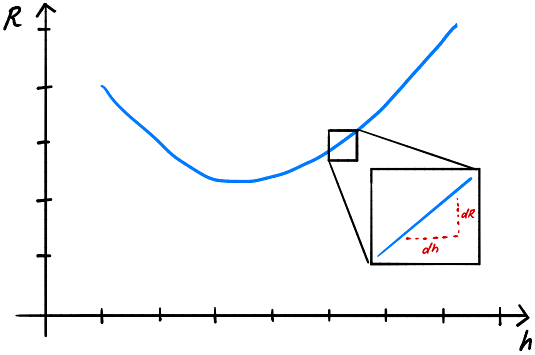


DSC 40A

Lecture 04

Learning via Optimization, pt IV

Announcements

- Remember: homework due tomorrow @ 5 pm.

Last Time: Empirical Risk Minimization

- To learn, pick a **loss function** L and minimize the **empirical risk**:

$$R(h) = \frac{1}{n}\sum_{i=1}^{n} L(h, y_i)$$

- Absolute loss: $L_{\text{abs}}(h, y) = |h - y|$ (gives the **median**)

- Square loss: $L_{\text{sq}}(h, y) = (h - y)^2$ (gives the **mean**)

- **Key Point**: Tradeoffs to each loss function.

Today

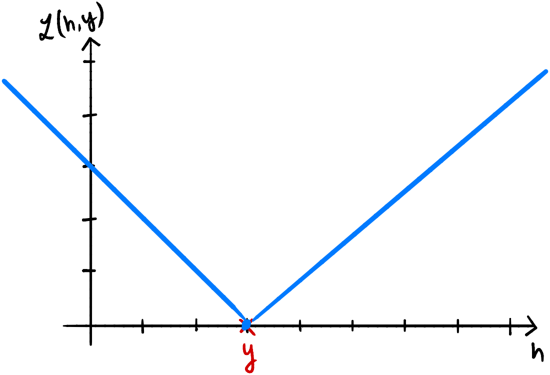
- We’ll design our own loss function.
- We’ll get stuck when trying to minimize.
- We’ll invent **gradient descent** as a general approach to minimizing functions.

Loss Functions

- A loss function $L(h, y)$ quantifies how "bad" a prediction is.
- Example: take $h = 4$ and $y = 6$.
- Absolute loss: $L_{\text{abs}}(h, y) = |4 - 6| = 2$
- Square loss: $L_{\text{sq}}(h, y) = (4 - 6)^2 = 4$

Plotting a Loss Function

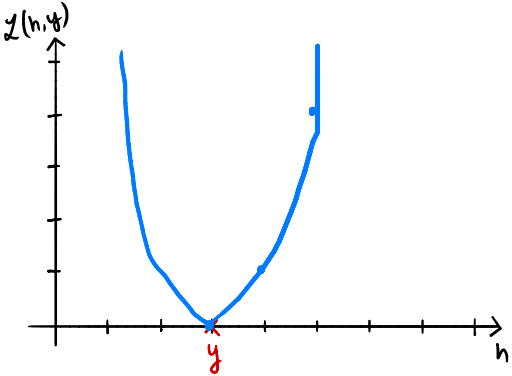
- The plot of a loss function tells us how it treats outliers.
- Consider y fixed. Plot $L_{\text{abs}}(h, y) = |h - y|$:



Plotting a Loss Function

- The plot of a loss function tells us how it treats outliers.

- Consider y fixed. Plot $L_{\text{sq}}(h, y) = (h - y)^2$:



Discussion Question

Suppose L considers all outliers to be equally as bad. What would it look like far away from y?

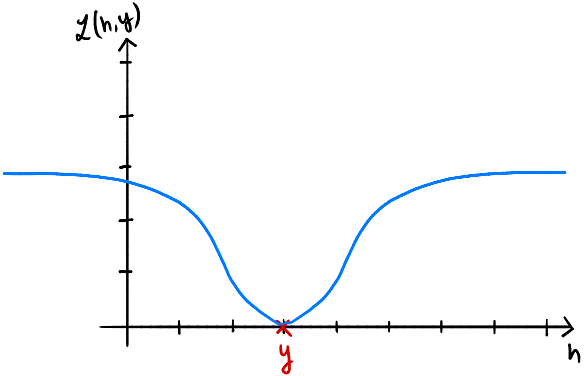
a) flat 60

b) rapidly decreasing 6

c) rapidly increasing 13

e) 1

A very insensitive loss

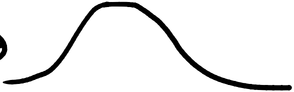


- We'll call this loss L_{ucsd} because it doesn't have a name.

Discussion Question

Which of these could be $L_{\text{ucsd}}(h, y)$?

a) $e^{-(h-y)^2}$ 15



b) $1 - e^{-(h-y)^2}$ 50

c) $1 - (h - y)^2$ 3

d) $1 - e^{-|h-y|}$ 12

Adding a scale parameter

- Problem: L_{ucsd} has a fixed scale.
- Won't work for all data sets (e.g., salaries).
- Fix: add a **scale parameter**, σ:

$$L_{\text{ucsd}}(h, y) = 1 - e^{(h-y)^2/\sigma^2}$$

Empirical Risk Minimization

- We have salaries $y_1, \ldots, y_n$.
- To find prediction, ERM says to minimize the mean loss:

$$
\begin{aligned}
R_{\text{ucsd}}(h) &= \frac{1}{n} \sum_{i=1}^{n} L_{\text{ucsd}}(h, y_i) \\
&= \frac{1}{n} \sum_{i=1}^{n} \left[1 - e^{-(h-y_i)^2/\sigma^2} \right]
\end{aligned}
$$

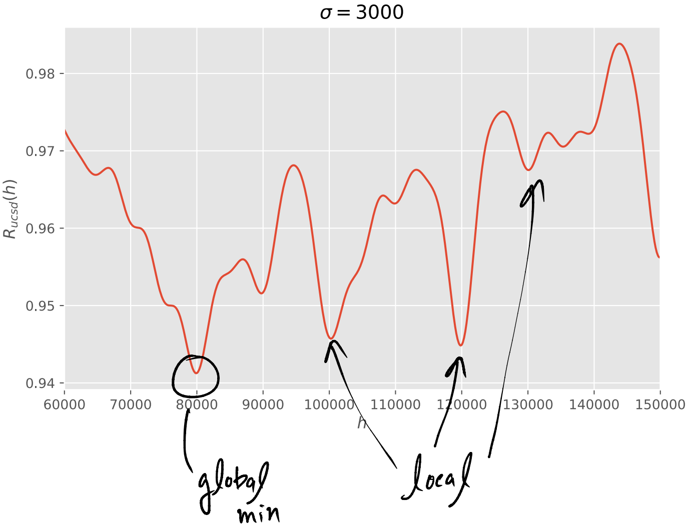
Let's plot $R_{\textbf{ucsd}}$

- Recall:

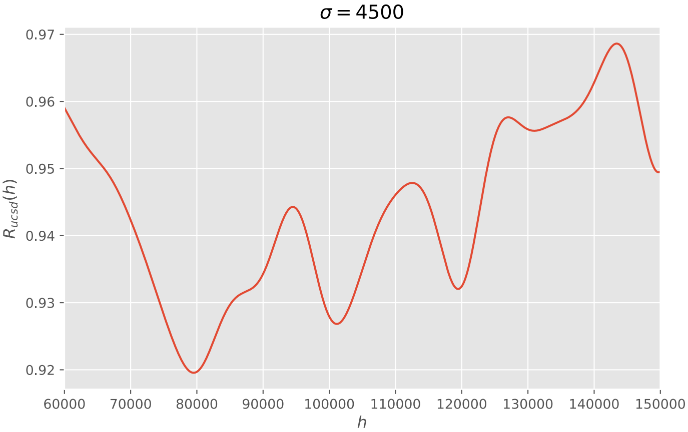
$$R_{\text{ucsd}}(h,) = \frac{1}{n} \sum_{i=1}^{n} \left[1 - e^{(h-y_i)^2/\sigma^2} \right]$$

- Once we have data $y_1, \ldots, y_n$ and a scale σ, we can plot $R_{\text{ucsd}}(h)$
- We'll use full StackOverflow data ($n = 1121$)
- Let's try several scales, σ.

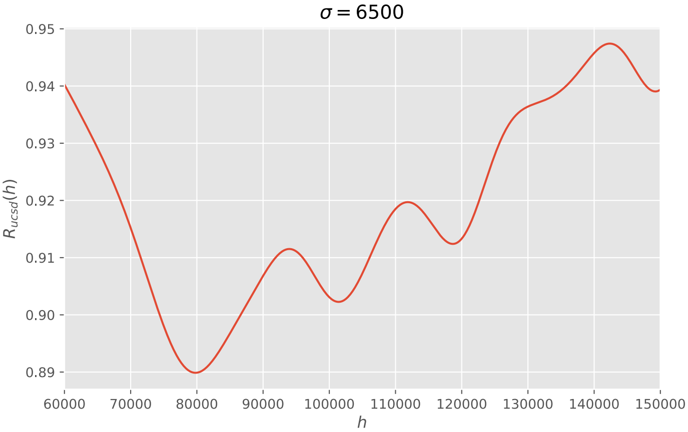
Plot of R_{ucsd}



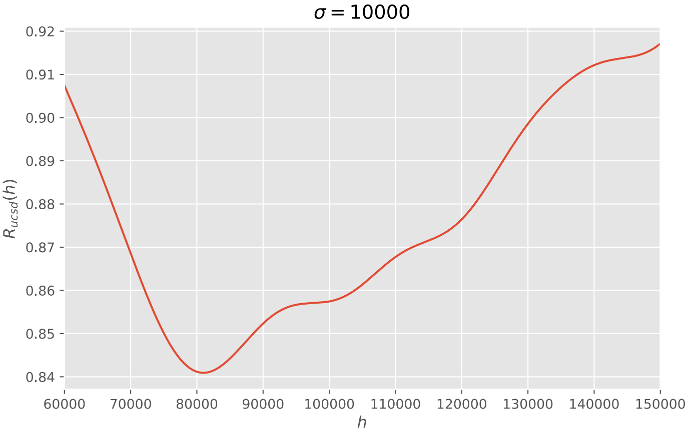
Plot of $R_{\mathbf{ucsd}}$



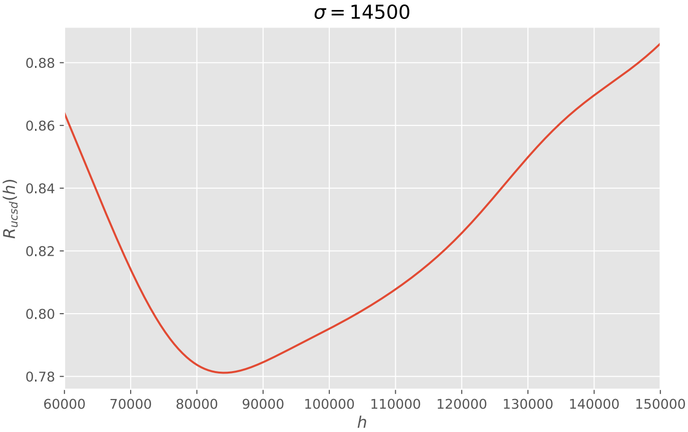
Plot of R_{ucsd}



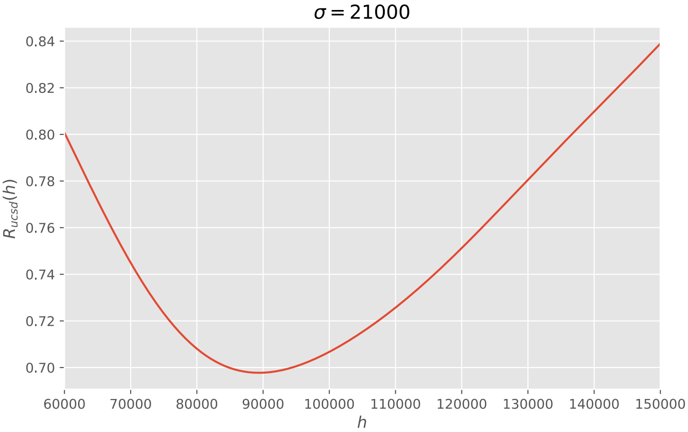
Plot of $R_{\mathbf{ucsd}}$



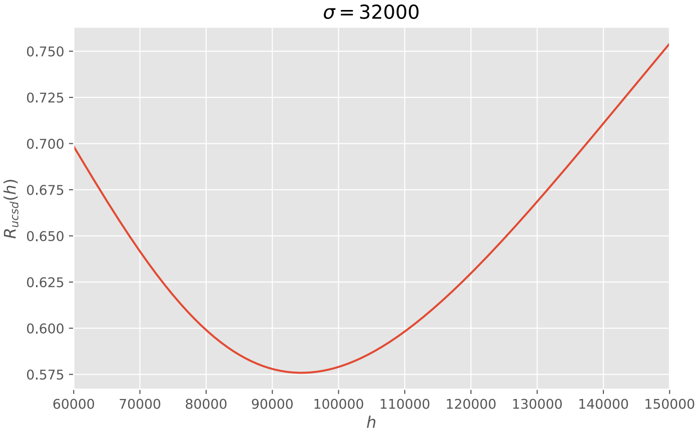
Plot of $R_{\mathbf{ucsd}}$



Plot of R_{ucsd}



Plot of $R_{\mathbf{ucsd}}$



Minimizing R_{ucsd}

- To make prediction, we find h^* minimizing $R_{\text{ucsd}}(h)$.
- R_{ucsd} is differentiable (no cusps).
- To minimize: take derivative, set to zero, solve.

Step 1) Taking the derivative

$$\frac{dR_{\text{ucsd}}}{dh} = \frac{d}{dh}\left(\frac{1}{n}\sum_{i=1}^{n}\left[1 - e^{-(h-y_i)^2/\sigma^2}\right]\right)$$

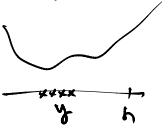
$$= \frac{1}{n}\sum_{i=1}^{n}\frac{d}{dh}\left[1 - e^{-(h-y_i)^2/\sigma^2}\right]$$

$$= \frac{1}{n}\sum -\frac{d}{dh}e^{-(h-y_i)^2/\sigma^2} \qquad u = \frac{-(h-y_i)^2}{\sigma^2}$$

$$= \frac{1}{n}\sum -\frac{d}{du}e^{u}\cdot\frac{du}{dh} \qquad \frac{du}{dh} = \frac{-2(h-y_i)}{\sigma^2}$$

$$= \frac{1}{n}\sum -e^{u}\cdot\left(\frac{-2(h-y_i)}{\sigma^2}\right)$$

$$= \frac{2}{n\sigma^2}\sum (h-y_i)\,e^{-(h-y_i)/\sigma^2}$$



Step 2) Setting to zero and solving

- We found (hopefully):

$$\frac{dR_{\text{ucsd}}}{dh}(h) = \frac{2}{n\sigma^2} \sum_{i=1}^{n} (h - y_i) \cdot e^{-(h-y_i)^2/\sigma^2}$$

- Now we just set to zero and solve for h:

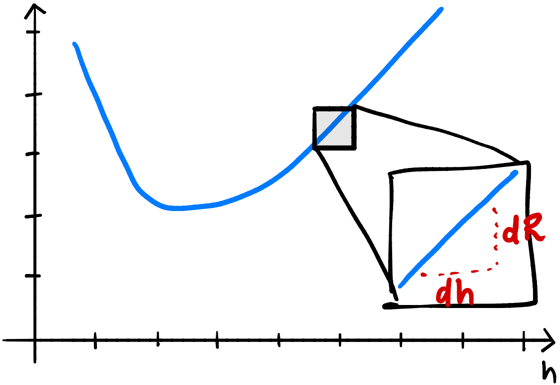
$$0 = \frac{2}{n\sigma^2} \sum_{i=1}^{n} (h - y_i) \cdot e^{-(h-y_i)^2/\sigma^2}$$

- We **can** calculate derivative, but we **can't** solve for h; we're stuck again.

Meaning of the Derivative

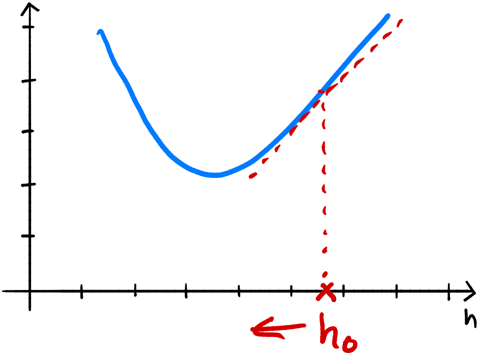
- We have the derivative; can we use it?

- $\frac{dR}{dh}(h)$ is a function; it gives the **slope** at h.



Key Idea Behind Gradient Descent

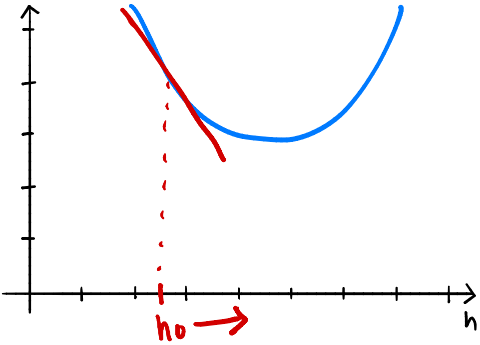
- If the slope of R at h is **positive** then moving to the **left** decreases the value of R.
- i.e., we should **decrease** h



Key Idea Behind Gradient Descent

- If the slope of R at h is **negative** then moving to the **right** decreases the value of R.

- i.e., we should **increase** h



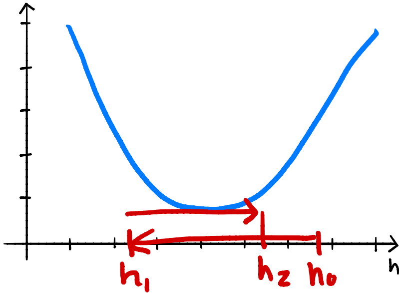
Key Idea Behind Gradient Descent

- Pick a starting place, h_0. Where do we go next?
- Slope at h_0 negative? Then increase h_0.
- Slope at h_0 positive? Then decrease h_0.
- This will work:

$$h_1 = h_0 - \frac{dR}{dh}(h_0)$$

Gradient Descent

- Pick α to be a positive number. It is the **learning rate**.
- Pick a starting prediction, h_0.
- On step i, perform update $h_i = h_{i-1} - \alpha \cdot \frac{dR}{dh}(h_{i-1})$
- Repeat until convergence (when h doesn't change much).



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

Example: Minimizing Mean Squared Error

- Recall the mean squared error and its derivative:

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

Discussion Question

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

Pick $h_0 = 4$ and $\alpha = 1/4$. What is h_1?

a) -1 8
b) 0 20
c) 1 7
d) 2 41

$$h_1 = h_0 - \alpha \frac{dR}{dh}(h_0)$$

Example $y_1 = -4 \quad y_2 = -2 \quad y_3 = 2 \quad y_4 = 4$

$$\frac{dR}{dh} = \frac{2}{n} \sum_i (h - y_i) \qquad \alpha = 1/4 \quad h_0 = 4$$

$$h_1 = h_0 - \alpha \frac{dR}{dh}(h_0)$$

$$= 4 - \frac{1}{4} \cdot 8$$

$$= 4 - 2$$

$$= 2$$

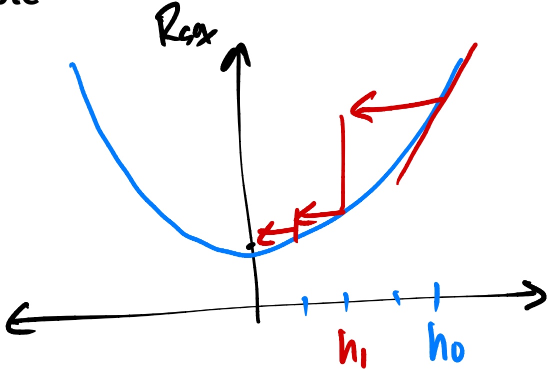
$$\frac{dR}{dh}(4) =$$

$$\frac{2}{4}\left[8 + 6 + 2 + 0\right]$$

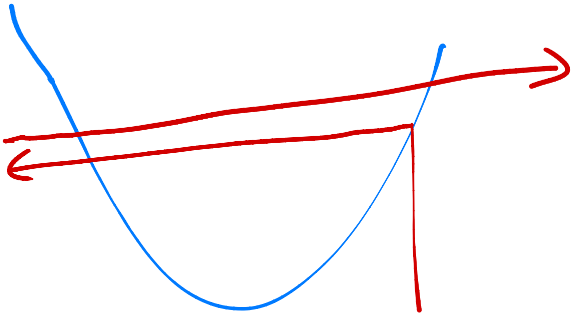
$$= \frac{1}{2} \cdot 16$$

$$= 8$$

Example



Example



Status Update

- We introduced the UCSD loss and got stuck trying to minimize.
- In response, we invented **gradient descent**.

What's Left?

- When does gradient descent work?
- When does it fail?