Training a Binary Classifier with the Quantum Adiabatic Algorithm

Hartmut Neven[1] Vasil S. Denchev[2]
Geordie Rose and William G. Macready[3]

[1]Google

[2]Purdue University

[3]D-Wave Systems

December 12, 2008

Outline

Background: Adiabatic quantum computing

Training a binary classifier

Modifications to allow the application of the quantum adiabatic algorithm

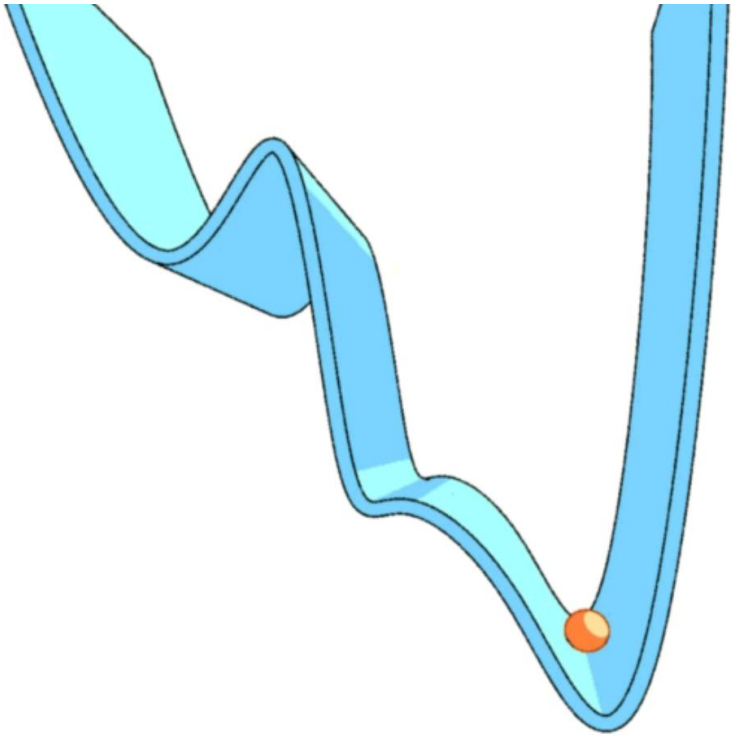
Implementation details

Performance measurements on benchmark problems

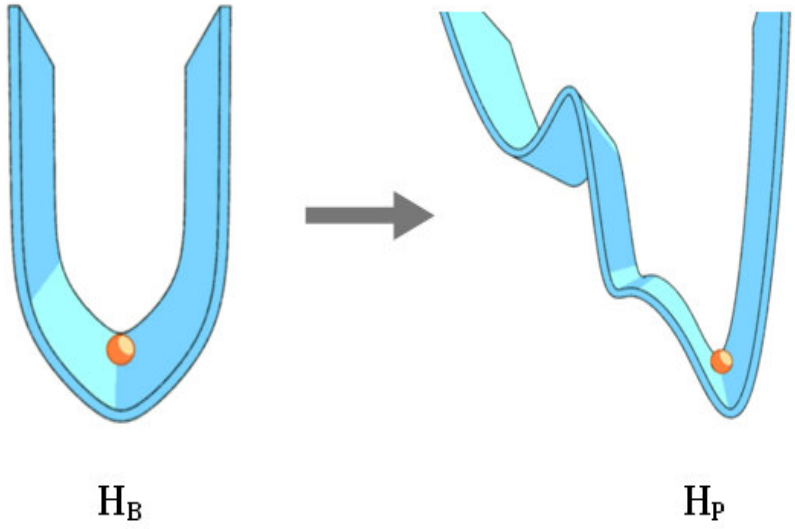
Conclusions

Solving hard optimization problems using adiabatic quantum computing

- Given an objective function $H : D \rightarrow \mathbb{R}$
- Find x_0 such that $\forall x : H(x_0) \leq H(x)$



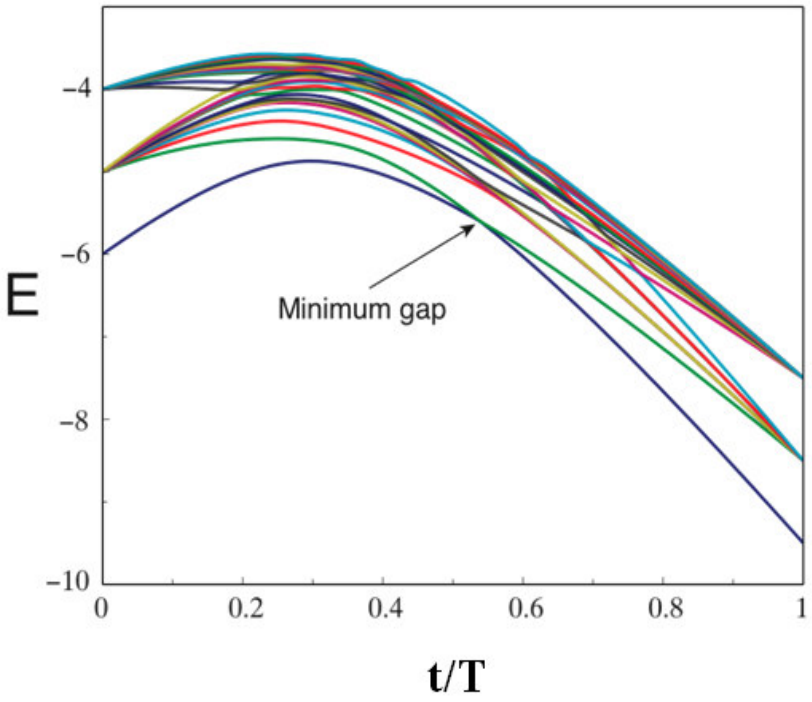
Solving hard optimization problems using adiabatic quantum computing



- Define: $H(t) = (1 - t/T)H_B + (t/T)H_P$
- H_B: initial Hamiltonian with known and easily preparable ground state
- H_P: Hamiltonian whose ground state encodes the solution to a given instance of an optimization problem

The adiabatic theorem

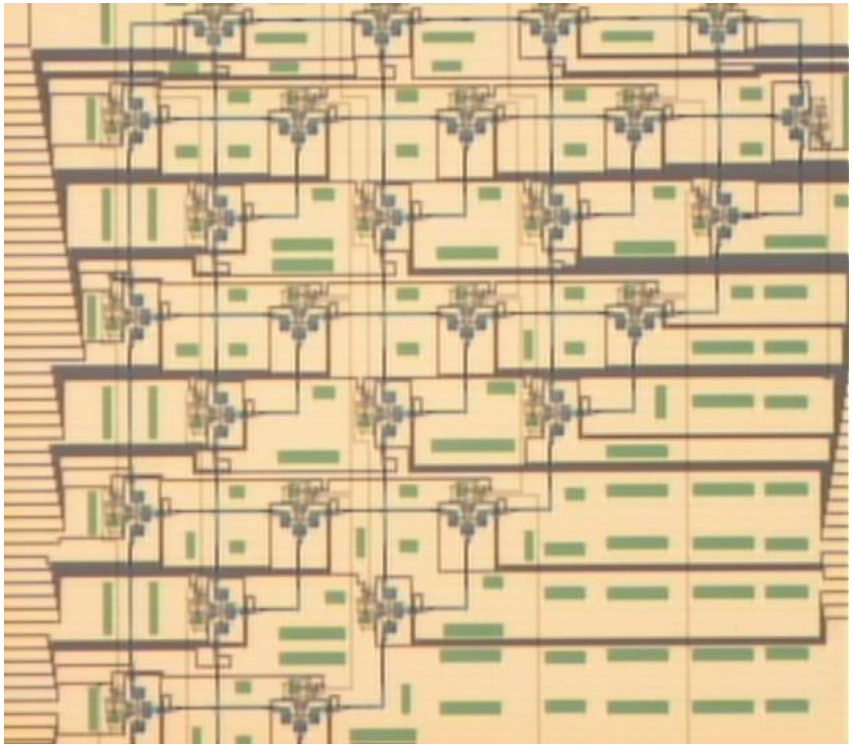
- Quantum evolution: $i\frac{d}{dt}|\psi(t)\rangle = H(t)|\psi(t)\rangle$
- $H(t)$: slowly varying Hamiltonian for evolution from $t = 0$ to $t = T$
- Provided T is "large enough", a quantum system starting in the ground state of $H(0)$ evolves to the ground state of $H(T)$
- Large enough: $T = \Omega(g_{min}^{-2})$, where g_{min} is the minimum gap between ground state and first excited state of $H(t)$

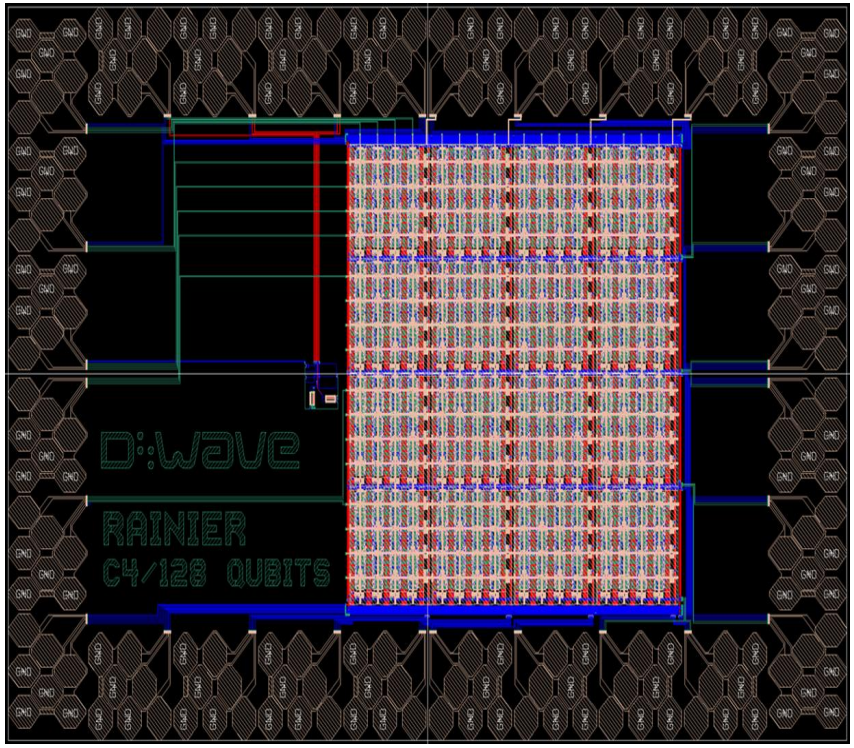


Why bother using AQC?

- Exponential speed-ups over classical approaches for certain typical-case NP-hard problems
- Could have large impact on fundamental machine learning problems as well

D-Wave's hardware





- Superconducting quantum processor
  - Current generation has 128 qubits
  - Scaling up to 1000's of qubits soon

D-Wave's hardware

- "Co-processor" to minimize an Ising function
- Optimization problems need to be formulated as **Q**uadratic **U**nconstrained **B**inary **O**ptimization QUBO
- Minimize E over binary variables x_i; h_i & $J_{ij} \in \mathbb{R}$

$$E(x_1, \cdots, x_N) = \sum_{i=1}^{N} h_i x_i + \sum_{i<j=1}^{N} J_{ij} x_i x_j$$

- Some additional limitations related to connectivity between qubits and precision of coefficients

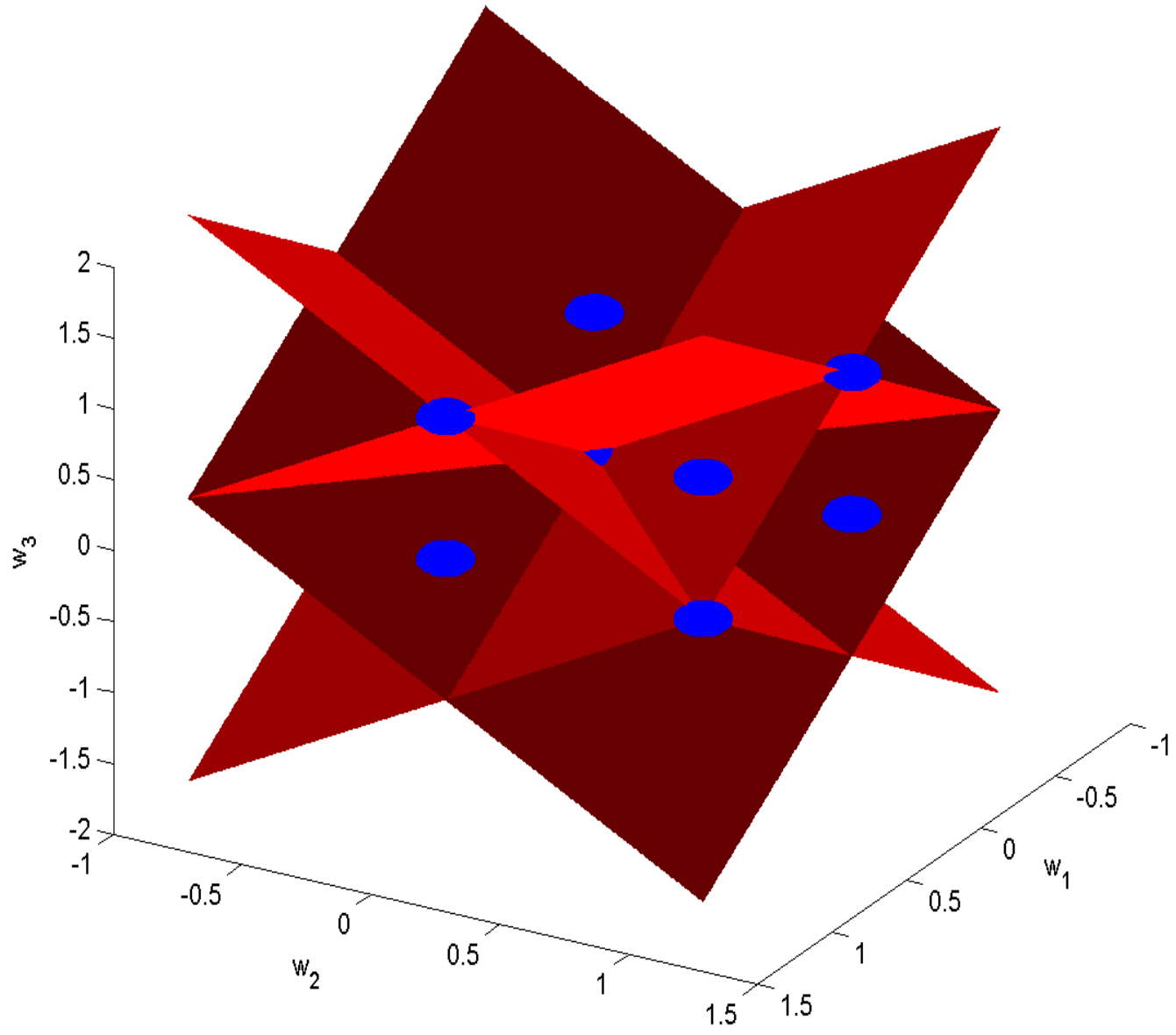
Training a binary classifier

- Strong classifier: $y = \operatorname{sign}\left(\sum_{i=1}^{N} w_i h_i(x)\right)$, where
  - $x \in \mathbb{R}^M$ (input patterns to be classified)
  - $y \in \{-1, 1\}$ (output)
  - $h_i : x \mapsto \{-1, 1\}$ (weak classifiers)
  - $w_i \in [0, 1]$ (weights to be optimized)
- Training: $w^{opt} = \arg\min_{w} \left(\sum_{s=1}^{S} \mathbf{H}\left(-y_s \sum_{i=1}^{N} w_i h_i(x_s)\right) + \lambda \sum_{i=1}^{N} {w_i}^0\right)$

Hyperplanes

- The per-sample loss, $\mathbf{H}\left(-y_s \sum_{i=1}^{N} w_i h_i(x_s)\right)$ enforces an inequality constraint:
  - $y_s \sum_{i=1}^{N} w_i h_i(x_s) \geq 0$ for $s = 1, \ldots, S$
- Each sample invokes a "diagonal hyperplane" in N-dimensional space
  - Hyperplane separates satisfying from non-satisfying weight configurations
- Possible solution subspaces divided into disjoint regions by the intersection of hyperplanes
- $N_{regions} \leq \sum_{k=0}^{N} \binom{S}{k}$

Hyperplanes in $N = 3$



Modifications I: reduce bits to represent weights

- Need discrete weights for hardware implementation
- Qubits very expensive
  - Use weights with only the necessary number of bits
- Finite-bit weight configurations lie on hypercubic lattice with $(2^{bits})^N$ vertices
- Goal: roughly 1 lattice point in each region

$$\frac{\textit{Vertices on Lattice}}{\textit{Regions in Positive Quadrant}} \approx \frac{(2^{bits})^N}{\frac{N_{regions}}{2^N}} \geq \frac{2^{(bits+1)N} N^N}{(eS)^N} \overset{!}{\geq} 1$$

$$\Rightarrow \quad \left(\frac{2^{(bits+1)} N}{eS}\right)^N = \left(\frac{2^{(bits+1)} N}{efN}\right)^N = \left(\frac{2^{(bits+1)}}{ef}\right)^N \overset{!}{\geq} 1$$

$$\Rightarrow \quad bits \geq \log_2(f) + \log_2(e) - 1 \qquad \text{with } f = \frac{S}{N}$$

Modifications II: quadratic loss

- Another modification imposed by hardware limitations: formulate optimization as QUBO

$$w^{opt} = \arg\min_{w} \left(\sum_{s=1}^{S} | \sum_{i=1}^{N} w_i h_i(x_s) - y_s|^2 + \lambda \parallel w \parallel_0 \right)$$

$$= \arg\min_{w} \left(\sum_{s=1}^{S} \left(\left(\sum_{i=1}^{N} w_i h_i(x_s) \right)^2 - 2 \sum_{i=1}^{N} w_i h_i(x_s) y_s + y_s^2 \right) + \lambda \sum_{i=1}^{N} {w_i}^0 \right)$$

$$= \arg\min_{w} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} w_i w_j \underbrace{\left(\sum_{s=1}^{S} h_i(x_s) h_j(x_s) \right)}_{Corr(h_i, h_j)} + \sum_{i=1}^{N} w_i \left(\lambda - 2 \underbrace{\sum_{s=1}^{S} h_i(x_s) y_s}_{Corr(h_i, y)} \right) \right)$$

Implementation details: dictionaries

- Dictonaries of weak classifiers:

$$
\begin{aligned}
h_l^{1+}(x) &= sign(x_l - \Theta_l^+) \text{ for } l = 1, \ldots, M \\
h_l^{1-}(x) &= sign(-x_l - \Theta_l^-) \text{ for } l = 1, \ldots, M \\
h_l^{2+}(x) &= sign(x_i x_j - \Theta_{i,j}^+) \text{ for } l = 1, \ldots, \binom{M}{2}; i, j = 1, \ldots, M; i < j \\
h_l^{2-}(x) &= sign(-x_i x_j - \Theta_{i,j}^-) \text{ for } l = 1, \ldots, \binom{M}{2}; i, j = 1, \ldots, M; i < j
\end{aligned}
$$

  - h_l^{1+}, h_l^{1-}, h_l^{2+}, h_l^{2-}: positive and negative stumps of orders 1 and 2
  - M: dimensionality of input vector x
  - Θ_l^+, Θ_l^-, $\Theta_{i,j}^+$, $\Theta_{i,j}^-$: thresholds that minimize training errors due to individual weak classifiers

Implementation details: classical optimization methods

- Simulated annealing used for minimizing 0-1 loss formulation
- Tabu search used for minimizing square error formulation
- Computing threshold for strong classifier:

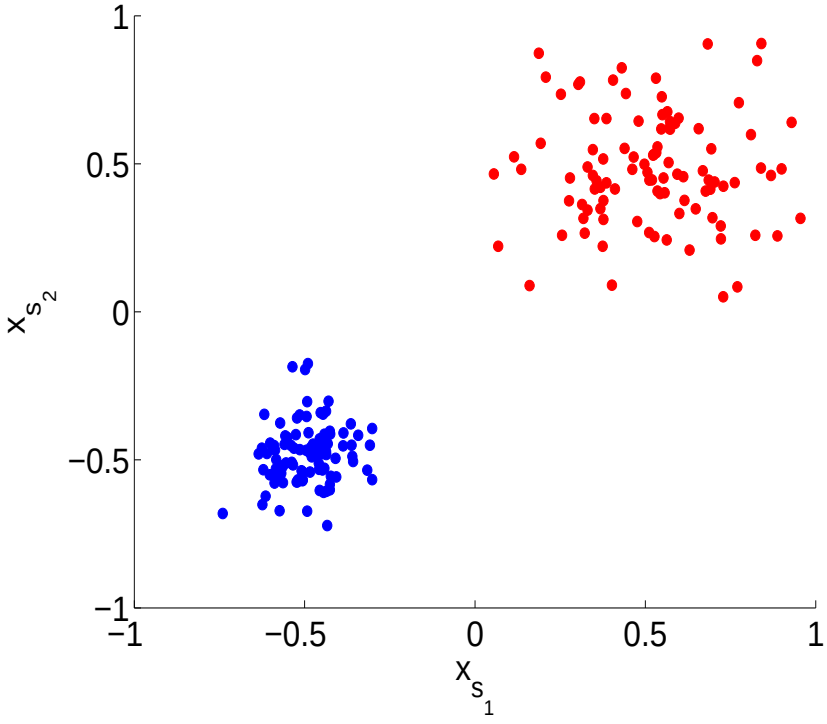
$$T = \frac{1}{S} \sum_{s=1}^{S} \sum_{i=1}^{N} w_i^{opt} h_i(x_s)$$

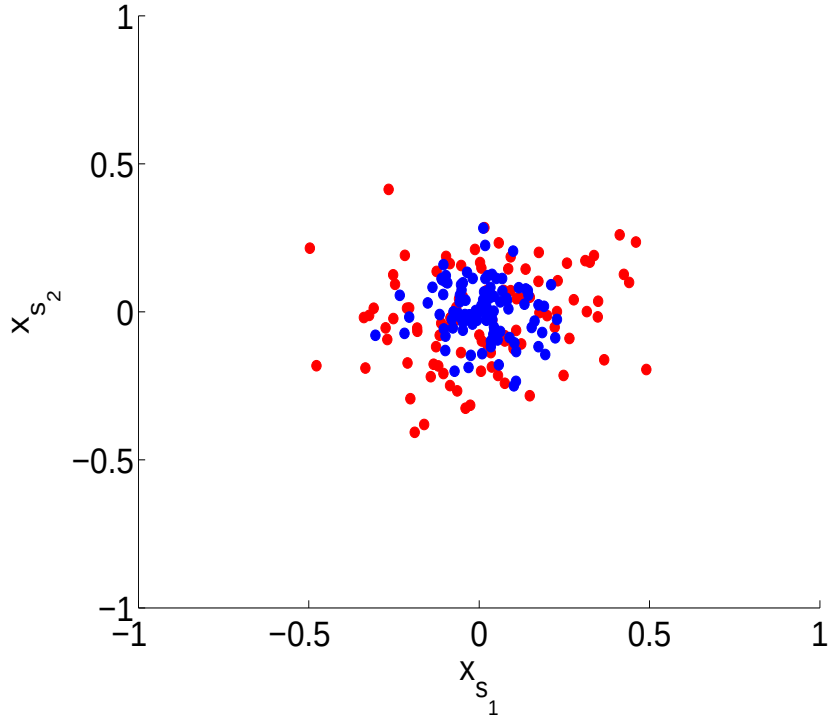
- Final classifier:

$$y = \text{sign} \left(\sum_{i=1}^{N} w_i^{opt} h_i(x) - T \right)$$

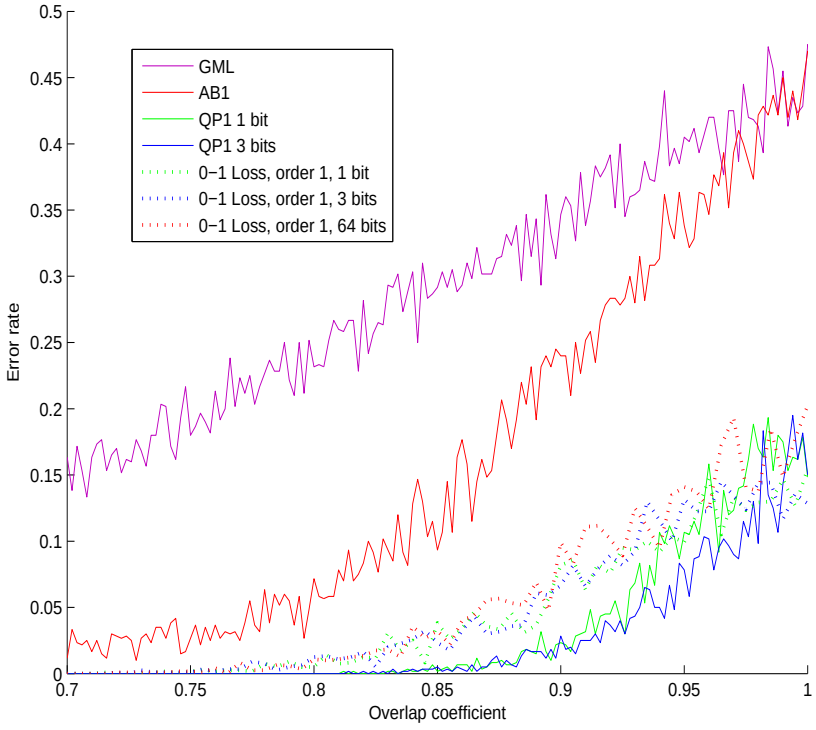
- 30-fold cross-validation to find regularization strength λ that gives best generalization
- Total number of non-zero weights limited to $N/2$

Experiments I: Synthetic data

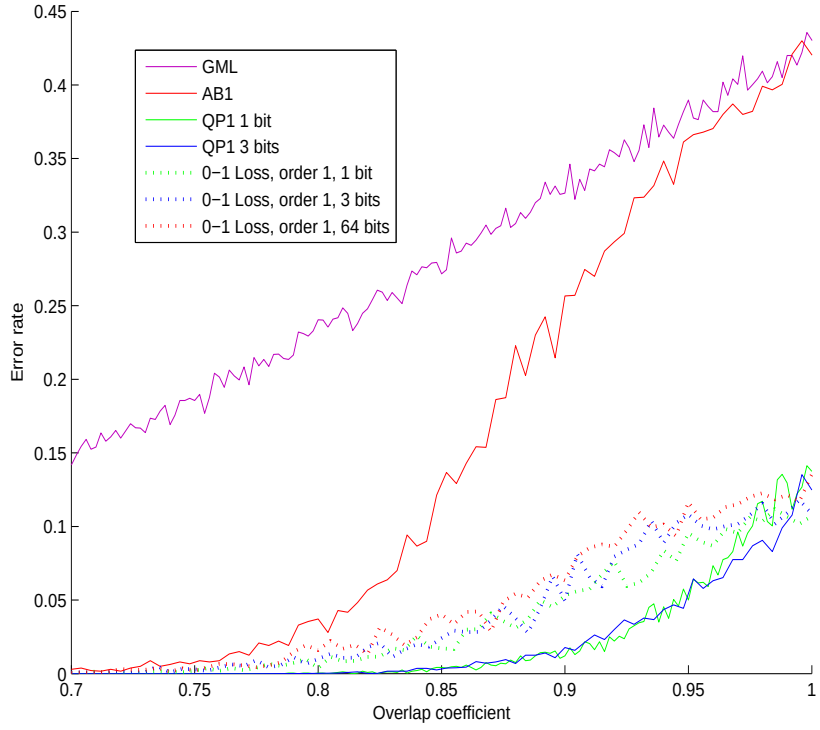




Results for synthetic data with order 1 dictionary

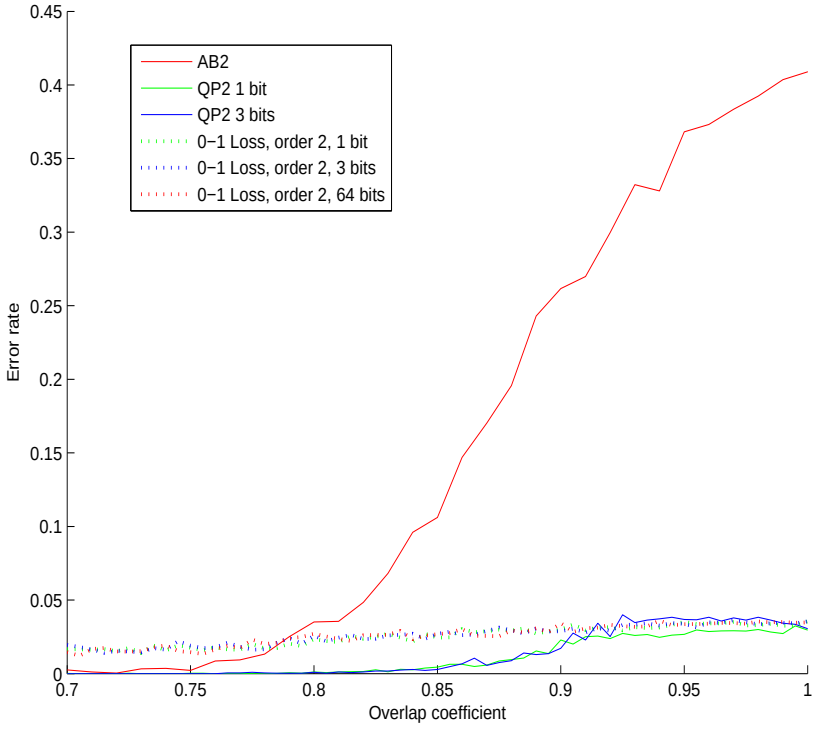


f = 1

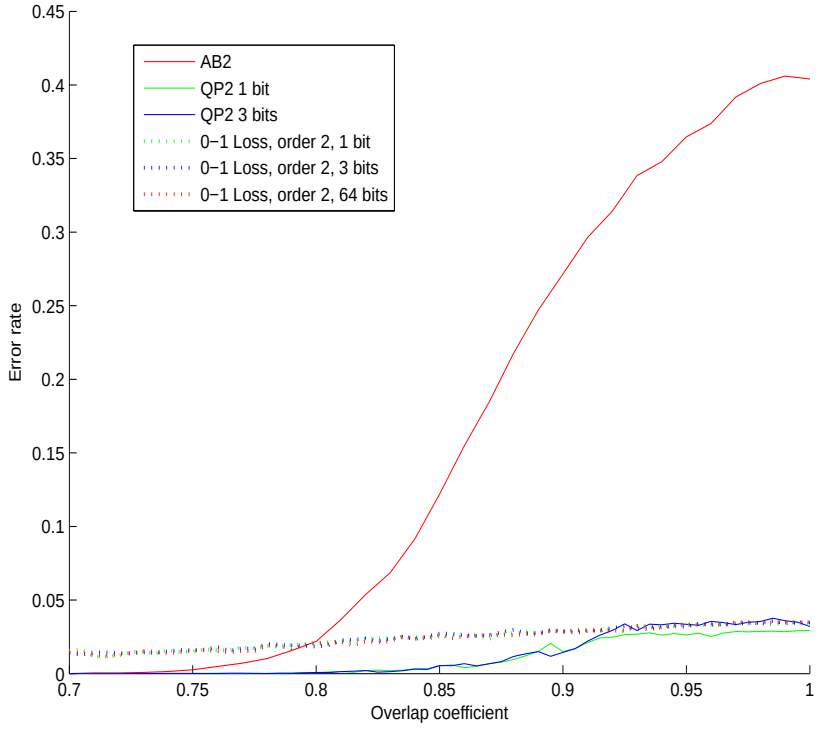


f = 8

Results for synthetic data with order 2 dictionary



f = 1



f = 8

Experiments II: Natural data

	f = 1	f = 8
	64 bits	**64 bits**
GML	0.3346, 30	0.3123, 30
AB1	0.3245, 30	0.2842, 30
AB2	0.2690, 465	0.2580, 465

	f = 1			f = 8		
	1 bit	**3 bits**	**64 bits**	**1 bit**	**3 bits**	**64 bits**
0-1 Loss, Order 1	0.3074, 22.98	0.3078, 21.86	0.3085, 21.9	0.3054, 22.3	0.3084, 21.78	0.3070, 20.9577
0-1 Loss, Order 2	0.2942, 218.53	0.2986, 196.25	0.3003, 192.19	0.2983, 170	0.2890, 183.75	0.2878, 173.33

	f = 1		f = 8	
	1 bit	**3 bits**	**1 bit**	**3 bits**
QP1	0.2936, 11.35	0.3126, 18.6	0.2794, 11.4	0.2912, 18.9
QP2	0.2684, 176.45	0.3035, 297.22	0.2579, 206	0.2825, 124.57

Conclusions

- Global optimization competes successfully with greedy methods
- Bit-constrained learning machines often exhibit lower generalization error
  - Intrinsic regularization
  - Required bit precision grows only logarithmically with $\frac{S}{N}$
- Training benefits from being treated as an integer program
  - Good news for cell phones, sensor networks, early quantum chips
  - Training problem manifestly NP-hard: motivates using AQC
- Next steps
  - Experiment with 128-qubit D-Wave hardware
  - Adaptive dictionaries
  - Co-training of classifiers with feature sharing