t} = -\nabla_w C(w)$$

Step A: define Lyapunov function:

$$h(t) \ \stackrel{\triangle}{=} \ (w(t) - w^*)^2$$

Continuous Gradient: step B.

Step B: Lyapunov function converges

$$\frac{\mathrm{d}h}{\mathrm{d}t} = 2(w - w^*)\frac{\mathrm{d}w}{\mathrm{d}t} = -2(w - w^*)\nabla_w C(w) \leq 0$$

Function $h(t)$ is positive, decreasing $\Longrightarrow$ converges.

Continuous Gradient: step C.

Step C: Continuous gradient converges

Since $h(t)$ converges,

$$\frac{\mathrm{d}h}{\mathrm{d}t} = -2(w - w^*)\nabla_w C(w) \rightarrow 0$$

General convexity $\quad \Longrightarrow \quad w(t) \rightarrow w^*$

Batch Gradient: step A.

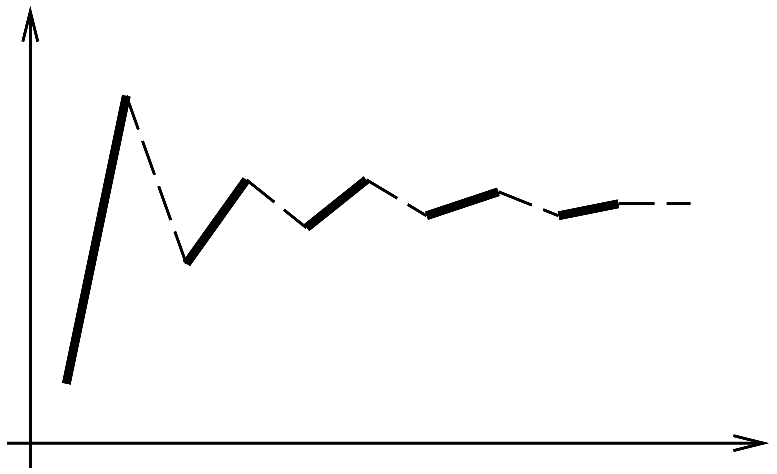
Update rule defines w_t:

$$w_{t+1} = w_t - \gamma_t \nabla_w C(w)$$

Step A: define Lyapunov sequence:

$$h_t \overset{\triangle}{=} (w_t - w^*)^2$$

Batch Gradient: lemma for step B.



$$S_t^+ = \sum_{i=1}^{t-1} [\, u_{t+1} - u_t \,]_+ \quad \text{with} \quad [\, x \,]_+ = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Bounded positive variations convergence theorem:

$$\left.\begin{array}{r} \forall t, \;\; u_t > 0 \\ S_t^+ \text{ converges} \end{array}\right\} \Longrightarrow \; u_t \text{ converges.}$$

Batch Gradient: step B.

Step B: Convergence of Lyapunov sequence:

⚠ Additional assumptions.

$$\left.\begin{array}{r} \sum \gamma_t^2 \text{ converges} \\ (\nabla_w C(w))^2 < A + B(w - w^*)^2 \end{array}\right\} \Longrightarrow h_t \text{ converges.}$$

Proof (assuming $B = 0$):

$$\begin{aligned} \left[\, h_{t+1} - h_t \,\right]_+ &= \left[-2\gamma_t \,(w_t - w^*)\nabla_w C(w_t) \,+\, \gamma_t^2 \,(\nabla_w C(w_t))^2 \right]_+ \\ &\leq \gamma_t^2 \,(\nabla_w C(w_t))^2 \quad \leq \quad \gamma_t^2 A \end{aligned}$$

and apply the bounded positive variations convergence theorem.

Proof sketch (assuming $B \neq 0$):

Define $\mu_t = \prod_{i=1}^{t} (1 - \gamma_i^2 B)$.
Show that $h'_t = h_t / \mu_t$ converges.
Then show that h_t converges as well.

Batch Gradient: step C.

Step C: Batch gradient converges.

⚠ Additional assumption:

$$\sum \gamma_t = \infty \quad \Longrightarrow \quad w_t \to w^*$$

Proof:

$$h_{t+1} - h_t \;=\; -2\gamma_t\,(w_t - w^*)\nabla_w C(w_t) \,+\, \gamma_t^2\,(\nabla_w C(w_t))^2$$

Both blue terms have convergent sums.

Therefore $\sum \gamma_i (w_i - w^*)\nabla_w C(w_i)$ converges.

The assumption then means that $(w_t - w^*)\nabla_w C(w) \to 0$.

The general convexity hypothesis then implies the convergence.

Stochastic Gradient: step A.

Stochastic update rule defines w_t:

$$w_{t+1} = w_t - \gamma_t J(\mathbf{z}_t, w_t)$$

Step A: define Lyapunov process:

$$h_t \stackrel{\triangle}{=} (w_t - w^*)^2$$

Stochastic Gradient: Quasi-martingales.

Let u_t be a stochastic process.
Let $\mathcal{P}_t$ represent what is known on time t, ($u_0, \ldots, u_t$.)

Define indicator of expected positive differences.

$$\delta_t = \begin{cases} 1 & \text{if } \mathbb{E}(u_{t+1} - u_t | \mathcal{P}_t) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

and the sum

$$S_t^+ = \sum_{i=1}^{t} \mathbb{E}(\delta_t \left(u_{t+1} - u_t\right))$$

Positive quasi-martingale convergence theorem:

$$\left. \begin{array}{r} \forall t,\ \ u_t > 0 \\ S_t^+ \text{ converges} \end{array} \right\} \implies u_t \text{ converges almost surely.}$$

Metivier: *Semi-martingales*, 1983, Theorem 9.4, Proposition 9.5

Stochastic Gradient: step B.

Step B: Convergence of Lyapunov process:

⚠ Additional assumptions.

$$\left.\begin{array}{r} \sum \gamma_t^2 \text{ converges} \\ \mathbb{E}\left(J(\mathbf{z},w)^2\right) < A + B(w-w^*)^2 \end{array}\right\} \Longrightarrow \quad h_t \text{ converges a.s.}$$

Proof (assuming $B=0$):

$$\begin{aligned} \mathbb{E}\left(h_{t+1}-h_t \mid \mathcal{P}_t\right) &= -2\,\gamma_t(w_t-w^*)\,\mathbb{E}\left(J(\mathbf{z}_t,w_t)\mid\mathcal{P}_t\right) + \gamma_t^2\,\mathbb{E}\left(J(\mathbf{z}_t,w_t)^2\mid\mathcal{P}_t\right) \\ &= -2\,\gamma_t(w_t-w^*)\nabla_w C(w_t) + \gamma_t^2\,\mathbb{E}_{\mathbf{z}}(J(\mathbf{z}_t,w_t)^2) \\ &\leq \gamma_t^2\,A \end{aligned}$$

$$\mathbb{E}(\delta_t(h_{t+1}-h_t)) = \mathbb{E}(\delta_t\mathbb{E}\left(h_{t+1}-h_t\mid\mathcal{P}_t\right)) \leq \mathbb{E}\left(\mathbb{E}\left(h_{t+1}-h_t\mid\mathcal{P}_t\right)\right) \leq \gamma_t^2\,A$$

and apply the quasi-martingale convergence theorem.

The case $B\neq 0$ can be treated as suggested for batch gradient.

Stochastic Gradient: step C.

Step B: Stochastic gradient converges almost surely.

⚠ Additional assumption:

$$\sum \gamma_t = \infty \quad \Longrightarrow \quad w_t \xrightarrow{a.s.} w^*$$

Proof:

$$\mathbb{E}\left(h_{t+1} - h_t\right) \;=\; -2\gamma_t\,\mathbb{E}\left((w_t - w^*)\nabla_w C(w_t)\right) + \gamma_t^2\,\mathbb{E}_{\mathbf{z}}(J(\mathbf{z}_t, w_t)^2)$$

Both blue terms have convergent sums.

Therefore $\sum \gamma_i \mathbb{E}\left((w_i - w^*)\nabla_w C(w_i)\right)$ converges.

The assumption then means that $\liminf (w_t - w^*)\nabla_w C(w) = 0$ (a.s.)

The general convexity hypothesis then implies the convergence.

Nonconvex case

Why worry about nonconvex objective functions?

– Multilayer networks.
– Clustering algorithms.
– Learning features.
– Semi-supervised learning.
– Mixture models.
– Hidden Markov Models.
– Selecting features (some).
– Transfer learning.

Nonconvexity issues

- Several local minima.
- Possibly several global minima.
- Critical subspaces.
- Saddle points.

Local confinement

- Partition space into attraction basins.
- Make sure w_t is confined to a specific basin.
- Define suitable Lyapunov functions on the basin

Reference: Krasovskii (1963)

Batch gradient: Confinement works well.
Stochastic gradient: Confinement works poorly.

Stochastic noise can always get w_t out of any
attraction basin with small but non zero probability.

Standard assumptions

$$C(w) > 0$$

$C(w)$ has continuous second derivatives.

$$\sum_i \gamma_i = \infty$$

$\sum_i \gamma_i$ converges.

$$\mathbb{E}_{\mathbf{z}}\left(J(\mathbf{z}, w)^2\right) \leq A + Bw^2$$

Global confinement

⚠ Additional Assumptions

$$\left.\begin{array}{lll} \exists D > 0, & \displaystyle\inf_{w^2>D} & w\, \nabla_w C(w) > 0 \\ \exists E > D, & \displaystyle\sup_{w^2<E} & J(\mathbf{z},w)^2 < K \end{array}\right\} \Longrightarrow \mathbf{P}\left\{\exists t_0, \forall t > t_0,\ w_t^2 < 2E\right\} = 1$$

Proof sketch:

Define Lyapunov process $h_t = \max(E, w_t^2)$.

Performs steps a, b, and c. Conclude.

Benefits:

We are now working in a compact set $\{w_t^2 < E\}$.

Continuity implies that all continous quantities are bounded a.s.

Convergence to Extremal Points

$$\ldots \Longrightarrow \begin{cases} C(w_t) \text{ converges to some value } C_\infty. \\ \nabla_w C(w_t) \xrightarrow{a.s.} 0 \end{cases}$$

Proof sketch:

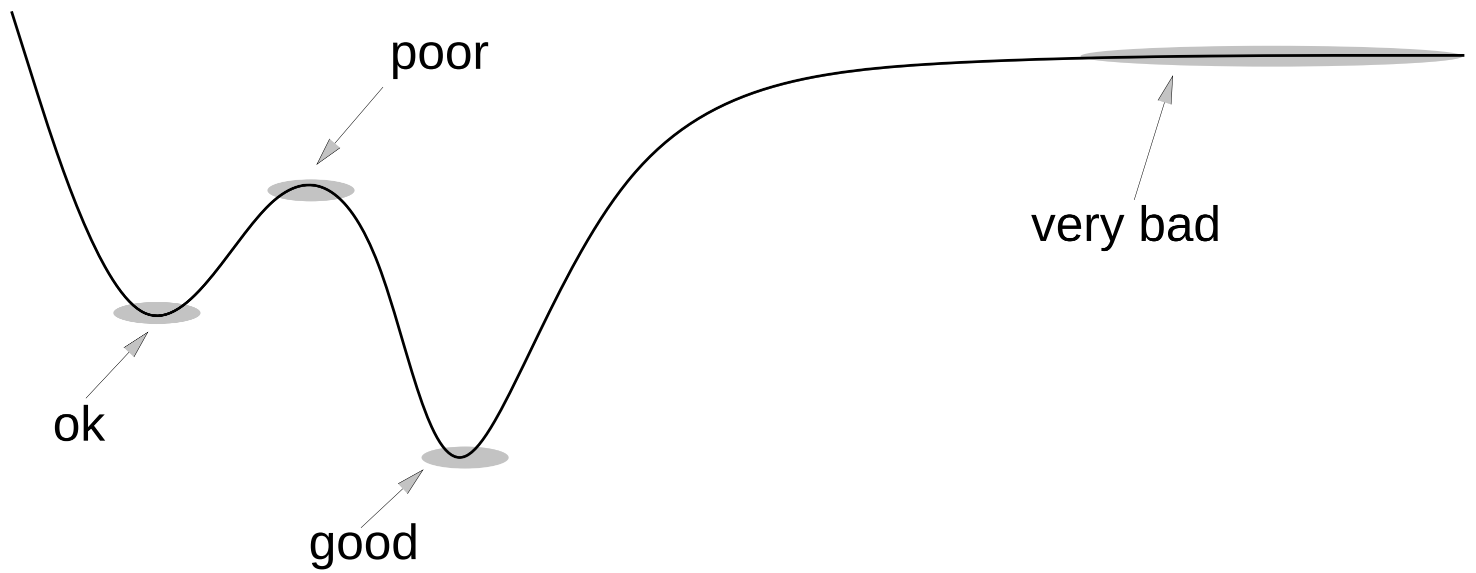
a- Define Lyapunov process $h_t = C(w_t)$.
b- Lyapunov process converges.
c- Define secondary Lyapunov process $g_t = (\nabla_w C(w_t))^2$.
Perform steps a, b, and c again.

Note:

We must either use global confinement

or assume suitable bounds on $||\nabla\nabla C(w)||$.

Convergence to Extremal Points



- Local maxima and saddle points are unstable.

  But one can construct cases whithout noise.

- Infinite plateaus are ruled out by global confinement.

  Otherwise they can spoil the day.

Second Order Stochastic Gradient Descent

Second Order Stochastic Gradient Update

$$\begin{aligned} w_{t+1} &= w_t - \Gamma_t J(\mathbf{z}_t, w_t) \\ \mathbb{E}_{\mathbf{z}} \, J(\mathbf{z}, w) &= \nabla_w \, C(w) \end{aligned}$$

Gain matrix

The gain is now a positive definite matrix Γ_t.
The idea is to have $\Gamma_t \approx \frac{1}{t} H(w^*)^{-1}$.
But convergence occurs under much weaker assumptions.

Assumptions

Gain Assumptions

Let $0 < \lambda_t^{\min} \leq \lambda_t^{\max}$ bracket the eigenvalues of Γ_t.

$$\sum_t \lambda_t^{\min} = +\infty$$

$$\sum_t (\lambda_t^{\max})^2 \quad \text{converges.}$$

More Assumptions

$$C(w) \quad \text{is positive}$$

$$C(w) \quad \text{has bounded second derivatives}$$

$$\mathbb{E}_{\mathbf{z}} \left(J(\mathbf{z}, w)^2 \right) \leq A + Bw^2$$

Convergence

$$\cdots \Longrightarrow \begin{cases} C(w_t) \text{ converges to some value } C_\infty. \\ \nabla_w C(w_t) \xrightarrow{a.s.} 0 \end{cases}$$

Proof sketch:

a- Define Lyapunov process $h_t = C(w_t)$.

b- Lyapunov process converges.

c- Define secondary Lyapunov process $g_t = (\nabla_w C(w_t))^2$.
Perform steps a, b, and c again.

Remarks:

– We did not say anything about the determination of Γ_t.

Any reasonable approximation of the inverse Hessian works!

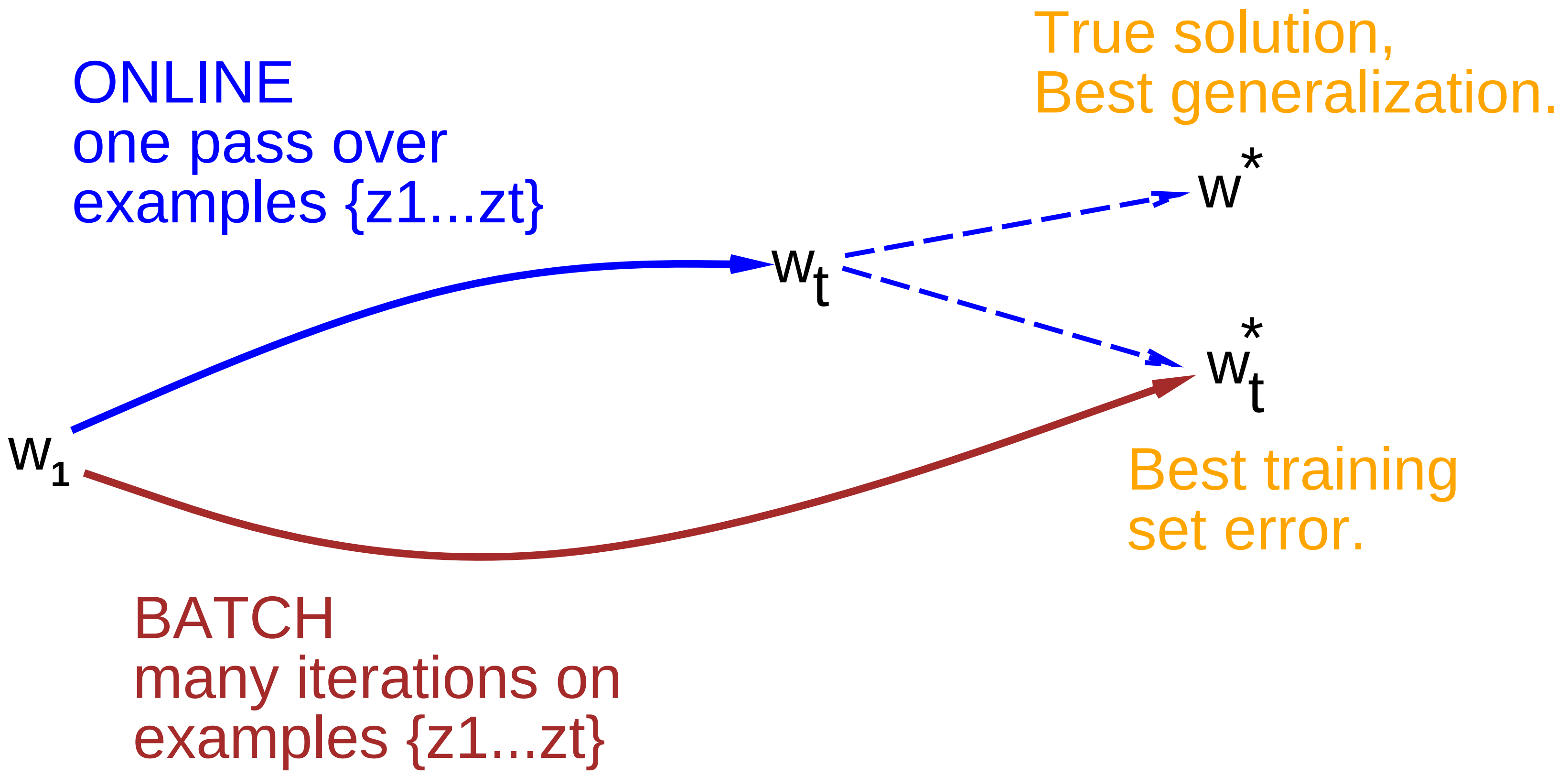
Conclusion

We have a very well oiled machinery to establish the
convergence of stochastic gradient algorithms
under mild conditions.

General convergence results apply to:
- adalines
- multilayer networks
- perceptrons
- kmeans
- lvq2
- . . .

VII. Learning with a Single Epoch

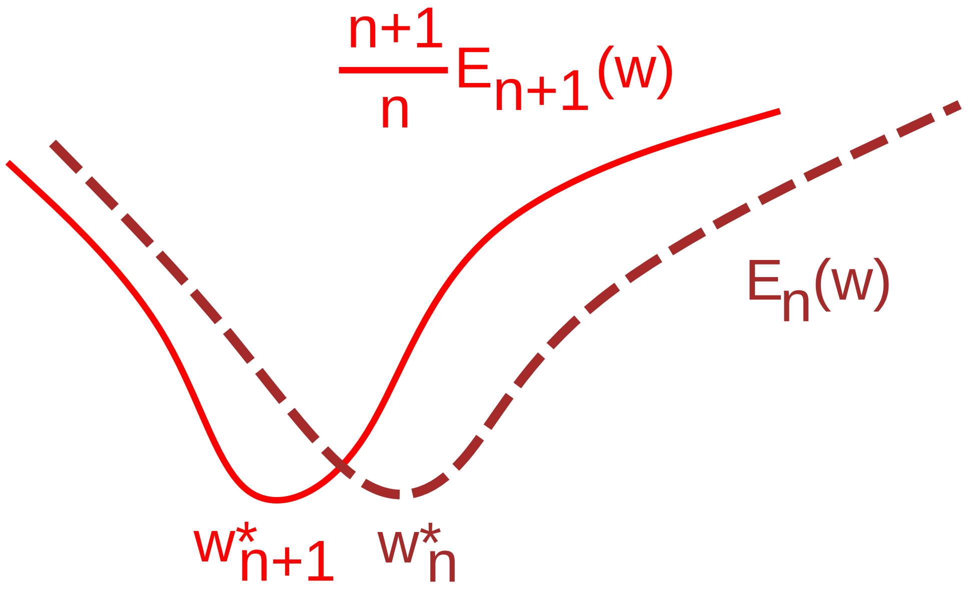
Batch and online paths



Effect of one Additional Example (i)

Compare

$$\begin{aligned} w_n^* &= \arg\min_w E_n(w) \\ w_{n+1}^* &= \arg\min_w E_{n+1}(w) = \arg\min_w \left[E_n(w) + \frac{1}{n}\ell(f(x_{n+1},w),y_{n+1}) \right] \end{aligned}$$



Effect of one Additional Example (ii)

First Order Calculation

$$w^*_{n+1} = w^*_n - \frac{1}{n} H^{-1}_{n+1} \frac{\partial\, \ell(f(x_n, w_n), y_n)}{\partial w} + \mathcal{O}\left(\frac{1}{n^2}\right)$$

where H_{n+1} is the empirical Hessian on $n+1$ examples.

Compare with Second Order Stochastic Gradient Descent

$$w_{t+1} = w_t - \frac{1}{t} H^{-1} \frac{\partial\, \ell(f(x_n, w_n), y_n)}{\partial w}$$

Could they converge with the same speed?

– We need to generalize our simple one-dimensional analysis.

Speed of Scaled Stochastic Gradient (i)

Scaled Stochastic Gradient

$$w_{t+1} \;=\; w_t \;-\; \frac{1}{t} B_t \frac{\partial\, \ell(f(x_n, w_t), y_n)}{\partial w} \;+\; \mathcal{O}\!\left(\frac{1}{t^2}\right)$$

with $B_t \to B \succ 0$, $BH \succ I/2$,

and H is the Hessian in w^*.

1- Establish convergence a.s.

– Using the general convergence results.

2- Define error term

– Let $U_t \;\overset{\triangle}{=}\; H\,(w_t - w^*)\,(w_t - w^*)^\top$.

– Observe $E(w_t) - E(w^*) = \mathrm{tr}(U_t) + \mathrm{o}(\mathrm{tr}(U_t))$

Speed of Scaled Stochastic Gradient (ii)

3- Derive error recursion

$$\mathbb{E}\left[U_{t+1}\right] = \left[I - \frac{2BH}{t} + \mathrm{o}\left(\frac{1}{t}\right)\right] \mathbb{E}\left[U_t\right] + \frac{HBGB}{t^2} + \mathrm{o}\left(\frac{1}{t^2}\right)$$

where the Fisher matrix

$$G \triangleq \int \left[\left(\frac{\partial \ell(f(x,w^*),y)}{\partial w}\right) \left(\frac{\partial \ell(f(x,w^*),y)}{\partial w}\right)^{\top} \right] dP(x,y)\,.$$

4- Establish lemma on sequences satisfying

$$u_{t+1} = \left(1 + \frac{\alpha}{t} + \mathrm{o}\left(\frac{1}{t}\right)\right) u_t + \frac{\beta}{t^2} + \mathrm{o}\left(\frac{1}{t^2}\right)$$

– When $\alpha > 1$ show $u_t = \frac{\beta}{\alpha-1}\frac{1}{t} + \mathrm{o}\left(\frac{1}{t}\right)$ (nasty proof!).

– When $\alpha < 1$ show $u_t \sim t^{-\alpha}$ (up to log factors).

Speed of Scaled Stochastic Gradient (iii)

5- Bracket $\mathbb{E}\left[\mathrm{trace}\left(()\, U_{t+1}\right)\right]$

– Using two sequences of the aforementioned type.

– The idea is to bracked the eigenvalues of BH.

6- Apply the lemma and conclude:

$$\boxed{\begin{aligned}&\frac{\mathrm{tr}(HBGB)}{2\lambda_{BH}^{\max}-1}\,\frac{1}{t}+\mathrm{o}\left(\frac{1}{t}\right)\\ &\qquad\leq\ \mathbb{E}\Big[E(w_t)-E(w^*)\Big]\ \leq\\ &\qquad\qquad\frac{\mathrm{tr}(HBGB)}{2\lambda_{BH}^{\min}-1}\,\frac{1}{t}+\mathrm{o}\left(\frac{1}{t}\right)\end{aligned}}$$

– Note that we know the constants.

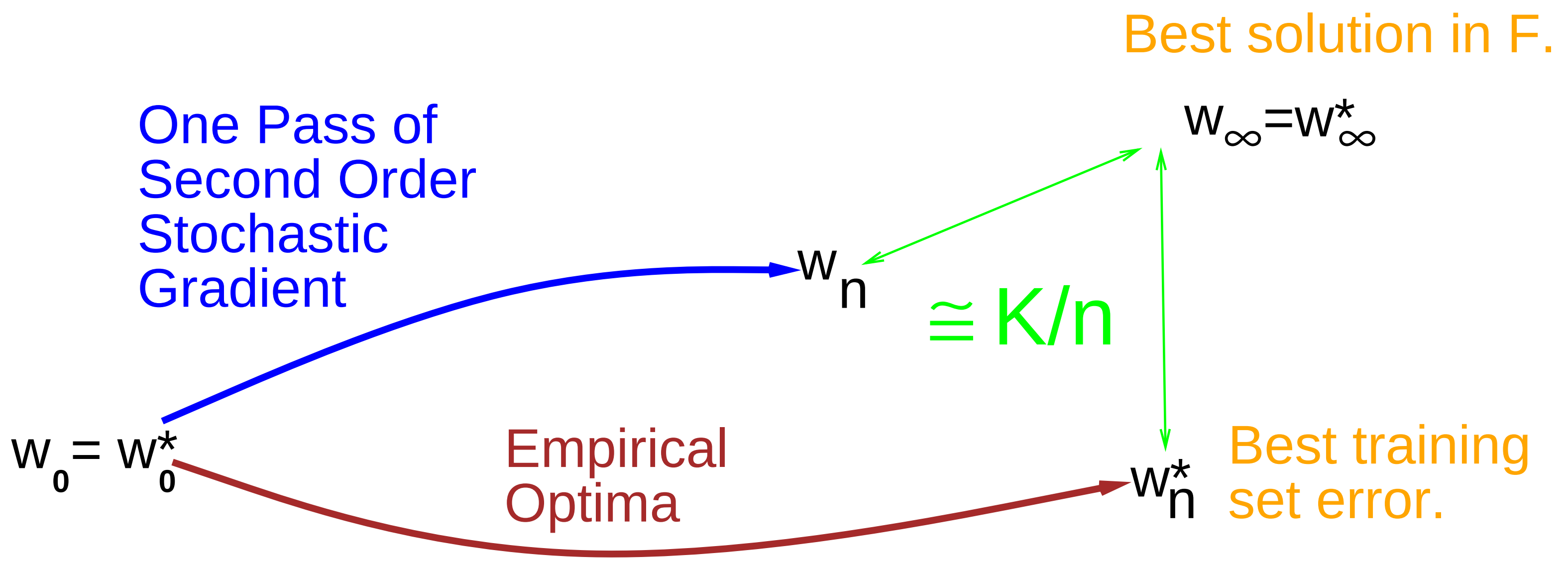
– Interesting special cases: $B = I/\lambda_H^{\min}$ and $B = H^{-1}$.

Asymptotic Efficiency of 2SGD.

“Empirical optima” “Second-order SGD”

$$n\,\mathbb{E}\big[E(f_{w_n^*}) - E(f_{\mathcal{F}})\big] = \lim_{t\to\infty} t\,\mathbb{E}\big[E(f_{w_t}) - E(f_{\mathcal{F}})\big]$$

$$\lim_{n\to\infty} n\,\mathbb{E}\big[\|w_\infty^* - w_n^*\|^2\big] = \lim_{t\to\infty} t\,\mathbb{E}\big[\|w_\infty - w_t\|^2\big]$$

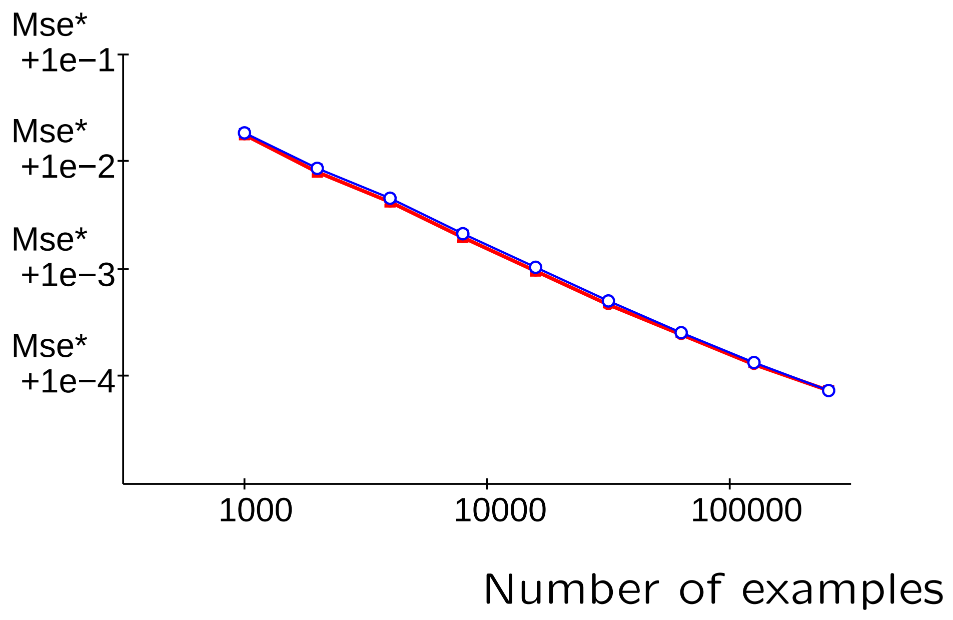


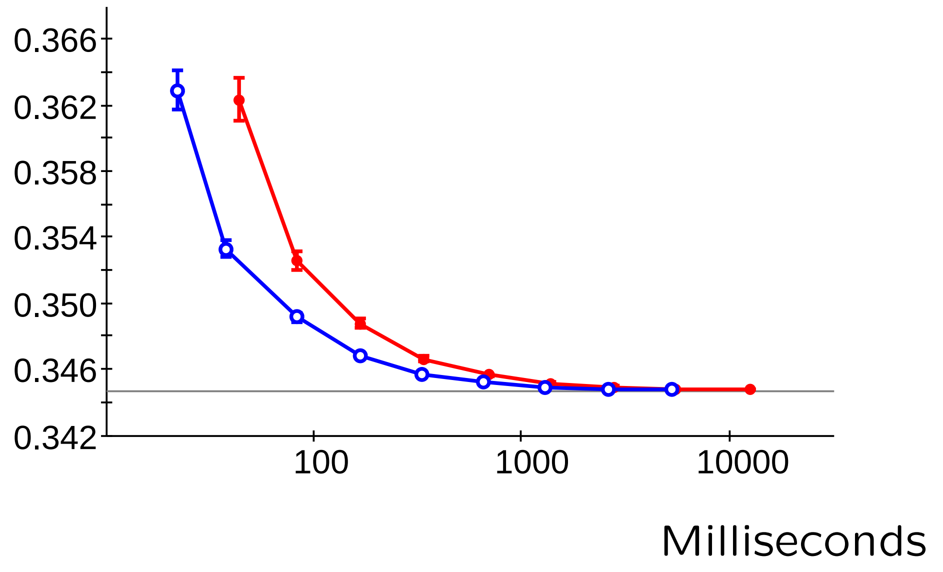
(Fabian, 1973; Murata & Amari, 1998; Bottou & LeCun, 2003).

Optimal Learning in One Pass

A Single Pass of Second Order Stochastic Gradient generalizes as well as the Empirical Optimum.

Experiments on synthetic data





Unfortunate Issue

Second order SGD is costly

Repeat: (a) Pick random example x_t, y_t

(b) $w \leftarrow w - \gamma_t H^{-1} \dfrac{\partial \ell(f(x_t, w), y_t)}{\partial w}$

– Estimate and store $d \times d$ matrix H^{-1}.

– Multiply the gradient for each example by this matrix H^{-1}.

Solutions

Limited storage approximations of H^{-1}

– Diagonal Gauss-Newton (Becker and Lecun, 1989)

– Low rank approximation [oLBFGS], (Schraudolph et al., 2007)

– Diagonal approximation [SGDQN], (Bordes et al., 2009)

Averaged stochastic gradient

– Perform SGD with slowly decreasing gains, e.g. $\gamma_t \sim t^{-0.75}$.

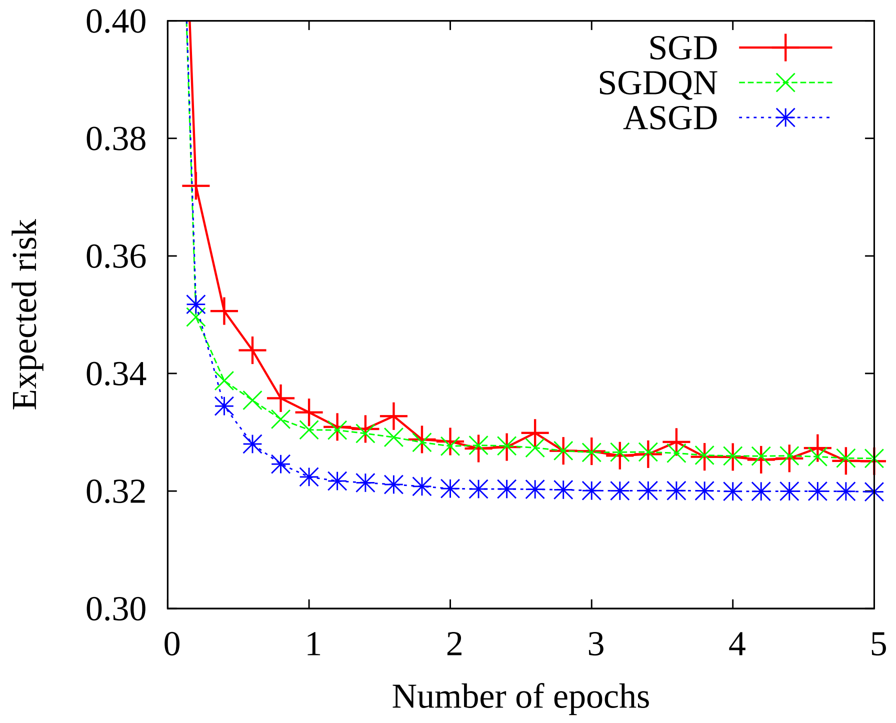
– Compute averages $\bar{w}_{t+1} = \frac{t}{t+1}\bar{w}_t + \frac{1}{t+1}w_{t+1}$

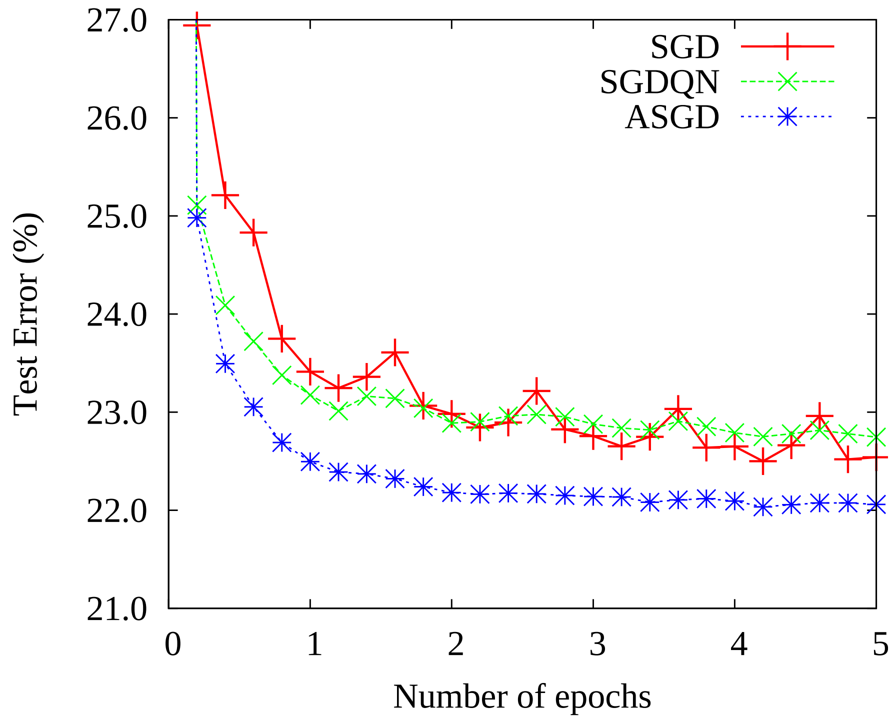
– Same asymptotic speed as second order SGD (Polyak and Juditsky, 92)

– Can take a while to "reach" the asymptotic regime.

Experiment: ALPHA dataset

– From the 2008 Pascal Large Scale Learning Challenge.
– Loss: $\ell(\hat{y}, y) = (\max\{0, 1 - y\,\hat{y})^2$.

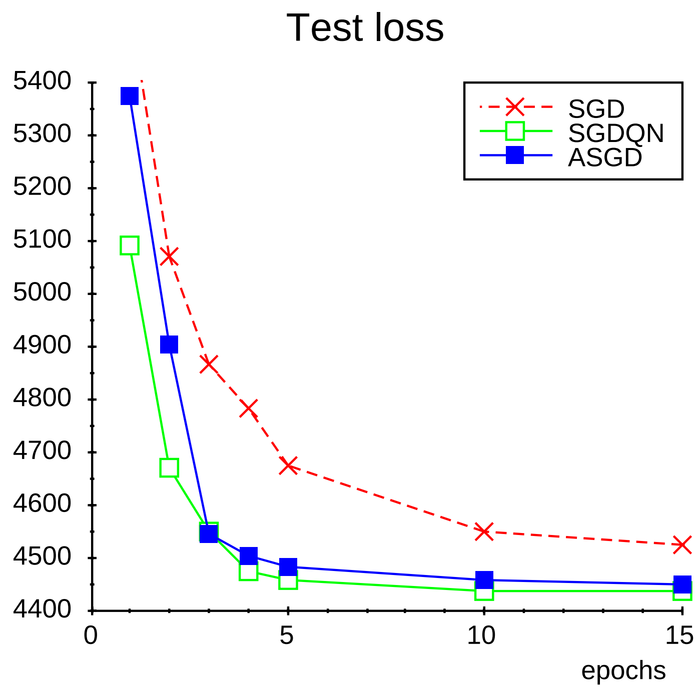


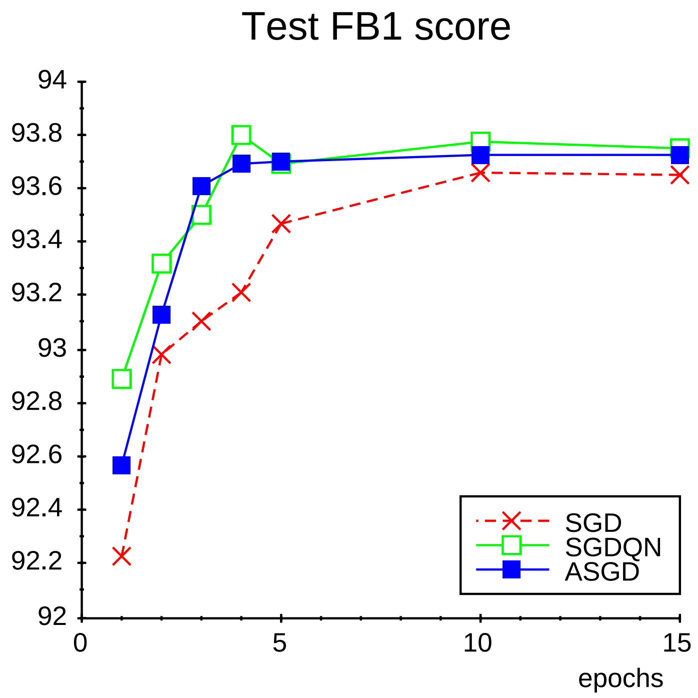


Optimal expected risk after a single epoch!

Experiment: Conditional Random Field

– CRF for the CONLL 2000 Chunking task.
– 1.7M parameters. 107,000 training segments.





SGDQN more attractive than ASGD.
Training times: 500s (SGD), 150s (ASGD), 75s (SGDQN).
Standard LBFGS optimizer needs 72 minutes.

Getting the Engineering Right

Gains

– SGD $\gamma_t = \gamma_0(1 + \gamma_0 \lambda t)^{-1}$.
– SGDQN $\gamma_t = \gamma_0(1 + \gamma_0 \lambda t)^{-1}$.
– ASGD: $\gamma_t = \gamma_0(1 + \gamma_0 \lambda t)^{-0.75}$

Sparsity preserving implementation

–SGD: Write $w_t = s_t W_t$ with $s_t = \prod_{i=0}^{t-1}(1 - \gamma_t \lambda)$.

–ASGD: Write $w_t = s_t W_t$ with $s_t = \prod_{i=0}^{t-1}(1 - \gamma_t \lambda)$
and $\bar{w}_t = (U_t + r_t W_t)/t$ with $r_t = \sum_{i=1}^{t} s_t$.

–SGDQN: Decoupled loss and regularization updates.

ASGD tricks: (Wei Xu, 2010)

Learning in a single epoch

One epoch learning in practice

– For convex objective functions,
  given a sufficiently large dataset,
  one pass learning can be achieved in practice.

Other considerations

– Shuffling the data can be hard.
– How large a dataset we need to reach this regime?

Reduced claim

– A couple epochs should suffice.
  (e.g., five. . .)

VIII. SGD for Neyman-Pearson classification

with Gilles Gasso, Aristidis Pappaiaonnou, and Marina Spivak.

ACM TIST 2(3), 2011.

Asymmetric cost problem

Binary classification.

– Positive class $y = +1$, negative class $y = -1$.

Examples of positive classes.

– fraudulent credit card transaction

– relevant document for a given query

– heart failure detection

Different kinds of errors have different costs.

– False positive, false detection, false alarm.

– False negative, non detection.

Imbalanced datasets.

Asymmetric cost problem

Misclassification probabilities

– Probability of non detection (false negatives)

$$\mathrm{P}_{\mathsf{nd}}(f) \;\triangleq\; \mathbb{P}\left\{ f(x) \leq 0 \,\middle|\, y = +1 \right\}$$

– Probability of false alarm (false positives)

$$\mathrm{P}_{\mathsf{fa}}(f) \;\triangleq\; \mathbb{P}\left\{ f(x) \geq 0 \,\middle|\, y = -1 \right\}$$

Asymmetric classification (AC)

$$\min_{f} C_+ \,\mathrm{P}_{\mathsf{nd}}(f) + C_- \,\mathrm{P}_{\mathsf{fa}}(f)$$

Neyman-Pearson Classification

Costs are difficult to assess

– Example: the pacemaker problem.

Neyman-Pearson classification (NP)

– It is often more convenient to solve

$$\min_f \mathrm{P}_{\text{nd}}(f) \quad \text{subject to} \quad \mathrm{P}_{\text{fa}}(f) < \rho$$

AC and NP classification share the same solutions

- Let $r^*(x) = \mathbb{P}\{y = +1|x\}$

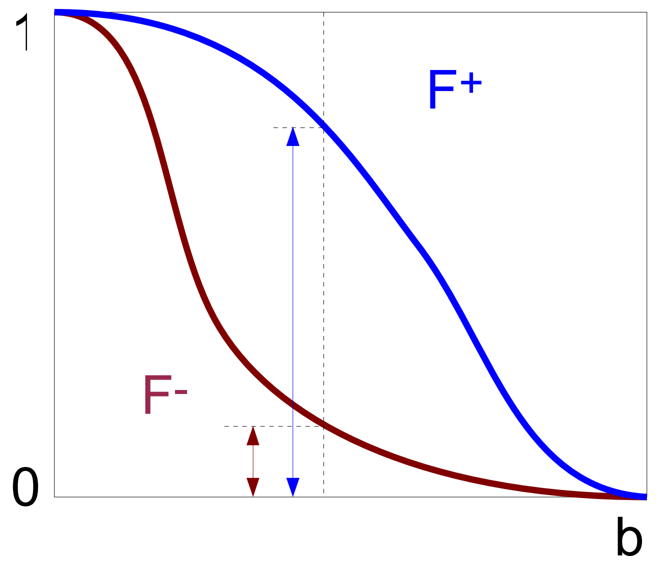
$$\begin{aligned} f^*_{AC}(x) &= r^*(x) - C_-/(C_+ + C_-) \\ f^*_{NP}(x) &= r^*(x) - \min\{r \ \text{ such that } \ \mathrm{P}_{\text{fa}}(r^* - r) < \rho\} \end{aligned}$$

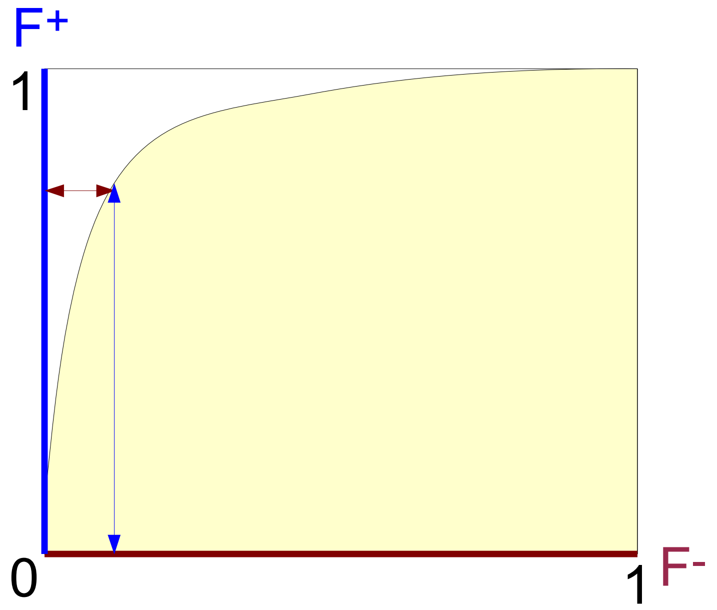
Adjusting only the Decision Threshold

Decision threshold

- Discriminant function $f_b(x) = f(x) - b$.
- The function f is fixed; we are only adjusting the threshold b.
- True positives: $F_+(b) = \mathbb{P}\{f(x) - b \geq 0 | Y = +1\} \; = \; 1 - \mathrm{P}_{\mathrm{nd}}(f_b)$
- False positives: $F_-(b) = \mathbb{P}\{f(x) - b \geq 0 | Y = -1\} \; = \; \mathrm{P}_{\mathrm{fa}}(f_b)$

ROC curve





Learning the classifier

Empirical counterparts of P_{nd} and P_{fa}

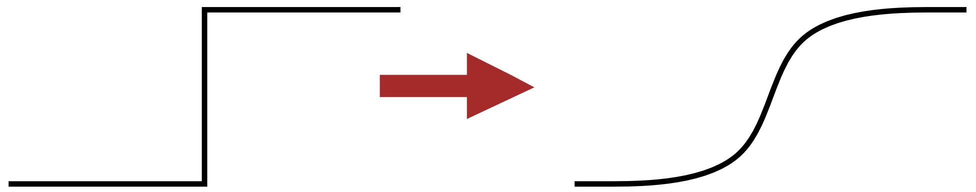
$$\tilde{\mathrm{P}}_{\mathsf{nd}}(f) = \frac{1}{n_+} \sum_{y_i > 0} \mathbb{I}\{f(x_i) \leq 0\} \qquad \tilde{\mathrm{P}}_{\mathsf{fa}}(f) = \frac{1}{n_-} \sum_{y_i < 0} \mathbb{I}\{f(x_i) \geq 0\}$$

Empirical Neyman-Pearson problem

$$\min_w \tilde{\mathrm{P}}_{\mathsf{nd}}(f_w) \;\; \text{subject to} \;\; \tilde{\mathrm{P}}_{\mathsf{fa}}(f_w) < \rho$$

The step functions make a very hard optimization problem.

Learning the classifier



$$\hat{\mathrm{P}}_{\mathsf{nd}}(f) = \frac{1}{n_+} \sum_{y_i>0} \sigma(y_i\, f(x_i))\} \qquad \hat{\mathrm{P}}_{\mathsf{fa}}(f) = \frac{1}{n_-} \sum_{y_i<0} \sigma(y_i\, f(x_i))\}$$

Empirical Neyman-Pearson problem revisited

$$\min_w \hat{\mathrm{P}}_{\mathsf{nd}}(f_w) \;\; \text{subject to} \;\; \hat{\mathrm{P}}_{\mathsf{fa}}(f_w) < \rho$$

Regularized variant

$$\min_w \frac{\lambda}{2}\|w\|^2 + \hat{\mathrm{P}}_{\mathsf{nd}}(f_w) \;\; \text{subject to} \;\; \hat{\mathrm{P}}_{\mathsf{fa}}(f_w) < \rho$$

Nonconvex objective function with nonconvex constraints

Nonconvex optimization with constraints

Lagrangian

$$\mathcal{L}(f, \mu) = \frac{\lambda}{2}\Omega(f) + \hat{\mathrm{P}}_{\mathsf{nd}}(f) + \mu\big(\hat{\mathrm{P}}_{\mathsf{fa}}(f) - \rho\big) \qquad \mu \geq 0$$

Two useful theorems

– The local saddle points of $\mathcal{L}$ are feasible local minima of the constrained problem.

– Assuming differentiability and KKT qualification the feasible local minima of the constrained problem are critical points of the Lagrangian $\mathcal{L}$. (Ciarlet, 1989)

Let us find local saddle points of $\mathcal{L}$ . . .

Uzawa algorithm

Uzawa algorithm

– Set initial values for f, $\mu > 0$.

– Choose very small gain ν.

– Repeat

$$\begin{aligned} f &\leftarrow \arg\min_f \mathcal{L}(f,\mu) \\ \mu &\leftarrow \mu\left(1+\nu\frac{\partial\mathcal{L}(f,\mu)}{\partial\nu}\right) \end{aligned}$$

Convergence of Uzawa algorithm

– Nonobvious because $f^*_\mu = \arg\min_f \mathcal{L}(f,\mu)$ can be noncontinuous.

– However we can show that $\hat{\mathrm{P}}_{\mathsf{fa}}(f^*_\mu)$ is a nonincreasing function of μ.

– Therefore the sign of $\frac{\partial\mathcal{L}(f,\mu)}{\partial\nu}$ indicates the right direction.

Stochastic NP algorithm

Stochastic NP algorithm

– Set initial values for f, $\mu > 0$.

– Choose decreasing gains γ_t.

– Choose very small gain ν.

– Repeat

- Pick random training example x_t, y_t
- Set $a_t = \begin{cases} n/n_+ & \text{if } y_i = +1 \\ \mu n/n_- & \text{if } y_i = -1 \end{cases}$
- Update $w \leftarrow (1 - \gamma_t \lambda) w - \gamma_t \, a_t \, \sigma'(y_t f_w(x_t)) \, \frac{\partial f}{\partial w}(x_t)$
- If $y_i = -1$ update $\mu \leftarrow \mu \, (1 + \nu \, (\sigma(y_t f_w(x_t)) - \rho))$

Convergence, etc. . .

– This is not a gradient descent algorithm (saddle point $\neq$ minimum.)

– Such stochastic approximations have been studied extensively

e.g. (Tsypkin, 1971; Andrieu et al. 2007)

Rebalancing the data

Imbalanced Dataset

– Suppose positive examples from a small proportion of the training set. (Asymmetric costs and imbalanced data often come together.)

Rebalancing

– Instead we draw positive and negative examples with equal probabilities.
– $\hat{\mathrm{P}}_{\text{nd}}$, $\hat{\mathrm{P}}_{\text{fa}}$ are not affected: they depend on $\mathbb{P}\{f(x)|y\}$, not $\mathbb{P}\{y\}$.
– Lagrange coefficient μ auto-adjusts to satisfy the constraint.

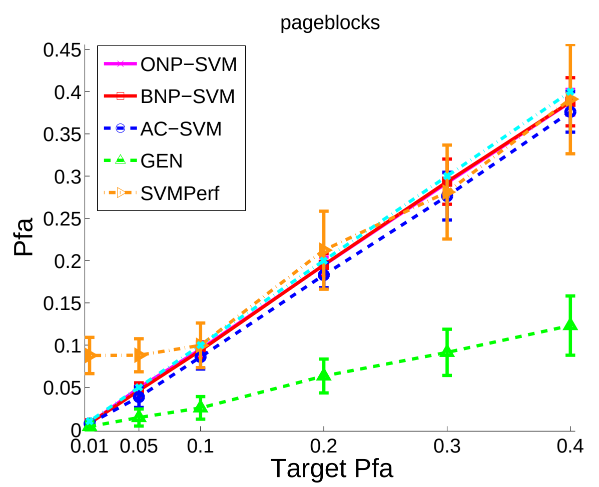
$$\mathcal{L}(f,\mu) = \frac{\lambda}{2}\Omega(f) + \hat{\mathrm{P}}_{\text{nd}}(f) + \mu\big(\hat{\mathrm{P}}_{\text{fa}}(f) - \rho\big) \qquad \mu \geq 0$$

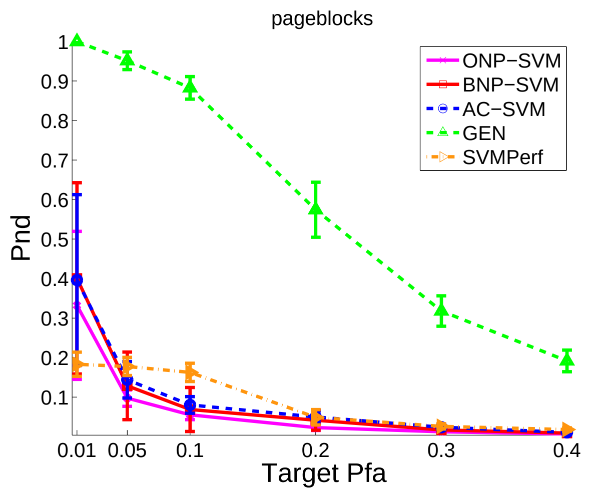
– Potential improvement: adjust the initial value of μ.
Equivalently: redefine $a_t = \begin{cases} 1 & \text{if } y_i = +1 \\ \mu & \text{if } y_i = -1 \end{cases}$

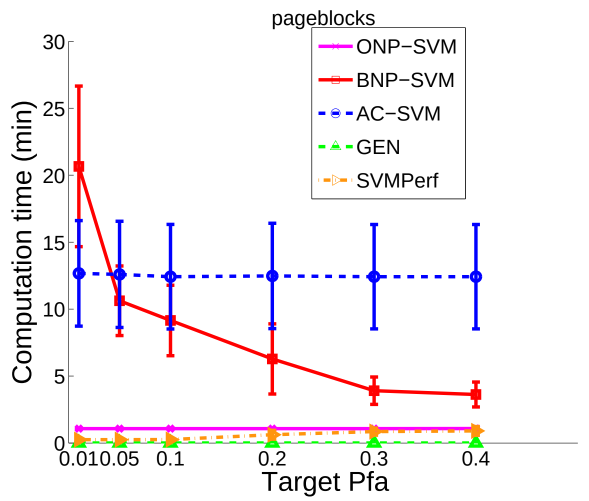
Results: Pageblocks

10 features, 4913 negatives, 560 positives.

Linear, 50% training set, 25% validation set, 25% test set.







ONP-SVM: stochastic Neyman Pearson classification.

BNP-SVM: batch Neyman Pearson.

AC-SVM: asymetric cost SVMs (Davenport et al.,2010)

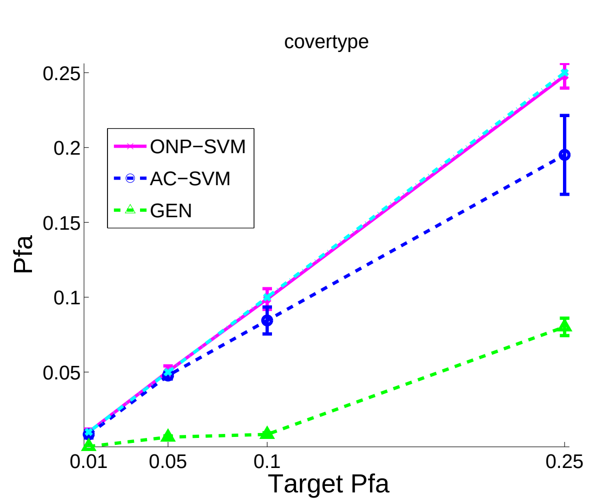
GEN: generative method (Huang et al.,2006)

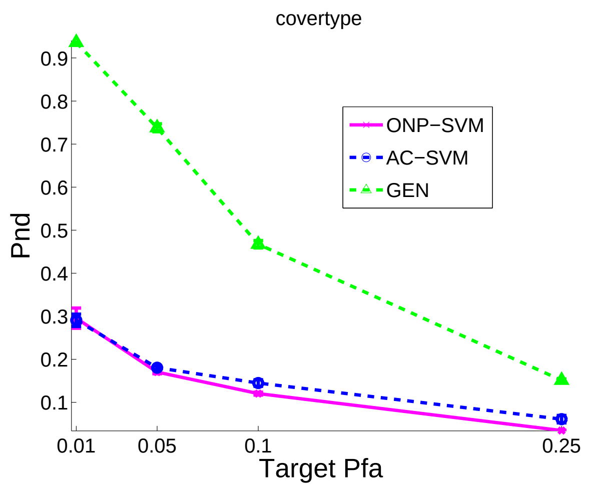
SVMPerf: (Joachims, 2005)

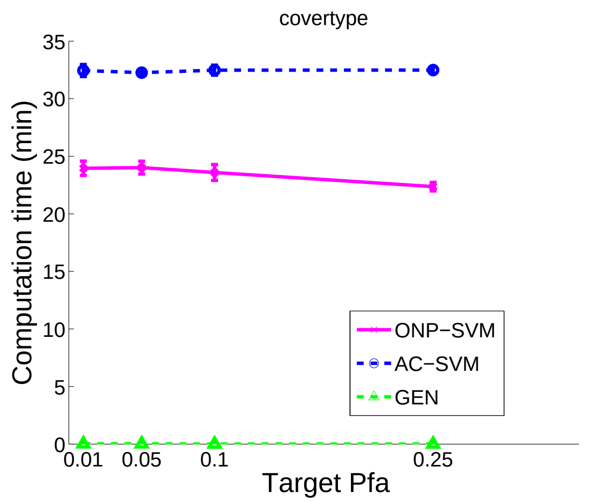
Results: Covertype

54 features, 211840 negatives, 20510 positives.

Linear, 50% training set, 25% validation set, 25% test set.







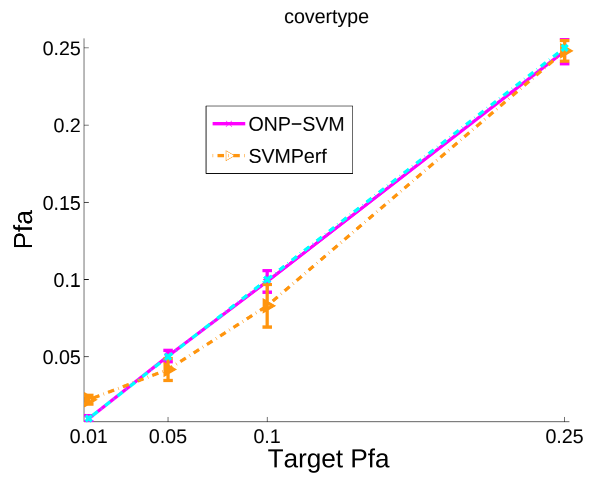
ONP-SVM: stochastic Neyman Pearson classification.
AC-SVM: asymetric cost SVMs (Davenport et al.,2010)
GEN: generative method (Huang et al.,2006)

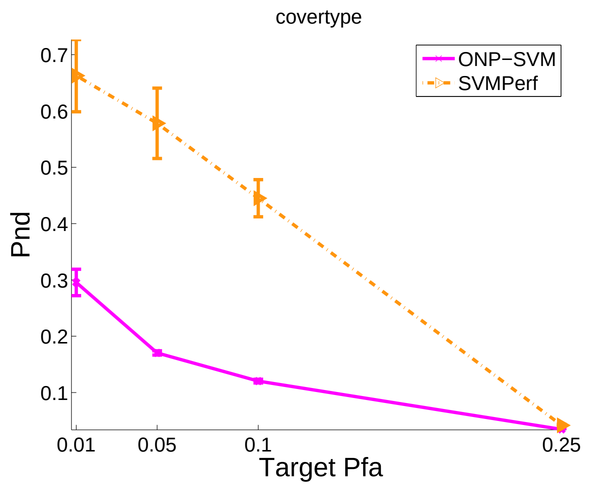
AC-SVM was modified to use the new liblinear solver (Hsieh et al.,2008) whose speed is comparable with SGD.

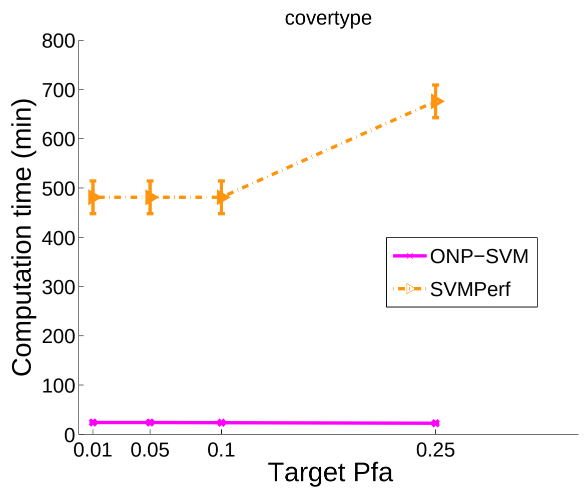
Results: Covertype

54 features, 211840 negatives, 20510 positives.

Linear, 50% training set, 25% validation set, 25% test set.







ONP-SVM: stochastic Neyman Pearson classification.

SVMPerf: (Joachims, 2005)

Extension: Q-value optimization

(Spivak et al., 2009, 2010)

Problem

– Mass spectrometer analyses a preparation and produces a lot of spectras.
– Spectras are matched against dictionaries of known peptides.
– Which matches are worth verifying in costly wet lab experiments?

Approach

– Augment dictionary with a large number of decoy peptides.
– Matches against decoys are known negatives.
– Construct a classifier that returns as many good matches as possible subject to a constraint on the proportion (q-value) of decoys returned among the posited good matches.

Extension: Q-value optimization

Q-value optimization

$$\min_{f} \mathrm{P}_{\mathsf{nd}}(f) \quad \text{subject to} \quad \mathrm{P}_{\mathsf{fa}}(f) < q(1 - \mathrm{P}_{\mathsf{nd}}(f))$$

Same approach as NP classification

– Write the Lagrangian

– Create a stochastic approximation to the Uzawa algorithm.

Results

– 70K true matches, 70K decoy matches.

– Same features and same model as the state-of-the-art QRanker system: A small multilayer network with 5 hidden units.

q	QRanker	This
0.0025	4449	**5005**
0.01	5462	**5707**
0.1	7473	7491

Number of positives returned for various q-values.