Stochastic optimization methods for minimizing training error and AUC

Katya Scheinberg,

LEHIGH UNIVERSITY

Joint work with J. Blanchet,
C. Cartis, H. Ghanbari and M. Menickelly

STEP FUNCTIONS IN MACHINE LEARNING

Supervised learning problem

- Given a sample data set S of n (input, label) pairs, written

$$S = \{(x_1, y_1), \ldots, (x_n, y_n)\}.$$

- each pair is an observation of the random variables (x, y) with some unknown distribution $P(x, y)$ over $\mathcal{X}, \mathcal{Y}$.

- each pair (x_i, y_i) is an independent sample

- Find a hypothesis (predictor) $p(w, \cdot)$ such that $p(w, x) \approx y$, i.e.,

$$\max_w \int_{\mathcal{X}\times\mathcal{Y}} \mathbb{1}[p(w, x) \approx y] dP(x, y).$$

Binary Classification Objective

Expected risk: ideal objective

$$\max_w \int_{\mathcal{X}\times\mathcal{Y}} \mathbb{1}[yp(w,x) > 0]dP(x,y).$$

$$\Updownarrow$$

$$\min_w f_{01}(w) = \mathbb{E}[\ell_{01}(p(w,x),y)]$$

$$\ell_{01}(p(w,x),y) = \begin{cases} 0 & \text{if } yp(w,x) > 0 \\ 1 & \text{if } yp(w,x) \leq 0, \end{cases}$$

Empirical risk: realizable objective

$$\min_w \hat{f}_{01}(w) = \frac{1}{n}\sum_{i=1}^n \ell_{01}(p(w,x_i),y_i)$$

Finite, but NP hard problem

Logistic Regression Model

Expected logistic loss

$$\min_w \ f(w) = \int_{\mathcal{X}\times\mathcal{Y}} \ell(p(w,x),y)dP(x,y) = \mathbb{E}[\ell(p(w,x),y)],$$

$$\ell(p(w,x),y) = \log(1+e^{-yp(w,x)}).$$

Empirical logistic loss: realizable objective

$$\min_w \hat{f}(w) = \frac{1}{n}\sum_{i=1}^{n} \log(1+e^{-yp(w,x)})$$

This is a convex function when *p(w,x)* is linear in *w*

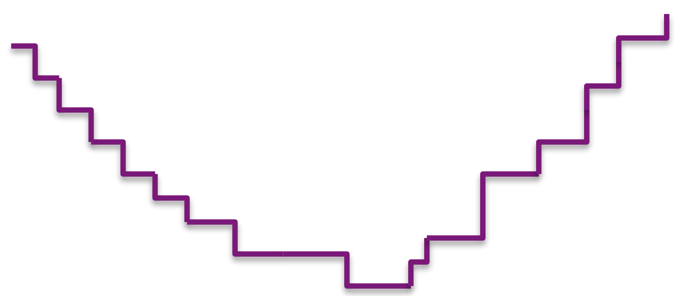
Classification error – smooth or not?

Given a sample set S

$$S = (X, Y) \sim (\mathcal{X}, \mathcal{Y})$$

For a given classifier *p(w,x),* classification error is computed as

$$Acc(w, X, Y) = \frac{\sum_{i=1}^{n} \mathbb{1}[y_i p(w, x_i) \geq 0]}{n}$$



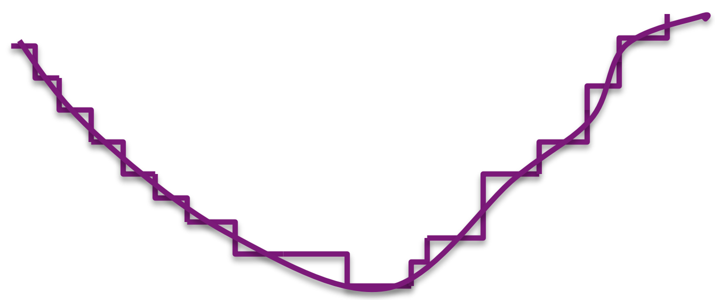
Classification error – smooth or not?

Given a sample set S

$$S = (X, Y) \sim (\mathcal{X}, \mathcal{Y})$$

For a given classifier *p(w,x),* classification error is computed as

$$Acc(w, X, Y) = \frac{\sum_{i=1}^{n} \mathbb{1}[y_i p(w, x_i) \geq 0]}{n}$$



$$F_{Acc}(w) = \mathbb{E}_{(x,y)}(Acc(w, x, y)) = \mathbb{P}[p(w, x) \geq 0]\mathbb{P}[y = 1] + \mathbb{P}[p(w, x) < 0]\mathbb{P}[y = -1]$$

F_{Acc} for a linear classifier, $p(w,x)=w^Tx$

$$F_{Acc}(w) = \mathbb{E}_{(x,y)}(Acc(w,x,y)) = \mathbb{P}[w^T x \geq 0]\mathbb{P}[y=1] + \mathbb{P}[w^T < 0]\mathbb{P}[y=-1]$$

Let positive set $(x|y=1) \sim \mathcal{N}(\mu_p, \Sigma_p)$
and negative set $(x|y=-1) \sim \mathcal{N}(\mu_n, \Sigma_n)$

Given a linear classifier w and a random vector x with label y,

$$F_{Acc}(w) = p(y=-1)\phi(-\mu_-/\sigma_-) + p(y=1)\phi(\mu_+/\sigma_+)$$

where $\phi(x) = \int_{-\infty}^{x} \frac{e^{-\frac{1}{2}t^2}}{\sqrt{2\pi}} dt$,

$$\mu_+ = w^T \mu_p, \quad \mu_- = w^T \mu_n$$
$$\sigma_+^2 = w^T(\Sigma_{pp})w, \quad \sigma_-^2 = w^T(\Sigma_{nn})w$$

Ghanbari and S, 2017

F_{Acc} for linear classifier, $p(w,x)=w^Tx$

$$F_{Acc}(w) = \mathbb{E}_{(x,y)}(Acc(w,x,y)) = \mathbb{P}[w^T x \geq 0]\mathbb{P}[y=1] + \mathbb{P}[w^T < 0]\mathbb{P}[y=-1]$$

$$\text{Let positive set } (x|y=1) \sim \mathcal{N}(\mu_p, \Sigma_p)$$
$$\text{and negative set } (x|y=-1) \sim \mathcal{N}(\mu_n, \Sigma_n)$$

Given a linear classifier w and a random vector x with label y,

$$F_{Acc}(w) = p(y=-1)\phi(-\mu_-/\sigma_-) + p(y=1)\phi(\mu_+/\sigma_+)$$

$$\text{where } \phi(x) = \int_{-\infty}^{x} \frac{e^{-\frac{1}{2}t^2}}{\sqrt{2\pi}} dt,$$

Smooth function in w
We can compute gradients

$$\mu_+ = w^T \mu_p, \quad \mu_- = w^T \mu_n$$
$$\sigma_+^2 = w^T(\Sigma_{pp})w, \quad \sigma_-^2 = w^T(\Sigma_{nn})w$$

Ghanbari and S, 2017

Optimizing accuracy on artificial data using gradient estimates

N = 1000, Separable means

	initial Accuracy		cdf_exact moments	cdf_approx moments	logreg
d = 2	0.065	Accuracy value	0.935	0.935	0.9375
		sol. Time(sec.)	0.11	0.08	0.06
d = 5	0.015	Accuracy value	0.9875	0.9875	0.9825
		sol. Time(sec.)	0.2	0.2	0.2
d = 10	0.0025	Accuracy value	1	1	1
		sol. Time(sec.)	0.2	0.19	0.32
d = 50	0.0125	Accuracy value	0.99	0.99	0.9725
		sol. Time(sec.)	0.22	0.22	0.57
d = 100	0.2725	Accuracy value	0.74	0.7025	0.725
		sol. Time(sec.)	0.27	0.26	0.4
d = 200	0.245	Accuracy value	0.8075	0.7475	0.79
		sol. Time(sec.)	0.48	0.49	2.05

Optimizing accuracy on real data using gradient estimates

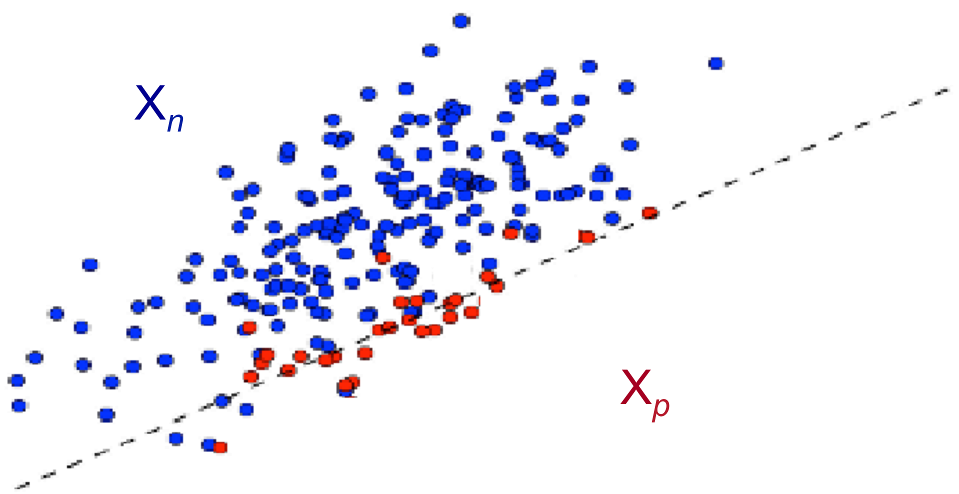
	d	N	accuracy_opt	LogReg_acc	sol time(01)	sol time (lreg)
Sonar	60	208	0.7329	0.875	0.01	0.22
fourclass	2	862	0.8453	0.8456	0.03	0.05
svm1	4	3089	0.9	0.9	0.06	0.11
magic04	10	19020	0.8914	0.8922	0.06	0.58
diabetes	8	768	0.8812	0.8848	0.06	0.05
german	24	1000	0.8526	0.8836	0.06	0.12
svm3	22	1243	0.8037	0.8156	0.06	0.07
shuttle	9	43500	0.9524	0.9193	0.06	1.24
segment	19	2310	0.9273		0.06	
ijcnn1	22	4691	0.9512	0.9512	0.06	1.5
satimage	36	4435	0.5467	0.9116	0.01	1.02
vowel	10	528	0.97071	0.9821	0.1	0.05
letter	16	5000	0.51835	0.991	0.01	5.45
poker	10	5010	0.5102	0.9897	0.01	3.02

"AUC" OPTIMIZATION

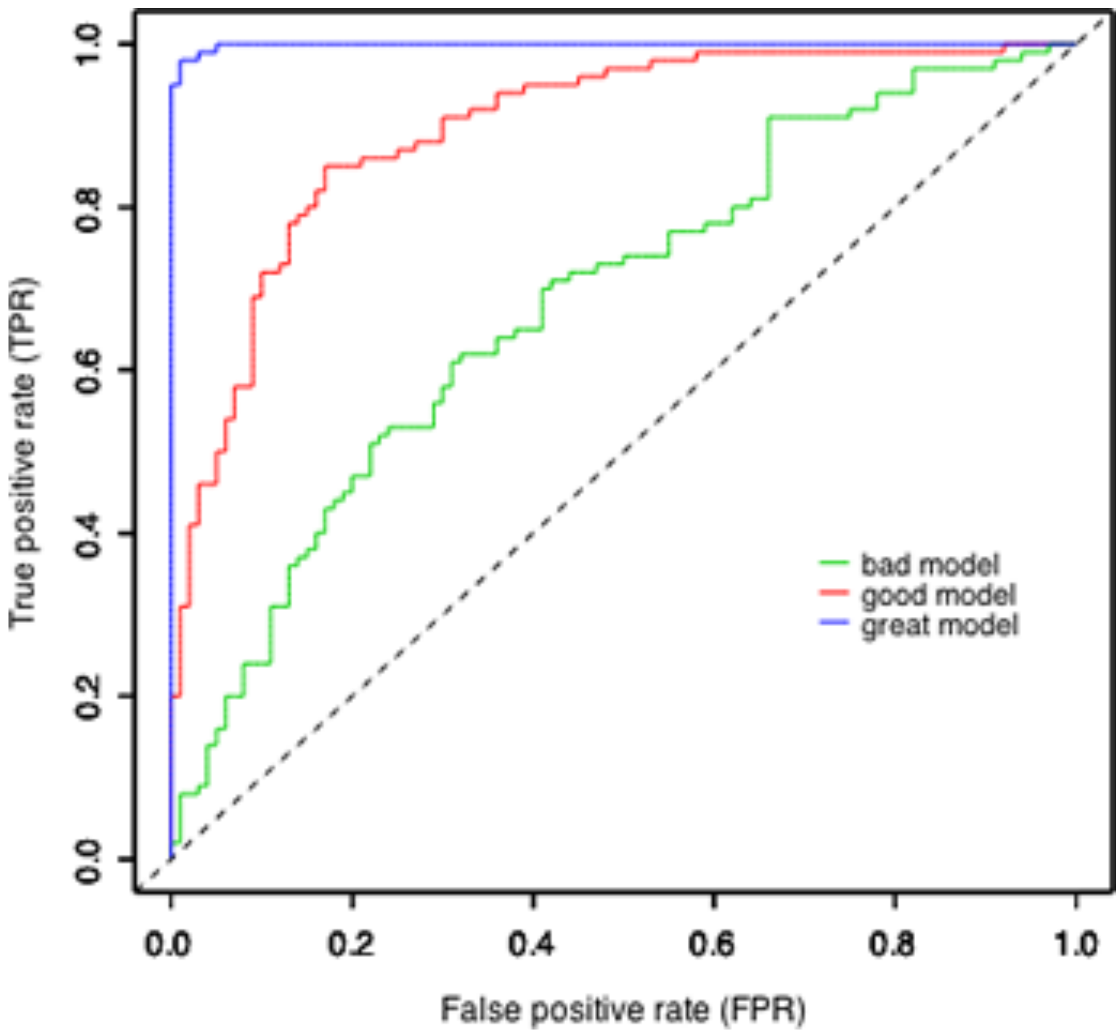
Learning From Imbalanced Data

$$\max_{w}\{\frac{1}{N}\sum_{i=1}^{N} C_i \mathbb{1}(y_i w^T x_i > 0) + \lambda\|w\|^2 \ w \in \mathrm{R}^d\}.$$

$$C_i = \begin{cases} C_p > 1 & \text{if } y_i > 0 \\ C_n \leq 1 & \text{otherwise} \end{cases}$$



AUC – area under the curve



How to compute AUC?

Given a positive sample set X_p and a negative sample set X_n

$$X_p \sim \mathcal{X}_p,\ X_n \sim \mathcal{X}_n$$

For a given classifier *p(w,x),* AUC can be computed as

$$AUC(w, X_p, X_n) = \frac{\sum_{i=1}^{|X_p|} \sum_{j=1}^{|X_n|} \mathbb{1}[p(w, x_i) > p(w, x_j)]}{|X_p|.|X_n|}.$$

$$F_{AUC}(w) = \mathbb{E}_{x_p \sim \mathcal{X}_p, x_n \sim \mathcal{X}_n}(AUC(w, x_p, x_n)) = \mathbb{P}[p(w, x_p) > p(w, x_n)]$$

the probability of correct ranking by classifier *p(w,x)*.

F_{AUC} for linear classifier, $p(w,x)=w^Tx$

Let random vectors $(x_p, x_n) \sim \mathcal{N}(\mu, \Sigma)$, such that

$$\mu = \begin{bmatrix} \mu_p \\ \mu_n \end{bmatrix}, \text{ and } \Sigma = \begin{bmatrix} \Sigma_{pp} & \Sigma_{pn} \\ \Sigma_{np} & \Sigma_{nn} \end{bmatrix}$$

Given a linear classifier w and random vectors X_p and X_n,

$$F_{AUC}(w) = P(w^T x_p > w^T x_n) = \phi(\frac{\mu_Z}{\sigma_Z}),$$

where $\phi(x) = \int_{\infty}^{x} \frac{e^{-\frac{1}{2}t^2}}{\sqrt{2\pi}} dt,$

$$\mu_Z = w^T(\mu_1 - \mu_2),$$
$$\sigma_Z^2 = w^T(\Sigma_{11} + \Sigma_{22} - \Sigma_{12} - \Sigma_{21})w.$$

Ghanbari and S, 2017

Optimizing AUC on artificial data using gradient estimates

N = 1000

	initial AUC		cdf_exact moments	cdf_approx moments	hinge
d = 2	0.5182	AUC value	0.6337	0.6326	0.6322
		sol. Time(sec.)	0.015	0.02	1.179
d = 5	0.570925	AUC value	0.8022	0.802225	0.804025
		sol. Time(sec.)	0.024	0.012	1.71
d = 10	0.5343	AUC value	0.785	0.7818	0.784725
		sol. Time(sec.)	0.013	0.023	2.77
d = 50	0.4581	AUC value	0.955575	0.9264	0.9465
		sol. Time(sec.)	0.03	0.029	70.89
d = 100	0.5491	AUC value	0.99925	0.9921	0.9961
		sol. Time(sec.)	0.15	0.18	205.6
d = 200	0.57175	AUC value	0.999425	0.997425	0.9992
		sol. Time(sec.)	0.65	0.65	379.38

Optimizing AUC on real data using gradient estimates

	d	N	CDF	Hinge
Sonar	60	208	0.8322	0.849
fourclass	2	862	0.8359	0.8363
svm1	4	3089	0.9526	0.9893
magic04	10	19020	0.8217	0.843
diabetes	8	768	0.8189	0.8308
german	24	1000	0.7848	0.7924
svm3	22	1243	0.705	0.7931
shuttle	9	43500	0.9813	0.9886
segment	19	2310	0.8661	0.9931
ijcnn1	22	4691	0.9244	0.9306
satimage	36	4435	0.7216	0.7695
vowel	10	528	0.9566	0.9755
letter	16	5000	0.9809	0.9866
poker	10	5010	0.5151	0.4735

F_{AUC} for linear classifier, $p(w,x)=w^T x$

Issues with using gradient estimates: we do not know how accurate there are because distribution may not be normal

Solution: optimize F_{AUC} only using function values, which are sufficiently accurate.

Difficulty: derivative free methods do not scale for large dimensions of w.

Reducing parameter space for AUC optimization

Idea: select a different training method (e.g. stochastic gradient for a deep neural network. Select parameters $\lambda \in R^l$ that affect output: $w(\lambda)$
Then optimize $F_{AUC}(w(\lambda))$ over λ

Hyperparameter optimization - poorly understood, expect in some cases.

- Given distributions $\mathcal{X}_p$ and $\mathcal{X}_n$ choose parameters C_p, C_n, λ and compute (apprpoximately)

$$w^*(C_p, C_n, \lambda) = \arg\min_w C_p \mathbb{E}_{x\sim\mathcal{X}_p} \log(1+\exp(-w^T x)) + C_n \mathbb{E}_{x\sim\mathcal{X}_n} \log(1+\exp(w^T x)) + \lambda\|w\|^2,$$

- Optimize over the choice of (C_p, C_n, λ)

$$(C_p^*, C_n^*, \lambda^*) = \arg\max F_{AUC}(w^*(C_p, C_n, \lambda))$$

- $F_{AUC}(w^*(C_p, C_n, \lambda))$ may be a smooth function but does not easily admit stochastic derivatives

Optimizing AUC as a black-box

Algorithm	sonar AUC	sonar fevals	fourclass AUC	fourclass fevals
Pair-wise Hinge	0.849057 ± 0.003061	274	0.836226 ± 0.000923	480
DFO-TR	0.840323 ± 0.002453	254	0.836311 ± 0.000921	254
Algorithm	**svmguide1** AUC	fevals	**magic04** AUC	fevals
Pair-wise Hinge	0.989319 ± 0.000008	334	0.843085 ± 0.000208	417
DFO-TR	0.989132 ± 0.000007	254	0.843511 ± 0.000213	254
Algorithm	**diabetes** AUC	fevals	**german** AUC	fevals
Pair-wise Hinge	0.830852 ± 0.001015	348	0.792402 ± 0.000795	421
DFO-TR	0.830402 ± 0.00106	254	0.791048 ± 0.000846	254
Algorithm	**svmguide3** AUC	fevals	**shuttle** AUC	fevals
Pair-wise Hinge	0.793116 ± 0.001284	368	0.988625 ± 0.000021	266
DFO-TR	0.775246 ± 0.002083	254	0.987531 ± 0.000035	254
Algorithm	**segment** AUC	fevals	**ijcnn1** AUC	fevals
Pair-wise Hinge	0.993134 ± 0.000023	753	0.930685 ± 0.000204	413
DFO-TR	0.99567 ± 0.00071	254	0.910897 ± 0.000264	254
Algorithm	**satimage** AUC	fevals	**vowel** AUC	fevals
Pair-wise Hinge	0.769505 ± 0.000253	763	0.975586 ± 0.000396	348
DFO-TR	0.757554 ± 0.000236	254	0.973785 ± 0.000506	254
Algorithm	**letter** AUC	fevals	**poker** AUC	fevals
Pair-wise Hinge	0.986699 ± 0.000037	517	0.519942 ± 0.001549	553
DFO-TR	0.985119 ± 0.000042	254	0.520517 ± 0.001618	254

Larger scale examples

a9a	w1a
d=123	d=300
N=32561	N=47272
AUC-cdf = 0.8666	AUC-cdf = 0.94
AUC-DFO=0.8991	AUC-DFO=0.9346
AUC-cdf-DFO=0.8997	AUC-cdf-DFO=0.9558
AUC-logreg = 0.902774	AUC-logreg=0.96

DFO vs. Bayesian optimization on AUC

www.automl.org/hpolib

Table 5. Comparing DFO-TR vs. BO algorithms.

Data	num. fevals	DFO-TR		TPE		SMAC		SPEARMINT	
		AUC	time	AUC	time	AUC	time	AUC	time
fourclass	100	0.835±0.019	0.31	**0.839**±0.021	12	0.839±0.021	77	0.838±0.020	5229
svmguide1	100	0.988±0.004	0.71	0.984±0.009	13	0.986±0.006	72	**0.987**±0.006	6435
diabetes	100	0.829±0.041	0.58	0.824±0.044	15	0.825±0.045	75	**0.829**±0.060	8142
shuttle	100	0.990±0.001	43.4	0.990±0.001	17	0.989±0.001	76	**0.990**±0.001	13654
vowel	100	0.975±0.027	0.68	0.965±0.029	16	0.965±0.038	77	**0.968**±0.025	9101
magic04	100	0.842±0.006	10.9	0.824±0.009	16	0.821±0.012	76	**0.839**±0.006	7947
letter	200	0.987±0.003	10.2	0.959±0.008	49	0.953±0.022	166	**0.985**±0.004	21413
segment	300	0.992±0.007	9.1	0.962±0.021	99	**0.997**±0.004	263	0.976±0.021	216217
ijcnn1	300	0.913±0.005	57.3	0.677±0.015	109	0.805±0.031	268	**0.922**±0.004	259213
svmguide3	300	0.776±0.046	13.5	0.747±0.026	114	**0.798**±0.035	307	0.7440±0.072	185337
german	300	0.795±0.024	9.9	0.771±0.022	120	0.778±0.025	310	**0.805**±0.020	242921
satimage	300	0.757±0.013	14.2	0.756±0.020	164	0.750±0.011	341	**0.761**±0.028	345398

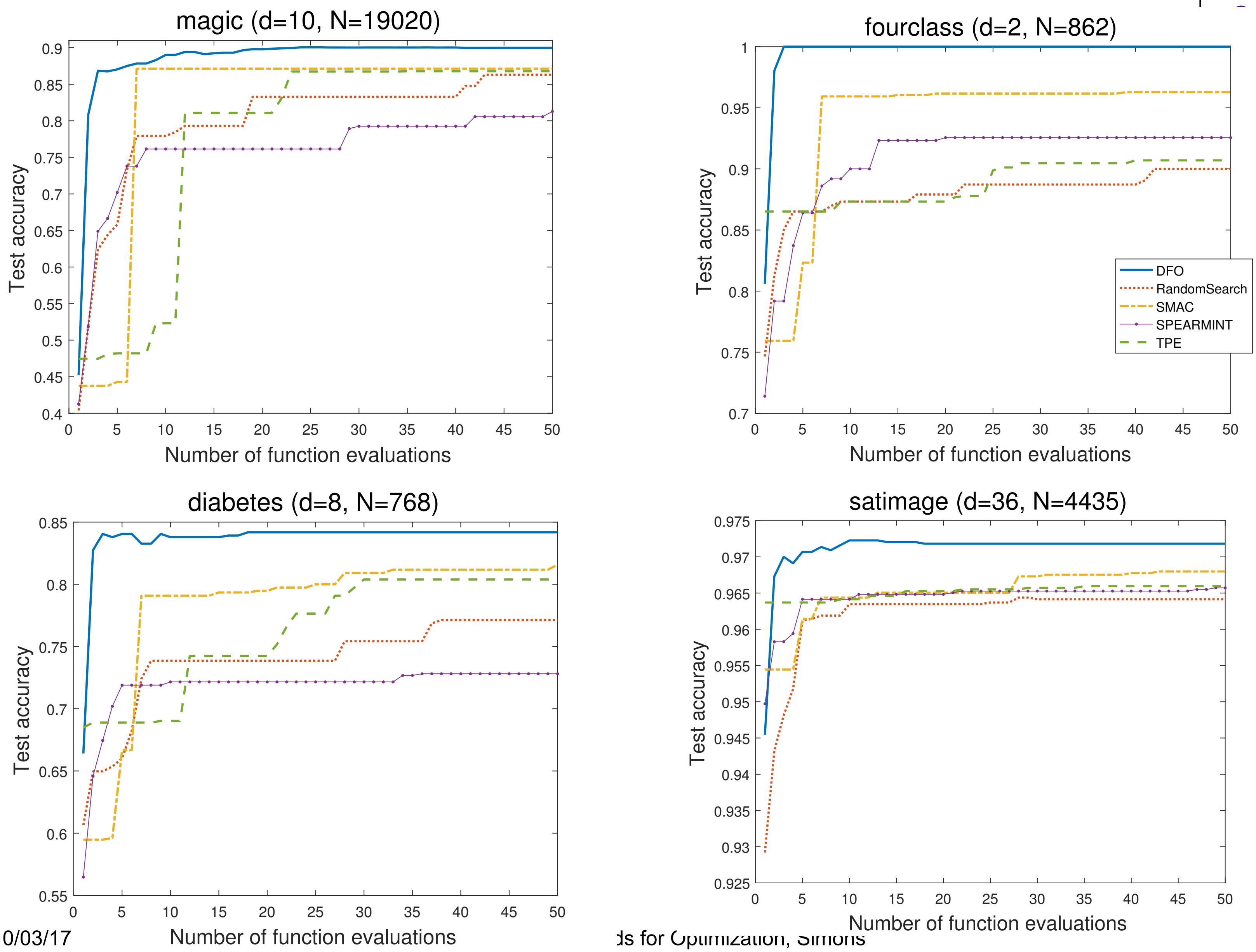
DFO vs. random search, optimizing AUC

Jamieson and Talwalkar

Table 6. Comparing DFO-TR vs. random search algorithm.

Data	DFO-TR		Random Search		Random Search	
	AUC	num. fevals	AUC	num. fevals	AUC	num. fevals
fourclass	0.835±0.019	100	0.836±0.017	100	0.839±0.021	200
svmguide1	0.988±0.004	100	0.965±0.024	100	0.977±0.009	200
diabetes	0.829±0.041	100	0.783±0.038	100	0.801±0.045	200
shuttle	0.990±0.001	100	0.982±0.006	100	0.988±0.001	200
vowel	0.975±0.027	100	0.944±0.040	100	0.961±0.031	200
magic04	0.842±0.006	100	0.815±0.009	100	0.817±0.011	200
letter	0.987±0.003	200	0.920±0.026	200	0.925±0.018	400
segment	0.992±0.007	300	0.903±0.041	300	0.908±0.036	600
ijcnn1	0.913±0.005	300	0.618±0.010	300	0.629±0.013	600
svmguide3	0.776±0.046	300	0.690±0.038	300	0.693±0.039	600
german	0.795±0.024	300	0.726±0.028	300	0.739±0.021	600
satimage	0.757±0.013	300	0.743±0.029	300	0.750±0.020	600

DFO vs. BO optimizing SVM hyperparameters



STOCHASTIC TRUST REGION METHODS

Stochastic optimization

- Unconstrained optimization problem

$$\min_w F(w)$$

- Function $F \in C^1$ *or* C^2 and bounded from below.
- $F(w)$ may not be computable, instead

$$\hat{F}(w) = F(w,\varepsilon),\ G(w) = \nabla F(w,\varepsilon),\ H(w) = \nabla^2 F(w,\varepsilon)$$

where $\epsilon\ is\ a\ random\ variable$

- We do not assume $unbiased\ estimators$

$E_\epsilon[F(w,\epsilon)]=F(w)$,

$E_\epsilon[\nabla F(w,\epsilon)]=\nabla F(w)$

$E_\epsilon[\nabla^2 F(w,\epsilon)]=\nabla^2 F(w)$

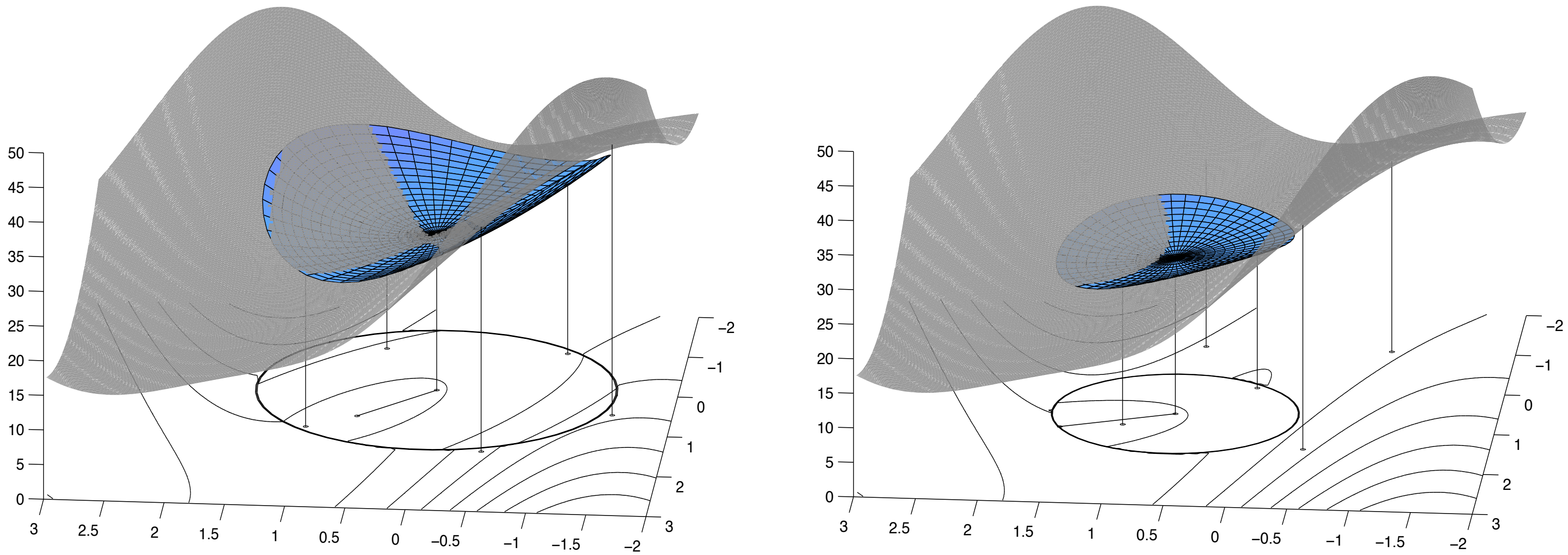
Deterministic trust region method

Algorithm 1 Trust region method

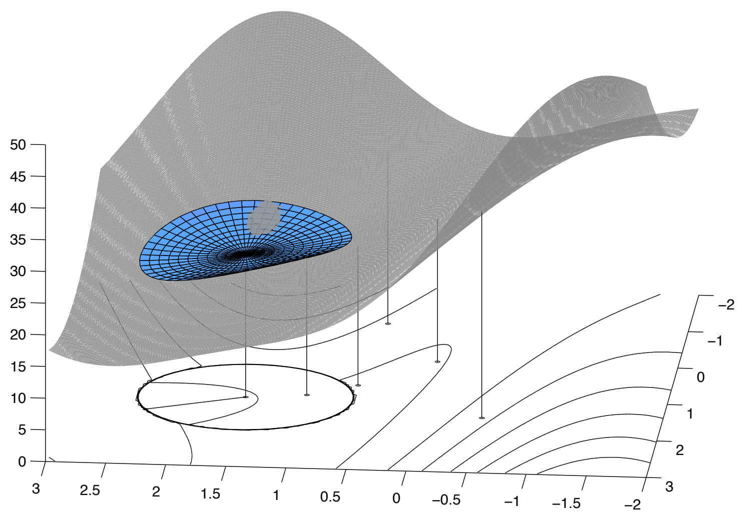
Parameters: $\gamma > 1$, $\eta_1 \in (0,1)$, $\eta_2 > 1$
Initialize: w_0, trust region radius Δ_0.
Iterate:
for $k = 1, 2, \ldots$ **do**
  Generate a model $m_k(w_k + s) = F(w_k) + G_k^\top s + \frac{1}{2} s^\top H_k s$
  $s_k = \arg \min_{s:\|s\| \leq \Delta_k} m_k(s)$ (approximately)
  Compute $\rho_k = \dfrac{F(w_k) - F(w_k + s_k)}{m_k(w_k) - m_k(w_k + s_k)}$.
  If $\rho_k \geq \eta_1$ set $w_{k+1} = w_k + s_k$ and $\Delta_{k+1} = \gamma \Delta_k$;
  Else, set $w_{k+1} = w_k$, and set $\Delta_{k+1} = \gamma^{-1} \Delta_k$;
end for

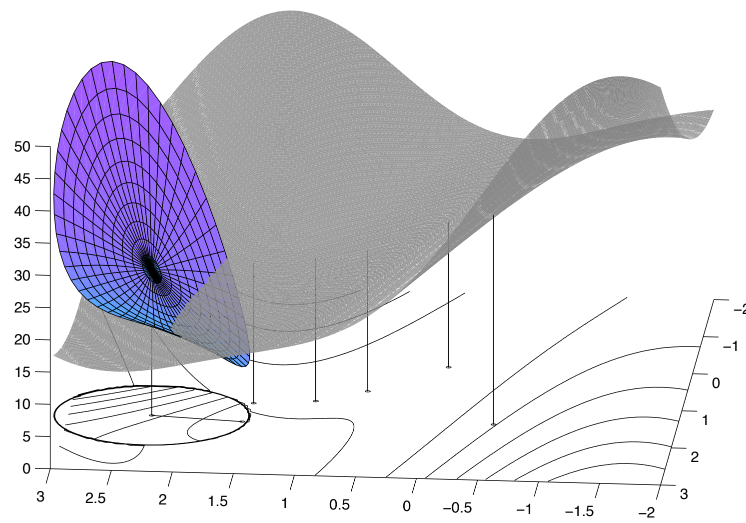
Conn Gould Toint, 2000
Conn, S, Vicente 2009

Examples of models and TR steps



Examples of models and TR steps





Model assumptions for trust region method

$$m_k(w_k + s) = F_k + G_k^\top s + \frac{1}{2} s^\top H_k s$$

Fully linear model

$$|F_k - F(w_k)| \le \kappa \Delta_k^2$$
$$\|G_k - \nabla F(w_k)\| \le \kappa \Delta_k$$
$$\|H_k - \nabla^2 F(w_k)\| \le \kappa$$

Fully quadratic model

$$|F_k - F(w_k)| \le \kappa \Delta_k^3$$
$$\|G_k - \nabla F(w_k)\| \le \kappa \Delta_k^2$$
$$\|H_k - \nabla^2 F(w_k)\| \le \kappa \Delta_k$$

Trust-region methods converge and achieve
1) $\|\nabla F(w_k)\| \le \epsilon$ at the rate of $1/\epsilon^2$
2) $\mathrm{Min}\{\lambda_{min}(\nabla^2 F(w_k)), \|\nabla F(w_k)\|\} \le \epsilon$ at the rate of $1/\epsilon^3$

Stochastic trust region method

Algorithm 1 Trust region method

Parameters: $\gamma > 1$, $\eta_1 \in (0,1)$, $\eta_2 > 1$
Initialize: w_0, trust region radius Δ_0.
Iterate:
for $k = 1, 2, \ldots$ **do**

Generate a random model $m_k(w_k + s) = F_k + G_k^\top s + \frac{1}{2} s^\top H_k s$

$s_k = \arg \min_{s:\|s\| \leq \Delta_k} m_k(s)$ (approximately)

Compute $\rho_k = \dfrac{\hat{F}_k - \hat{F}_k^+}{m_k(w_k) - m_k(w_k + s_k)}$.

where $\hat{F}_k \approx F(w_k)$ and $\hat{F}_k^+ \approx F(w_k + s_k)$

If $\rho_k \geq \eta_1$ and $\|G_k\| \geq \eta_2 \Delta_k$

set $w_{k+1} = w_k + s_k$ and $\Delta_{k+1} = \gamma \Delta_k$;

Else, set $w_{k+1} = w_k$, and set $\Delta_{k+1} = \gamma^{-1} \Delta_k$;

end for

Chen, Menickelly, S, 2015
Blanchet, Cartis, Menickelly, S, 2017

Convergence rates

For nonconvex F(w), for first order convergence, we aim to achieve

$$\|\nabla F(w_k)\| \leq \epsilon$$

Convergence rate for our methods

For nonconvex F(w), for first order convergence, we aim to achieve

$$\|\nabla F(w_k)\| \leq \epsilon$$

Define a stopping time T_ϵ

$$T_\epsilon = \inf\{k \geq 0 : \|\nabla F(w_k)\| \leq \epsilon\}.$$

Bound it in expectation

$$\mathbf{E}[T_\epsilon] \leq O(\frac{1}{\epsilon^2})$$

Stochastic trust region method

$$m_k(w_k + s) = F_k + G_k^\top s + \frac{1}{2} s^\top H_k s$$

$$\left.\begin{aligned} |F_k - F(w_k)| &\le \kappa \Delta_k^2 \\ \|G_k - \nabla F(w_k)\| &\le \kappa \Delta_k \\ \|H_k - \nabla^2 F(w_k)\| &\le \kappa \end{aligned}\right\}$$

w.p. p suff. large

Trust-region method converges and achieves $\|\nabla F(w_k)\|^2 \le \epsilon$ at the rate of $1/\epsilon$

Chen, Menickelly, S. 2016,

Blanchet, Cartis, Menickelly, S, 2017,

Stochastic second order TR method

Random second order model

$$m_k(w_k + s) = F_k + G_k^\top s + \frac{1}{2} s^\top H_k s$$

$$|F_k - F(w_k)| \leq \kappa \Delta_k^3$$

$$\|G_k - \nabla F(w_k)\| \leq \kappa \Delta_k^2$$

$$\|H_k - \nabla^2 F(w_k)\| \leq \kappa \Delta_k$$

w.p. p>0
suff. large

$$|\mathbb{E}[F_k] - F(w_k)| \leq \kappa \Delta_k^3$$

Trust-region method converges and achieves
Min{λ_{min} (∇^2F(w_k)), || ∇ F(w_k)||} $\leq \epsilon$ at the rate of *$1/\epsilon^3$*
Cartis, S. 2017

Employing sample average approximation with S_k dependent on Δ_k

For function accuracy $F_k = \frac{1}{S_k} \sum_{i \in S_k} F_i(w^k) \quad |S_k| = O\left(\frac{V_F}{\Delta_k^4}\right)$

For gradient accuracy $G_k = \frac{1}{S_k} \sum_{i \in S_k} \nabla F_i(w^k) \quad |S_k| = O\left(\frac{V_G}{\Delta_k^2}\right)$

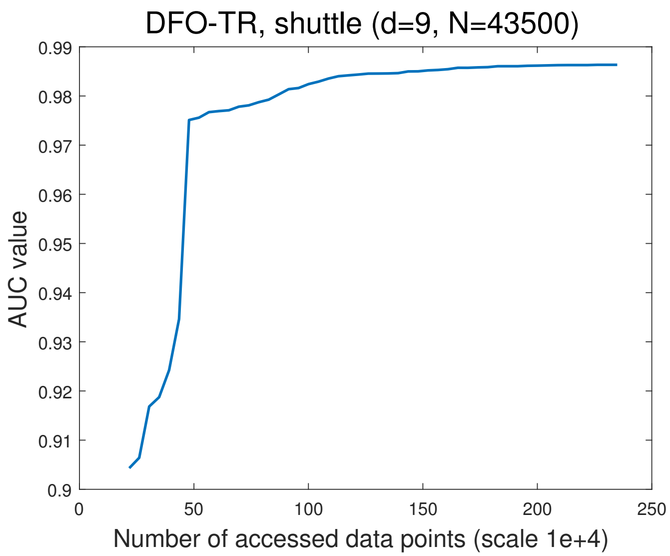
Employing sample average approximation with S_k dependent on Δ_k

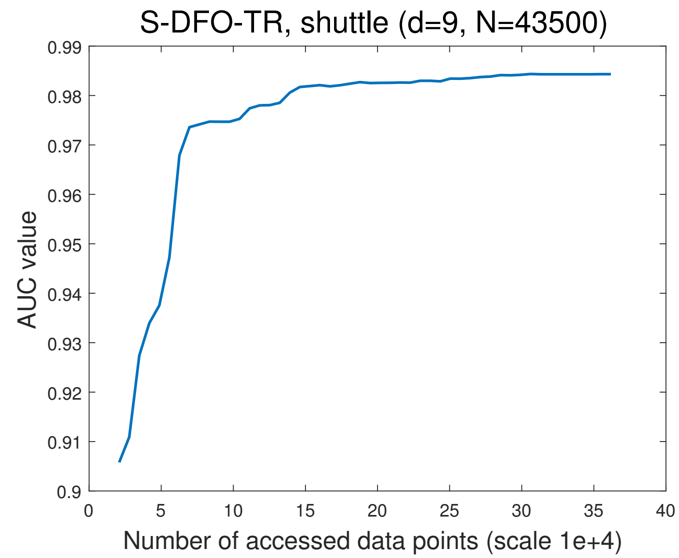
For function accuracy $F_k = \frac{1}{S_k} \sum_{i \in S_k} F_i(w^k) \quad |S_k| = O\left(\frac{V_F}{\Delta_k^6}\right)$

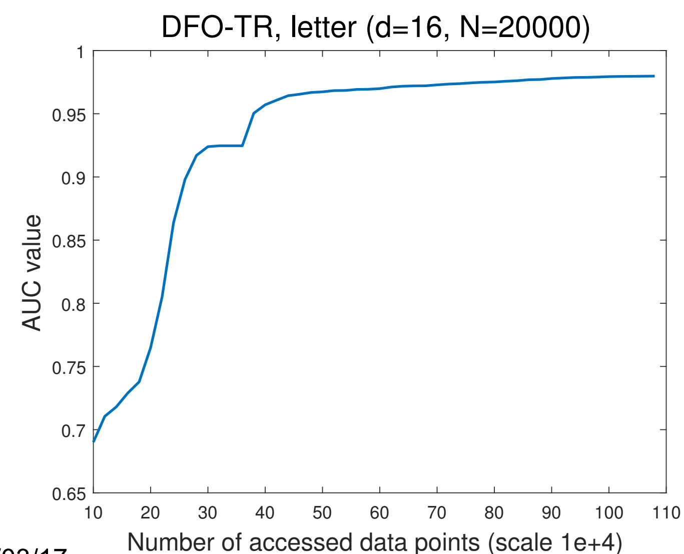
For gradient accuracy $G_k = \frac{1}{S_k} \sum_{i \in S_k} \nabla F_i(w^k) \quad |S_k| = O\left(\frac{V_G}{\Delta_k^4}\right)$

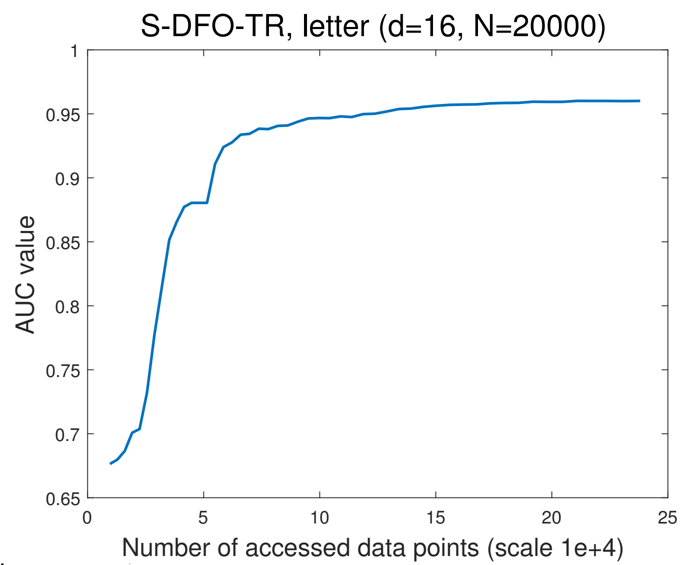
For Hessian accuracy $H_k = \frac{1}{S_k} \sum_{i \in S_k} \nabla^2 F_i(w^k) \quad |S_k| = O\left(\frac{V_H}{\Delta_k^2}\right)$

Stochastic vs. deterministic TR method









Conclusions

- Optimizing accuracy and AUC directly is possible.
- If underlining expected values functions are smooth, then convergent methods exist.
- Scaling DFO methods up will be useful.
- Studying/engineering data distributions may lead to new efficient methods.

Thank you!