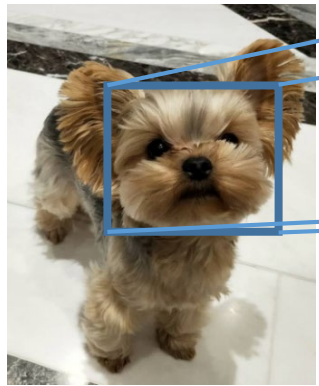


计算机视觉与多媒体信息处理

Lecture 3: 损失函数和优化



L02-图像分类



255	255	255	255	245	200	186	187	192	222	255	255	255
255	255	255	239	162	154	172	173	172	155	197	227	236
255	255	255	173	209	206	194	204	200	184	155	241	253
255	242	267	193	192	193	194	205	182	157	250	247	255
255	255	255	189	157	196	207	199	167	241	186	219	248
251	255	255	179	213	193	193	193	160	241	255	255	255
247	251	251	233	193	196	196	196	135	255	255	255	255
255	255	255	245	255	255	254	250	181	177	231	220	146
255	207	245	255	255	245	165	187	179	156	236	254	255
82	255	255	253	274	169	205	192	205	156	197	253	255
39	255	255	253	129	161	198	192	192	204	154	192	255
255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	272	189	202	186	145	206	255	255
255	249	251	254	254	200	153	167	171	215	254	252	255
252	253	255	255	255	255	233	221	245	255	250	250	255
255	255	255	172	175	255	255	255	253	249	249	253	255

一组[0, 255]的数字

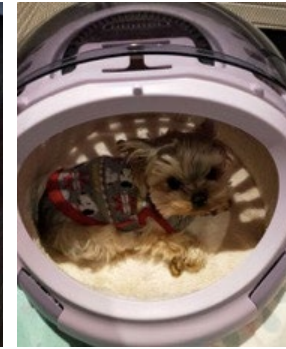


狗

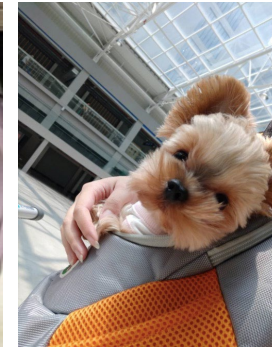
Semantic Gap



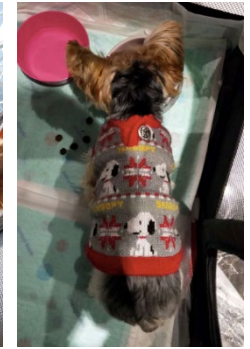
illumination



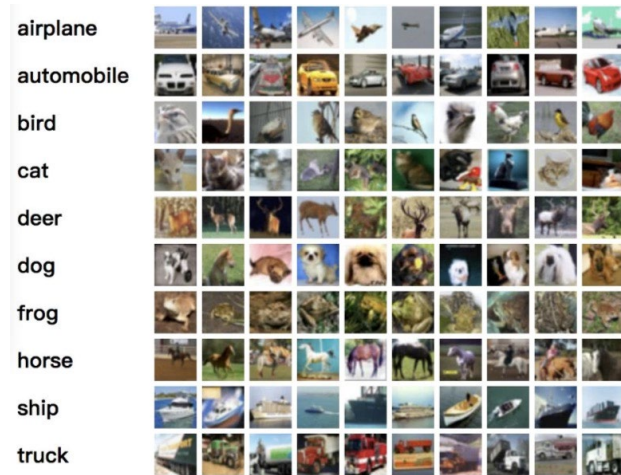
background clutter



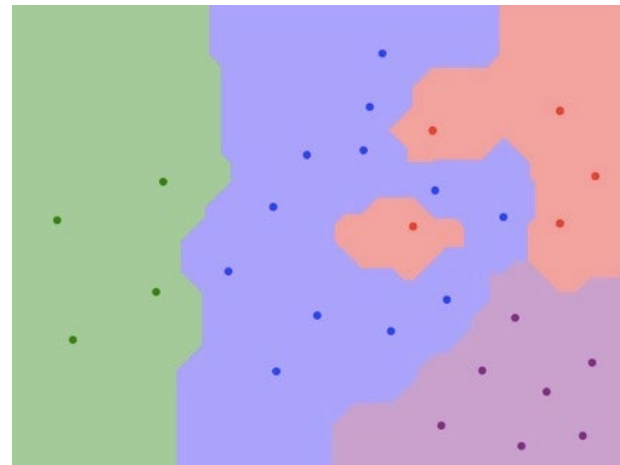
occlusion



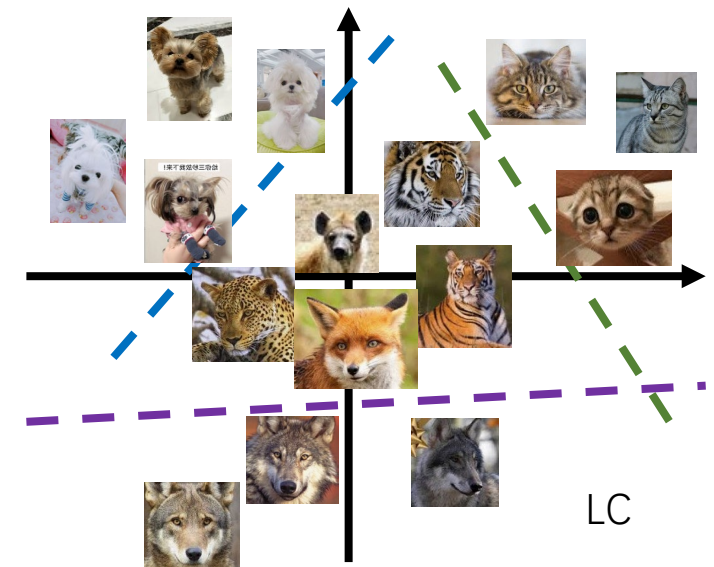
deformation



CIFAR-10

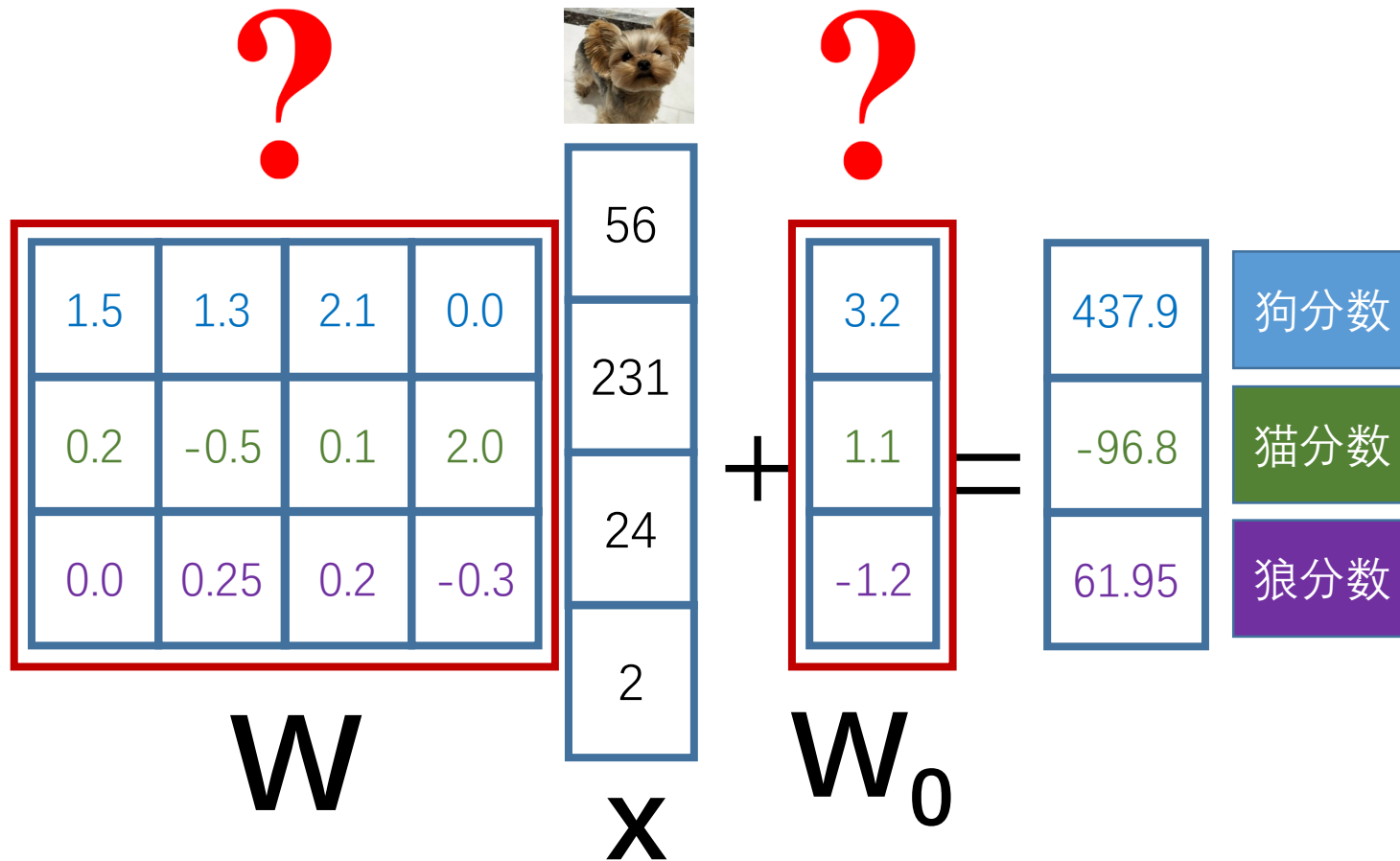


kNN



LC

线性分类器的权重



- ✓ 从模型输出分数的好坏反推权重的好坏
- ✓ 定义一个衡量输出分数好坏的函数：损失函数（目标函数）
- ✓ 设计一个反推好的权重的方法，即能够最小化损失函数的计算方法：优化

损失函数



狗分数	7.9	3.3	-1.1	2.3
猫分数	5.8	-0.8	3.4	1.5
狼分数	-1.9	6.5	2.5	4.4

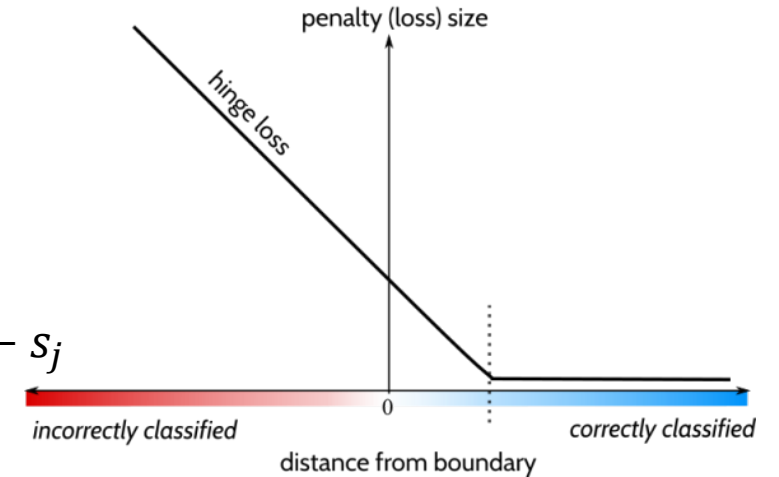
- ✓ 训练集 $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$
 - \mathbf{x}_i 为第 i 个图片, $y_i \in Z$ 为它的类别标签
 - $f(\mathbf{W}, \mathbf{x}_i)$ 为第 i 个图片的输出分数

- ✓ 损失函数
 - 第 i 条数据的损失函数 $L_i = l(f(\mathbf{W}, \mathbf{x}_i), y_i)$
 - 训练样本总体损失 $L = \frac{1}{N} \sum_{i=1}^N L_i$

假定训练集有 N 个图片 (这里 $N = 4$) , 经过线性分类器 $f(\mathbf{W}, \mathbf{x})$ 计算后分别得到如上分数

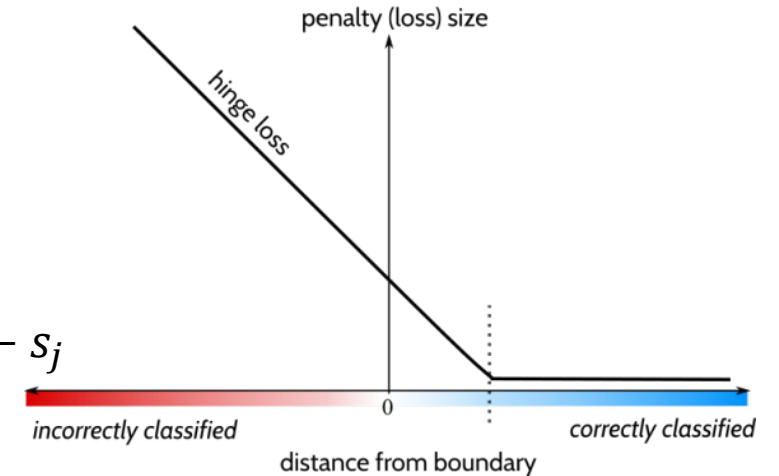
线性分类器常用损失函数

- 令 $\mathbf{s} = f(\mathbf{W}, \mathbf{x}_i)$, 即图片 i 的所有类别分数 $s_{y_i} - s_j$
- multiclass SVM loss (Hinge loss): $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$
 ✓ SVM 分类器, y_i 是图片 i 的类别标签



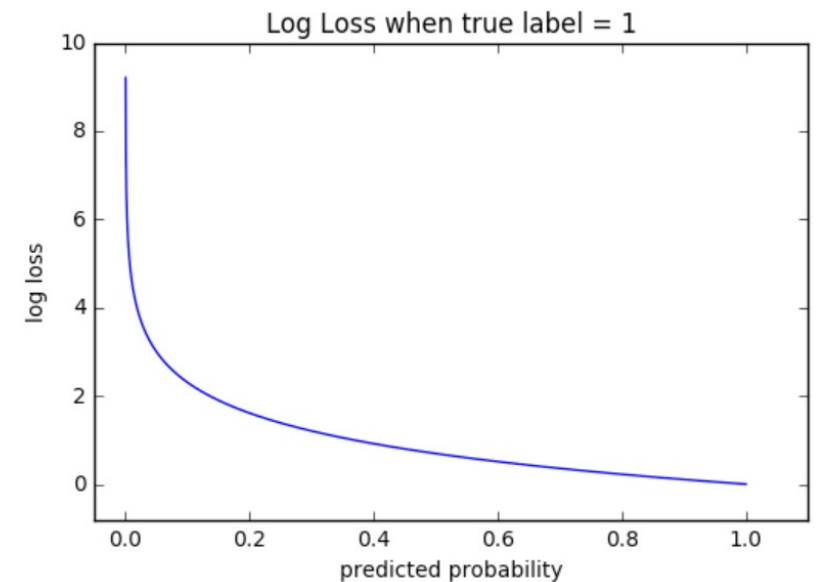
线性分类器常用损失函数

- 令 $\mathbf{s} = f(\mathbf{W}, \mathbf{x}_i)$, 即图片 i 的所有类别分数 $s_{y_i} - s_j$

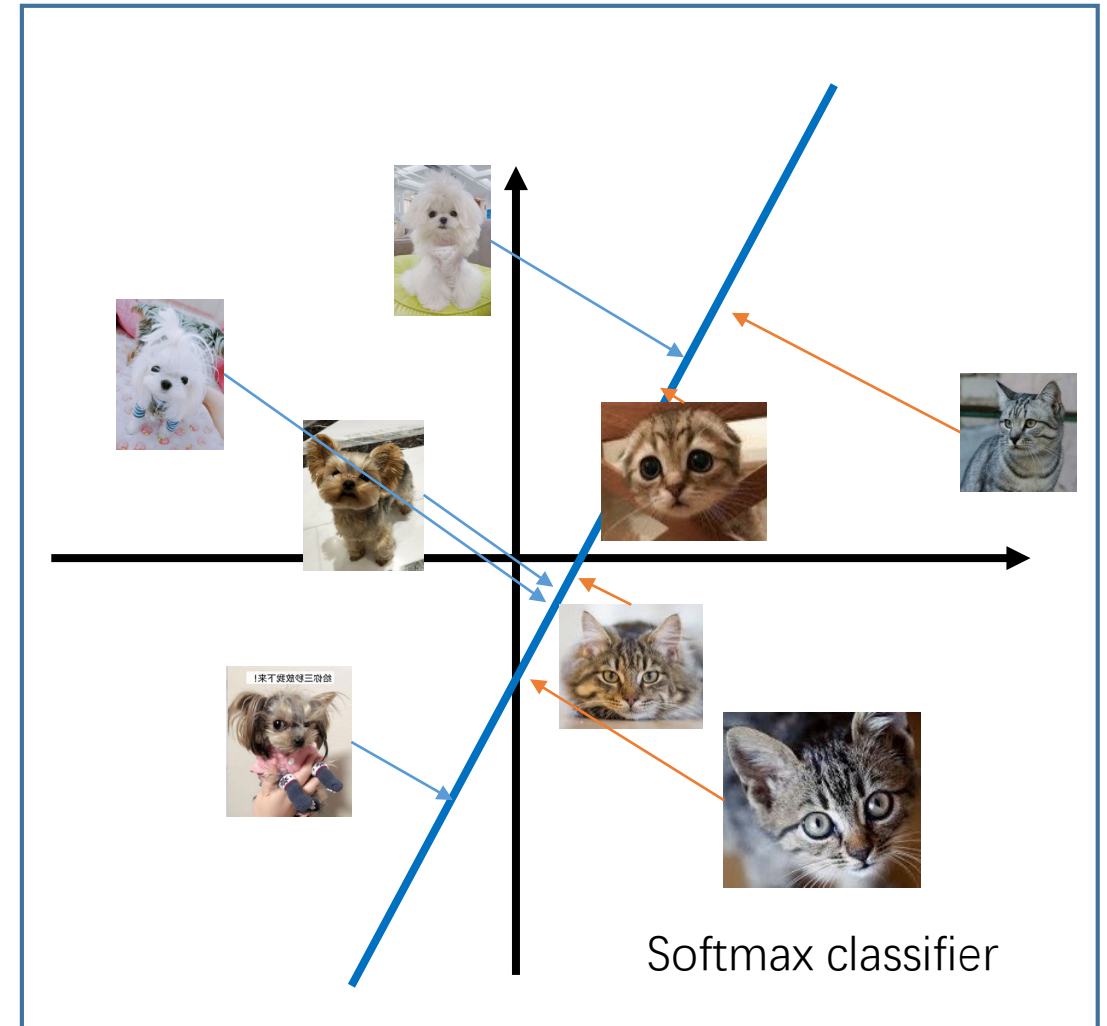
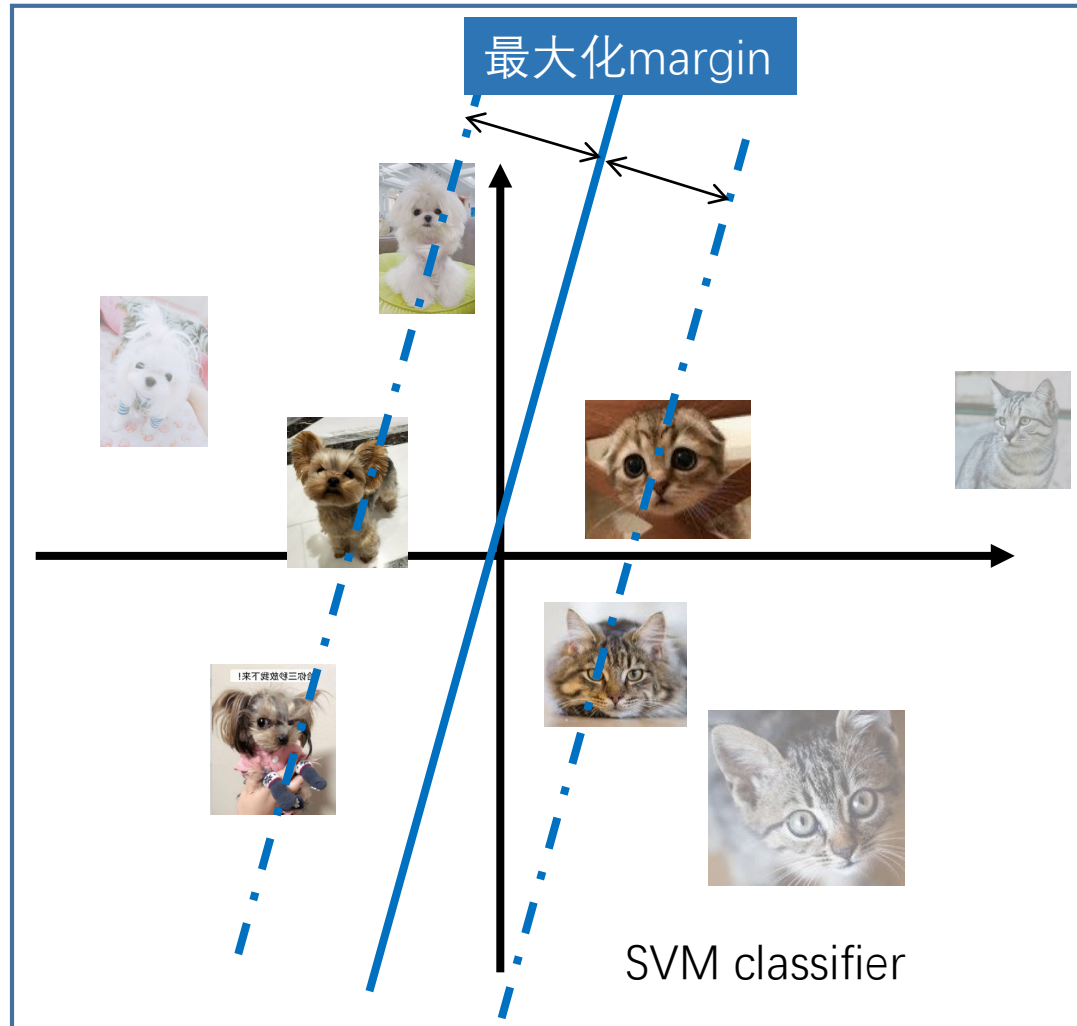


- multiclass SVM loss (Hinge loss): $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$
 ✓ SVM分类器, y_i 是图片 i 的类别标签

- cross-entropy loss: $L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$
 ✓ Softmax分类器 (多类别逻辑回归)



SVM分类器 Vs. Softmax分类器



SVM分类器



狗分数

7.9 3.3 -1.1 2.3

猫分数

5.8 -0.8 3.4 1.5

狼分数

-1.9 6.5 2.5 4.4

Multiclass SVM loss:

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

第一个图片的loss:

$$\begin{aligned} L_1 &= \max(0, 5.8 - 7.9 + 1) + \max(0, -1.9 - 7.9 + 1) \\ &= \max(0, -1.1) + \max(0, -8.8) \\ &= 0 \end{aligned}$$

SVM分类器



狗分数

7.9

3.3

-1.1

2.3

猫分数

5.8

-0.8

3.4

1.5

狼分数

-1.9

6.5

2.5

4.4

Multiclass SVM loss:

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

第二个图片的loss:

$$\begin{aligned} L_2 &= \max(0, 3.3 - (-0.8) + 1) + \max(0, 6.5 - (-0.8) + 1) \\ &= \max(0, 5.1) + \max(0, 8.3) \\ &= 5.1 + 8.3 \\ &= 13.4 \end{aligned}$$

SVM分类器



狗分数

7.9 3.3 -1.1 2.3

猫分数

5.8 -0.8 3.4 1.5

狼分数

-1.9 6.5 2.5 4.4

Multiclass SVM loss:

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

第三个图片的loss:

$$\begin{aligned} L_3 &= \max(0, -1.1 - 2.5 + 1) + \max(0, 3.4 - 2.5 + 1) \\ &= \max(0, -2.6) + \max(0, 1.9) \\ &= 0 + 1.9 \\ &= 1.9 \end{aligned}$$

SVM分类器



狗分数

7.9 3.3 -1.1 2.3

猫分数

5.8 -0.8 3.4 1.5

狼分数

-1.9 6.5 2.5 4.4

Multiclass SVM loss:

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

第四个图片的loss:

$$\begin{aligned} L_4 &= \max(0, 1.5 - 2.3 + 1) + \max(0, 4.4 - 2.3 + 1) \\ &= \max(0, 0.2) + \max(0, 3.1) \\ &= 0.2 + 3.1 \\ &= 3.3 \end{aligned}$$

SVM分类器



狗分数

7.9

3.3

-1.1

2.3

猫分数

5.8

-0.8

3.4

1.5

狼分数

-1.9

6.5

2.5

4.4

0

13.4

1.9

3.3

Multiclass SVM loss:

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

总的loss:

$$L = \frac{1}{N} \sum_{i=1}^N L_i$$

$$L = (0 + 13.4 + 1.9 + 3.3) / 4 = 4.65$$

SVM分类器



狗分数

7.9

3.3

-1.1

2.3

猫分数

5.8

-0.8

3.4

1.5

狼分数

-1.9

6.5

2.5

4.4

0

13.4

1.9

3.3

Multiclass SVM loss:

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

Q: 如果有输出分数发生微小改变 (比如 ± 0.001)，如何影响损失函数?

A: $s_j - s_{y_i} < -1$ 时，没有影响。反之，会略微改变损失函数的值。

SVM分类器



狗分数

7.9

3.3

-1.1

2.3

猫分数

5.8

-0.8

3.4

1.5

狼分数

-1.9

6.5

2.5

4.4

0

13.4

1.9

3.3

Multiclass SVM loss:

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

Q: 损失函数的最大值和最小值分别是多少?

A: 最小值为0, 最大值为正无穷 (理论上)

可用于代码正确性检查 (Sanity Check)

SVM分类器



狗分数

7.9

3.3

-1.1

2.3

猫分数

5.8

-0.8

3.4

1.5

狼分数

-1.9

6.5

2.5

4.4

0

13.4

1.9

3.3

Multiclass SVM loss:

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

Q: 假如初始化 \mathbf{W} 接近0, 导致所有输出分数都 ≈ 0 , 那么 L_i 约等于多少?

A: $C - 1$, C 为类别数, 这里为 $3 - 1 = 2$

可用于代码正确性检查

SVM分类器



狗分数

7.9

3.3

-1.1

2.3

猫分数

5.8

-0.8

3.4

1.5

狼分数

-1.9

6.5

2.5

4.4

0

13.4

1.9

3.3

Multiclass SVM loss:

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

Q: 假如去掉 $j \neq y_i$ 的限制, 损失函数如何变化?

A: 增加 1

可用于代码正确性检查

SVM分类器



狗分数

7.9

3.3

-1.1

2.3

猫分数

5.8

-0.8

3.4

1.5

狼分数

-1.9

6.5

2.5

4.4

0

13.4

1.9

3.3

Multiclass SVM loss:

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

Q: 假如在 L_i 中使用求平均代替求和, 模型如何变化?

A: 不变

SVM分类器



狗分数

7.9

3.3

-1.1

2.3

猫分数

5.8

-0.8

3.4

1.5

狼分数

-1.9

6.5

2.5

4.4

0

13.4

1.9

3.3

Multiclass SVM loss:

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

Q: 假如在 L_i 中使用 $\max(0, s_j - s_{y_i} + 2)$ 代替 $\max(0, s_j - s_{y_i} + 1)$, 有什么影响?

A: 没有影响, SVM loss只关注输出分数之间的差异, 这里的常数只起到scale权重的作用。

SVM分类器



狗分数

7.9

3.3

-1.1

2.3

猫分数

5.8

-0.8

3.4

1.5

狼分数

-1.9

6.5

2.5

4.4

0

13.4

1.9

3.3

Multiclass SVM loss:

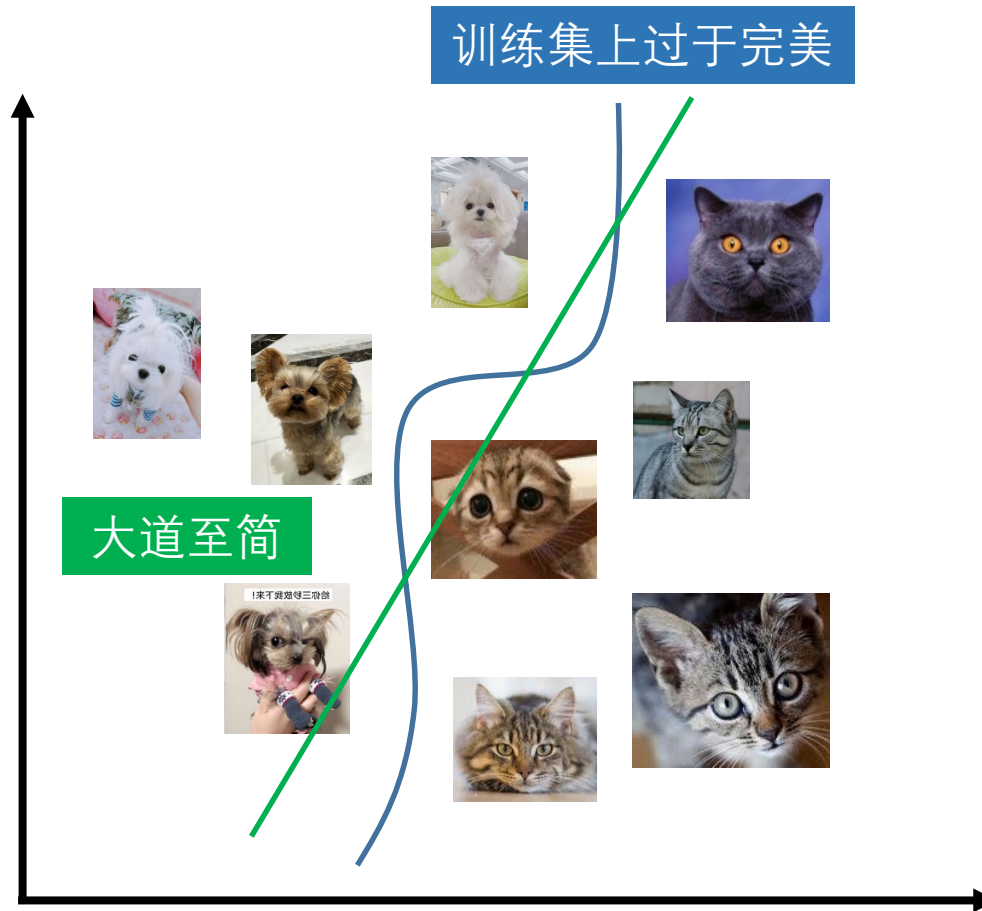
$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Q: 假如 \mathbf{W} 使得 $L = 0$ (完美), 请问 \mathbf{W} 是否唯一?

A: 不是, $\mathbf{W} \times c$ 也使得 $L = 0$, c 为任意正整数

选取更好的 \mathbf{W} : 添加正则项!

正则化 (Regularization)



将复杂模型简单化

正则化

$$L(\mathbf{W}) = \underbrace{\frac{1}{N} \sum_{i=1}^N l(f(\mathbf{W}, \mathbf{x}_i), y_i)}_{\text{数据损失}} + \lambda R(\mathbf{W})$$

数据损失

✓ 使得模型尽可能拟合训练集

正则化

正则项

✓ 防止模型过度拟合训练集

$$L(\mathbf{W}) = \underbrace{\frac{1}{N} \sum_{i=1}^N l(f(\mathbf{W}, \mathbf{x}_i), y_i)}_{\text{数据损失}} + \underbrace{\lambda R(\mathbf{W})}_{\text{正则项}}$$

数据损失

✓ 使得模型尽可能拟合训练集

正则化

$$L(\mathbf{W}) = \underbrace{\frac{1}{N} \sum_{i=1}^N l(f(\mathbf{W}, \mathbf{x}_i), y_i)}_{\text{数据损失}} + \underbrace{\lambda R(\mathbf{W})}_{\text{正则项}}$$

数据损失

✓ 使得模型尽可能拟合训练集

正则项

✓ 防止模型过度拟合训练集

超参数, 正则化强度

L1: $\sum_k \sum_l |w_{k,l}|$

L2: $\sum_k \sum_l w_{k,l}^2$

L1+L2(elastic net): $\sum_k \sum_l |w_{k,l}| + \beta w_{k,l}^2$

Dropout

Batch normalization

Stochastic depth

Fractional pooling

等

正则化

正则项

✓ 防止模型过度拟合训练集

超参数, 正则化强度

$$L(\mathbf{W}) = \underbrace{\frac{1}{N} \sum_{i=1}^N l(f(\mathbf{W}, \mathbf{x}_i), y_i)}_{\text{数据损失}} + \underbrace{\lambda R(\mathbf{W})}_{\text{正则项}}$$

数据损失

✓ 使得模型尽可能拟合训练集

L1: $\sum_k \sum_l |w_{k,l}|$

L2: $\sum_k \sum_l w_{k,l}^2$

L1+L2(elastic net): $\sum_k \sum_l |w_{k,l}| + \beta w_{k,l}^2$

Dropout

Batch normalization

Stochastic depth

Fractional pooling

等

添加正则项的意义:

- ✓ 缩小权重空间
- ✓ 调整权重偏好的分布
- ✓ 提高模型泛化能力

正则化: L1 Vs. L2

$$\mathbf{x} = [1, 1, 1]$$

$$\mathbf{W}_1 = [1, 0, 0]$$

$$\mathbf{W}_2 = [0.3, -0.1, 0.8]$$

$$\mathbf{W}_1^T \mathbf{x} = \mathbf{W}_2^T \mathbf{x} = 1$$

数据损失相同

正则项

$$\text{L1: } \sum_k \sum_l |W_1| = 1, \sum_k \sum_l |W_2| = 1.2$$

$$\text{L2: } \sum_k \sum_l W_1^2 = 1, \sum_k \sum_l W_2^2 = 0.74$$

- ✓ L1偏向于使权重集中在少数输入像素上
- ✓ L2偏向于使权重分布在所有像素上

Softmax分类器

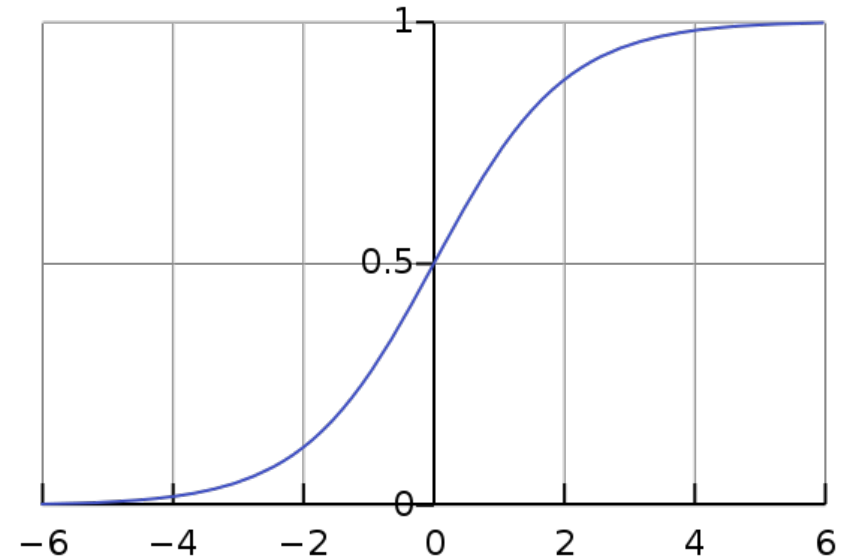
- 将分数通过softmax函数转化为概率

$$\checkmark s = f(W, x_i)$$

$$\checkmark P(y = k | x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

$$f(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid



- 利用交叉熵 (cross-entropy) 计算第*i*个图片的损失

$$\checkmark L_i = -\log P(y = y_i | x_i) = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

✓ 这里 y_i 代表正确的标签



Claude Shannon

交叉熵 (cross-entropy)



真实天气: 50%

30%

20%

$$H(\text{真实}) = -(0.5 \times \log(0.5) + 0.3 \times \log(0.3) + 0.2 \times \log(0.2)) = 0.45$$

- Entropy: 衡量概率分布 Q 的不确定性

$$\checkmark H(Q) = -\sum_i q_i \log q_i$$

天气预报: 40%

30%

30%

$$H(\text{真实}, \text{预报}) = -(0.5 \times \log(0.4) + 0.3 \times \log(0.3) + 0.2 \times \log(0.3)) = 0.46$$

- Cross-entropy: 衡量概率分布 P 服从概率分布 Q 的不确定性

$$\checkmark H(Q, P) = -\sum_i q_i \log p_i$$

$$\checkmark D_{KL}(Q||P) = H(Q, P) - H(Q) \quad \text{KL散度 (相对熵): 衡量分布Q和P的差异性}$$

Softmax分类器



$$p(y = k|x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

狗分数

7.9

2697.3

猫分数

5.8

330.3

狼分数

-1.9

0.15

exp

归一化

0.89

0.11

0.00

预测分布P

$H(Q, P)$

1

0

0

真实分布Q

$H(Q, P) = -$

$(1 \times \log(0.89) +$

$0 \times \log(0.11) +$

$0 \times \log(0.00))$

$= -\log(0.89)$

第*i*个图片的损失为:

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

Softmax分类器



$$p(y = k|x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

狗分数

7.9

2697.3

猫分数

5.8

330.3

狼分数

-1.9

0.15

exp

归一化

0.89

0.11

0.00

预测分布P

$H(Q, P)$

1

0

0

真实分布Q

第*i*个图片的损失为:

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

Q: 如果有输出分数发生微小改变 (比如 ± 0.1), 损失函数是否发生改变?

A: 是的, 正确类别和错误类别输出分数差距越大, 损失函数走

注意和SVM loss的区别

Softmax分类器



$$p(y = k|x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

狗分数

7.9

2697.3

猫分数

5.8

330.3

狼分数

-1.9

0.15

exp

归一化

0.89

0.11

0.00

$H(Q, P)$

1

0

0

预测分布P

真实分布Q

第*i*个图片的损失为:

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

Q: 损失函数 L_i 的最大值和最小值分别是多少?

A: 最小值为0 (理论上), 最大值为正无穷 (理论上)

Softmax分类器



$$p(y = k|x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

第*i*个图片的损失为:

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

狗分数

7.9

2697.3

猫分数

5.8

330.3

狼分数

-1.9

0.15

exp

归一化

0.89

0.11

0.00

$H(Q, P)$

1

0

0

预测分布P

真实分布Q

Q: 假如初始化 W 接近0, 导致所有输出分数都 ≈ 0 , 那么 L 约等于多少?

A: $\log C$, C 为类别数, 这里为 $\log 3 \approx 0.477$

可用于代码正确性检查

损失函数计算流程

- ✓ 训练集 $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$
 - \mathbf{x}_i 为第 i 个图片, $y_i \in Z$ 为它的类别标签
 - $\mathbf{s} = f(\mathbf{W}, \mathbf{x}_i)$ 为第 i 个图片输出的所有分数
 - L_i 为第 i 个图片的预测损失

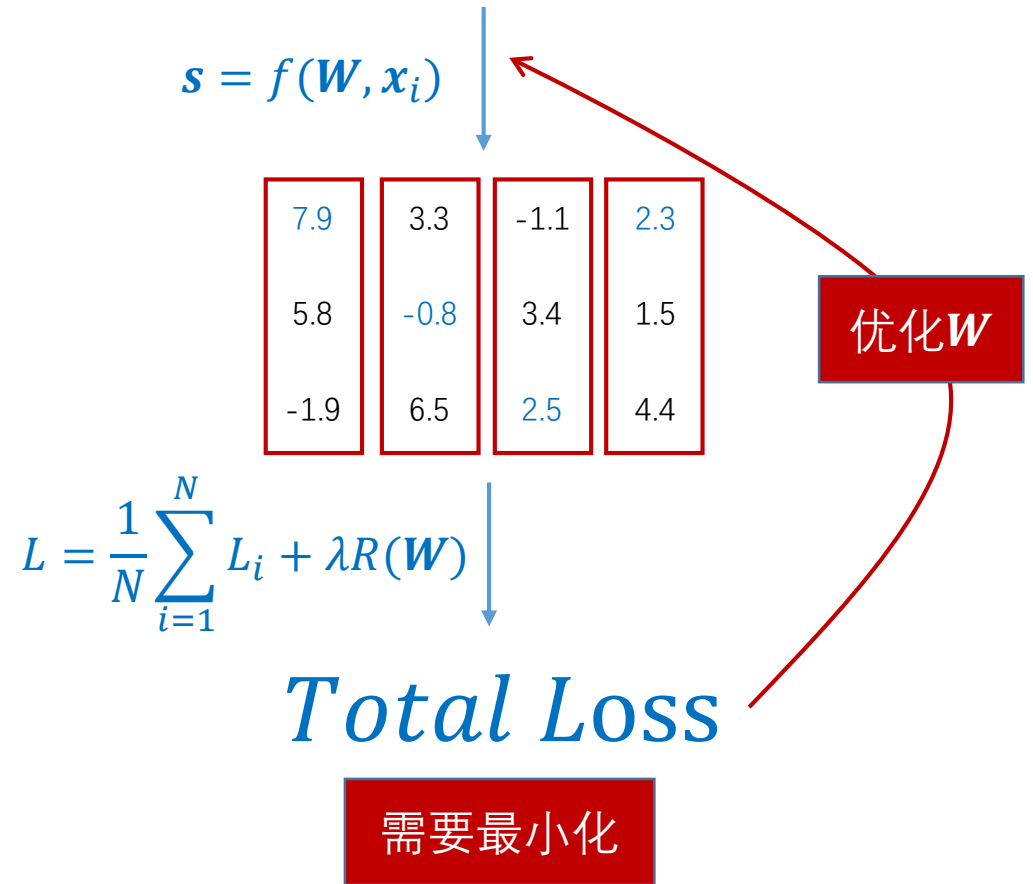
Multiclass SVM loss: $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

SVM分类器

Cross-entropy loss : $L_i = -\log P(y_i | \mathbf{x}_i) = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$

Softmax分类器

总体损失加上正则项: $L = \frac{1}{N} \sum_{i=1}^N L_i + \lambda R(\mathbf{W})$



Wait a minute。。。

- 线性分类器并非直接在原始像素值上进行训练
- 回想线性分类器的传统应用场景

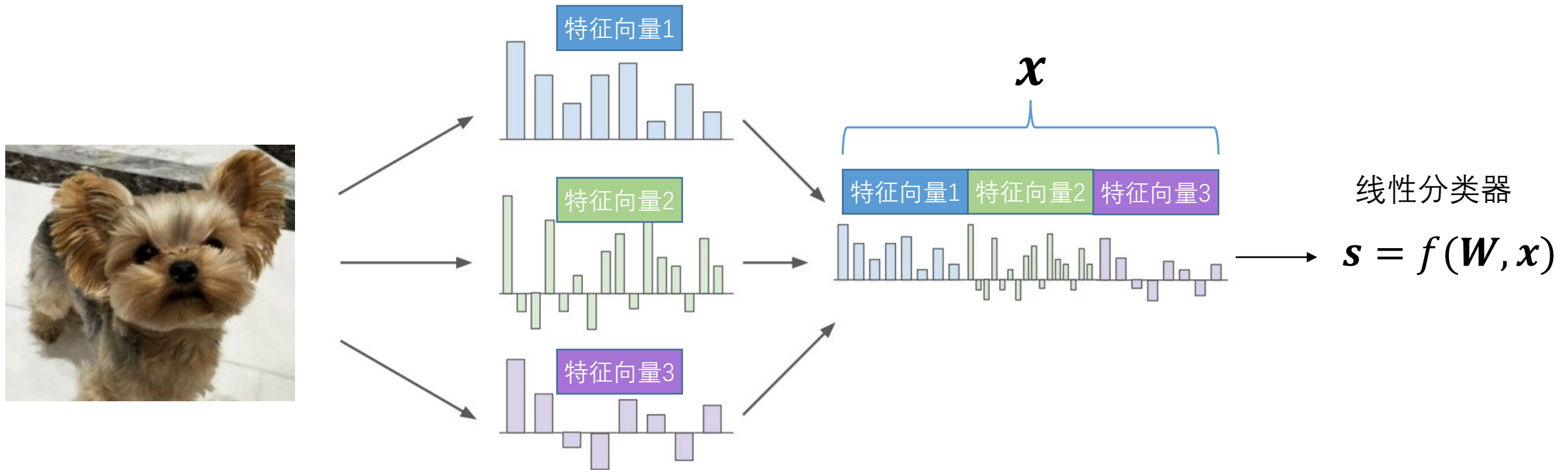
PatientId	Age	Gender	X	ASA	RF
1	45	1	True	True	True
2	50	2	False	True	False
3	45	1	False	True	True
4	59	2	False	True	False
5	22	2	True	False	True
...					

抽取特定的特征，生成结构化数据

feature
engineering

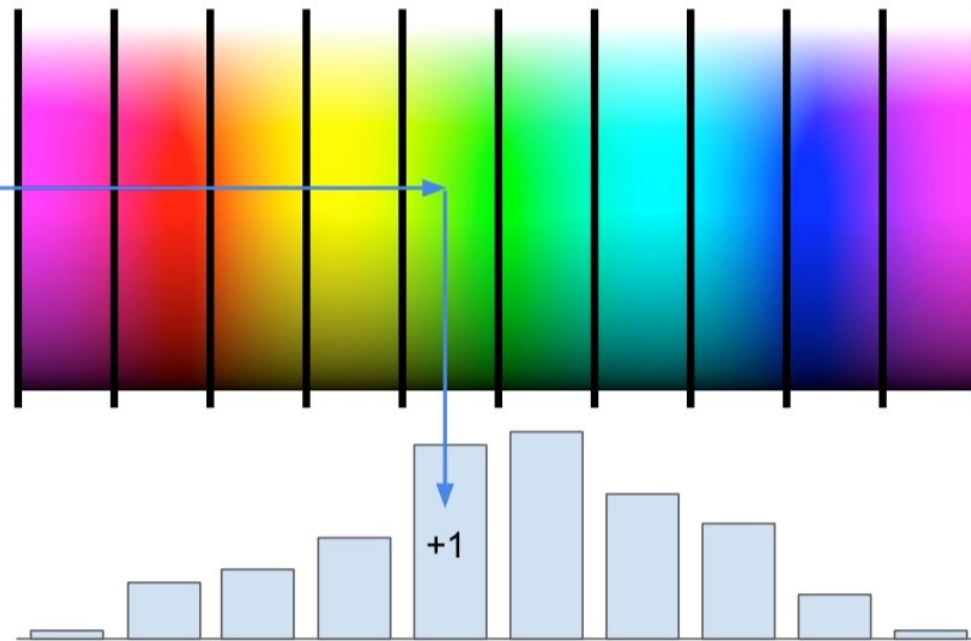
线性分类器应用于图像，往往也需要对原始像素做特征抽取，利用抽取的特征训练模型，提高预测性能

图像特征抽取



- ✓ Color Histogram
- ✓ Histogram of Oriented Gradients (HoG)

Color Histogram (Hue Histogram)

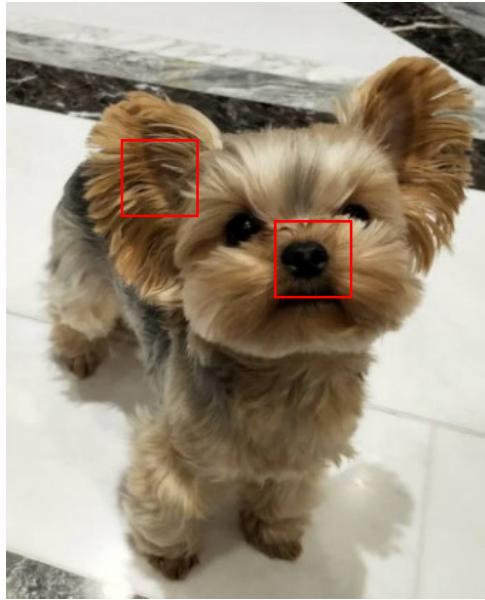


1、建立色相哈希表

2、哈希每个像素值，并计算每个key中像素的个数

3、将哈希结果作为模型输入

Histogram of Oriented Gradients (HoG)



HoG



$$\text{len}(\mathbf{x})=1152 \longrightarrow f(\mathbf{W}, \mathbf{x})$$

将图像切割成 8×8 的小区域，计算每个区域内9个（梯度）方向上边的条数，即每个区域生成9个数值

假设图像为 128×64 ，则切分成 $16 \times 8 = 128$ 个区域，生成的feature长度为 $128 \times 9 = 1152$

优化

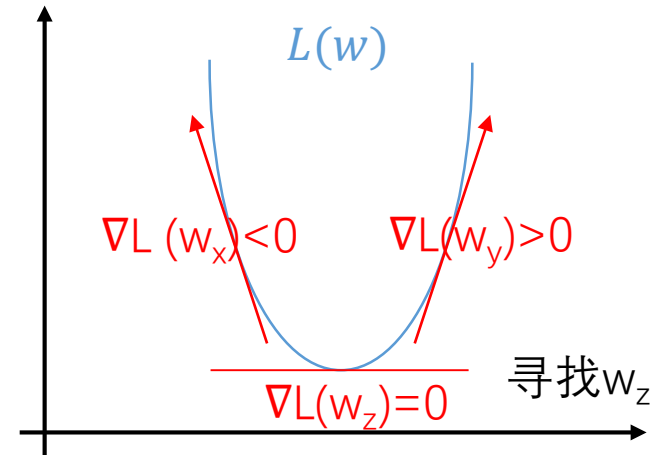
- 目标：最小化损失函数 $L(\mathbf{W})$

- 假设只有一个权重 w

- ✓ 导数 $\frac{dL(w)}{dw} = \lim_{h \rightarrow 0} \frac{L(w+h)-L(w)}{h}$ 代表 L 在 w 的切线斜率，即 $L(w)$ 在该点的变化速率和方向
- ✓ 沿反方向微调 w 即可减小 $L(w)$

- 多维情况下，即 \mathbf{W} 为向量

- ✓ 偏导数 $[\frac{\partial L(\mathbf{W})}{\partial w_1}, \frac{\partial L(\mathbf{W})}{\partial w_2}, \dots, \frac{\partial L(\mathbf{W})}{\partial w_n}]$ 代表 L 在 \mathbf{W} 处沿每个维度的变化速率和方向，称为梯度(gradient)，记为 $\nabla_{\mathbf{W}}L$ 或 $grad(L(\mathbf{W}))$
- ✓ $\nabla_{\mathbf{W}}L$ 和方向向量 \mathbf{v} 的点积即为该方向的斜率（方向导数）
- ✓ 负梯度 $-\nabla_{\mathbf{W}}L$ 的方向即为 L 在 \mathbf{W} 处下降最快的方向
- ✓ 沿 $-\nabla_{\mathbf{W}}L$ 方向微调 \mathbf{W} 即可快速减小 $L(\mathbf{W})$ （梯度下降）



$L(\mathbf{W})$ 沿任意方向 \mathbf{v} 的斜率：
 $\nabla_{\mathbf{W}}L \cdot \mathbf{v} = |\nabla_{\mathbf{W}}L| |\mathbf{v}| \cos \theta$

当 $\cos \theta = 1$ 的时候达到最大值，即方向 \mathbf{v} 和 $\nabla_{\mathbf{W}}L$ 同方向，所以负梯度 $-\nabla_{\mathbf{W}}L$ 代表 L 下降最快的方向（最陡峭）

梯度下降：

$\mathbf{W}_{new} = \mathbf{W} - \lambda \nabla_{\mathbf{W}}L$
 超参数 λ 为step size或learning rate

优化

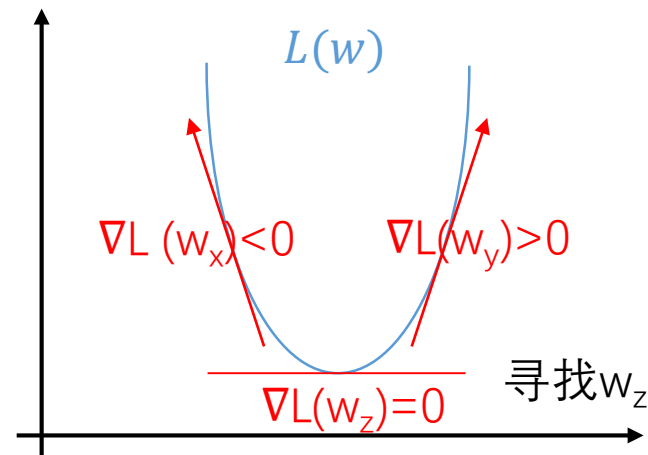
- 目标：最小化损失函数 $L(\mathbf{W})$

- 假设只有一个权重 w

- ✓ 导数 $\frac{dL(w)}{dw} = \lim_{h \rightarrow 0} \frac{L(w+h) - L(w)}{h}$ 代表 L 在 w 的切线斜率，即 $L(w)$ 在该点的变化速率和方向
- ✓ 沿反方向微调 w 即可减小 $L(w)$

- 多维情况下，即 \mathbf{W} 为向量

- ✓ 偏导数 $[\frac{\partial L(\mathbf{W})}{\partial w_1}, \frac{\partial L(\mathbf{W})}{\partial w_2}, \dots, \frac{\partial L(\mathbf{W})}{\partial w_n}]$ 代表 L 在 \mathbf{W} 处沿每个维度的变化速率和方向，称为梯度(gradient)，记为 $\nabla_{\mathbf{W}}L$ 或 $grad(L(\mathbf{W}))$
- ✓ $\nabla_{\mathbf{W}}L$ 和方向向量 \mathbf{v} 的点积即为该方向的斜率（方向导数）
- ✓ 负梯度 $-\nabla_{\mathbf{W}}L$ 的方向即为 L 在 \mathbf{W} 处下降最快的方向
- ✓ 沿 $-\nabla_{\mathbf{W}}L$ 方向微调 \mathbf{W} 即可快速减小 $L(\mathbf{W})$ （梯度下降）



$L(\mathbf{W})$ 沿任意方向 \mathbf{v} 的斜率：
 $\nabla_{\mathbf{W}}L \cdot \mathbf{v} = |\nabla_{\mathbf{W}}L| |\mathbf{v}| \cos \theta$

当 $\cos \theta = 1$ 的时候达到最大值，即方向 \mathbf{v} 和 $\nabla_{\mathbf{W}}L$ 同方向，所以负梯度 $-\nabla_{\mathbf{W}}L$ 代表 L 下降最快的方向（最陡峭）

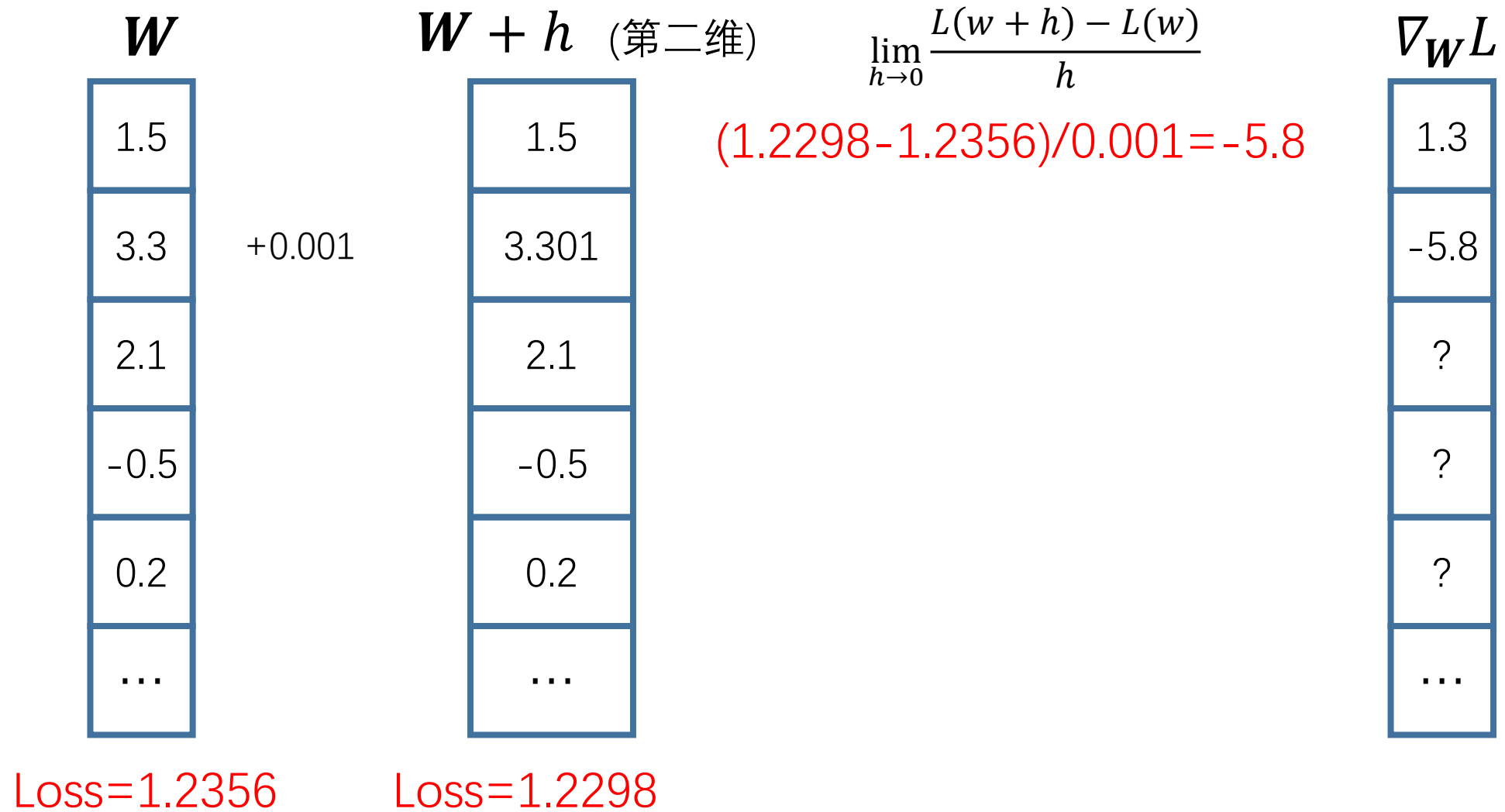
如何计算梯度 $\nabla_{\mathbf{W}}L$

$\mathbf{w}_{new} = \mathbf{w} - \lambda \nabla_{\mathbf{W}}L$
 超参数 λ 为step size或learning rate

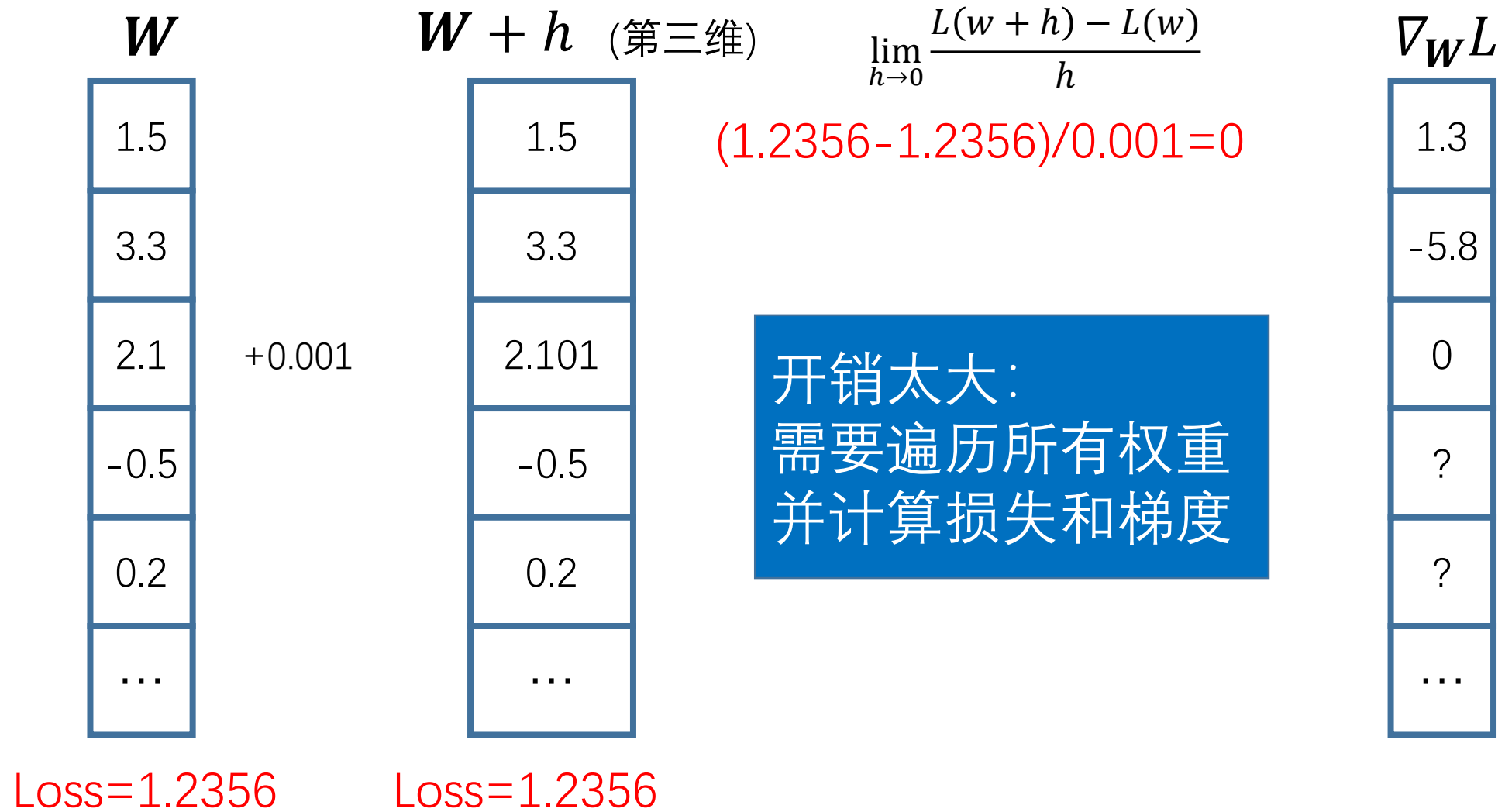
数值梯度 (Numerical Gradient)

W		$W + h$ (第一维)	$\lim_{h \rightarrow 0} \frac{L(w + h) - L(w)}{h}$	$\nabla_W L$
1.5	+0.001	1.501	$(1.2369 - 1.2356) / 0.001 = 1.3$	1.3
3.3		3.3		?
2.1		2.1		?
-0.5		-0.5		?
0.2		0.2		?
...	
Loss=1.2356		Loss=1.2369		

数值梯度 (Numerical Gradient)



数值梯度 (Numerical Gradient)



解析梯度 (Analytic Gradient)

- 损失函数是关于 \mathbf{W} 的函数

$$\mathbf{s} = f(\mathbf{W}, \mathbf{x})$$

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

$$\text{Cross-entropy loss : } L_i = -\log P(y_i | x_i) = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

$$\text{损失函数: } L = \frac{1}{N} \sum_{i=1}^N L_i + \lambda R(\mathbf{W})$$

可以使用数值梯度的结果检验解析梯度是否正确，即所谓gradient check

- 对损失函数求偏导，编写梯度公式，直接计算梯度

$$\text{梯度: } \nabla_{\mathbf{W}} L = \left[\frac{\partial L(\mathbf{W})}{\partial w_1}, \frac{\partial L(\mathbf{W})}{\partial w_2}, \dots, \frac{\partial L(\mathbf{W})}{\partial w_n} \right]$$

解析梯度 (Analytic Gradient)

Hinge loss和Cross-entropy loss梯度的推导

请自行推导

第一次作业要求编写解析梯度！！！！

溜了溜了

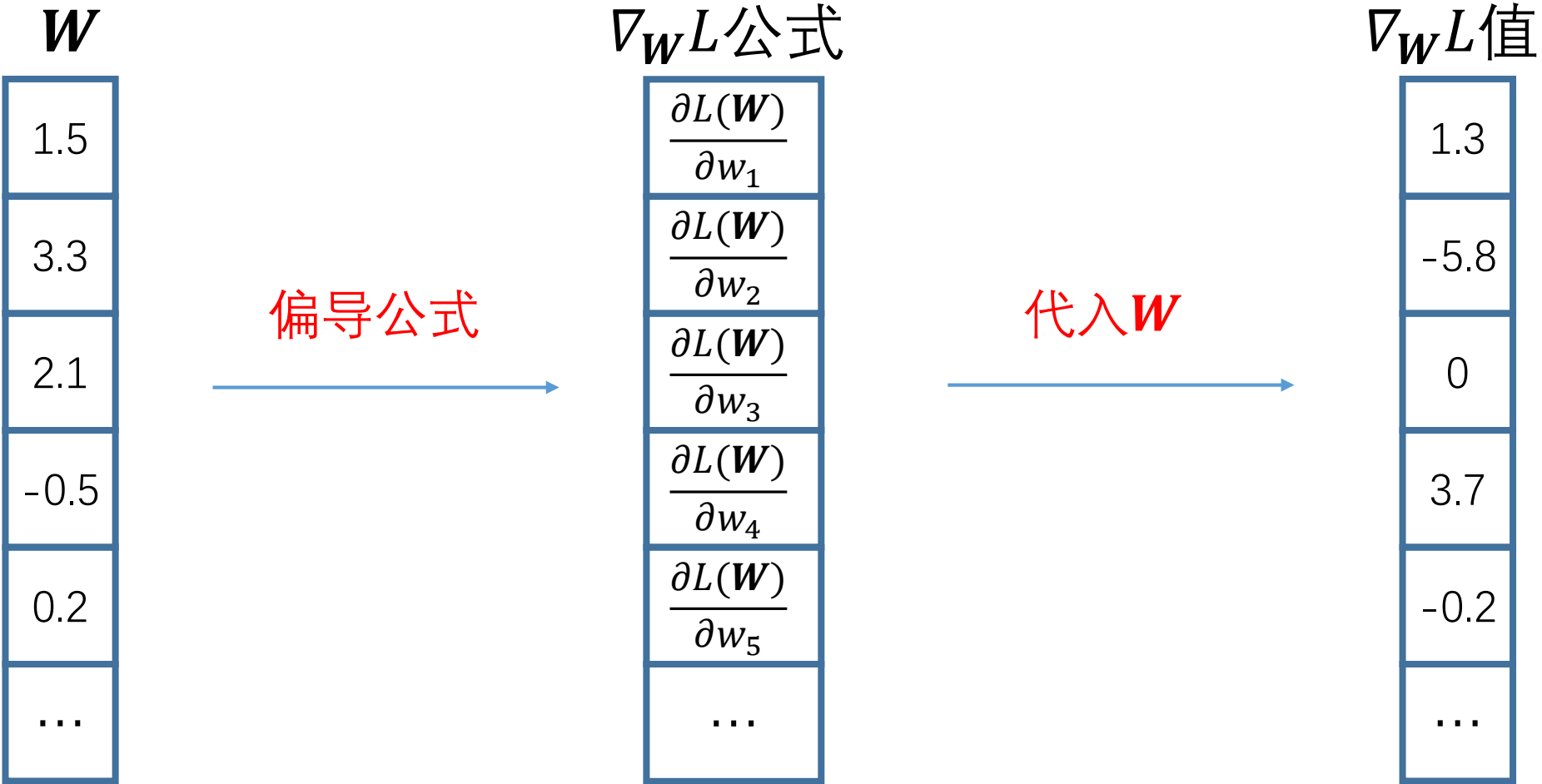


!



刺不刺激

解析梯度 (Analytic Gradient)



梯度下降 vs 随机梯度下降

- 梯度下降：每次更新 W 需要遍历所有数据！迭代效率低！
- 随机梯度下降（Stochastic Gradient Descent, SGD）
 - ✓ 每次选取一个sample集（minibatch，大小一般为32/64/128/256）
 - ✓ 利用在sample集上的损失计算近似梯度

```
# Vanilla Minibatch Gradient Descent
```

```
while True:
```

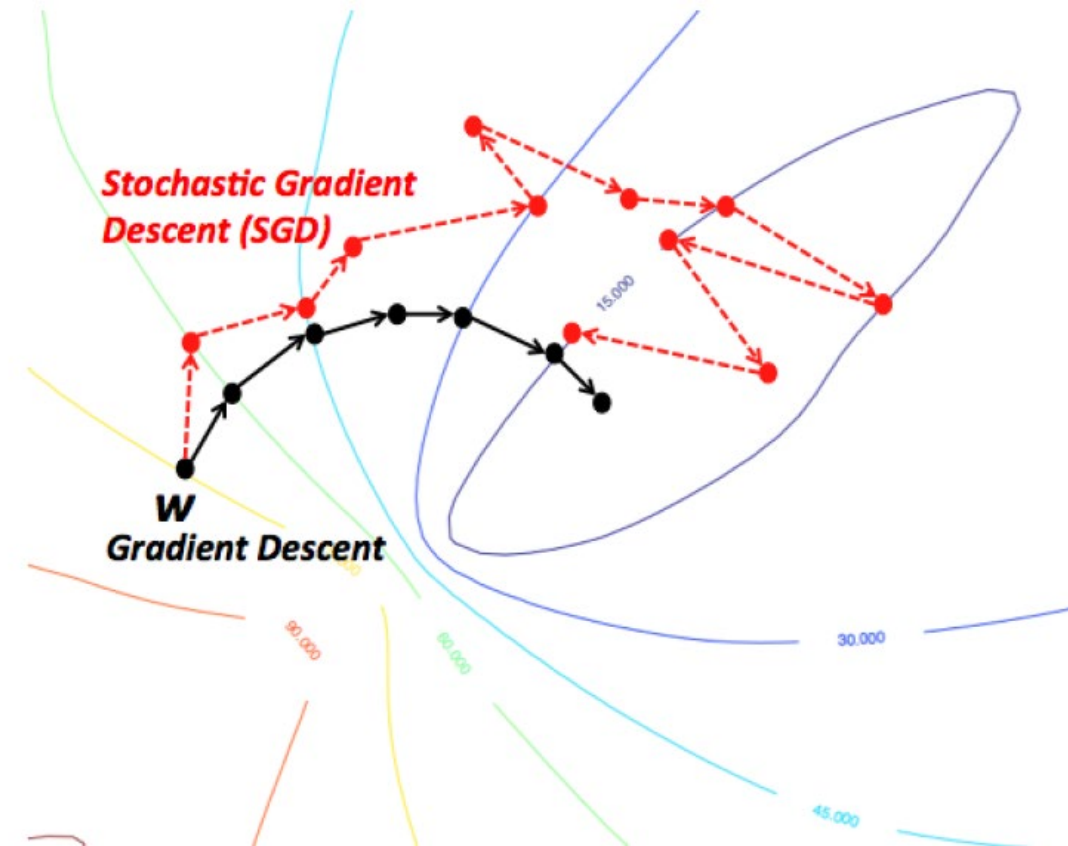
```
    data_batch = sample_training_data(data, 256) # sample 256 examples  
    weights_grad = evaluate_gradient(loss_fun, data_batch, weights)  
    weights += - step_size * weights_grad # perform parameter update
```

随机选取256个图片，并计算梯度

梯度下降

GD Vs. SGD

- Gradient descent
 - ✓优势：每次迭代loss下降快
 - ✓劣势：一次迭代需要遍历所有数据，并且容易陷入local minima
- Stochastic gradient descent
 - ✓优势：迭代更新速度快，并且往往因为minibatch含有噪声而避开local minima
 - ✓劣势：每次迭代loss下降较慢



由于数据量较大，训练深度神经网络基本都使用SGD，以及其他性能更佳的优化方法（L07神经网络训练2）

小结

- 损失函数
 - ✓ multiclass SVM loss: SVM分类器
 - ✓ Cross-entropy loss: Softmax分类器
 - ✓ L1和L2正则化

- 图像特征抽取

- 优化
 - ✓ 梯度计算、梯度下降、随机梯度下降

L04

- 神经网络
- 优化：反向传播