Class 5 - Sequential Regression

Continue the introduction of 1 new network/class

Sequential Regression

- Least Squares
- Logistic

Cost Functions

- Squared error
- Negative Log Likelihood

The homework will compare the behavior of sequential logistic regression using negative log likelihood (NLL) and squared error (SE) cost functions

Optimization

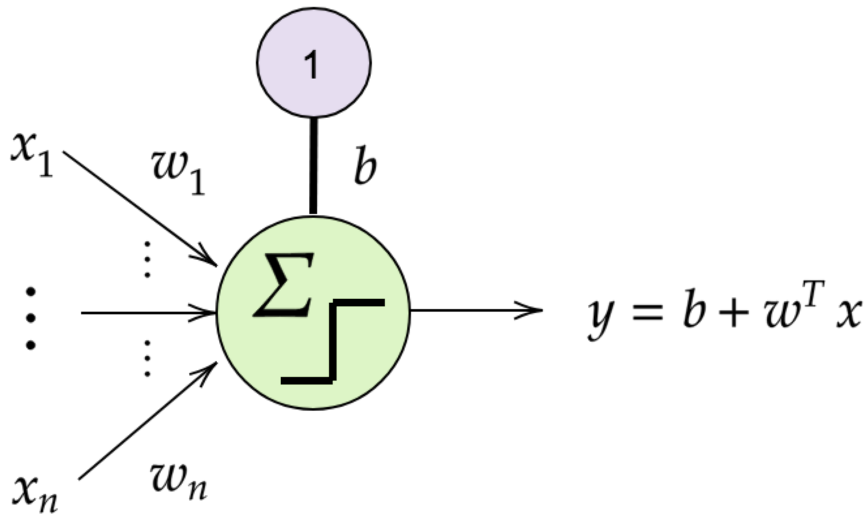
- Gradient descent
- Convexity

Perceptron

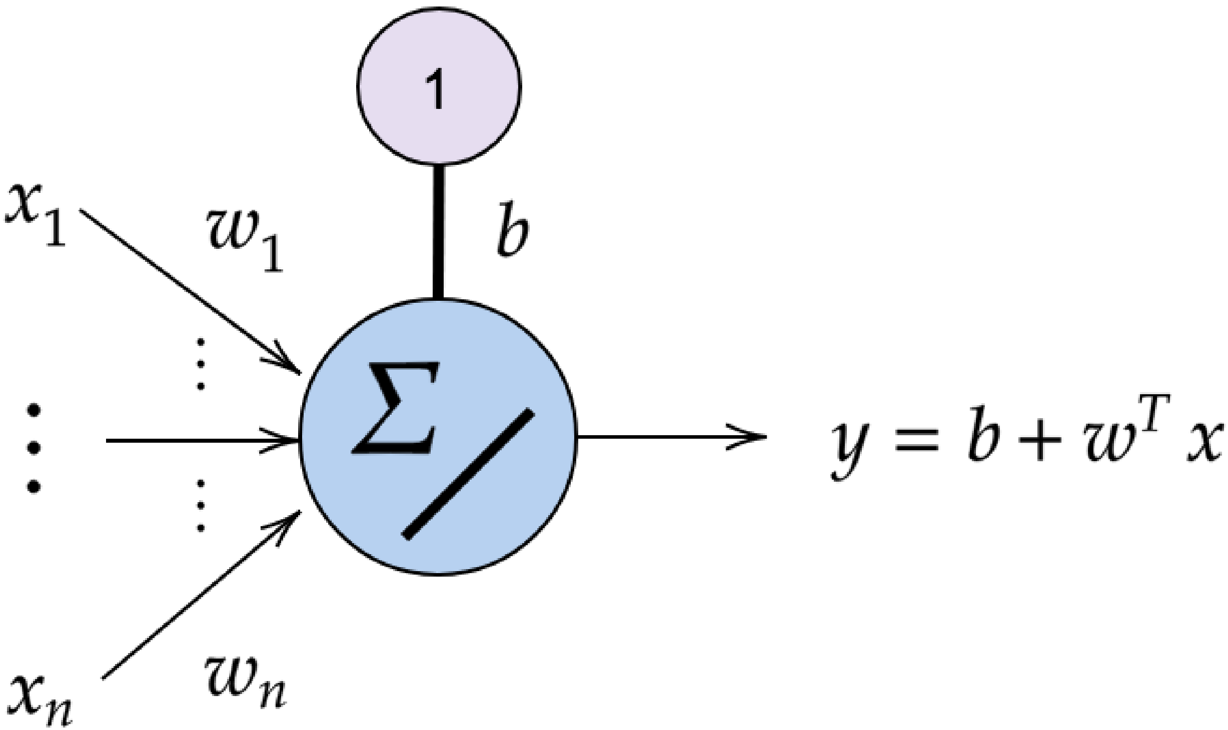
Last time we learned:

- Perceptron is trained using error correcting algorithm
- No gradient calculation
- Needs separable data

Today we'll look at 2 new networks



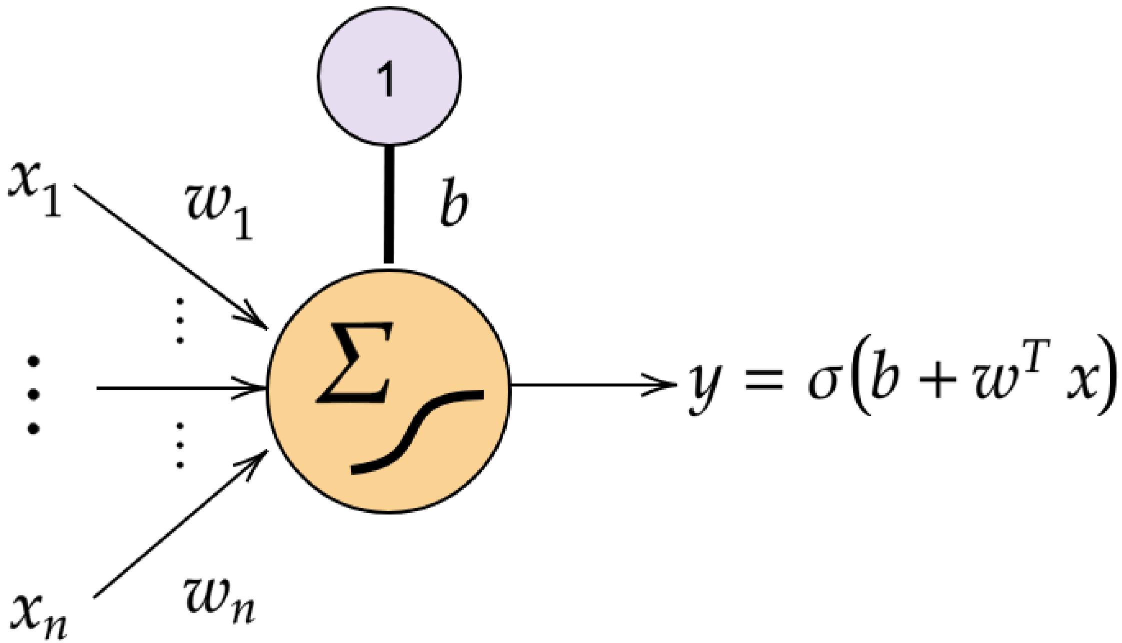
Least Squares Regression



$$y^{(i)} = b + \mathbf{w} \cdot \mathbf{x}^{(i)} \qquad \text{Explicit Bias}$$

$$\begin{aligned} y^{(i)} &= \mathbf{w}_0 + \mathbf{w} \cdot \mathbf{x}^{(i)} \qquad \text{Implicit Bias} \\ &= \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}}^{(i)} \end{aligned}$$

Logistic Regression



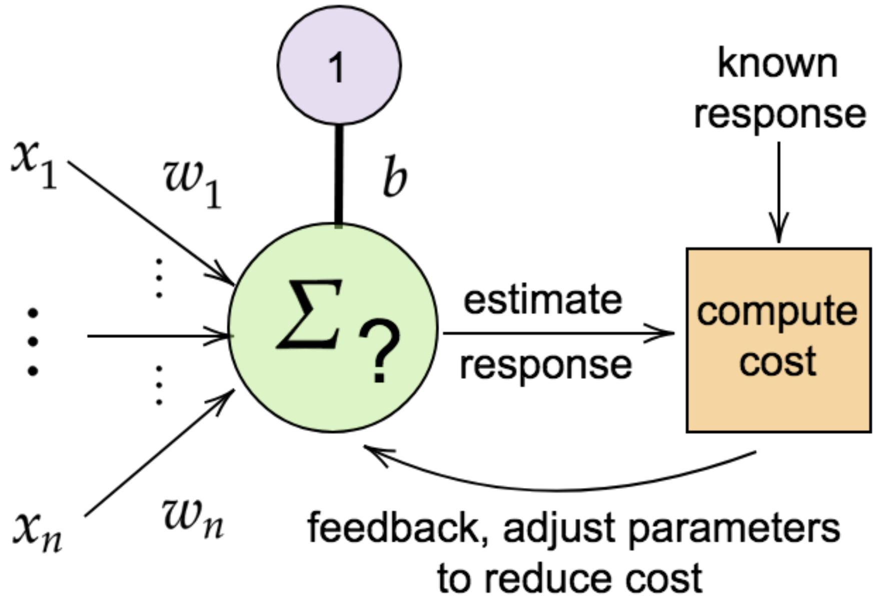
$\sigma(\mathbf{x})$ is the sigmoid (logistic) function. Commonly used for the output layer for binary classification

$\sigma(\mathbf{x}) \in [0, 1]$ will be interpreted as a probability

Adjustable Parameters

Generic model with adjustable parameters

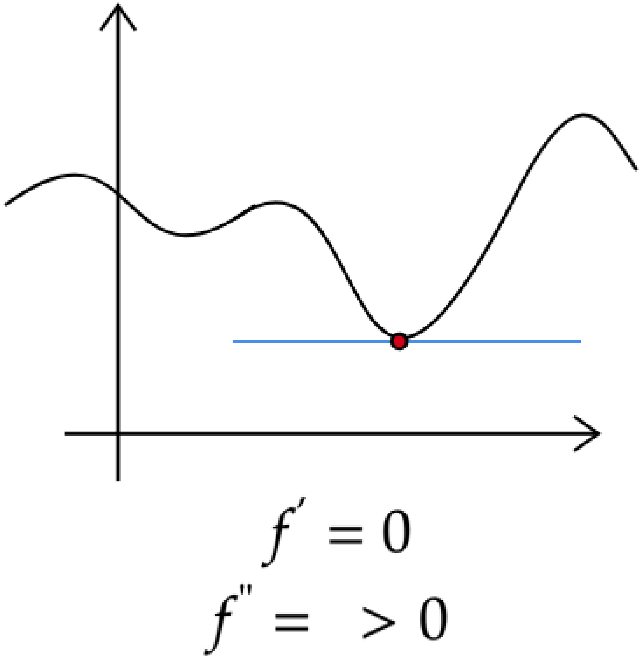
- The final output of the process is the model parameters
- Neural networks can have millions of parameters



Optimization in 1-dimension

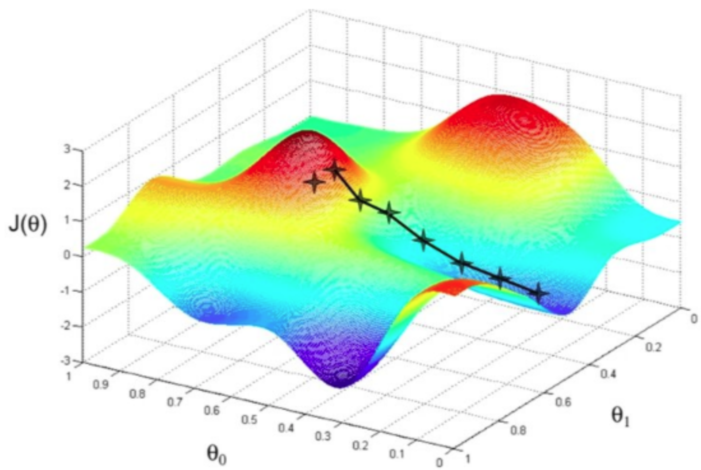
From 1^d calculus

- If unconstrained, minimum satisfies $f'(x) = 0$ and $f''(\mathbf{x}) > 0$
- Points $\mathbf{x}$ where $f'(\mathbf{x}) = 0$ are called stationary or critical points
- If constrained a minimum could also be on a constraint surface, like a boundary where $f' \neq 0$
- Networks have many parameters, so the cost function is minimized in a very high dimensional space.

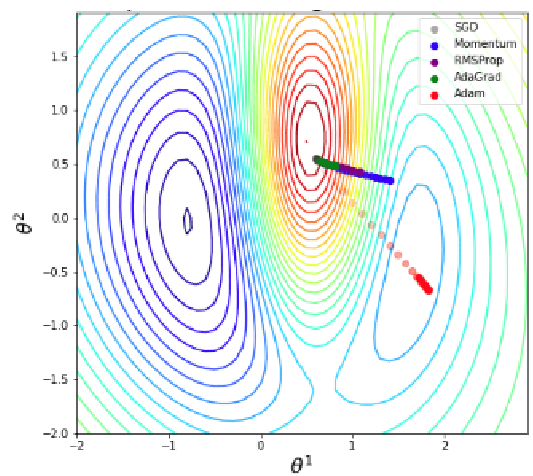


Geometry of Gradient

A function of $f : \mathbb{R}^n \to \mathbb{R}$ (scalar valued) defines a hypersurface in $\mathbb{R}^n$



Hypersurface defined by $f(\mathbf{x})$



Level curves of f define a topographical map

Geometry of Gradient

Directional Derivative

The directional derivative of $f : \mathbb{R}^n \to \mathbb{R}$ in the direction of a unit vector $\mathbf{u}$ is defined as:

$$\begin{aligned}
\nabla_u f = \partial_u f &= \lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{u}) - f(\mathbf{x})}{h} \\
&= \lim_{h \to 0} \frac{f(\mathbf{x}_1 + h\mathbf{u}_1, \cdots, \mathbf{x}_n + h\mathbf{u}_n) - f(\mathbf{x})}{h}
\end{aligned}$$

Gradient Vector

The gradient of f is a collection of directional derivatives. The components are equal to the directional derivatives in the coordinate basis directions:

$$\frac{\partial f}{\partial \mathbf{x}_i} = \lim_{h \to 0} \frac{f(\mathbf{x} + h\hat{e}_i) - f(\mathbf{x})}{h}$$

Geometry of Gradient

Will show that derivatives in a direction $\mathbf{u}$ can be expressed in terms of the gradient

$$\nabla_{\mathbf{u}} f = \mathbf{u} \cdot \nabla_{\mathbf{x}} f$$

Define $g(h) = f(\mathbf{x} + h\mathbf{u})$ for scalar h and unit vector $\mathbf{u}$

$$\begin{aligned} g^{'}\Big|_0 &= \lim_{h \to 0} \frac{g(h) - g(0)}{h} \\ &= \lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{u}) - f(\mathbf{x})}{h} \\ &\triangleq \partial_{\mathbf{u}} f \triangleq \nabla_{\mathbf{u}} f \end{aligned}$$

Now use chain rule on g to set:

$$\begin{aligned} g^{'}\Big|_h &= (\frac{\partial f}{\partial \mathbf{x}_1}\mathbf{u}_1 + \cdots + \frac{\partial f}{\partial \mathbf{x}_n}\mathbf{u}_n)\Big|_{\mathbf{x}+h\mathbf{u}} \\ g^{'}\Big|_0 &= \mathbf{u} \cdot \nabla_f\Big|_{\mathbf{x}} \end{aligned}$$

$$\nabla_{\mathbf{u}} f = \mathbf{u} \cdot \nabla f$$

Geometry of Gradient

The rate of change of the function f in the direction $\mathbf{u}$ is the scalar value $\nabla_{\mathbf{u}} f$

Want to show that the maximum value of $\|\nabla_{\mathbf{u}} f\|$ is in the direction of $\nabla_{\mathbf{x}} f$

$$\begin{aligned} \nabla_{\mathbf{u}} f &= \mathbf{u} \cdot \nabla_{\mathbf{x}} f \\ |\nabla_{\mathbf{u}} f| &= \|\mathbf{u}\| \|\nabla f\| \cos \theta \\ &= \|\nabla f\| \cos \theta \qquad \text{because} \|\mathbf{u}\| = 1 \end{aligned}$$

Where θ is the angle between $\mathbf{u}$ and the gradient vector. It follows that $\|\nabla_{\mathbf{u}} f\|$ is maximized when $\cos(\theta) = 1$

$$\max_{\|\mathbf{u}\|=1} \|\nabla_{\mathbf{u}} f\| = \|\nabla_x f\|$$

$$\mathbf{u} = \frac{\nabla f}{\|\nabla f\|}$$

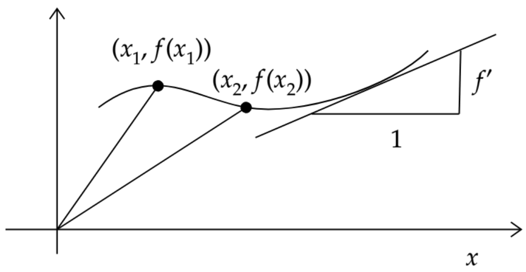
Gradient gives direction of most rapid increase.

Parametric Curves

Scalar valued functions are typically 'parameterized' by x. The graph of f can be written as:

$$(x, f(x))$$

$$\frac{d}{dx}(x, f(x)) = (1, f')$$



So the vector described by the derivative of the parameterized expression is tangent to the curve provided that f' is defined.

Parametric Curves

Example

The equation of a circle can be parametrized as $(r\cos(\theta), r\sin(\theta))$. Want to show that the derivative of this $(-r\sin(\theta), r\cos(\theta))$ is tangent.

$$y = \sqrt{r^2 - x^2}$$

$$y' = \frac{-x}{\sqrt{r^2 - x^2}}$$

Now use $x = r\cos(\theta), y = r\sin(\theta)$ to get

$$y' = -\frac{\cos\theta}{\sin(\theta)}$$

In general, if a curve in $\mathbb{R}^n$ is described by a the parametric equation $\mathbf{x}(t)$, then $\mathbf{x}'$ is tangent to $\mathbf{x}(t)$

Geometry of Gradient

Level Curves

A curve C in $\mathbb{R}^n$ can be parametrized as $(\mathbf{x_1}(t), \cdots, \mathbf{x_n}(t))$. The value of the function f along this curve is given by $f(\mathbf{x}(t))$ if the curve is a level curve of f then

$$f(\mathbf{x}(t)) = c$$

$$\frac{df}{dt} = \frac{\partial f}{\partial \mathbf{x_1}} \frac{d\mathbf{x_1}}{dt} + \cdots + \frac{\partial f}{\partial \mathbf{x_n}} \frac{d\mathbf{x_n}}{dt} = \mathbf{x}^{'} \cdot \nabla f$$

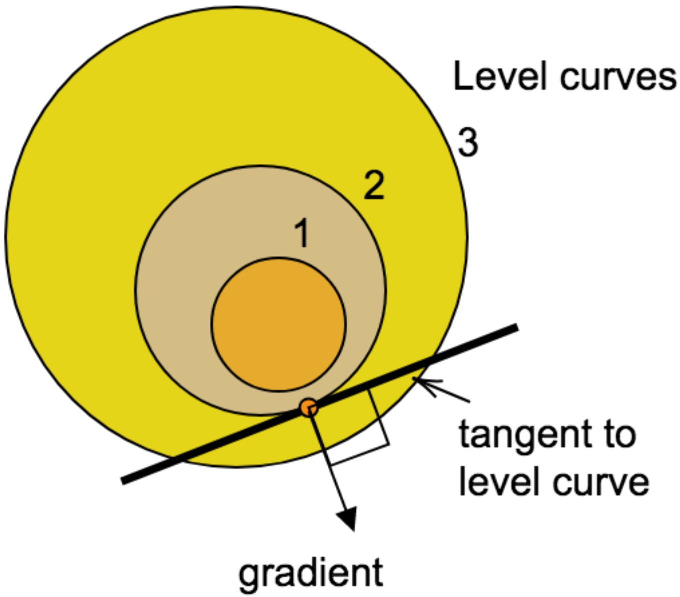
$$= \frac{d\,c}{dt} = 0$$

$\mathbf{x}^{'} \cdot \nabla f = 0$ shows that ∇f is $\perp$ to the tangent vector $(\mathbf{x}^{'})$ to the level curve.

∇f is the direction of most rapid increase

∇f is perpendicular to level curves

Geometry of Gradient



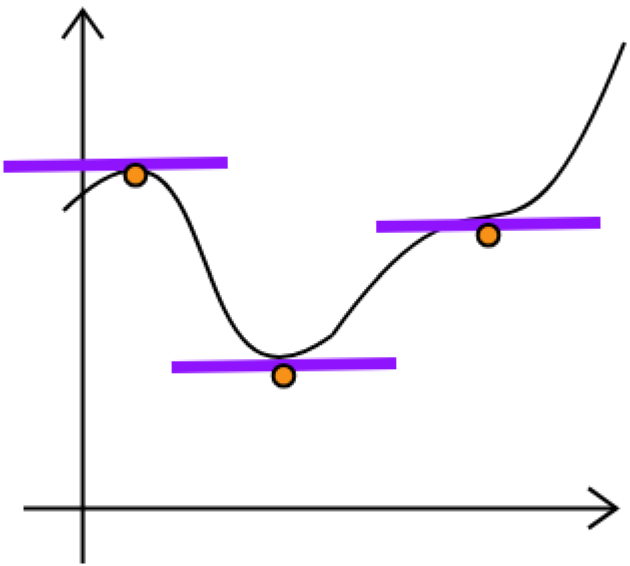
Gradient

- Points in direction of most rapid increase
- Is orthogonal to level curves

To move towards minimum, go in direction opposite to gradient.

Question: Is going down the steepest slope always the best choice?

Optimization



From calculus

In 1^d stationary, or critical points, are defined as points c where $f'(c) = 0$

A critical point is:

- A (local or global) minimum if $f'' > 0$
- A (local or global) maximum if $f'' < 0$
- An inflection or saddle point if $f'' = 0$

One-term Taylor approximation:

$$f(\mathbf{x} + \epsilon) \approx f(\mathbf{x}) + \epsilon f^{'}(\mathbf{x})$$

It follows that a smaller value of f can be found if $\mathrm{sign}(\epsilon) = -\,\mathrm{sign}(f')$ and $f' \neq 0$

Optimization

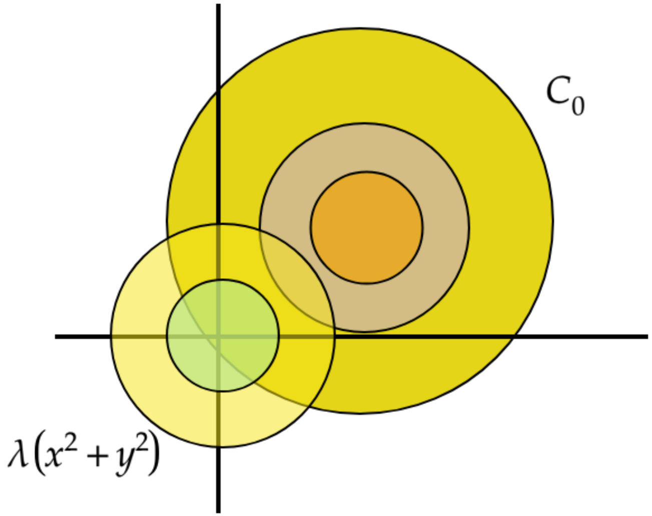
With each type of network, a scalar cost (objective) function C is defined. The goal will be to optimize cost by minimization.

- Neural Networks minimize a cost function. Minimization is just a convention.
    - Minimizing C is the same as maximizing the negative of C.
- Minimization will be over a set of adjustable parameters.
    - C (**params**; data)
- When a cost function is linear in the adjustable **parameters**, can just solve f' = 0.

Gradient Descent

- For neural networks C will depend on nonlinear functions and have no explicit solution. Iterative solutions are needed.

Regularized Cost Function



For cost C, want to find point where
$\nabla C = (\partial_x, \partial_y)C = 0$

Consider cost function C:

$$C_0 = \frac{1}{2}\left[(x-1)^2 + (y-1)^2\right] \quad \text{base cost}$$

$$C = C_0 + \lambda(x^2 + y^2) \quad \text{cost plus penalty}$$

$$\nabla C = (x - 1 + 2\lambda x \,, y - 1 + 2\lambda y)$$

Assume the penalized minimum lies on the line $x = y = t$

$$C(t) = (t-1)^2 + 2\lambda t^2 = (2\lambda + 1)t^2 - 2t + 1$$

$$C' = 0 \rightarrow t^* = \frac{1}{(2\lambda + 1)}$$

$$\nabla C(t^*) = 0$$

- The positive sum of convex functions is convex. Intuitive?
- By symmetry, the solution lies on the line connecting the individual minima

Optimization

Multi-dimensional optimization

Neural network cost functions are defined over a high-dimensional parameter space. Critical points are defined using the gradient w.r.t the adjustable parameters.

Gradient over vector of parameters $\mathbf{w}$:

$$\nabla_{\mathbf{w}} C = (\frac{\partial C}{\partial \mathbf{w}_1}, \cdots, \frac{\partial C}{\partial \mathbf{w}_n})$$

Critical point

$$\nabla_{\mathbf{w}} C = \frac{\partial C}{\partial \mathbf{w}_i} = 0 \quad \forall i$$

In general, will minimize C over adjustable parameters b and $\mathbf{w}$

Optimization

Classification of critical points in higher dimensions

The Hessian is the matrix of second derivatives:

$$H = \nabla^2_{\mathbf{w}} C =$$

$$\begin{bmatrix} \frac{\partial^2 C}{\partial \mathbf{w}_1^2} & \cdots & \frac{\partial^2 C}{\partial \mathbf{w}_1 \partial \mathbf{w}_n} \\ \vdots & & \vdots \\ \frac{\partial^2 C}{\partial \mathbf{w}_n \partial \mathbf{w}_1} & \cdots & \frac{\partial^2 C}{\partial \mathbf{w}_n^2} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial (\nabla_w C)}{\partial \mathbf{w}_1} \\ \vdots \\ \frac{\partial (\nabla_w C)}{\partial \mathbf{w}_n} \end{bmatrix}$$

- **Positive Definite** $\mathbf{x}^T H \mathbf{x} > 0$
  - Local minimum -- Positive eigenvalues
- **Negative Definite** $\mathbf{x}^T H \mathbf{x} < 0$
  - Local maximum -- Negative eigenvalues
- **Not Pos or Neg Definite**
  - Saddle point -- Both positive and negative eigenvalues

Optimization

Directional second derivative

$$\partial_{\mathbf{u}} f = \mathbf{u} \cdot \nabla \mathbf{f} = (\mathbf{u}_1 \partial_{x_1} + \ldots \mathbf{u}_n \partial_{x_n}) f$$
$$\partial^2_{\mathbf{u},\mathbf{u}} f = \partial_{\mathbf{u}} (\mathbf{u} \cdot \nabla f) = (\mathbf{u}_1 \partial_{x_1} + \ldots \mathbf{u}_n \partial_{x_n})(\mathbf{u}_1 \partial_{x_1} + \ldots \mathbf{u}_n \partial_{x_n}) f$$

Consider the i^{th} term

$$\mathbf{u}_i \partial_{\mathbf{x}_i} (\mathbf{u} \cdot \nabla f) = \mathbf{u}_i (\mathbf{u}_1 \partial^2_{\mathbf{x}_i . \mathbf{x}_1} f + \cdots + \mathbf{u}_n \partial^2_{\mathbf{x}_i, \mathbf{x}_n} f)$$

The term in parenthesis is the i-th row of the Hessian dotted with the direction vector $\mathbf{u}$. Since the total directional derivative is each of these terms weighted by $\mathbf{u}_i$ we get:

$$\partial^2_{\mathbf{u},\mathbf{u}} f = \mathbf{u}^T H \mathbf{u} \qquad \text{where } H \text{ is the Hessian of } f$$

This form connects the multidimensional case with the one-dimensional case

Note: If the cost function depends on n adjustable parameters then H is $n \times n$ dimensional. For neural networks n can be extremely large (> 1,000,000).

Optimization

Classification of critical points in n dimensions:

- **Minima**: H is positive definite

$$\mathbf{u}^T H \mathbf{u} > 0 \ \forall \mathbf{u} \neq 0$$

- **Maxima**: H is negative definite

$$\mathbf{u}^T H \mathbf{u} < 0 \ \forall \mathbf{u} \neq 0$$

- **Saddle point**: if H is neither positive definite or negative definite

Will cover positive definite matrices in future lecture. Useful facts:

- Positive definite matrices have positive eigenvalues
- Direction of maximum curvature will be the direction of the eigenvector of the largest eigenvalue

Optimization

Newton's method successively minimizes a quadratic approximation to f given by 2 terms in a Taylor Series expansion

$$f(\mathbf{x}) \approx f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T H(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k)$$

$$d\mathbf{x} = -H^{-1}(\mathbf{x}_k)(\nabla f(\mathbf{x}_k))^T$$

This is the multidimensional form of solving $y = \frac{1}{2}ax^2 + b\mathbf{x} + c$ for $y' = ax + b = 0$

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k + d\mathbf{x} \\ &= \mathbf{x}_k - H^{-1}(\mathbf{x}_k)(\nabla f(\mathbf{x}_k))^T \end{aligned}$$

Newton's method converges much faster than steepest descent

- typically impractical for neural networks
- The Hessian can be an extremely large matrix
- Solving $H^{-1}(\mathbf{x}_k)(\nabla f(\mathbf{x}_k))^T$ is computationally expensive

Convexity

A **global minimum** is the point $\mathbf{x}_0$ such that

$$Cost(\mathbf{x}) \geq Cost(\mathbf{x}_0)\ \forall \mathbf{x}$$

A **local minimum** is a point $\mathbf{x}_0$ such that there is an $\epsilon > 0$, such that

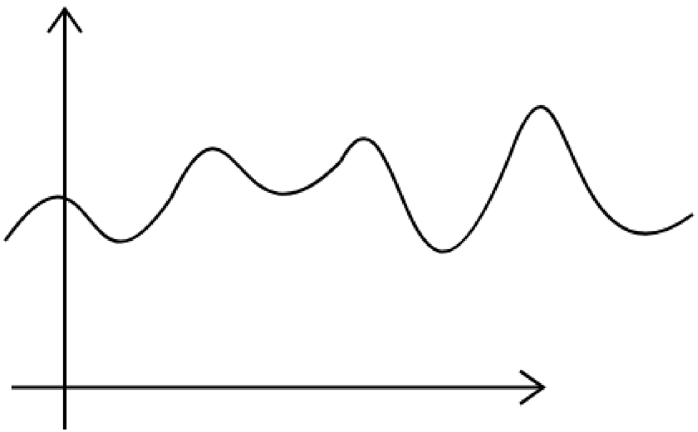
$$Cost(x) \geq Cost(x_0) \text{when } |x - x_0| < \epsilon$$

- Functions can have many local minima.

Convexity

- When the cost function is convex there are no local minima.
    - Any minimum is a global minimum.
- Will show later that
    - Linear regression with squared error cost is convex
    - Logistic regression with negative log likelihood cost is convex
- In general, Neural Network cost functions are not convex

Convexity



Lack of convexity for neural networks could be a deal breaker.

- In 1^d, finding global minimum of non-convex functions is NP-hard.
- For NN want to find the global minimum of the cost function in a high-dimensional parameter space
    - seems hopeless

Optimization

It is not yet clear why neural networks work as well as they do.

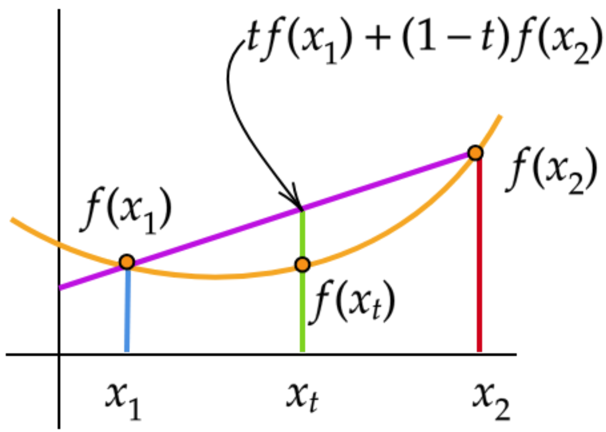
A plausibility argument:

- Gradient descent gets stuck at local minima where $\nabla f = 0$
  - In 1-dimension, there's a 1 in 3 chance that a critical point is a local minima
  - In n dimensons there is only a 1 in 3^n chance that a critical point is a local minimum
  - In higher dimensions, it can be argued that almost all critical points are saddle points, so some component of the gradient will move the estimate away from the critical point.
  - Descent algorithms probably don't get stuck a saddle points
    - Convergence can be extremely slow

Convexity

A function $f : X \to \mathbb{R}^n$ is called convex if $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$, $t \in [0, 1]$

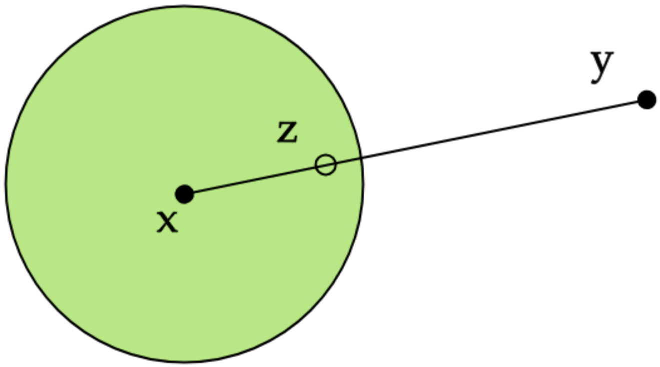
$$f(t\mathbf{x}_1 + (1-t)\mathbf{x}_2) \leq tf(\mathbf{x}_1) + (1-t)f(\mathbf{x}_2)$$



Local minima are global minima

if $\mathbf{x}$ is a local minimum then $f(\mathbf{z}) \geq f(\mathbf{x})$ for some neighborhood of $\mathbf{x}$. For any $\mathbf{y} \neq \mathbf{x}$ there is a $\lambda \in (0, 1)$ such that $\mathbf{z} = \lambda\mathbf{y} + (1-\lambda)\mathbf{x}$

$$\begin{aligned} f(\mathbf{z}) &= f(\lambda\mathbf{y} + (1-\lambda)\mathbf{x}) \\ &\leq (1-\lambda)f(\mathbf{x}) + \lambda f(\mathbf{y}) \\ &= f(\mathbf{x}) + \lambda(f(\mathbf{y}) - f(\mathbf{x})) \end{aligned}$$



this implies

$$\lambda(f(\mathbf{y}) - f(\mathbf{x})) \geq 0$$

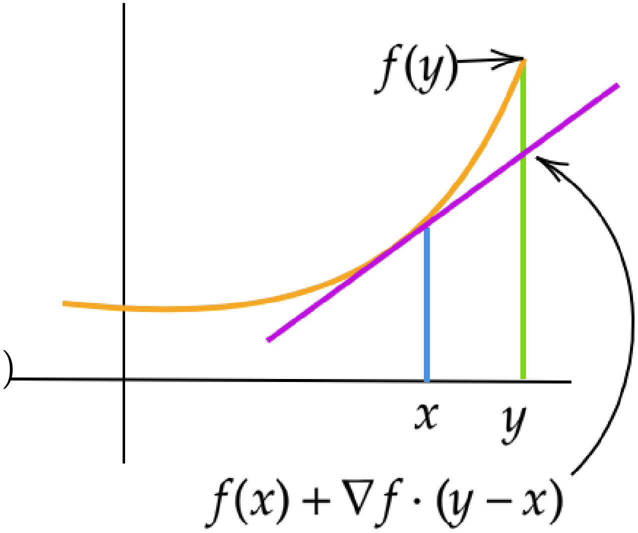
Convexity

Want to show that if f is convex and differentiable at $\mathbf{x}$ then

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot (\mathbf{y} - \mathbf{x})$$

Convexity condition

$$\begin{aligned} f(\alpha\mathbf{y} + (\mathbf{1} - \alpha)\mathbf{x}) &\leq \alpha f(\mathbf{y}) + (1 - \alpha) f(\mathbf{x}) \\ f(\mathbf{x} + \alpha(\mathbf{y} - \mathbf{x})) &\leq f(\mathbf{x}) + \alpha(f(\mathbf{y}) - f(\mathbf{x})) \\ \frac{f(\mathbf{x} + \alpha(\mathbf{y} - \mathbf{x})) - f(\mathbf{x})}{\alpha} &\leq f(\mathbf{y}) - f(\mathbf{x}) \end{aligned}$$



Example:
Left-hand side is derivative in direction $\mathbf{y} - \mathbf{x}$ or

$$\nabla f(\mathbf{x}) \cdot (\mathbf{y} - \mathbf{x})$$

Maximum Likelihood (MLE)

Maximum Likelihood is one way to estimate model parameters from data.

- A parametric model choice $p(\mathbf{x};\theta)$ is needed (Binomial, Gaussian, ...)
- An optimization process estimates the parameters θ given the data $\mathbf{x}$
- Has the intuitively appealing property that the estimated parameters maximize the likelihood of the observed data
- The assumption is that the data $\mathbf{x}^{(i)}$ are a representative sample from the distribution $p(\mathbf{x};\theta)$
- MLE often has other desirable properties, such as being unbiased.

Maximum Likelihood

Parametric method that adjusts the parameters of an assumed model to maximize likelihood of the data. Given a set of data $\mathbf{x}^{(i)}$, assume the samples are drawn independently from a probability density $p(\mathbf{x}; \theta)$ with parameters θ.

The $(\mathbf{x}^i)$ are independent, so their probability of occurring is

$$p(X; \theta) = \prod_{i=1}^{m} p(\mathbf{x}^{(i)}; \theta)$$

When considered as a function of θ instead of $\mathbf{x}$ it is called the likelihood function.

- Likelihood is not a distribution over θ

Estimation

- Adjust θ to make the observed $\mathbf{x}^{(i)}$ most likely.

Log likelihood:

- Log is monotonic so can take the logarithm and not affect the location of the maximum.

$$\begin{aligned} \mathcal{L}(\theta; X) &= \ln l(\theta; X) \\ &= \sum_{i=1}^{m} \ln p(\mathbf{x}^{(i)}; \theta) \end{aligned}$$

Minimization:

- The convention in neural networks is to minimize a cost function
  - So will maximize likelihood by minimizing negative log likelihood

Bernoulli

For $\mathbf{x}^{(i)} \in \{0, 1\}$

$$\begin{aligned} p^{(i)} = p(\mathbf{x}^{(i)}) &= p^{\mathbf{x}^i}(1-p)^{1-\mathbf{x}^i} \\ \mathcal{L}(p|\mathbf{x}) &= \ln \prod_{i=1}^{M} p^{\mathbf{x}^{(i)}}(1-p)^{1-\mathbf{x}^{(i)}} \\ &= \sum_{i=1}^{M} \left(\mathbf{x}^{(i)} \ln p^{(i)} + (1-\mathbf{x}^{(i)}) \ln(1-p^{(i)}) \right) \\ &= \sum_{i=1}^{M} \mathbf{x}^{(i)} \ln p + (M - \sum_{i=1}^{M} \mathbf{x}^{(i)}) \ln(1-p) \end{aligned}$$

$$\frac{d\mathcal{L}}{dp} = 0 \Rightarrow p = \frac{1}{M} \sum_{i=1}^{M} \mathbf{x}^{(i)}$$

Log Likelihood

- If $\mathbf{x}^{(i)} = 1$ then i^{th} term is maximized by making $p = 1$
- If $\mathbf{x}^{(i)} = 0$ then i^{th} term is maximized by making $p = 0$