Generalizations of SAT

- Constraint programming (CP)
  - allows integer variables
  - more operators (besides $\neg, \lor, \land$), such as all-different($x_1, \ldots, x_k$).
- Satisfiability modulo theories (SMT)
  - other types of operators,
  - introduces bit vectors, etc.
Nonlinear optimization

Consider

$$\begin{align*}
\text{min } & f(x) \\
\text{s.t. } & g_i(x) \leq 0, \text{ for } i=1, \ldots, m \\
& x \in \mathbb{R}^n
\end{align*}$$

where $f$ and $g_i$ are functions $\mathbb{R}^n \rightarrow \mathbb{R}$, not necessarily linear.

Black-box/oracle model

$f$ and $g_i$ can be any functions, but we can evaluate them and their derivatives at any point $x \in \mathbb{R}^n$. 
Theorem: In the absence of further assumptions, (OPT) is unsolvable.

Example: find an optimal solution \( x^* \) to

\[
\min f(x) \quad \text{s.t.} \quad x \in \mathbb{R}
\]

\[
f(x) = \begin{cases} 
0 & \text{if } x = \lambda \text{ for some } \lambda \in \mathbb{R} \\
1 & \text{otherwise}
\end{cases}
\]

This problem is equivalent to "guess \( 1 \), \( \lambda \in \mathbb{R} \)."
Definition: Consider $f: \mathbb{D} \to \mathbb{R}$. The point $x^* \in \mathbb{D}$ is

- a **global minimizer** (optimal solution) if $f(x^*) \leq f(x)$, $\forall x \in \mathbb{D}$.

- a **local minimizer** if $\exists \delta > 0$

  $f(x^*) \leq f(x)$, $\forall x \in \mathbb{D} : \|x - x^*\| < \delta$.
Definition  Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable function. Its gradient $\nabla f \in \mathbb{R}^n \to \mathbb{R}^n$ is given by:

$$\nabla f (x) = \begin{bmatrix}
\frac{\partial f}{\partial x_1} (x) \\
\vdots \\
\frac{\partial f}{\partial x_n} (x)
\end{bmatrix}$$

Example  Let $f(x) = x_1^2 + 2x_2^2 - 3x_1x_2 + x_1$.

Then $\nabla f (x) = \begin{bmatrix}
\frac{\partial f}{\partial x_1} (x) \\
\frac{\partial f}{\partial x_2} (x)
\end{bmatrix} = \begin{bmatrix}
2x_1 - 3x_2 + 1 \\
4x_2 - 3x_1
\end{bmatrix}$
What are the values of $f$ and $\nabla f$ at $x = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$?

$f(\begin{bmatrix} 2 \\ 1 \end{bmatrix}) = 2^2 + 2.1^2 - 3.2.1 + 2 = 2$

$\nabla f(\begin{bmatrix} 2 \\ 1 \end{bmatrix}) = \begin{bmatrix} 2.2 - 3.1 + 1 \\ 4.1 - 3.2 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$
Assumption: \( f \) is differentiable and its derivatives are continuous.

Theorem: If \( p^k \cdot \nabla f(x^k) < 0 \), then there exists \( d^k > 0 \) such that
\[
    f(x^k + d^k p^k) < f(x^k)
\]

Theorem: If \( p^k = -\frac{\nabla f(x^k)}{||\nabla f(x^k)||} \), then there exists \( d^k > 0 \) such that
\[
    f(x^k + d^k p^k) = \min \{ f(x) : ||x - x^k|| \leq d^k \}
\]
**Note:** If \( x^k \) is a local minimizer, \( \nabla f(x^k) = 0 \). (But \( \nabla f(x^k) = 0 \) does not imply \( x^k \) is a local minimizer)

**Note:** Choosing always \( p^k = -\frac{\nabla f(x^k)}{\|\nabla f(x^k)\|} \) yields "steepest descent".
Algorithm to find local minimizers for unconstrained problems.

**Gradient descent**

Choose any $x^0 \in \mathbb{R}^n$

for $k = 0, 1, 2, ...$

choose search direction $p^k \in \mathbb{R}^n$:
- $\|p^k\| = 1$, and
- $p^k \cdot \nabla f(x^k) < 0$

choose step length $d^k > 0$

let $x^{k+1} = x^k + d^k p^k$
Classification / labelling problem

We are given a training set: many input vectors $x^i \in [0, 1]^n$, already labelled into $k$ categories. Find "good" labels to more vectors of $[0, 1]^n$. 
Neural network: A (trained) neural network (NN) provides a function $F(x) \in \mathbb{R}^n \rightarrow \mathbb{R}^k$, such that if $x \in [0, 1]^n$ is an input vector, then the largest component

$$k' = \arg\max_j \{ (F(x))_j \}$$

of $F(x)$ is the predicted category.
Example: If $x = \begin{bmatrix} 0.3 \\ 0.5 \end{bmatrix}$ and $F(x) = \begin{bmatrix} 0.1 \\ 0.9 \\ 0.2 \\ 0.2 \end{bmatrix}$, then the predicted category for $x$ is 2.

1. Given a NN, how do we compute $F(x)$?

2. How do we get a NN to be a good classifier.
One neuron

\[ \text{inputs} \rightarrow \text{output} = \sigma \text{ (linear combination of the inputs)} \]

Typical choice of \( \sigma \):
- Sigmoid function: \( \sigma(x) = \frac{1}{1 + e^{-x}} \)
rectified linear unit (ReLU):
\[
\sigma = \begin{cases} 
0, & \text{if } x \leq 0 \\
 x, & \text{if } x > 0 
\end{cases}
\]
A neural network:

- the neural graph is fixed
- the $\sigma$ function is fixed
- what can change is the linear combination of inputs in each neuron

\[ y = \sigma (w^T x + b) \]