DSC 40A

Theoretical Foundations of Data Science I

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.

In This Video

We'll design our own loss function. We'll find that it's hard to minimize using the methods we've learned so far, which will motivate a new approach to minimizing functions.

Recommended Reading

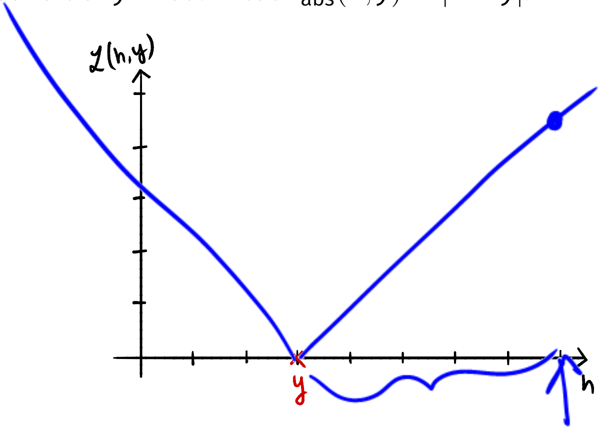
Course Notes: Chapter 1, Section 2

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: $L_{\text{sq}}(h, y) = (4 - 6)^2 = 4$

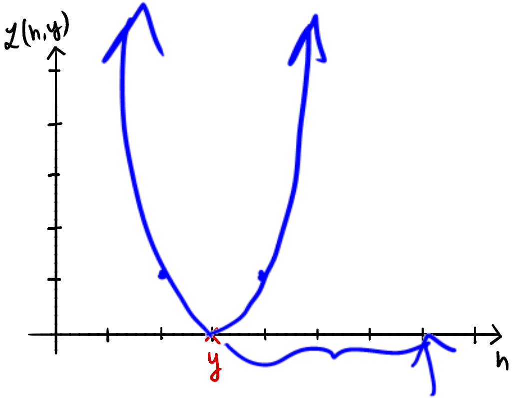
Plotting a Loss Function

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



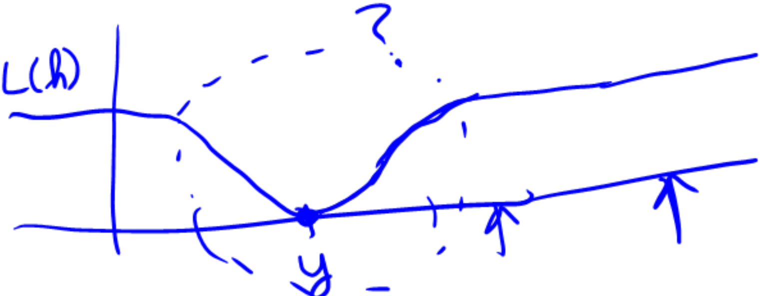
Question

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

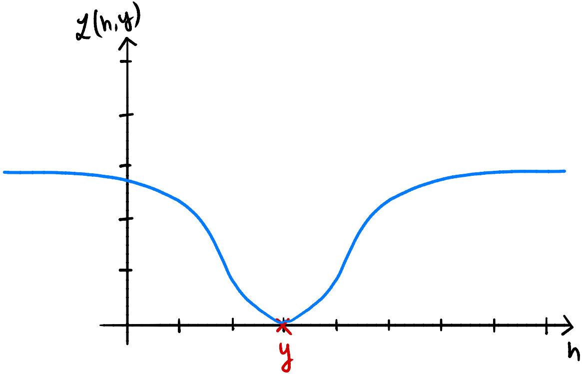
a) flat

b) rapidly decreasing

c) rapidly increasing



A very insensitive loss



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

Question

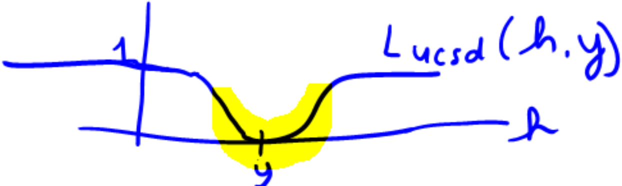
Which of these could be $L_{\text{ucsd}}(h, y)$?

~~a)~~ $e^{-\overbrace{(h-y)^2}^{\text{big}}}$

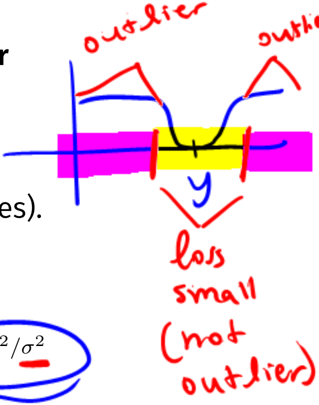
b) $1 - \underbrace{e^{-(h-y)^2}}_{\text{close to 0}}$

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

d) $1 - \underbrace{e^{-\overbrace{|h-y|}^{\text{big}}}}_{\text{close to 0}}$

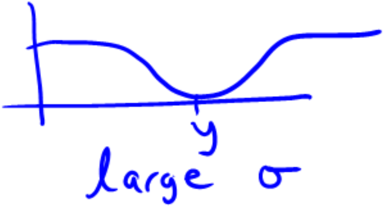


Adding a scale parameter



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

$$L_{\text{ucsd}}(h, y) = 1 - \underbrace{e^{-(h-y)^2/\sigma^2}}_{\text{similar form as bell curve}}$$



Empirical Risk Minimization

- We have salaries $\underbrace{y_1, \ldots, y_n}$.
- 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:

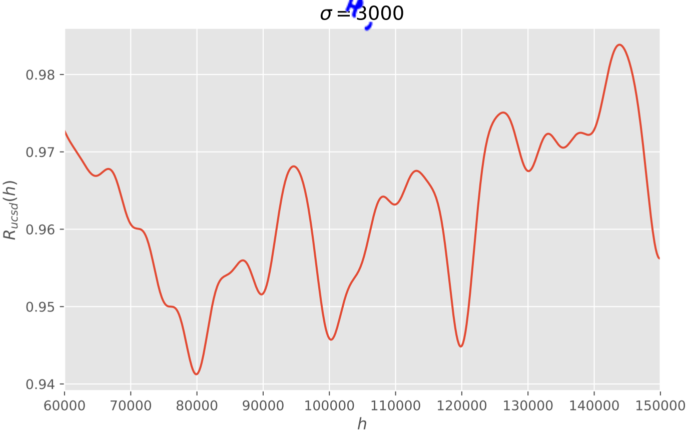
$$R_{\text{ucsd}}(h) = \frac{1}{n} \sum_{i=1}^{n} \left[1 - e^{-(h-y_i)^2/\sigma^2}\right]$$

- Once we have data $y_1, \ldots, y_n$ and a scale σ, we can plot $R_{\text{ucsd}}(h)$

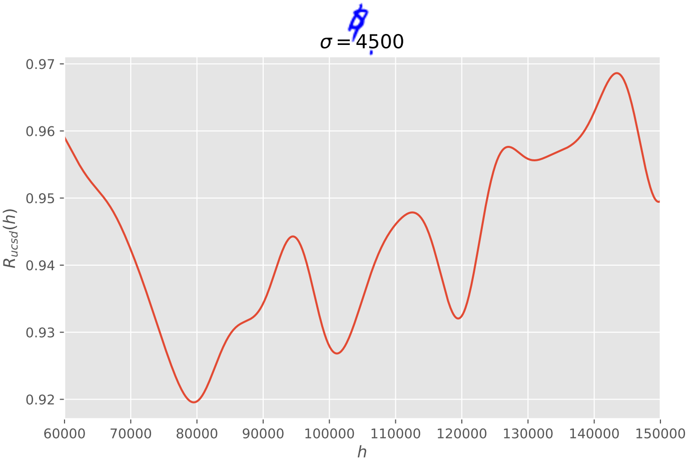
- We'll use full StackOverflow data ($n = 1121$)

- Let's try several scales, σ.

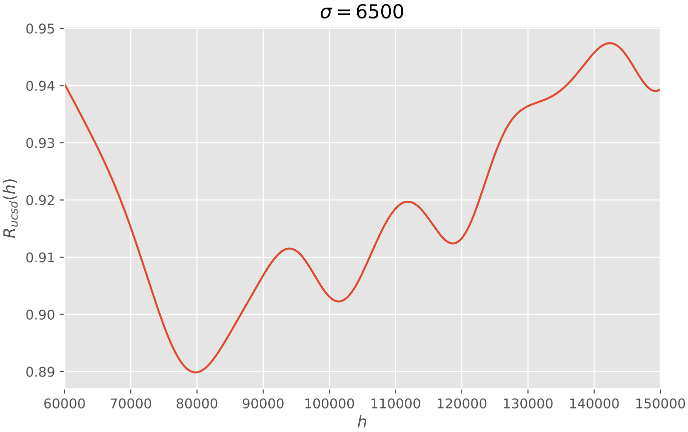
Plot of R_{ucsd}



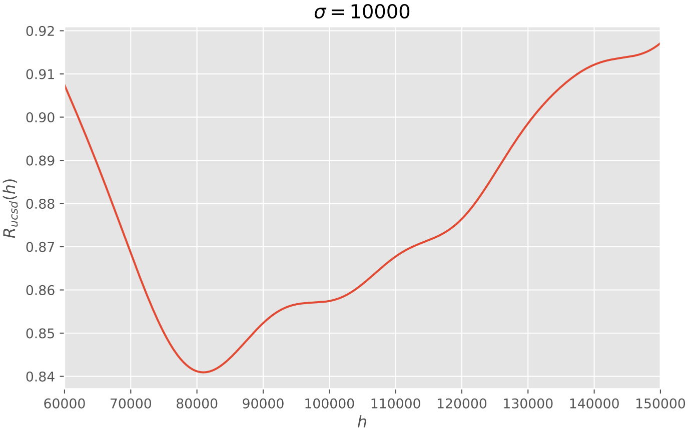
Plot of $R_{\mathbf{ucsd}}$

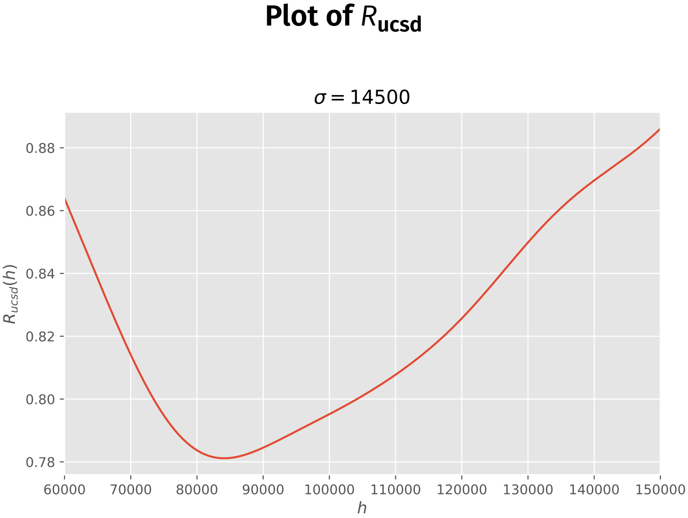


Plot of $R_{\mathbf{ucsd}}$

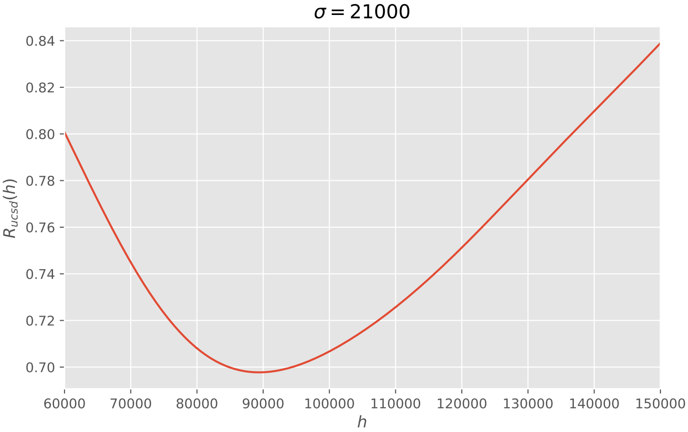


Plot of $R_{\mathbf{ucsd}}$

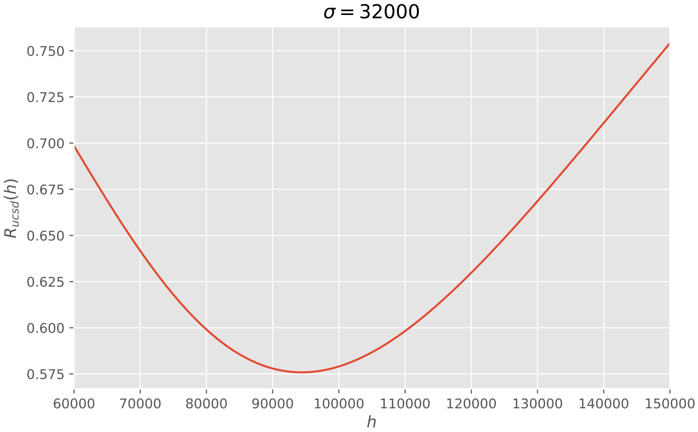



Plot of $R_{\mathbf{ucsd}}$
$\sigma = 14500$
$R_{ucsd}(h)$
0.88
0.86
0.84
0.82
0.80
0.78
60000
70000
80000
90000
100000
110000
120000
130000
140000
150000
h

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



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



Minimizing R_{ucsd}

- To make prediction, we find h^* minimizing $R_{\text{ucsd}}(h)$.
- R_{ucsd} is **differentiable**.
- To minimize: take derivative, set to zero, solve.

Step 1) Taking the derivative

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

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

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

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

Step 2) Setting to zero and solving

- We found:

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

- 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.

Summary

- We created our own loss function, which was designed to treat all outliers in much the same way.

- Our loss function was differentiable, but we still couldn't minimize it

- **Next Time:** We'll invent a general algorithm called **gradient descent** for minimizing differentiable functions.