Quantum-Classical Sentiment Analysis

Mario Bifulco

Relatore: Luca Roversi

Università degli Studi di Torino

Goal

Testing the effectiveness of quantum computation on machine learning tasks

Using the QPU extensively while avoiding the closed source alternatives of D-Wave

Quantum Annealing

QA is an optimization process for finding the global minimum of a given objective function over a given set of candidate solutions.

Quantum Annealing
QA is an optimization process for finding the global minimum of a given objective function over a given set of candidate solutions.

Sentiment Analysis
SA is the use of natural language processing to systematically identify, extract, quantify, and study affective states and subjective information.

Applying Quantum Annealing to Sentiment Analysis

Quantum Annealing
QA is an optimization process for finding the global minimum of a given objective function over a given set of candidate solutions.

Sentiment Analysis
SA is the use of natural language processing to systematically identify, extract, quantify, and study affective states and subjective information.

Goal of Machine Learning
Minimizing a loss function.

Using Quantum Annealing for Sentiment Analysis

From Machine Learning to Quantum Annealing

Model Selection

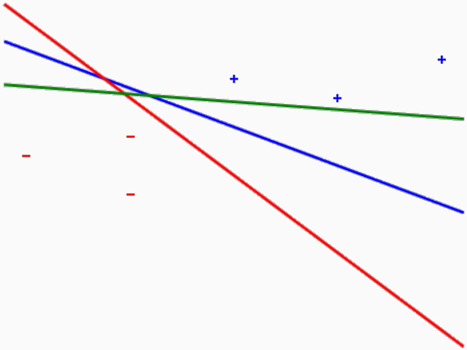
↓

Reduction to Minimization of Quadratic Programming

↓

Solve with D-Wave Hybrid Solver

Support Vector Machine

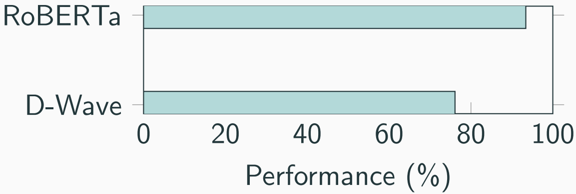


Why using SVM?

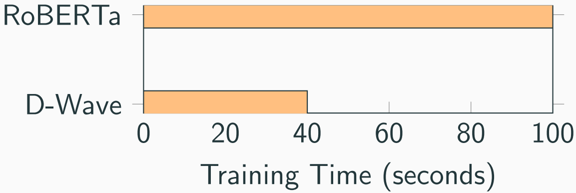
- Machine learning model for binary classification
- Native formulation as quadratic programming problem

Binary Sentiment Analysis Results

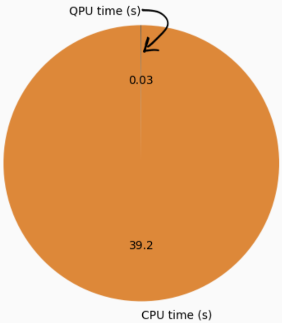
D-Wave is worse than RoBERTa



D-Wave converges in only 40s



Usage of QPU to Obtain the Above BSA



- Only 0.08% of the time is spent on the QPU
- The performance boost should comes from quantum annealing
- Is it possible to increase the QPU usage?
- Is it possible to avoid the closed-source procedures of D-Wave?

Increase QPU usage

From Quadratic Programming to QUBO

Quadratic Programming Problem

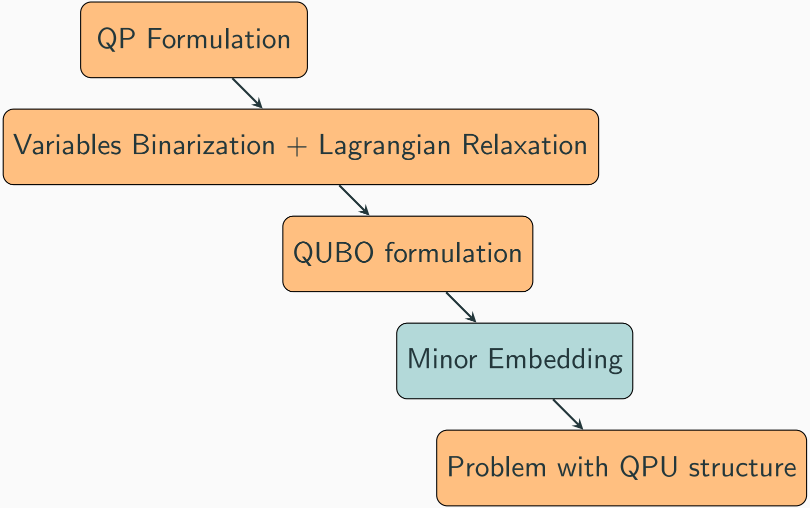
$$
\begin{aligned}
\min_{x} \quad & \frac{1}{2} x^T Q x + c^T x \\
\text{s.t.} \quad & Ax \preceq b \\
& x_i \in \mathbb{R}, \forall i
\end{aligned}
$$

$$\Longrightarrow$$

QUBO Problem

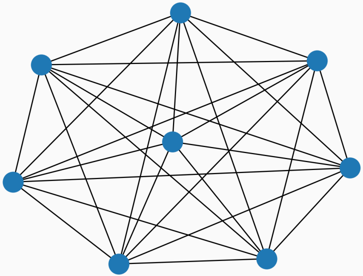
$$
\begin{aligned}
\min_{x} \quad & x^T Q x \\
& x_i \in \{0, 1\}, \forall i
\end{aligned}
$$

From Quadratic Problem to QPU

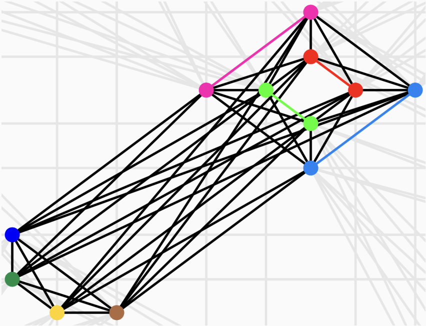


From Quadratic Problem to QPU

SVM graph for 8 variables $\Longrightarrow$ Mapped problem on Pegasus QPU

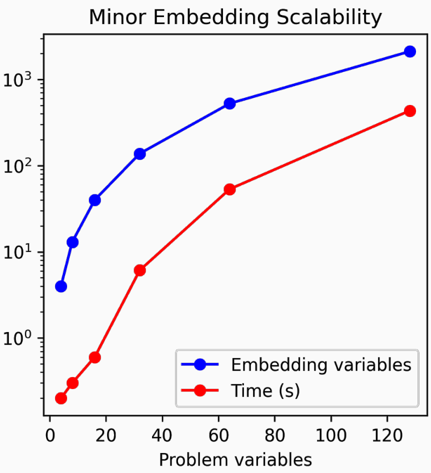


$\Longrightarrow$



Pegasus QPU is not enough

For SVMs with 128 binary variables, the QPU is almost completely used



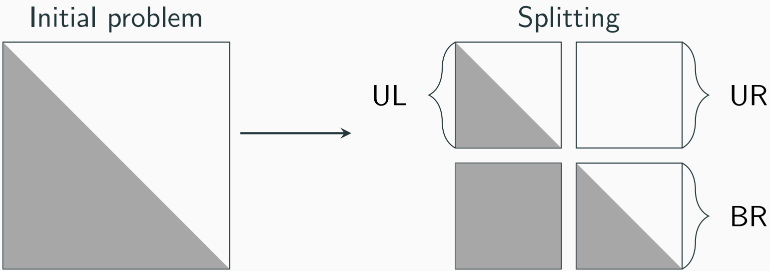
QUBO Formulation

The QPU process directly problems in QUBO form:

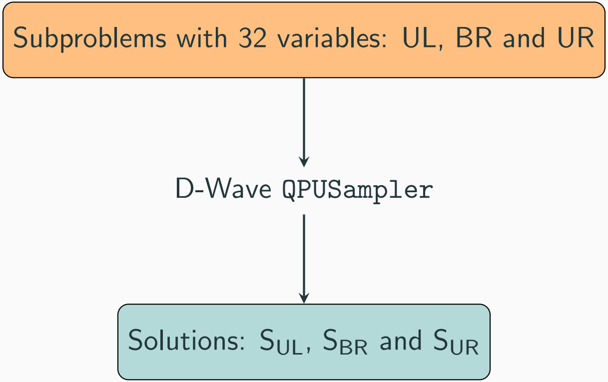
$$\min_{x} \quad x^T Q x$$
$$x_i \in \{0, 1\}, \forall i$$

Example with 4 binary variables:

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} \begin{bmatrix} -3 & 5 & 1 & -7 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 9 & -4 \\ 0 & 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

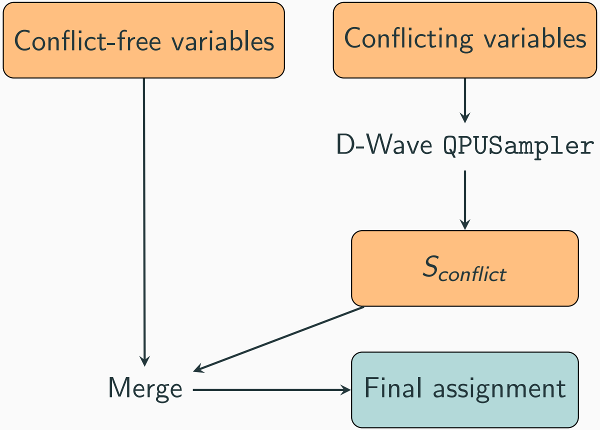

Initial problem
Splitting
UL
UR
BR

QSplitSampler - Solving Subproblems



QSplitSampler - **Aggregate Solutions**

Aggregating S_{UL}, S_{BR} and S_{UR} gives two sets of variables



QSplitSampler Results

	QSplitSampler		QPUSampler	
Cut Dim	Classical time	QPU time	Classical time	QPU time
2	470.71	0.45	142.54	0.02
4	223	0.35	116.02	0.02
8	94.97	0.25	271.36	0.02
16	68.33	0.17	190.29	0.02
32	48.46	0.1	207.86	0.02

Table 1: Results for randomly generated QUBO problems with 128 variables

Conclusion

Using only the **free version** of the D-Wave solvers, we showed:

- How hybrid solvers can be used for real problems
- How closed source infrastructure can be avoided
- How the use of QPU can be increased

Thanks for your attention