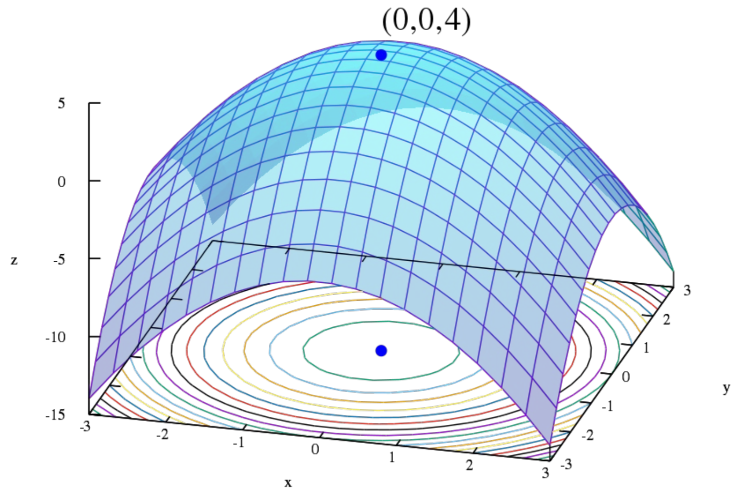
CS440/ECE448 Lecture 11: Optimization

Mark Hasegawa-Johnson, 2/2023



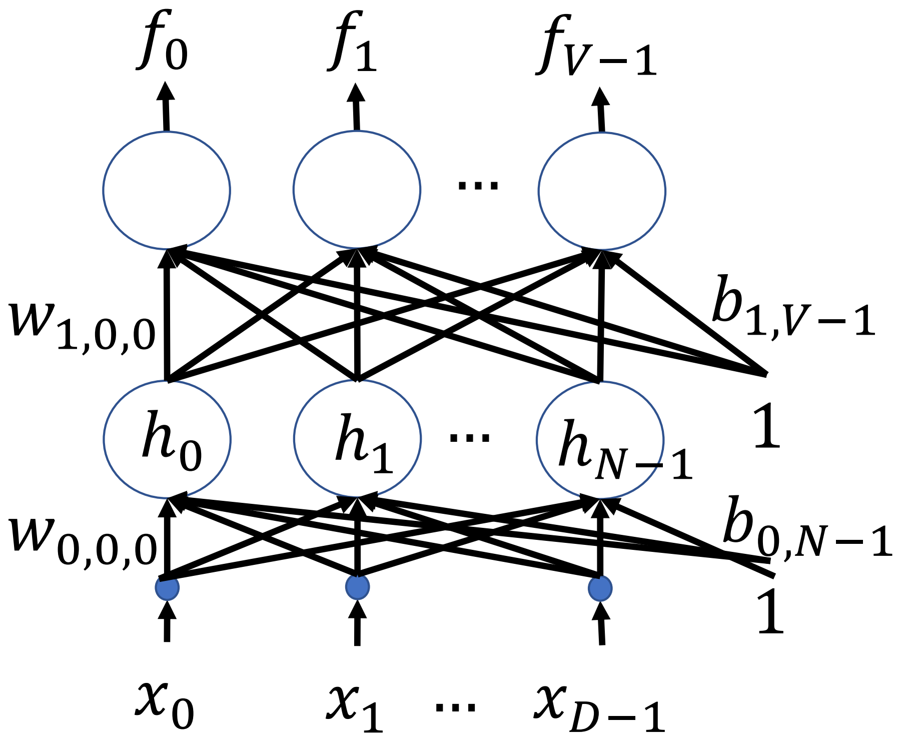
By IkamusumeFan - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=42043175

Outline

- The three types of machine learning error
- The optimization problem
- Local optimization with a known gradient
    - Gradient descent
    - Newton’s method
    - Line search
- Local optimization with unknown gradient
    - Empirical gradients
    - Coordinate search
- Global optimization
    - Random restarts
    - Simulated annealing

Review: Neural net is a universal approximator

- Suppose we have some function, $f(x)$, that we want to approximate using a neural net
- In the limit as $N \rightarrow \infty$, the network shown here can approximate $f(x)$ with zero error

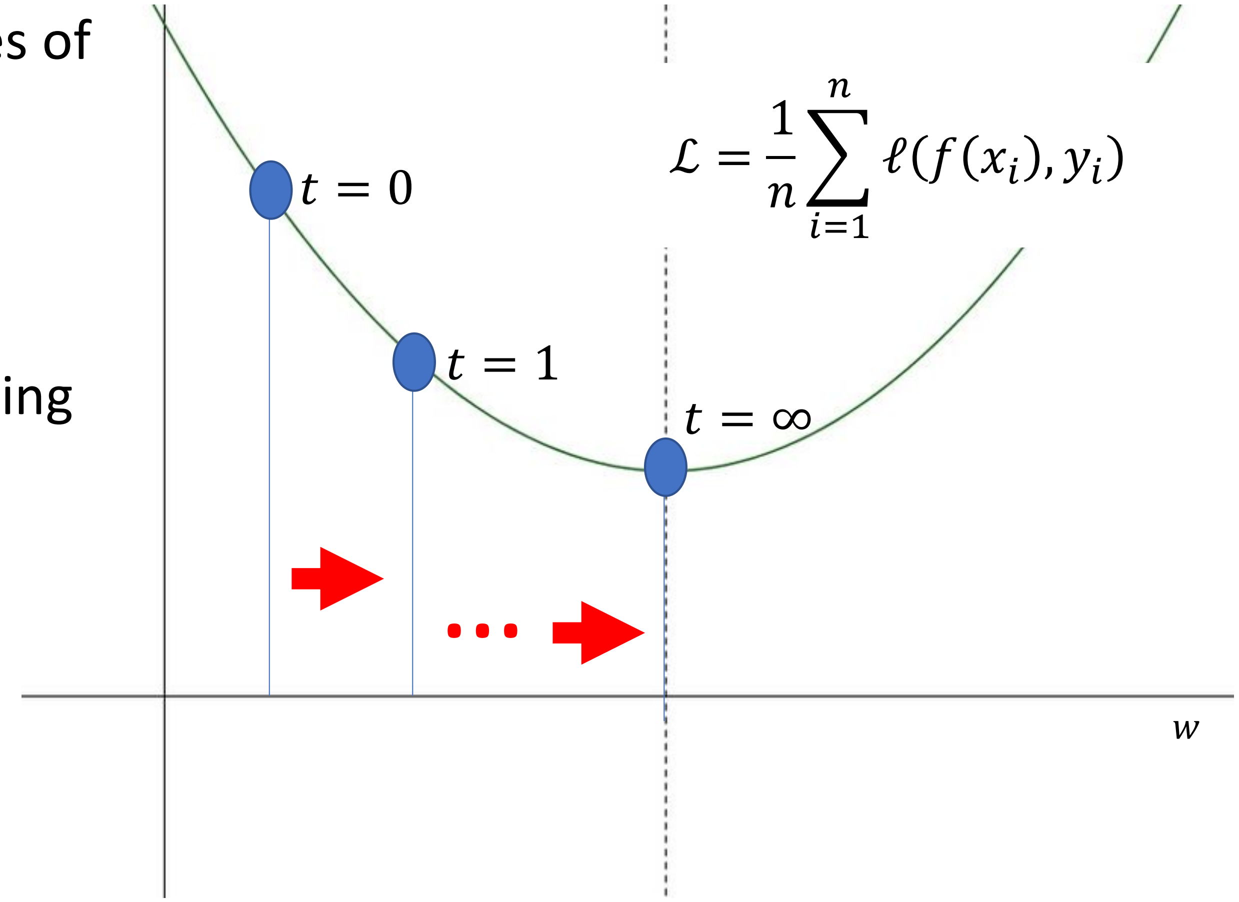


Review: Machine learning

- Start from random initial values of w.
- Adjust w according to:

$$w \leftarrow w - \eta \nabla_w \mathcal{L}$$

- Continue until $\mathcal{L}$ stops decreasing



The three types of machine learning error

- **Approximation error/Underfitting**: You don’t have enough hidden nodes to achieve small $\mathcal{L}$ on the training corpus
- **Generalization error/Overfitting**: You don’t have enough training data; optimum w on the training corpus is not also optimum on the test corpus
- **Optimization error**: There is a value of w that achieves low $\mathcal{L}$ on the training corpus, but finding it is too computationally expensive

Outline

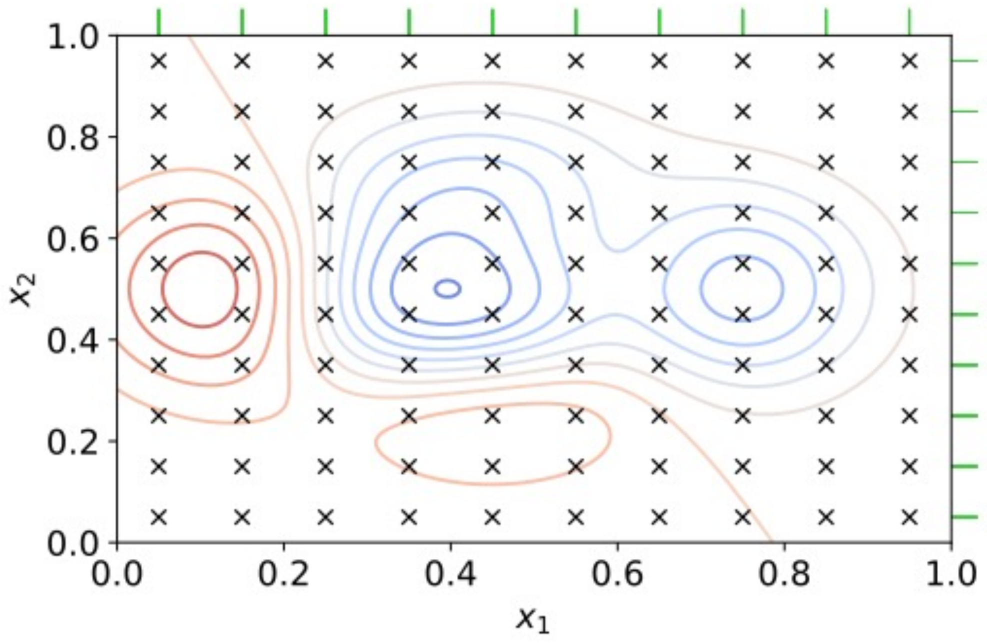
- The three types of machine learning error
- The optimization problem
- Local optimization with a known gradient
    - Gradient descent
    - Newton’s method
    - Line search
- Local optimization with unknown gradient
    - Empirical gradients
    - Coordinate search
- Global optimization
    - Random restarts
    - Simulated annealing

The optimization problem

- Given: a function $\mathcal{L}: \mathcal{W} \rightarrow \mathbb{R}$ that maps from a set of possible weight vectors, $\mathcal{W}$, to the set of real numbers, $\mathbb{R}$
- Find: a value $\widehat{w}$ such that $\mathcal{L}(\widehat{w}) \leq \mathcal{L}(w)$ for all $w \in \mathcal{W}$.

Example of the problem: Grid search

- Suppose we have M network weights
- Suppose we test K possible values of each weight
- Then the computational complexity is $\mathcal{O}\{K^M\}$.



Grid search of 2 hyperparameters, with 10 values each, requires testing $10^2 = 100$ combinations. CC-SA 4.0, Alexander Elvers,

https://commons.wikimedia.org/wiki/File:Hyperparameter_Optimization_using_Grid_Search.svg

Outline

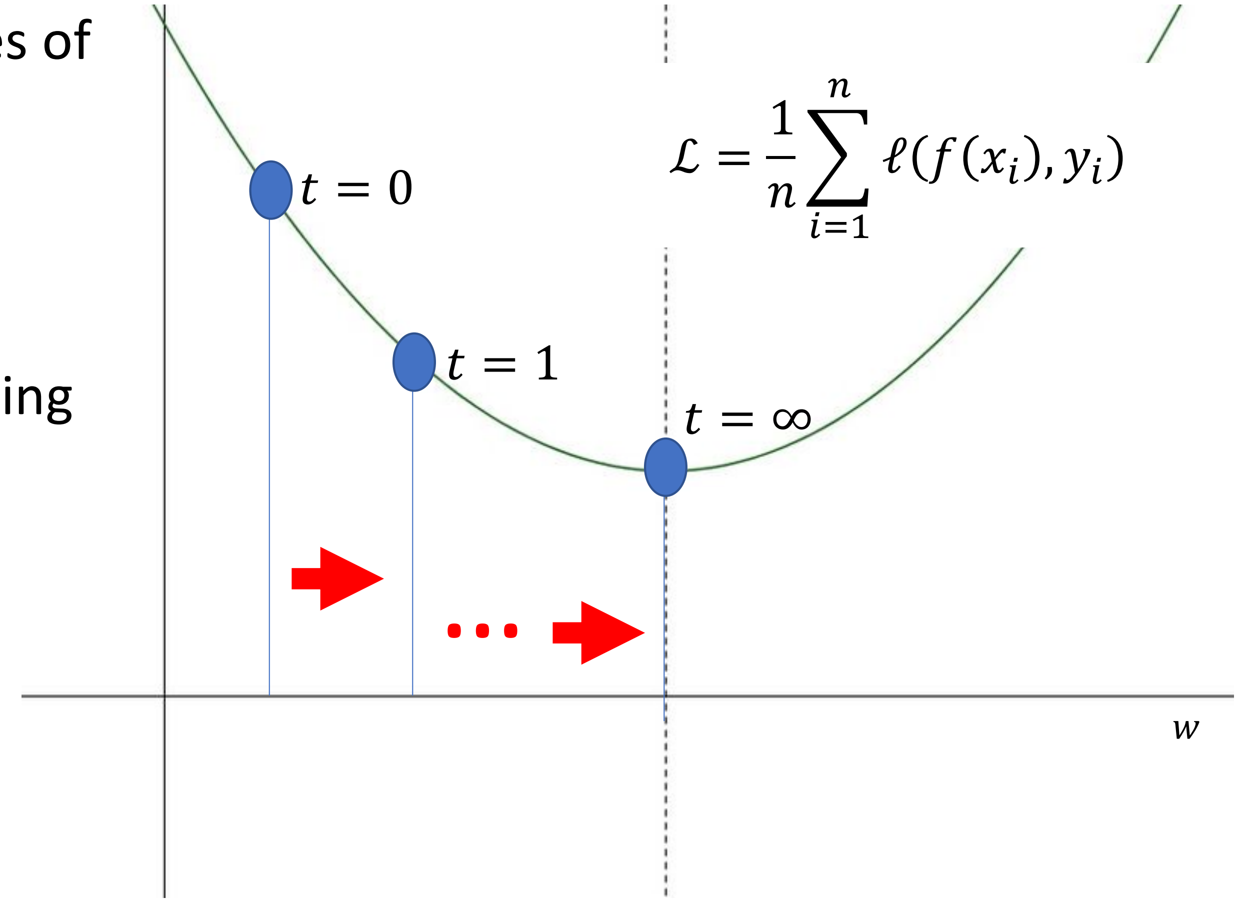
- The three types of machine learning error
- The optimization problem
- Local optimization with a known gradient
    - Gradient descent
    - Newton’s method
    - Line search
- Local optimization with unknown gradient
    - Empirical gradients
    - Coordinate search
- Global optimization
    - Random restarts
    - Simulated annealing

Review: Gradient descent

- Start from random initial values of w.
- Adjust w according to:

$$w \leftarrow w - \eta \nabla_w \mathcal{L}$$

- Continue until $\mathcal{L}$ stops decreasing



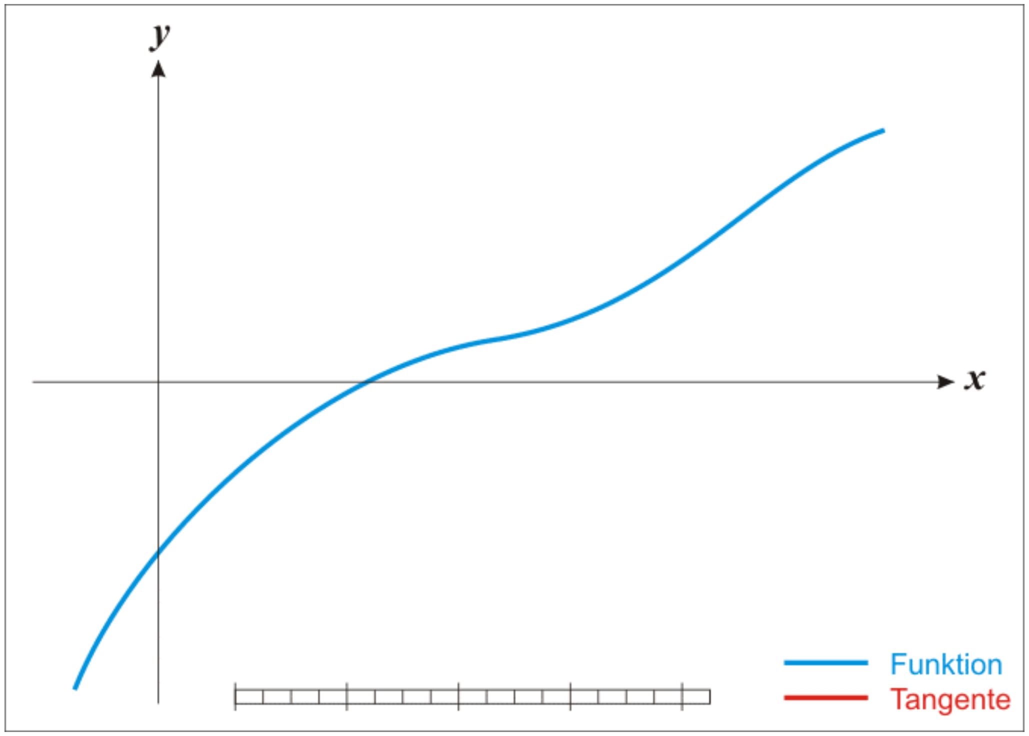
Problems with gradient descent, and a program for fixing them during today's lecture

1. If we choose η too large, the sequence of w's might diverge. If we choose η too small, the sequence of w's might take a very long time to converge.
   - Solutions: Newton's method, quasi-second-order methods, line search
2. It might not be possible to compute $\nabla_w \mathcal{L}$
   - Solutions: empirical gradients, coordinate search
3. Gradient descent finds a local optimum, not a global optimum
   - Solutions: simulated annealing, random restarts

Newton’s method

Newton proposed an iterative method to find the zeros of a function:

$$w \leftarrow w - \frac{f(w)}{f'(w)}$$



Newton's method for machine learning: scalar w

- In machine learning, we're trying to find minima of $\mathcal{L}(w)$, which are zero-crossings of $\mathcal{L}'(w) = \nabla_w \mathcal{L}$, so Newton's method becomes

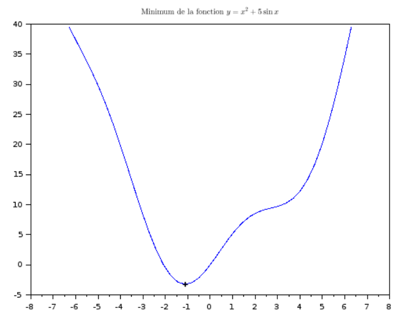
$$w \leftarrow w - \frac{\mathcal{L}'}{\mathcal{L}''}$$

- Notice the similarity to Gradient descent: $w \leftarrow w - \eta \mathcal{L}'$. Newton's method is the same thing as setting the learning rate to $\eta = 1/\mathcal{L}''$.
- If $\mathcal{L}(w)$ is quadratic, Newton's method finds the minimum in one step:

$$\mathcal{L}(w) = a(w - \hat{w})^2 + \mathcal{L}_{min}$$

$$w \leftarrow w - \frac{\mathcal{L}'}{\mathcal{L}''} = w - \frac{a(w - \hat{w})}{a} = \hat{w}$$

- Near a local optimum, many loss functions are approximately quadratic, so Newton's method can often find an optimum in ≈ 1 step



CC-SA 3.0, Cdang, https://commons.wikimedia.org/wiki/File:Minimum_fonction_scilab.svg

Newton’s method for machine learning: vector w

- If $w=[w_0,\ldots,w_{M-1}]$, then $\mathcal{L}'(w)=\nabla_w\mathcal{L}=\left[\frac{\partial\mathcal{L}}{\partial w_0},\ldots,\frac{\partial\mathcal{L}}{\partial w_{M-1}}\right]$, and the second derivative is called the “Hessian:”

$$H=\begin{bmatrix}\frac{\partial^2\mathcal{L}}{\partial w_0^2} & \cdots & \frac{\partial^2\mathcal{L}}{\partial w_0\partial w_{M-1}}\\ \vdots & \ddots & \vdots\\ \frac{\partial^2\mathcal{L}}{\partial w_{M-1}\partial w_0} & \cdots & \frac{\partial^2\mathcal{L}}{\partial w_{M-1}^2}\end{bmatrix}$$

- Newton’s method becomes:

$$w\leftarrow w-\nabla_w\mathcal{L}@H^{-1}$$

… which has a computational complexity of $\mathcal{O}\{M^3\}$. If M is millions, then gradient descent ($\mathcal{O}\{M\}$) is reasonable, but Newton’s method is not.

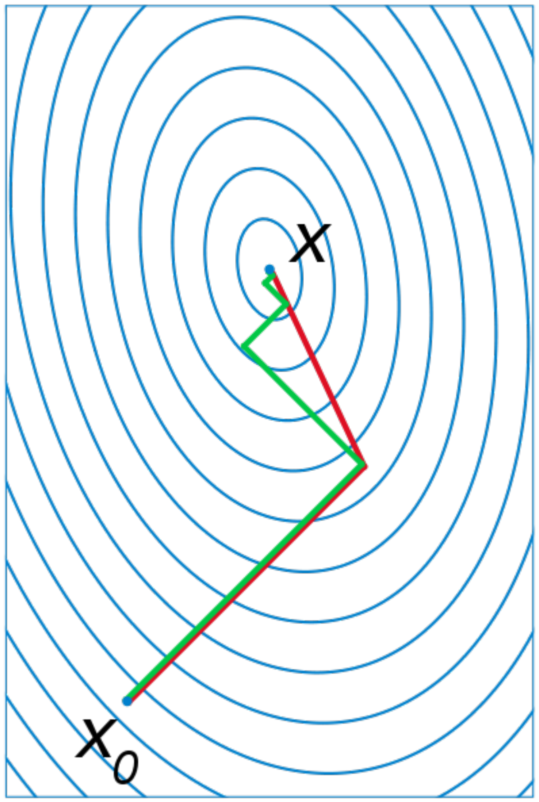
Approximate 2nd-order methods

The most successful optimization algorithms in machine learning, currently, are approximate 2nd-order methods. Basically, these are algorithms that approximate $\nabla_w\mathcal{L}@H^{-1}$ using an $\mathcal{O}\{M\}$ approximation. Some examples include:

- BFGS (Broyden-Fletcher-Goldfarb-Shanno) estimates the second derivative by computing first-differences of recent values of $\nabla_w\mathcal{L}$. LBFGS (limited-memory BGFS) limits the memory.
- Adam (Adaptive moment estimation) estimates both $\mathcal{L}'$ and $\mathcal{L}''$ for each weight, independently of all other weights, using a running average of recent values of $\nabla_w\mathcal{L}$ and $(\nabla_w\mathcal{L})^2$.

Line search

- A "line search" chooses a particular direction, d, and then finds an optimum scalar gain, g, that minimizes $\mathcal{L}(w + gd)$.
- Finding the best value of g requires testing many values, but not exponentially many. Using a golden-section search, you can often find a good g by testing only 6-10 values.
- If the directions are chosen well, this method can converge very quickly



Conjugate gradient descent uses a series of line searches with well-chosen directions. Public domain image, https://commons.wikimedia.org/wiki/File:Conjugate_gradient_illustration.svg

Outline

- The three types of machine learning error
- The optimization problem
- Local optimization with a known gradient
  - Gradient descent
  - Newton’s method
  - Line search
- Local optimization with unknown gradient
  - Empirical gradients
  - Coordinate search
- Global optimization
  - Random restarts
  - Simulated annealing

Empirical gradients

Suppose that $\mathcal{L}(w)$ is not differentiable. Can it be optimized?

The method of empirical gradients estimates the gradient as:

$$\frac{\partial \mathcal{L}}{\partial w_m} \approx \frac{1}{2\varepsilon}\big(\mathcal{L}(w + \varepsilon e_m) - \mathcal{L}(w - \varepsilon e_m)\big)$$

…where e_m is a one-hot vector with a 1 in its m^{th} element, and ε is some small number (a hyperparameter!)

This method requires doing 2M forward-props per update step, instead of only one forward-prop and one back-prop.

It works pretty well if ε is small enough.

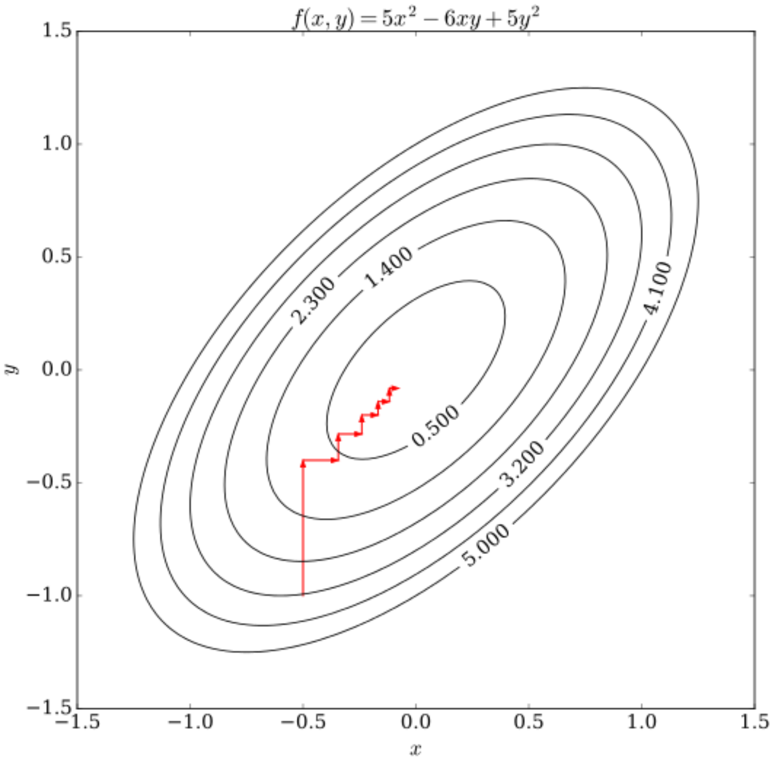
Coordinate search

The method of coordinate search skips over the gradient entirely. Instead, it finds:

$$\hat{m}, \hat{g} = \underset{m,g}{\operatorname{argmin}} \mathcal{L}(w + g e_m)$$

$$w \leftarrow w + \hat{g} e_{\hat{m}}$$

- Do a line-search on every coordinate direction, e_m, to find the value of that coordinate that minimizes the loss keeping all other coordinates unchanged.
- Update w to have the best value of that coordinate.
- If each line search has complexity K, then the total complexity is only $\mathcal{O}\{KM\}$.



CC-BY 4.0, https://commons.wikimedia.org/wiki/File:Coordinate_descent.svg

Quiz

Go to https://us.prairielearn.com/pl/course_instance/129874/assessment/2331863, and try the quiz!

quiz

$L=(w0-w1)^2 + w0^2 + w1^2$

$W=[9,1]$

X $dL/dw0 = 2(w0-w1) + 2w0=0$ -> best w0 is $w0=w1/2$ -> $[1/2,1]$

X $dL/dw1 = 2(w1-w0) + 2w1=0$ -> best w1 is $w1=w0/2$ -> $[9,9/2]$

$L([1/2,1]) = (1/2)^2 + (1/2)^2 + 1^2$

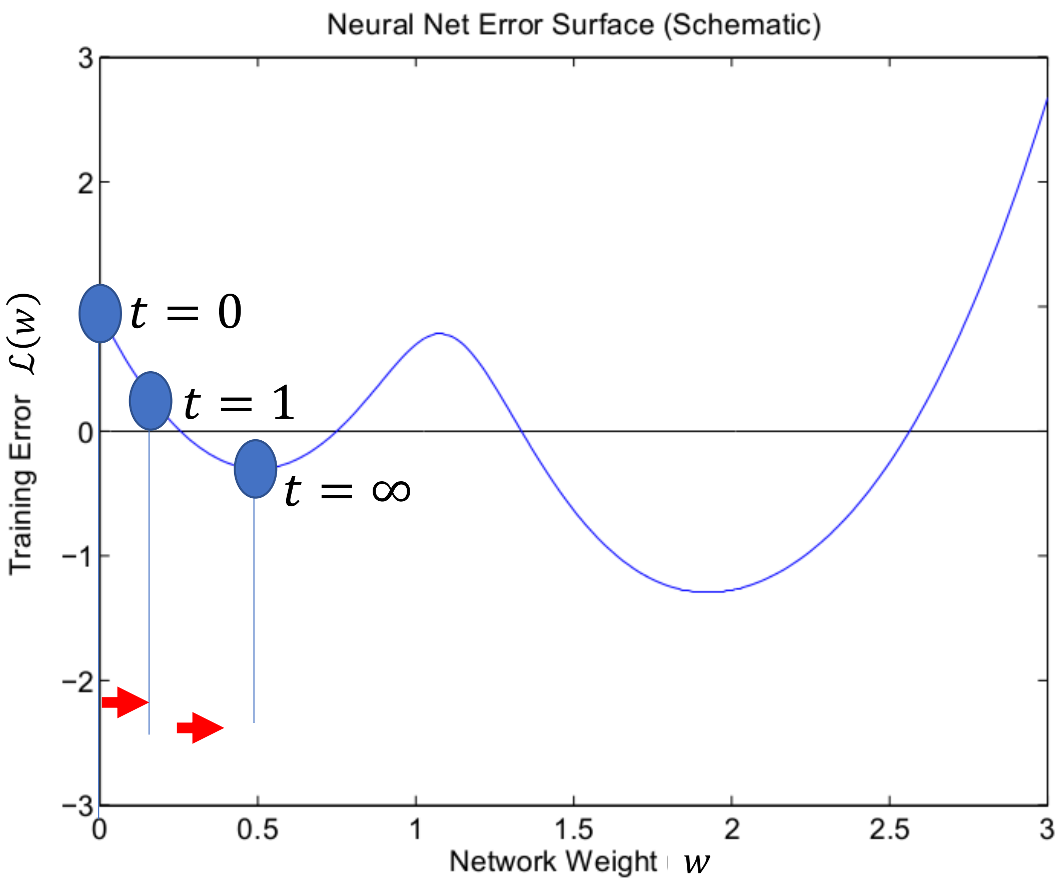
$L([9,9/2]) = (9/2)^2 + 9^2 + (9/2)^2$

Outline

- The three types of machine learning error
- The optimization problem
- Local optimization with a known gradient
  - Gradient descent
  - Newton's method
  - Line search
- Local optimization with unknown gradient
  - Empirical gradients
  - Coordinate search
- Global optimization
  - Random restarts
  - Simulated annealing

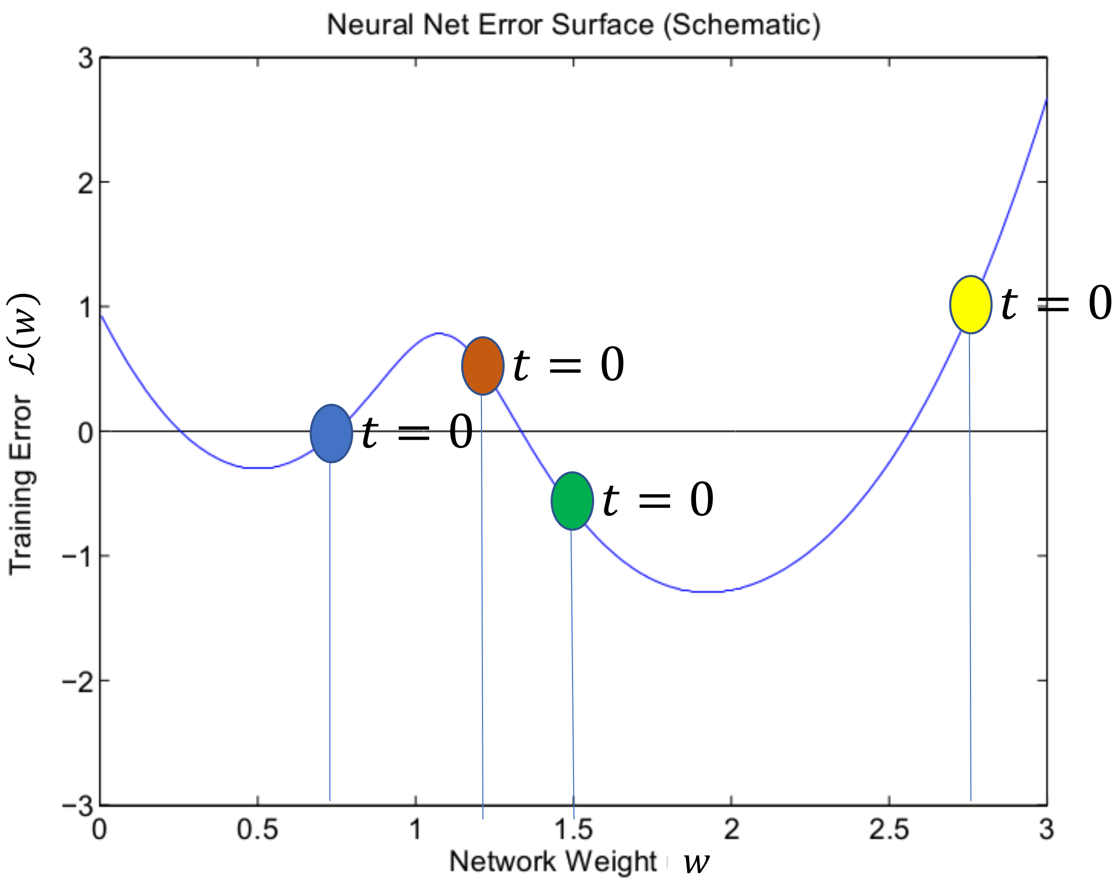
The problem with local optimization

- All of the methods we've seen so far are "local optimization" methods: starting from some initial vector w, they converge to the nearest local minimum of $\mathcal{L}(w)$
- **<u>Oops</u>**: The nearest local minimum of $\mathcal{L}(w)$ might not be very good



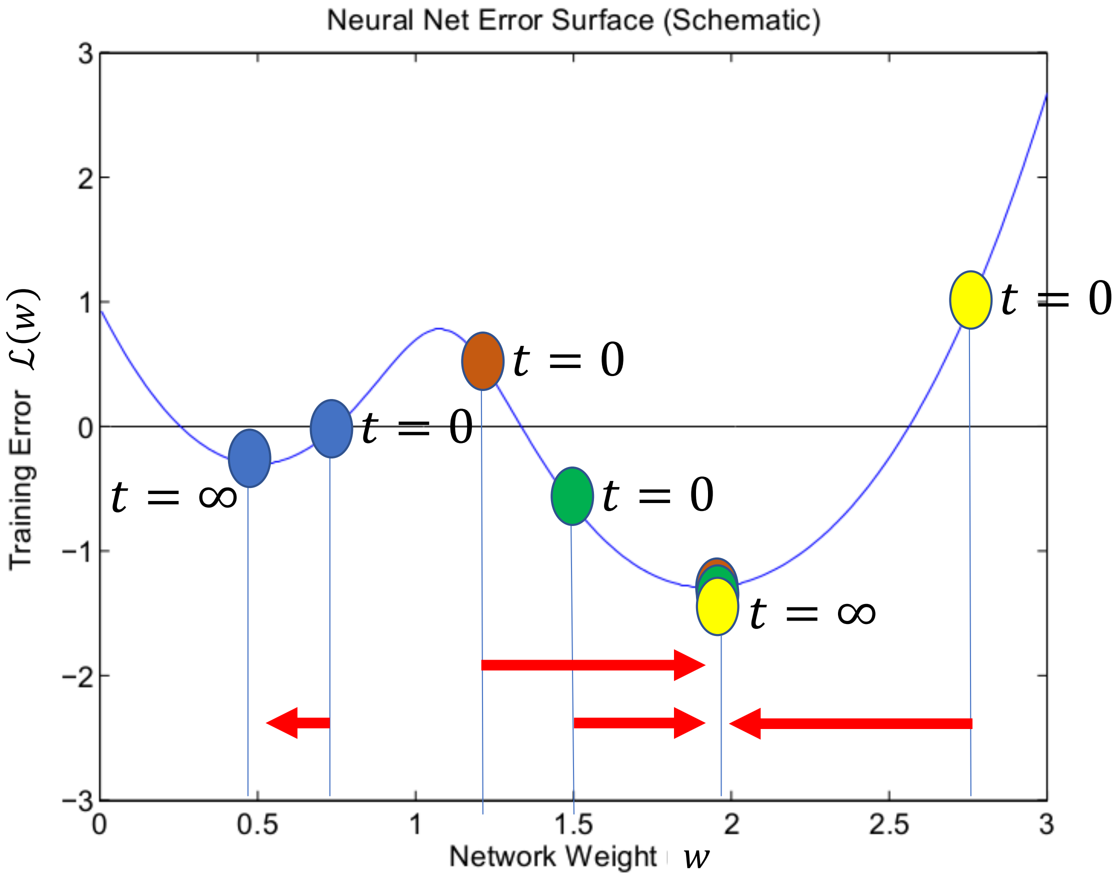
Random restarts

- The most common solution to the problem of bad local optima is the method of random restarts
- Basically, we just choose N different random starting locations (e.g., N might be 4, as shown, or maybe as large as 40)



Random restarts

- From each of those random starting locations, use gradient descent to find the local optimum.
- Choose the best local optimum, and call it the global optimum.



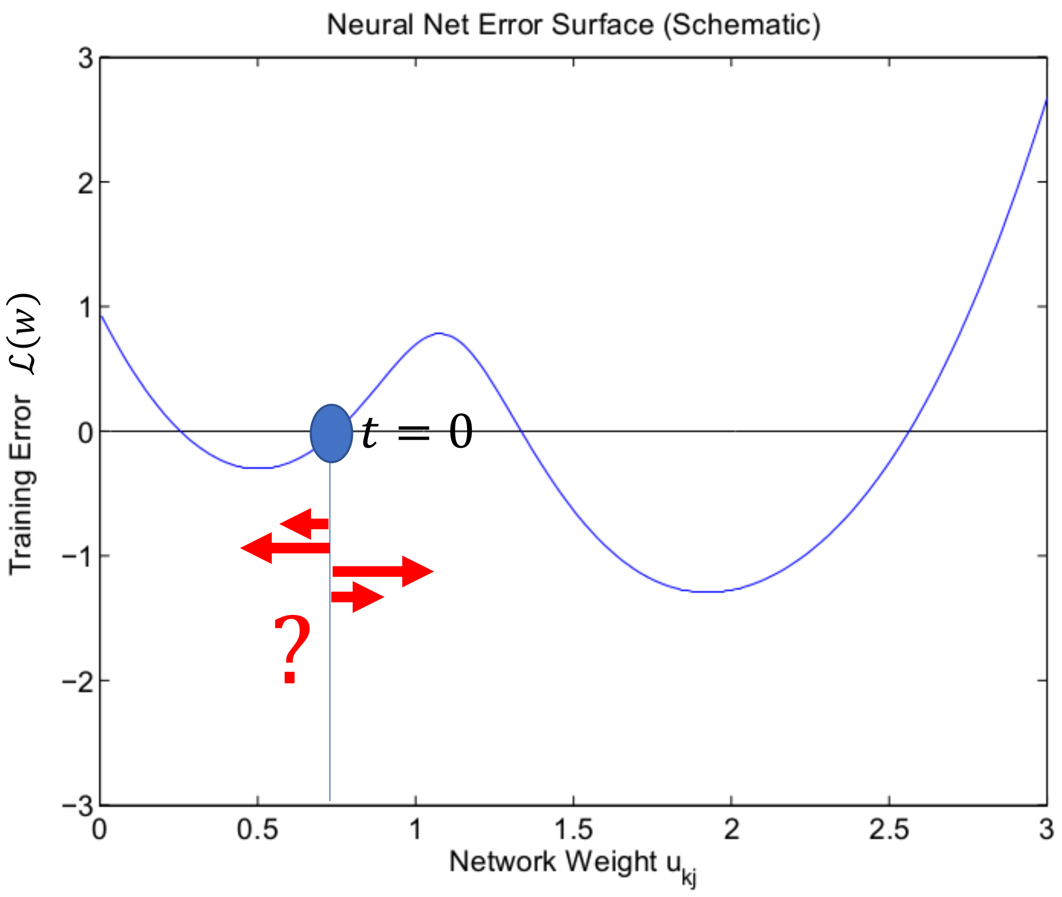
Advantages and disadvantages of the method of random restarts

- Advantage: Controllable computational complexity. Decide how many random restarts you can afford based on the amount of computation you have available.
- Disadvantage: Not provably optimal. There is no proof that this method finds the global optimum.

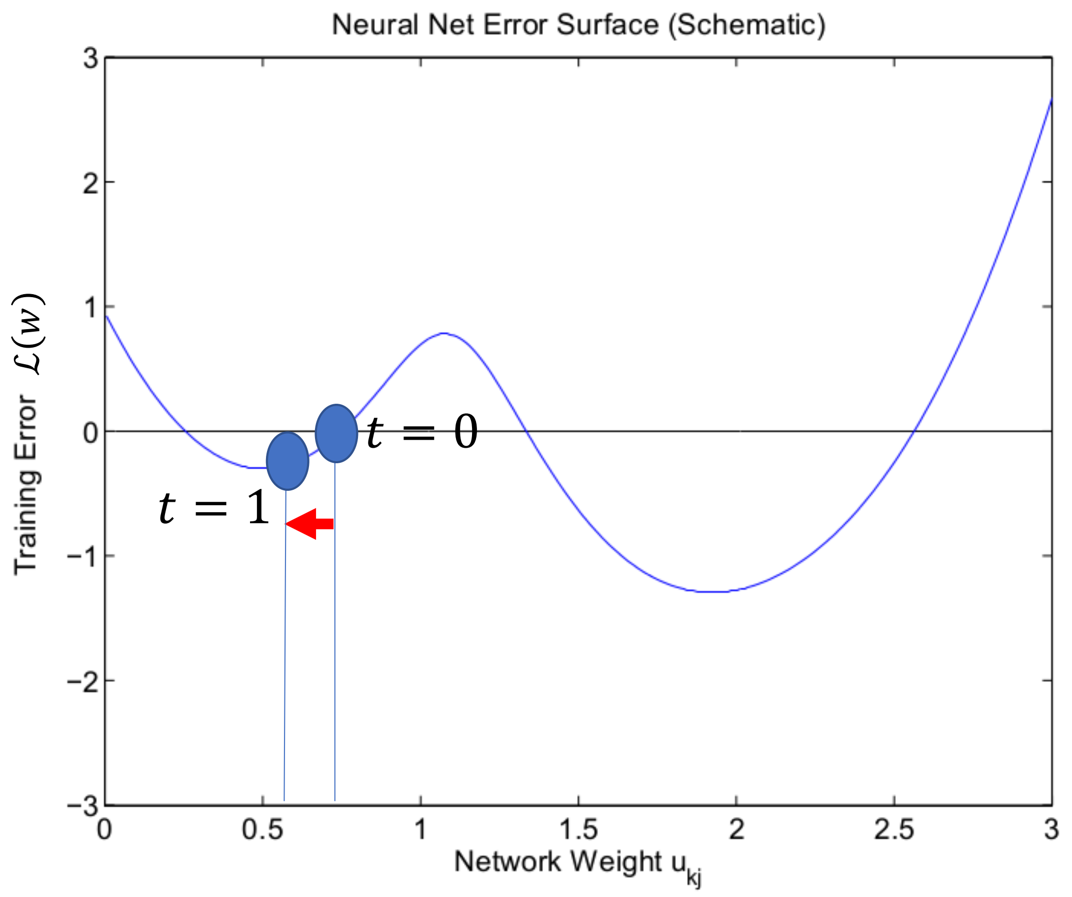
A provably optimal method: Simulated annealing

- Simulated annealing can be proven to find the global optimum.
- How it works:
    1. Instead of choosing $-\eta \nabla_w \mathcal{L}$ as our update step, simulated annealing chooses an update step, dw, at random
    2. If $\mathcal{L}(w + dw) \leq \mathcal{L}(w)$, then set $w \leftarrow w + dw$.
    3. If $\mathcal{L}(w + dw) > \mathcal{L}(w)$, then ***sometimes*** take the update step anyway, and sometimes don't. The probability of setting $w \leftarrow w + dw$ is
$$P(w \leftarrow w + dw) = \min\left(1, e^{\left(\mathcal{L}(w) - \mathcal{L}(w+dw)\right)/T}\right)$$
- At first, the "temperature," T, is large, so the step is usually taken. As time goes on, $T \rightarrow 0$, so bad steps become increasingly unlikely.

1. Choose an update step at random

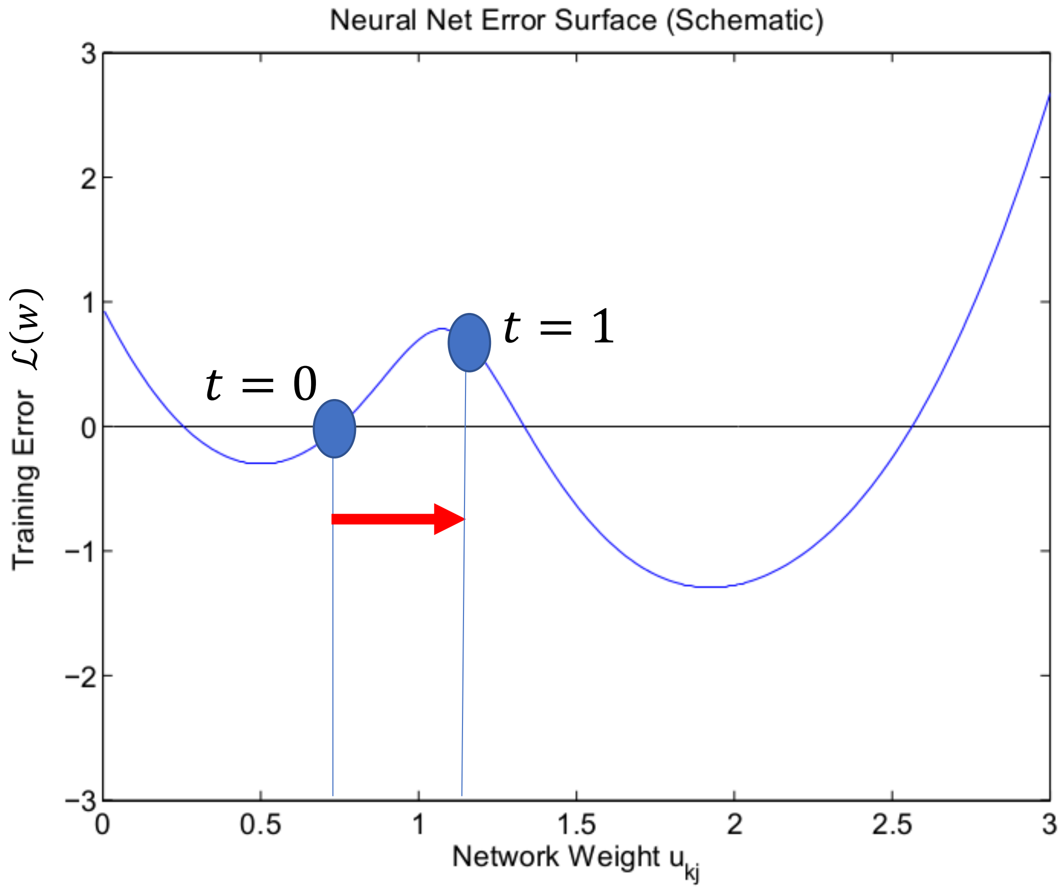


2. If the update improves the loss, then take it.



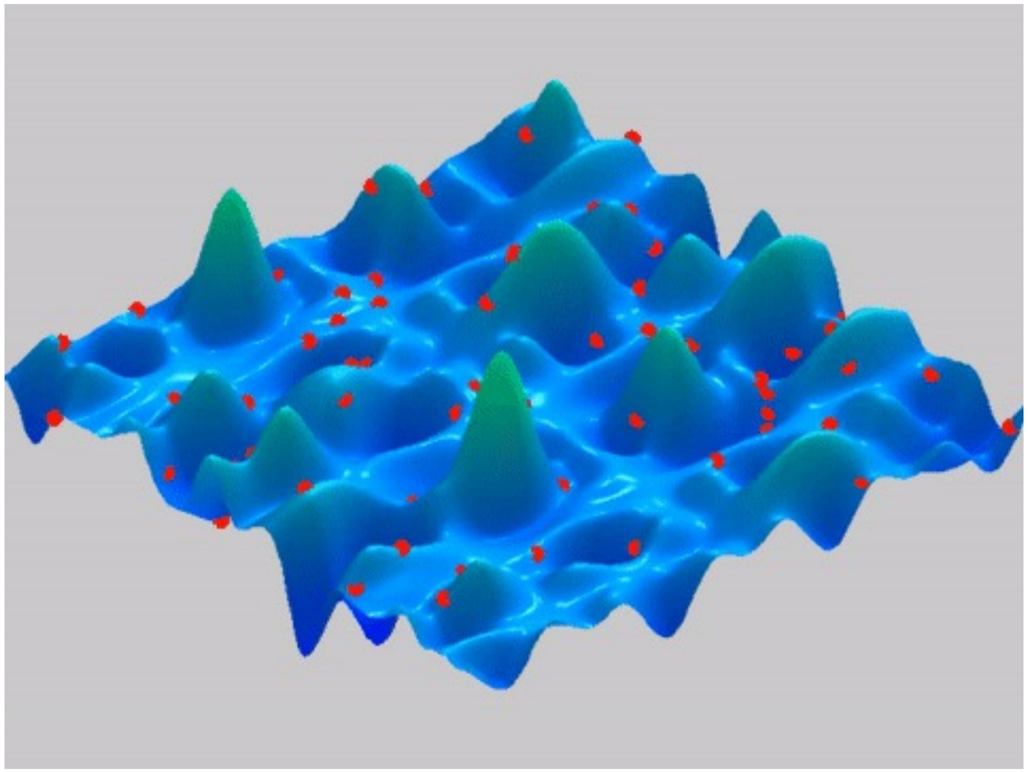
3. If the update makes the loss worse, then take it anyway, with a probability of

$$P = e^{\left(\mathcal{L}(w)-\mathcal{L}(w+dw)\right)/T}$$



Particle Swarm Optimization

- PSO combines random restarts with simulated annealing. Like simulated annealing, it is guaranteed to converge to a global optimum, if you run it long enough.
- Here is a video of PSO converging.



CC-SA 4.0, Hennegrolsch, https://commons.wikimedia.org/wiki/File:PSO_swarm_behaviour_optimized_for_presentation.gif

Outline

- The three types of machine learning error
- The optimization problem
- Local optimization with a known gradient
  - Gradient descent
  - Newton’s method
  - Line search
- Local optimization with unknown gradient
  - Empirical gradients
  - Coordinate search
- Global optimization
  - Random restarts
  - Simulated annealing