Gradient ascent / descent

• Gradient descent procedure for finding the $\arg_x \min f(x)$
  – choose initial $x_0$ randomly
  – repeat
    • $x_{i+1} \leftarrow x_i - \eta f'(x_i)$
  – until the sequence $x_0, x_1, \ldots, x_i, x_{i+1}$ converges
• Step size $\eta$ (eta) is small (perhaps 0.1 or 0.05)
Gradient methods vs. Newton’s method

• A reminder of Newton’s method from Calculus:
  \[ x_{i+1} \leftarrow x_i - \eta f'(x_i) / f''(x_i) \]

• Newton’s method uses 2\textsuperscript{nd} order information (the second derivative, or, curvature) to take a more direct route to the minimum.

• The second-order information is more expensive to compute, but converges quicker.
Exploring the Landscape

- **Local Maxima**: peaks that aren’t the highest point in the space

- **Plateaus**: the space has a broad flat region that gives the search algorithm no direction (random walk)

- **Ridges**: flat like a plateau, but with drop-offs to the sides; steps to the North, East, South and West may go down, but a step to the NW may go up.
Simulated annealing

• Simulated annealing (SA) exploits an analogy between the way in which a metal cools and freezes into a minimum-energy crystalline structure (the annealing process) and the search for a minimum [or maximum] in a more general system.

• SA can avoid becoming trapped at local minima.

• SA uses a random search that accepts changes that increase objective function $f$, as well as some that decrease it.

• SA uses a control parameter $T$, which by analogy with the original application is known as the system “temperature.”

• $T$ starts out high and gradually decreases toward 0.
Simulated annealing (cont.)

- A “bad” move from $A$ to $B$ is accepted with a probability
  
  $$P(\text{move}_{A \rightarrow B}) = e^{(f(B) - f(A)) / T}$$

- The higher the temperature, the more likely it is that a bad move can be made.

- As $T$ tends to zero, this probability tends to zero, and SA becomes more like hill climbing

- If $T$ is lowered slowly enough, SA is complete and admissible.
The simulated annealing algorithm

function SIMULATED-ANNEALING(problem, schedule) returns a solution state

inputs: problem, a problem
schedule, a mapping from time to “temperature”

static: current, a node
next, a node
T, a “temperature” controlling the probability of downward steps

current ← MAKE-NODE(INITIAL-STATE[problem])
for t ← 1 to ∞ do
  T ← schedule[t]
  if T=0 then return current
  next ← a randomly selected successor of current
  ΔE ← VALUE[next] – VALUE[current]
  if ΔE > 0 then current ← next
  else current ← next only with probability $e^{ΔE/T}$