DSC 190

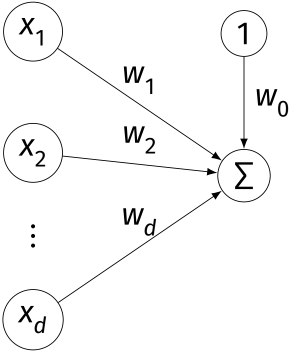
Machine Learning: Representations

Lecture 12 | Part 1

Neural Networks

Recall: Linear Predictor

- **Input**: features $\vec{x} = (x_1, \dots, x_d)^T$
- **Parameters**:
  $\vec{w} = (w_0, w_1, \dots, w_d)^T$
- **Output**: $w_0 + w_1 x_1 + \dots + w_d x_d$

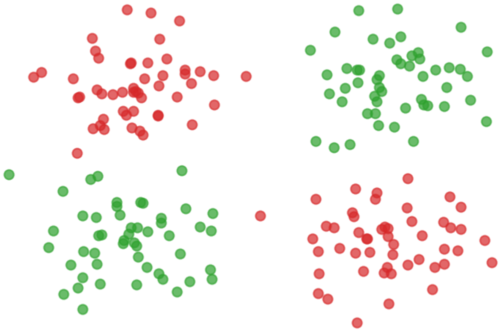


Linear Predictors

- **Pro**: simple, usually easy to optimize $\vec{w}$
  - With square loss, solution given by normal equations

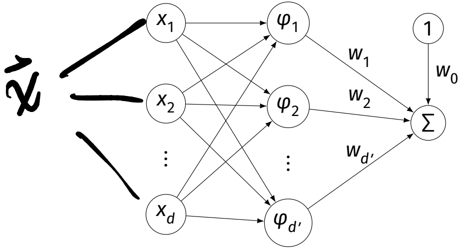
- **Con**: Decision boundary is linear

Example



Recall: Basis Functions

- **Input**: features $\vec{x}$, basis functions $\varphi_1, \ldots, \varphi_d : \mathbb{R}^d \to \mathbb{R}$
- **Parameters**: $\vec{w} = (w_0, w_1, \ldots, w_d)^T$
- **Output**: $w_0 + w_1\varphi_1(\vec{x}) + \ldots + w_d\varphi_d(\vec{x})$



Basis Functions

- **Note**: the basis functions and the weights $\vec{w}$ are **not** chosen at the same time
- Two step process
- First, basis functions are chosen and fixed
  - By hand, by k-means clustering, etc.
- *Then* the weights $\vec{w}$ are learned

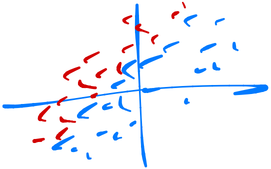
Exercise

Why do this in two steps as opposed to one?

Answer

- By fixing basis functions *then* finding best $\vec{w}$, optimization is easy again
- Using square loss, normal equations still work

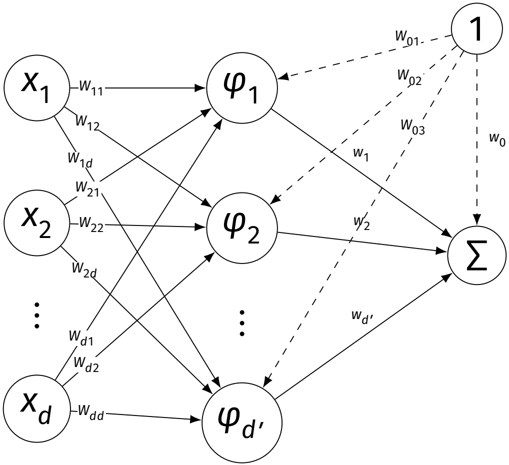
Idea



- Try to learn basis functions at same time as weights, $\vec{w}$
- Attempt #1: linear basis functions?

$$\varphi_i(\vec{x}) = W_{1i}x_1 + \ldots + W_{di}x_d$$

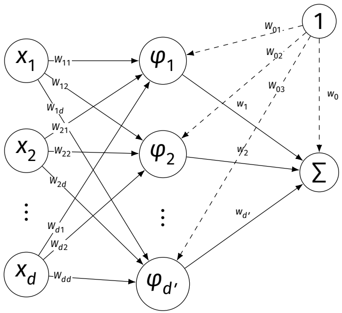
The Model



$$\varphi_i(\vec{x}) = W_{1i}x_1 + \dots + W_{di}x_d$$

Neural Network

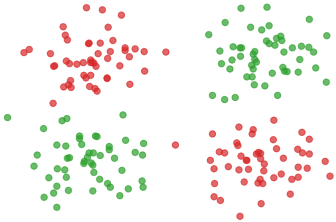
- **Input**: features $\vec{x}$,
- **Parameters**:
  $\vec{w} = (w_0, w_1, \ldots, w_d)^T$,
  $(d+1) \times d'$ matrix W
- **Output**:
  $w_0 + w_1\varphi_1(\vec{x}) + \ldots + w_d\varphi_d(\vec{x})$
- This is a **neural network**



$f \quad g \qquad h = f(g(\vec{x}))$

Problem

- If φ_i is linear, so is the decision boundary!

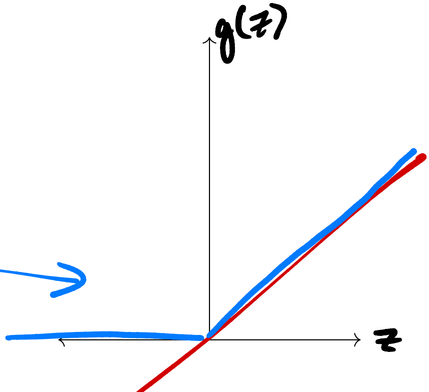


Activation Function

- To make φ_i nonlinear, we often apply a **activation function**.

- Very commonly: **rectified linear unit** (ReLU)

$$g(z) = \max\{0, z\}$$



$$\varphi_i(\vec{x}) = g(W_{0i} + W_{1i}x_1 + W_{2i}x_2 + \ldots + W_{di}x_d)$$
$$= \max\{0, W_{0i} + W_{1i}x_1 + W_{2i}x_2 + \ldots + W_{di}x_d\}$$

$f(x) = x^3 \cdot 3x^2$

Neural Networks as Functions

- A neural network is simply a special kind of **function**.

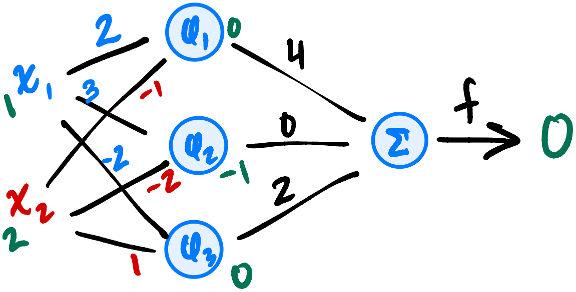
- $f(\vec{x}; \vec{w}, W)$

What is $f(\vec{x})$?

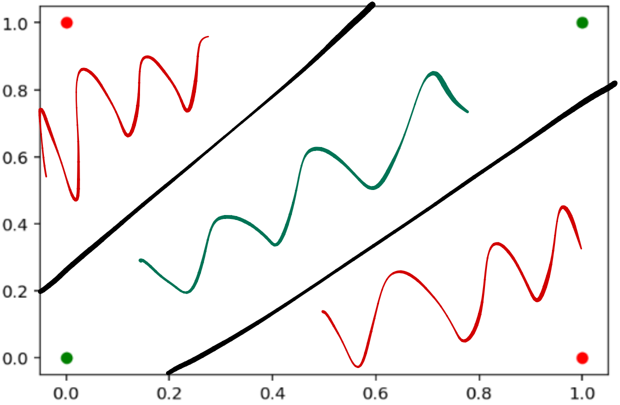
Example

$$\varphi_1(\vec{x}) = 2\times 1 + (-1)\times 2$$
$$= 0$$

$$W \quad \begin{array}{c|cc} & x_1 & x_2 \\ \hline \varphi_1 & 2 & -1 \\ \varphi_2 & 3 & -2 \\ \varphi_3 & -2 & 1 \end{array} \qquad \vec{w} = \begin{pmatrix} 4 \\ 0 \\ 2 \end{pmatrix} \qquad \vec{x} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$



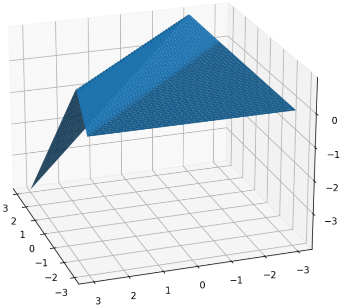
The Xor Problem



A Solution

$$W = \begin{pmatrix} 0 & -1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \qquad \vec{w} = \begin{pmatrix} 0 \\ 1 \\ -2 \end{pmatrix}$$

Prediction Surface



Learning with NNs

- We can **learn** weights by gathering data, picking a loss function and minimizing loss.

- The square loss works:

$$R(\vec{w}, W) = \frac{1}{n} \sum_{i=1}^{n} (f(\vec{x}^{(i)}; \vec{w}, W) - y_i)^2$$

Problem

- Now that the basis function weights are learnable, too, there is no simple solution for the best weights.
- We must instead use **gradient descent**.

DSC 190

Machine Learning: Representations

Lecture 12 | Part 2

Gradient Descent

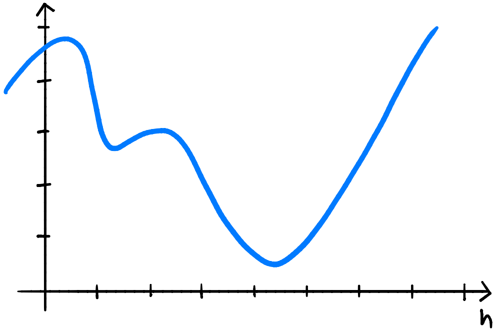
$f(x) = x^{23} - 3x^2 + 5$

Gradient Descent

- We have a function $f : \mathbb{R} \to \mathbb{R}$
- We can't solve for the x that minimizes (or maximizes) $f(x)$
- Instead, we use the derivative to "walk" towards the optimizer

Meaning of the Derivative

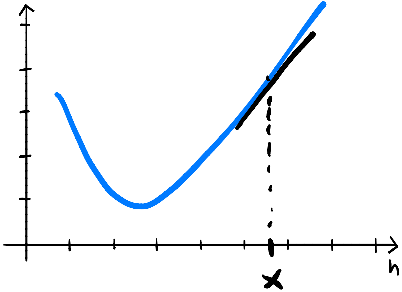
- We have the derivative; can we use it?
- $\frac{df}{dx}(x)$ is a function; it gives the **slope** at x.



Key Idea Behind Gradient Descent

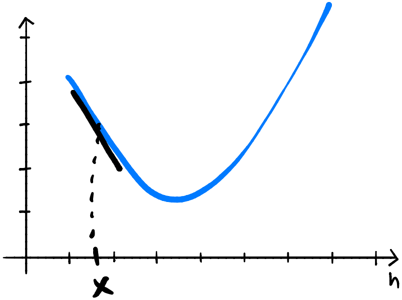
- If the slope of f at x is **positive** then moving to the **left** decreases the value of f.

- i.e., we should **decrease** x



Key Idea Behind Gradient Descent

- If the slope of f at x is **negative** then moving to the **right** decreases the value of f.
- i.e., we should **increase** x



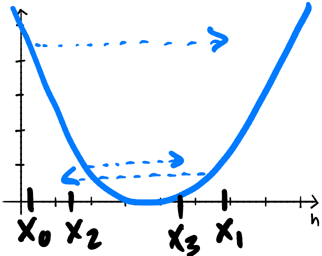
Key Idea Behind Gradient Descent

- Pick a starting place, x_0. Where do we go next?
- Slope at x_0 negative? Then increase x_0.
- Slope at x_0 positive? Then decrease x_0.
- This will work:

$$x_1 = x_0 - \frac{df}{dx}(x_0)$$

Gradient Descent

- Pick α to be a positive number. It is the **learning rate**.
- Pick a starting prediction, x_0.
- On step i, perform update $x_i = x_{i-1} - \alpha \cdot \frac{df}{dx}(x_{i-1})$
- Repeat until convergence (when x doesn't change much).



```python
def gradient_descent(derivative, x, alpha, tol=1e-12):
    """Minimize using gradient descent."""
    while True:
        x_next = x - alpha * derivative(x)
        if abs(x_next - x) < tol:
            break
        x = x_next
    return h
```

Example: Minimizing Mean Squared Error

- Recall the mean squared error and its derivative:

$$R_{sq}(x) = \frac{1}{n}\sum_{i=1}^{n}(x - y_i)^2 \qquad \frac{dR_{sq}}{dx}(x) = \frac{2}{n}\sum_{i=1}^{n}(x - y_i)$$

Exercise

Let $y_1 = -4, \quad y_2 = -2, \quad y_3 = 2, \quad y_4 = 4.$

Pick $x_0 = 4$ and $\alpha = 1/4$. What is x_1?

a) -1
b) 0
c) 1
d) 2

$$x_1 = x_0 - \alpha \frac{dR}{dx}(x_0) = 4 - \frac{1}{4} 8 = 4 - 2 = 2$$

$$\frac{dR}{dx}(4) = \frac{2}{n}\sum_{i=1}^{n}(4 - y_i) = \frac{2}{4}\big([4+4] + [4+2] + [4-2] + [4-4]\big) = \frac{1}{2}(8+6+2) = \frac{1}{2}16 = 8$$

Example

Gradient Descent in > 1 dimensions

- The derivative of f becomes the gradient:

$$\frac{df}{dx} \to \nabla f(\vec{x})$$

- Meaning of **differentiable**: locally, f looks linear.
- **Key**: $\nabla f(\vec{w})$ is a function; it returns a vector pointing in direction of steepest ascent.

Gradient Descent in > 1 dimensions

- Pick α to be a positive number.
    - It is the **learning rate**.
- Pick a starting guess, $\vec{w}^{(0)}$.
- On step i, update $\vec{w}^{(i)} = \vec{w}^{(i-1)} - \alpha \cdot \nabla f(\vec{w}^{(i-1)})$
- Repeat until convergence
    - when $\vec{w}$ doesn't change much
    - equivalently, when $\|\nabla f(\vec{w}^{(i)})\|$ is small

```python
def gradient_descent(gradient, w, alpha, tol=1e-12):
    """Minimize using gradient descent."""
    while True:
        w_next = w - alpha * gradient(x)
        if np.linalg.norm(w_next - w) < tol:
            break
        w = w_next
    return w
```

DSC 190

Machine Learning: Representations

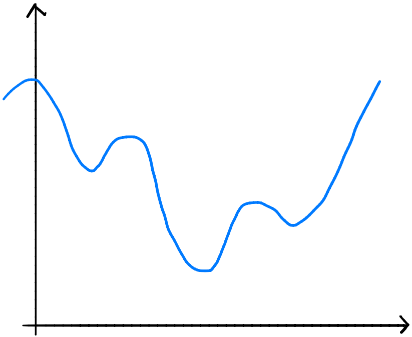
Lecture 12 | Part 3

Convexity in 1-d

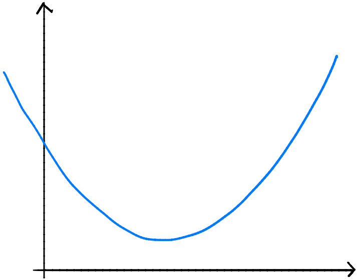
Question

When is gradient descent guaranteed to work?

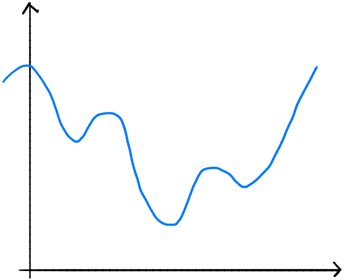
Not here...



Convex Functions



Convex



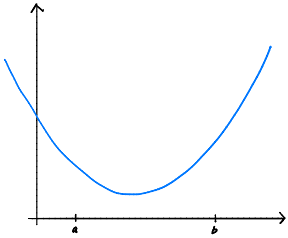
Non-convex

Convexity: Definition

- f is **convex** if for **every** a, b the line segment between

$$(a, f(a)) \qquad \text{and} \qquad (b, f(b))$$

does not go below the plot of f.

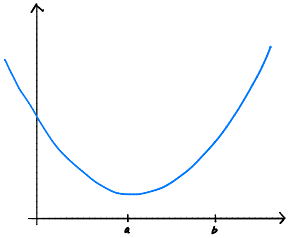


Convexity: Definition

- f is **convex** if for **every** a, b the line segment between

$$(a, f(a)) \qquad \text{and} \qquad (b, f(b))$$

does not go below the plot of f.

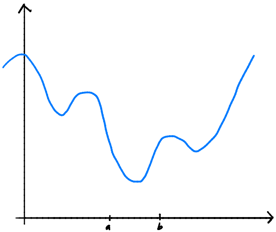


Convexity: Definition

- f is **convex** if for **every** a, b the line segment between

$$(a, f(a)) \qquad \text{and} \qquad (b, f(b))$$

does not go below the plot of f.

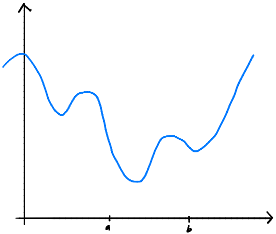


Convexity: Definition

- f is **convex** if for **every** a, b the line segment between

$$(a, f(a)) \qquad \text{and} \qquad (b, f(b))$$

does not go below the plot of f.



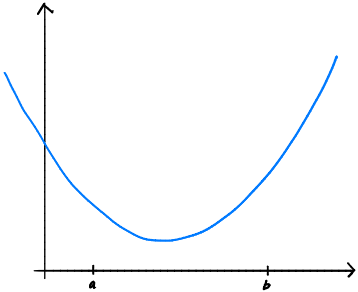
Other Terms

- If a function is not convex, it is **non-convex**.
- **Strictly convex**: the line lies strictly above curve.
- **Concave**: the line lines on or below curve.

Convexity: Formal Definition

- A function $f : \mathbb{R} \to \mathbb{R}$ is **convex** if for every choice of $a, b \in \mathbb{R}$ and $t \in [0, 1]$:

$$(1 - t)f(a) + tf(b) \geq f((1 - t)a + tb).$$

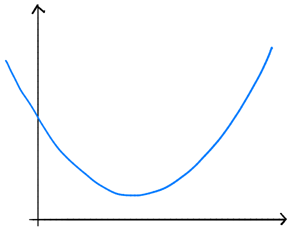


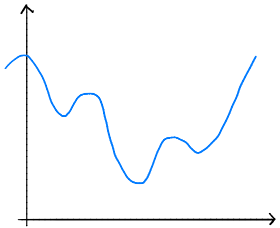
Example

Is $f(x) = |x|$ convex?

Another View: Second Derivatives

- If $\frac{d^2 f}{dx^2}(x) \geq 0$ for all x, then f is convex.
- Example: $f(x) = x^4$ is convex.
- **Warning!** Only works if f is twice differentiable!





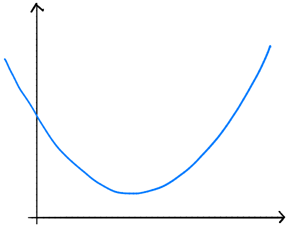
Another View: Second Derivatives

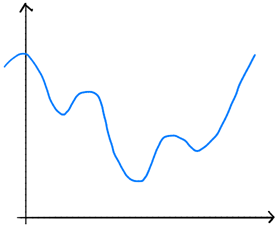
- "Best" straight line at x_0:
  - $h_1(z) = f'(x_0) \cdot z + b$

- "Best" parabola at x_0:
  - At x_0, f looks likes $h_2(z) = \frac{1}{2} f''(x_0) \cdot z^2 + f'(x_0) z + c$
  - Possibilities: upward-facing, downward-facing.

Convexity and Parabolas

- Convex if for **every** x_0, parabola is upward-facing.
  - That is, $f''(x_0) \geq 0$.





Convexity and Gradient Descent

- Convex functions are (relatively) easy to optimize.

- **Theorem**: if $R(x)$ is convex and differentiable[1][2] then gradient descent converges to a **global optimum** of R *provided* that the step size is small enough[3].

---

[1] and its derivative is not too wild

[2] actually, a modified GD works on non-differentiable functions

[3] step size related to steepness.

Nonconvexity and Gradient Descent

- Nonconvex functions are (relatively) hard to optimize.
- Gradient descent can still be useful.
- But not guaranteed to converge to a global minimum.

DSC 190

Machine Learning: Representations

Lecture 12 | Part 4

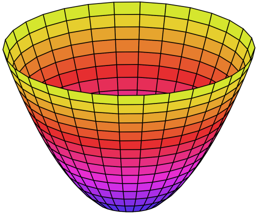
Convexity in Many Dimensions

Convexity: Definition

- $f(\vec{x})$ is **convex** if for **every** $\vec{a}, \vec{b}$ the line segment between

$$(\vec{a}, f(\vec{a})) \qquad \text{and} \qquad (\vec{b}, f(\vec{b}))$$

does not go below the plot of f.



Convexity: Formal Definition

- A function $f : \mathbb{R}^d \to \mathbb{R}$ is **convex** if for every choice of $\vec{a}, \vec{b} \in \mathbb{R}^d$ and $t \in [0, 1]$:

$$(1 - t)f(\vec{a}) + tf(\vec{b}) \geq f((1 - t)\vec{a} + t\vec{b}).$$

The Second Derivative Test

- For 1-d functions, convex if second derivative ≥ 0.
- For 2-d functions, convex if ???

The Hessian Matrix

- Create the **Hessian** matrix of second derivatives:

$$H(\vec{x}) = \begin{pmatrix} \frac{\partial f^2}{\partial x_1^2}(\vec{x}) & \frac{\partial f^2}{\partial x_1 x_2}(\vec{x}) \\ \frac{\partial f^2}{\partial x_2 x_1}(\vec{x}) & \frac{\partial f^2}{\partial x_2^2}(\vec{x}) \end{pmatrix}$$

In General

- If $f : \mathbb{R}^d \to \mathbb{R}$, the **Hessian** at $\vec{x}$ is:

$$H(\vec{x}) = \begin{pmatrix} \frac{\partial f^2}{\partial x_1^2}(\vec{x}) & \frac{\partial f^2}{\partial x_1 x_2}(\vec{x}) & \cdots & \frac{\partial f^2}{\partial x_1 x_d}(\vec{x}) \\ \frac{\partial f^2}{\partial x_2 x_1}(\vec{x}) & \frac{\partial f^2}{\partial x_2^2}(\vec{x}) & \cdots & \frac{\partial f^2}{\partial x_2 x_d}(\vec{x}) \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f^2}{\partial x_d x_1}(\vec{x}) & \frac{\partial f^2}{\partial x_d^2}(\vec{x}) & \cdots & \frac{\partial f^2}{\partial x_d^2}(\vec{x}) \end{pmatrix}$$

The Second Derivative Test

- A function $f : \mathbb{R}^d \to \mathbb{R}$ is **convex** if for any $\vec{x} \in \mathbb{R}^d$, the Hessian matrix $H(\vec{x})$ is **positive semi-definite**.
- That is, all eigenvalues are ≥ 0

Next Time

- Backpropagation and gradient descent for training neural networks.