Structural Extensions of Support Vector Machines

Mark Schmidt

March 30, 2009

Outline

- Formulation:
  - Binary SVMs
  - Multiclass SVMs
  - Structural SVMs
- Training:
  - Subgradients
  - Cutting Planes
  - Marginal Formulations
  - Min-Max Formulations

Topics Not Covered

- Optimal separating hyper-planes
- Deriving Wolfe duals of quadratic programs
- The kernel trick, Mercer/Representer theorems
- Generalization bounds

Logistic Regression

- Model probabilities of binary labels as:

$$p(y_i = 1|w, x_i) \propto \exp(w^T x_i),$$
$$p(y_i = -1|w, x_i) \propto \exp(-w^T x_i),$$

- Train by maximizing likelihood, or minimizing negative log-likelihood:

$$\min_w - \sum_i \log p(y_i|w, x_i).$$

- To make solution unique, add an L2 penalty:

$$\min_w - \sum_i \log p(y_i|w, x_i) + \lambda ||w||_2^2,$$

- Make decisions using the rule:

$$\hat{y} = \begin{cases} 1 & \text{if } p(y_i = 1|w, x_i) > p(y_i = -1|w, x_i) \\ -1 & \text{if } p(y_i = 1|w, x_i) < p(y_i = -1|w, x_i) \end{cases}.$$

Linear Separability

- If we just want to get the decisions right, then the we require (for some arbitrary c > 1):

$$\forall_i \ \frac{p(y_i|w, x_i)}{p(-y_i|w, x_i)} \geq c,$$

- Taking logarithms

$$\forall_i \ \log p(y_i|w, x_i) - \log p(-y_i|w, x_i) \geq \log c,$$

- Plugging in probabilities (canceling normalizing constants):

$$\forall_i \ 2y_i w^T x_i \geq \log c.$$

- Choose c such that log(c)/2 = 1:

$$\forall_i \ y_i w^T x_i \geq 1.$$

Fixing

- We can solve this as a linear feasibility problem:

$$\forall_i \; y_i w^T x_i \geq 1.$$

- This problem either has no solution, or an infinite number
- To make the solution unique with add an L2 penalty:

$$\min_w \lambda ||w||_2^2,$$

$$s.t. \;\; \forall_i \; y_i w^T x_i \geq 1,$$

- To make the solution exist we allow 'slack' in the constraints, but penalize the L1-norm of this slack:

$$\min_{w,\xi} \sum_i \xi + \lambda ||w||_2^2,$$

$$s.t. \;\; \forall_i \; y_i w^T x_i \geq 1 - \xi_i, \;\; \forall_i \; \xi_i \geq 0,$$

Support Vector Machine

- This is the primal form of 'soft-margin' SVMs:

$$\min_{w,\xi} \sum_i \xi + \lambda ||w||_2^2,$$

$$s.t. \;\; \forall_i \; y_i w^T x_i \geq 1 - \xi_i, \;\; \forall_i \; \xi_i \geq 0,$$

- We can also eliminate the slacks and write it as an unconstrained problem:

$$\min_w \sum_i (1 - y_i w^T x_i)^+ + \lambda ||w||_2^2,$$

- The 'hinge' loss is an upper bound on the classification errors
- It is very similar to logistic regression with L2-regularization:

$$\min_w - \sum_i \log p(y_i | w, x_i) + \lambda ||w||_2^2,$$

Outline

- Formulation:
  - Binary SVMs
  - **Multiclass SVMs**
  - Structural SVMs
- Training:
  - Subgradients
  - Cutting Planes
  - Marginal Formulations
  - Min-Max Formulations

Multinomial Logistic

- We extend binary logistic regression to multi-class data by giving each class 'k' its own weight vector:

$$p(y_i = k|w_k, x_i) \propto \exp(w_k^T x_i).$$

- Training is the same as before, and we make decisions using:

$$\hat{y}_i = \max_k p(y_i = k|w_k, x_i).$$

NK-Slack Multiclass SVMs

- Making the right decisions corresponds to satisfying:

$$\forall_i \forall_{k \neq y_i},\ \frac{p(y_i | w, x_i)}{p(y_i = k | w_k, x_i)} \geq c$$

- Following the same steps as before, we can write this as:

$$\forall_i \forall_{k \neq y_i},\ w_{y_i}^T x_i - w_k^T x_i \geq 1.$$

- Adding slacks and L2-regularization yields the 'NK'-slack multi-class SVM:

$$\min_{w,\xi} \sum_i \sum_{k \neq y_i} \xi_{i,k} + \lambda ||w||_2^2,$$

$$\forall_i \forall_{k \neq y_i},\ w_{y_i}^T x_i - w_k^T x_i \geq 1 - \xi_{i,k},\ \ \forall_i \forall_{k \neq y_i} \xi_{i,k} \geq 0,$$

- This can also be written as:

$$\min_w \sum_i \sum_{k \neq y_i} (1 - w_{y_i}^T x_i + w_k^T x_i)^+ + \lambda ||w||_2^2,$$

N-Slack Multiclass SVMs

- If instead of writing the constraint on the decision rul as:

$$\forall_i \forall_{k \neq y_i}, \; \frac{p(y_i | w, x_i)}{p(y_i = k | w_k, x_i)} \geq c$$

- We wrote it as:

$$\forall_i \; \frac{p(y_i | w, x_i)}{\max_{k \neq y_i} p(y_i = k | w_k, x_i)} \geq c.$$

- Then following the same procedure we obtain the 'N'-slack multiclass SVM:

$$\min_{w,\xi} \sum_i \xi_i + \lambda ||w||_2^2,$$

$$\forall_i \forall_{k \neq y_i}, \; w_{y_i}^T x_i - w_k^T x_i \geq 1 - \xi_i, \;\; \forall_i \xi_i \geq 0,$$

- Which can be written as the unconstrained optimization:

$$\min_w \sum_i \max_{k \neq y_i} (1 - w_{y_i}^T x_i + w_k^T x_i)^+ + \lambda ||w||_2^2,$$

Outline

- Formulation:
  - Binary SVMs
  - Multiclass SVMs
  - **Structural SVMs**
- Training:
  - Subgradients
  - Cutting Planes
  - Marginal Formulations
  - Min-Max Formulations

Conditional Random Fields

- The extension of logistic regression to data with multiple (dependent) labels is known as a conditional random field.
- For example, a binary chain-CRF with Ising-like potentials and tied parameters could use:

$$p(Y_i|w, X_i) \propto \exp(\sum_{j=1}^{S} y_{i,j} w_n^T x_{i,j} + \sum_{j=1}^{S-1} y_{i,j} y_{i,j+1} w_e^T x_{i,j,j+1}),$$

- A concise notation for the general case is:

$$p(Y_i|w, X_i) \propto exp(w^T F(X_i, Y_i)),$$

- One possible decision rule is:

$$\hat{Y}_i = \max_{Y_i} p(Y_i|w, X_i).$$

- In the case of chains, this is Viterbi decoding

Hidden Markov SVMs

- Making the right decisions with Viterbi decoding corresponds to satisfying:

$$\forall_i \forall_{Y_i' \neq Y_i} \frac{p(Y_i|w, X_i)}{p(Y_i'|w, X_i)} \geq c,$$

- This is equivalent to the set of constraints:

$$\forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq 1.$$

- Adding the L2-penalty and using the N-slack penalty:

$$\min_{w,\xi} \sum_i \xi_i + \lambda ||w||_2^2,$$

$$s.t. \;\; \forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq 1 - \xi_i, \;\; \forall_i \; \xi_i \geq 0$$

- The ‘hidden Markov support vector machine’

Max-Margin Markov Networks

- The constraints in the HMSVM don't care about the number of differences between Yi and Yi':

$$\forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq 1.$$

- We might to be the difference in probability to be higher when the difference in labels is higher:

$$\forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq \Delta(Y_i, Y_i').$$

- Leading to the QP: $\min_{w,\xi} \sum_i \xi_i + \lambda ||w||_2^2,$

$$s.t.\ \ \forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq \Delta(Y_i, Y_i') - \xi_i, \ \ \forall_i \xi_i \geq 0,$$

- This is known as a 'max-margin Markov networks', or 'structural SVM' with 'margin-rescaling'

Structural SVMs

- Rescaling the constant might make us concentrate on being much better than sequences that differ at many positions:

$$\forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq \Delta(Y_i, Y_i').$$

- An alternative is to rescale the slacks based on the difference between sequences:

$$\forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq 1 - \xi_i/\Delta(Y_i, Y_i'), \ \ \forall_i \xi_i \geq 0.$$

- Leading to the QP:

$$\min_{w,\xi} \sum_i \xi_i + \lambda ||w||_2^2,$$

$$s.t. \ \ \forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq 1 - \xi_i/\Delta(Y_i, Y_i'), \ \ \forall_i \xi_i \geq 0.$$

- This is known as a ‘structural SVM’ with ‘slack-rescaling’

Summary

- Unconstrained formulations of structural extensions:

$$\text{(HMSVM)} \qquad \min_w \sum_i \max_{Y_i' \neq Y_i} (1 - \log p(Y_i|w, X_i) + \log p(Y_i'|w, X_i))^+ + \lambda||w||_2^2,$$

$$\text{(MMMN)} \qquad \min_w \sum_i \max_{Y_i' \neq Y_i} (\Delta(Y_i, Y_i') - \log p(Y_i|w, X_i) + \log p(Y_i'|w, X_i))^+ + \lambda||w||_2^2,$$

$$\text{(SSVM)} \qquad \min_w \sum_i \max_{Y_i' \neq Y_i} (\Delta(Y_i, Y_i')(1 - \log p(Y_i|w, X_i) + \log p(Y_i'|w, X_i)))^+ + \lambda||w||_2^2.$$

- Since delta(Yi,Yi)=0, we simplify MMMN and SSVM:

$$\text{(MMMN)} \qquad \min_w \sum_i \max_{Y_i'} (\Delta(Y_i, Y_i') + \log p(Y_i'|w, X_i)) - \log p(Y_i|w, X_i) + \lambda||w||_2^2,$$

$$\text{(SSVM)} \qquad \min_w \sum_i \max_{Y_i'} (\Delta(Y_i, Y_i')(1 - \log p(Y_i|w, X_i) + \log p(Y_i'|w, X_i))) + \lambda||w||_2^2.$$

- This allows us to use Viterbi decoding with a modified input to compute the max values.

Beyond Chains

$$\text{(MMMN)} \qquad \min_w \sum_i \max_{Y_i'} (\Delta(Y_i, Y_i') + \log p(Y_i'|w, X_i)) - \log p(Y_i|w, X_i) + \lambda ||w||_2^2,$$

$$\text{(SSVM)} \qquad \min_w \sum_i \max_{Y_i'} (\Delta(Y_i, Y_i')(1 - \log p(Y_i|w, X_i) + \log p(Y_i'|w, X_i))) + \lambda ||w||_2^2.$$

- We can compute these objective value anytime we can do decoding in the model:
  - Trees and low-treewidth graphs
  - Context-free grammars
  - General graphs with sub-modular potentials*
  - Weighted bipartite matching*
- *: #P-hard to train conditional random field
- We can also plug in an approximate decoding or convex relaxation of decoding

Outline

- Formulation:
  - Binary SVMs
  - Multiclass SVMs
  - Structural SVMs
- Training:
  - **Subgradients**
  - Cutting Planes
  - Marginal Formulations
  - Min-Max Formulations

Subgradients

- Our objective function is:

$$f(w) \triangleq \sum_i \max_{Y_i'}(\Delta(Y_i, Y_i') + \log p(Y_i'|w, X_i)) - \log p(Y_i|w, X_i) + \lambda||w||_2^2.$$

- If Yi'' is an argmax of the max, a subgradient is:

$$g(w) \triangleq \sum_i \nabla_w \log p(Y_i''|w, X_i) - \nabla_w \log p(Y_i|w, X_i) + 2\lambda w.$$

- Consider the step:

$$w_{k+1} = w_k - \eta_k g(w_k).$$

- For small enough eta, this will:
  - always move us toward the optimal solution
  - decrease the objective function when the argmax is unique

Subgradient descent

- We can therefore consider optimization algorithms of the form:

$$w_{k+1} = w_k - \eta_k g(w_k).$$

- Common choices of step size are constant, or a sequence satisfying:

$$\sum_{k=1}^{\infty} \eta_k = \infty, \quad \sum_{k=1}^{\infty} \eta_k^2 < \infty.$$

- Update based on a single training example:

$$g_i(w) \triangleq \nabla_w \log p(Y_i''|w, X_i) - \nabla_w \log p(Y_i|w, X_i) + (2/N)\lambda w,$$

- Average the iterations:

$$w_{k+1} = w_k - \eta g_i(w_k),$$

$$\tilde{w}_{k+1} = \frac{k-1}{k}\tilde{w}_k + \frac{1}{k} w_{k+1}.$$

- Project onto a compact set containing the solution:

$$w_{k+1} = \pi(w_k - \eta_k g(w_k)),$$

Some Convergence Rates

- Projected batch SD (diminishing step sizes): O(1/eps)
- Averaged stochastic SD (constant step sizes): O(1/eps^2), asymptotic variance
- Stochastic projected SD (dimishing step sizes): O(1/(d eps)) w.p. 1-d
- Averaged stochastic projected SD (constant step sizes): ?, asymptotic variance
- Batch SD (constant step sizes): O(log(1/eps)) to get within bounded region of optimal (bound depends on lambda and bound on sub-differential)

Outline

- Formulation:
  - Binary SVMs
  - Multiclass SVMs
  - Structural SVMs
- Training:
  - Subgradients
  - **Cutting Planes**
  - Marginal Formulations
  - Min-Max Formulations

Cutting Plane Methods

- The problem with the QP formulation is that it has an exponential number of constraints:

$$\min_{w,\xi} \sum_i \xi_i + \lambda ||w||_2^2,$$

$$s.t. \;\; \forall_i \forall_{Y_i'} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq \Delta(Y_i, Y_i') - \xi_i, \;\; \forall_i \xi_i \geq 0.$$

- But, there always exists a polynomial-sized set that satisfies all constraints up to an accuracy of eps.

- Basic idea behind cutting plane method:

    - use decoding to find out if all constraints are satisfied

    - if not, greedily add a constraint

QP Cutting Plane Method

- Cutting plane method:
  - we have a working set of constraints
  - iterate over training examples:
    - if decoding does not violate constraints, continue
    - otherwise, add constraint to working set and solve QP
  - stop if no changes in working set
- Solving these QPs in the dual is efficient, as long as the working set is small.
- At most O(1/eps) constraints are required.

Convex Cutting Plane

- There also exist 'cutting plane' methods for solving (non-smooth) convex optimization problems
- We can apply these to the unconstrained formulation:

$$f(w) \triangleq \sum_i \max_{Y_i'}(\Delta(Y_i, Y_i') + \log p(Y_i'|w, X_i)) - \log p(Y_i|w, X_i) + \lambda||w||_2^2.$$

- Basic idea: any subgradient gives a lower bounding hyper-plane

$$f(w) \geq f(w_0) + (w - w_0)^T g(w_0),$$

- Cutting plane for non-smooth optimization:
  - Find minimum over these lower bounds
  - Use minimum to make better lower bound

Bundle Methods

- Problem: minimum of lower bound may be far away from current solution.
- Bundle method: minimize lower bound subject to L2-penalty on distance from current solution $||w_{k+1} - w_k||_2^2$
- Combined cutting-plane/bundle-method: use the L2-penalty already present in the objective, and build a lower bound on the hinge loss
- Combined method requires at most O(1/eps) iterations.

Outline

- Formulation:
  - Binary SVMs
  - Multiclass SVMs
  - Structural SVMs
- Training:
  - Subgradients
  - Cutting Planes
  - **Marginal Formulations**
  - Min-Max Formulations

Poly-sized Formulations

- The previous two strategies use the graph structure to allow efficient decoding.
- An alternative strategy is to use the graph structure to re-parameterize our quadratic program.
- Although we can also do this for the primal, this can be shown more directly for the dual problem...

Dual MMMN

- Solving the MMMN QP is equivalent to solving the following QP:

$$\max_{\alpha} \sum_{i} \sum_{Y_i'} \alpha_i(Y_i')\Delta(Y_i, Y_i') - \frac{1}{2} \sum_{i} \sum_{Y_i'} \sum_{j} \sum_{Y_j'} \alpha_i(Y_i')\alpha_j(Y_j')\Delta F_i(Y_i')^T \Delta F_j(Y_j'),$$

$$s.t.\ \ \forall_i \sum_{Y_i'} \alpha_i(Y_i') = \frac{1}{2\lambda},\ \ \forall_i \forall_{Y_i'} \alpha_i(Y_i') \geq 0.$$

- Notes:
  - this QP has an exponential number of constraints/variables
  - the constraints take the form of an unnormalized distribution over label configurations
- We are going to write this QP in terms of marginals of this distribution

Marginal Representation

- If the distribution factorizes into node and edge potentials, we can write the marginals of the distribution as:

$$\mu_i(y_{ij}) = \sum_{Y_i' \sim [y_{ij}]} \alpha_i(Y_i'),$$

$$\mu_i(y_{ij}, y_{ik}) = \sum_{Y_i' \sim [y_{ij}, y_{ik}]} \alpha_i(Y_i'),$$

- We must satisfy the constraints of the original problem:

$$\forall_i \forall_j \ \mu_i(y_{ij}) \geq 0, \ \ \forall_i \ \sum_j \mu_i(y_{ij}) = \frac{1}{2\lambda}.$$

- We also need the node and edge marginals to lie in the 'marginal polytope'. For chains/trees/forests, it is sufficient to enforce a local consistency condition:

$$\forall_i \forall_{(j,k) \in E} \sum_{y_i j} \mu_i(y_{ij}, y_{ik}) = \mu_i(y_{ik}), \ \ \forall_i \forall_{(j,k) \in E} \ \mu_i(y_{ij}, y_{ik}) \geq 0.$$

Polynomial-Sized Dual

- We can re-write the first set of terms in the dual using these marginals:

$$\sum_{Y_i'} \alpha_i(Y_i')\Delta(Y_i, Y_i') = \sum_{Y_i'} \sum_j \alpha_i(Y_i')\Delta_j(y_{ij}, y_{ij}') = \sum_j \sum_{y_{ij}'} \Delta_j(y_{ij}, y_{ij}')\mu_i(y_{ij}').$$

- We can similarly write the second set of terms, yielding a polynomial-sized version of the dual problem:

$$\max_\mu \sum_i \sum_j \sum_{y_{ij}'} \Delta_j(y_{ij}, y_{ij}')\mu_i(y_{ij}') -$$

$$\frac{1}{2} \sum_i \sum_{i'} \sum_{(j,k)\in E} \sum_{(j',k')\in E} \sum_{y_{ij}', y_{ik}'} \sum_{y_{i'j'}', y_{i'k'}'} \mu_i(y_{ij}', y_{ik}')\mu_{i'}(y_{i'j'}', y_{i'k'}')F_i(X_i, y_{ij}', y_{ik}')^T F_i(X_i, y_{i'j'}', y_{i'k'}'),$$

$$s.t.\ \ \forall_i \forall_j\ \mu_i(y_{ij}) \geq 0,\ \ \forall_i\ \sum_j \mu_i(y_{ij}) = \frac{1}{2\lambda},$$

$$\forall_i \forall_{(j,k)\in E} \sum_{y_i j} \mu_i(y_{ij}, y_{ik}) = \mu_i(y_{ik}),\ \ \forall_i \forall_{(j,k)\in E}\ \mu_i(y_{ij}, y_{ik}) \geq 0.$$

- 'Structural SMO'; coordinate descent on this problem

Exponentiated Gradient

- An alternative to using an explicity formulation of the dual is to use an implicit formulation apply the exponentiated gradient (EG) algorithm.
- The EG algorithm solves optimization problem where the variables take the form of a distribution:

$$\forall_i x_i \geq 0, \quad \sum_i x_i = 1.$$

- EG steps take the form:

$$x_i = \frac{x_i \exp(-\eta \nabla_i f(x))}{\sum_{i'} x_{i'} \exp(-\eta \nabla_{i'} f(x))},$$

Exponentiated Gradient

- It is possible to derive a dual where the variables alpha represent a normalized distribution.
- In this case, we can apply the batch or online EG algorithm.
- To make the iterations efficient, an implicit representation for alpha that factorizes according to the graph is used.
- The algorithm requires O(1/eps) iterations to reach an eps-accurate solution.
- Performing the updates using this implicit representation requires inference, instead of just decoding
(so it can’t be applied in general)

Outline

- Formulation:
  - Binary SVMs
  - Multiclass SVMs
  - Structural SVMs
- Training:
  - Subgradients
  - Cutting Planes
  - Marginal Formulations
  - **Min-Max Formulations**

Min-Max Formulations

- Rather than dealing with the exponential number of constraints linear:

$$\min_{w,\xi} \sum_i \xi_i + \lambda ||w||_2^2,$$

$$s.t.\ \ \forall_i \forall_{Y_i'} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq \Delta(Y_i, Y_i') - \xi_i, \ \ \forall_i \xi_i \geq 0.$$

- We could just use one non-linear constraint for each training example:

$$\min_{w,\xi} \sum_i \xi_i + \lambda ||w||_2^2,$$

$$s.t.\ \ \forall_i \log p(Y_i|w, X_i) + \xi_i \geq \max_{Y_i'} \log p(Y_i'|w, X_i) + \Delta(Y_i, Y_i'), \ \ \forall_i \xi_i \geq 0.$$

- In this formulation, we have a constraint on the optimal decoding.

Linear Programming

- The min-max formulation is useful is when the optimal decoding can be formulated as a linear program:

$$\max_{z} \;\; wBz \;\; s.t. \;\; z \geq 0, \;\; Az \leq b,$$

- In this case we can write out the dual of this problem:

$$\min_{z} \;\; b^T z \;\; s.t. \;\; z \geq 0, \;\; A^T z \geq (wB)^T.$$

- And plug it in to the min-max formulation:

$$\min_{w, \xi_z} \sum_i \xi_i + \lambda ||w||_2^2,$$

$$s.t. \;\; \forall_i \log p(Y_i | w, X_i) + \xi_i \geq b^T z, \;\; \forall_i \xi_i \geq 0, \;\; z \geq 0, \;\; A^T z \geq (wB)^T.$$

- This is like changing the max over Z into a max over R

Extragradient Method

- An alternative to plugging the linear program into the QP formulation is to plug it into the unconstrained formulation:

$$\min_{w \in \mathcal{W}} \max_{z \in \mathcal{Z}} \quad \lambda ||w||_2^2 + \sum_i w^T F_i z_i + c_i^T z_i - w^T F(X_i, Y_i),$$

- This problem can be solved using the extragradient method:

$$\begin{aligned} w^p &= \pi_{\mathcal{W}}(w - \eta \nabla_w L(w, z)), \\ z_i^p &= \pi_{\mathcal{Z}}(z_i + \eta \nabla_{z_i} L(w, z)), \\ w^c &= \pi_{\mathcal{W}}(w - \eta \nabla_w L(w^p, z^p)), \\ z_i^c &= \pi_{\mathcal{Z}}(z_i + \eta \nabla_{z_i} L(w^p, z^p)). \end{aligned}$$

- The projection onto Z can be formulated as a quadratic-cost network flow problem.

- The step size is chosen by backtracking, and the algorithm has a linear convergence rate, O(log(1/eps))

Comments on rates of convergence

- $O(1/eps^2)$ is incredibly, incredibly slow
- $O(1/eps)$ is still incredibly slow ('sub-linear' convergence)
- $O(\log(1/eps))$ can be fast, slow, or somewhere in between ('linear convergence')
- $O(\log\log(1/eps))$ is fast ('quadratic' convergence)

- Open question: can we get a practical $O(\log\log(1/eps))$ algorithm, or an $O(\log(1/eps))$ algorithm with a provably nice constant in the rate of convergence.