Ch. 3. Loss Functions and Optimiz

Date /

loss Function

<loss function> tells how good the current classifier is $\Rightarrow$ measure the badness of weight.

given a dataset of examples: $\{(x_i, y_i)\}_{i=1}^{N}$ where $\begin{cases} x_i : \text{image} \\ y_i : \text{(integer) label} \end{cases}$

$\Rightarrow$ loss over the dataset = average of sum of loss:

$$L = \frac{1}{N}\sum L_i(f(x_i, W), y_i)$$

<Multiclass SVM loss>

- scores vector: $s = f(x_i, W)$ if true score is higher than other scores $\uparrow$ by a margin value. $\Rightarrow$ T.

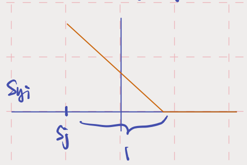
SVM loss: $L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases} = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

mean or sum doesn't really matter.

Hinge Loss

$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Hinge loss:



$\begin{cases} \text{min loss} = 0 \\ \text{max loss} = \infty \end{cases}$

Q: L = 0 ? is W unique? A: No! 2W also achieve 0 Loss!

- $$\underbrace{L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{} + \underbrace{\boxed{\lambda R(W)}}_{}$$ — soft penalty.

$\hookrightarrow$ Regularization: "simpler" W.

Data loss: model predictions should match training data.

- Occam's Razor: 奥卡姆剃须刀 (1285-1347)

"Among competing hypotheses, the simplest is the best."

- Regularization: peneliẕe the complex of the model.

① L_1, $R(W) = \sum_k \sum_l |W_{k,l}|$ $\Rightarrow$ encourage sparsity.

② L_2: $R(W) = \sum_k \sum_l W_{k,l}^2$ $\Rightarrow$ euclidean norm,

③ Elastic net ($L_1 + L_2$): $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

④ max norm. Dropout, Francier: Batch normalization stochastic depth.

* Bayesian: L_2 also corresponds MAP inference using a Gaussian prior on W.

Softmax

(cross-entropy loss)

$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$

- Softmax Classifier: Multinomial Logistic Regression.

scores = unnormalized log prob. of the classes.

$$P(Y = k \mid X = x_i) = \boxed{\frac{e^{s_k}}{\sum_j e^{s_j}}} \quad \text{where } s = f(x_i; W)$$

$\uparrow$ Softmax function

$\hookrightarrow$ want to max the log likelihood, or (for loss func) to min the $-\log$ likelihood:

$$L_i = -\log P(Y = y_i \mid X = x_i)$$

$\uparrow$ loss measure badness not goodness.

$$\Rightarrow L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

Q. What is min & max value of softmax loss? due to computation limit, never get 0 loss!

A: min = 0, max = ∞. ⇒ 0 is theoretical minimum loss.

Optimization

How to find W that minimize the loss? ⇒ iterative method.

- Strategy 1: Random Search ⇒ very bad idea solution
- Strategy 2: Follow the slope

1D: the derivative of a function: $\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$

(linear, 1st order approx.)

multi-D: the $\boxed{\text{gradient}}$ is the vector of (partial derivatives) along each dimensions.

- The slope in any direction is the $\boxed{\text{dot product}}$ of the direction with the gradient.

The direction of steepest descent is the $\boxed{\text{negative gradient}}$ (gradient · direction unit vector)

Analytic Gradient:

check implementation with numerical gradient.

```
while True:
    weights_grad = evaluate_grad(loss_func, data, weights)
    weights += - step_size * weights_grad  # perform parameter update
```

SGD (Stochastic Gradient Descent)

$$L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(x_i, y_i, W) + \lambda R(W)$$

$$\nabla_W L(W) = \frac{1}{N} \sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

- Approximate sum using a $\boxed{\text{minibatch}}$ of examples of 32/64/128 common.