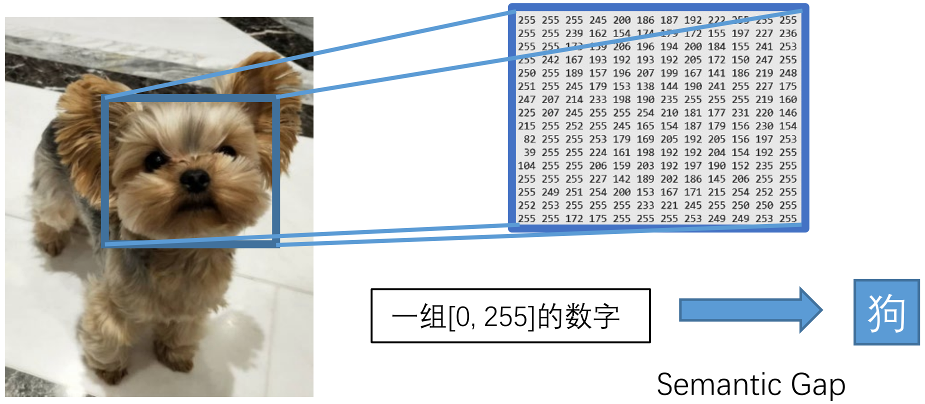
计算机视觉
Computer Vision

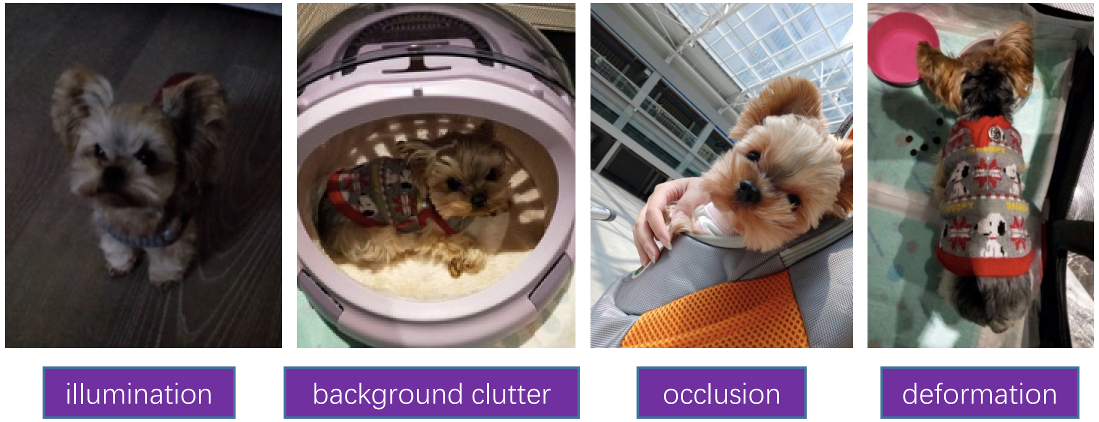
Lecture 3: 损失函数和优化

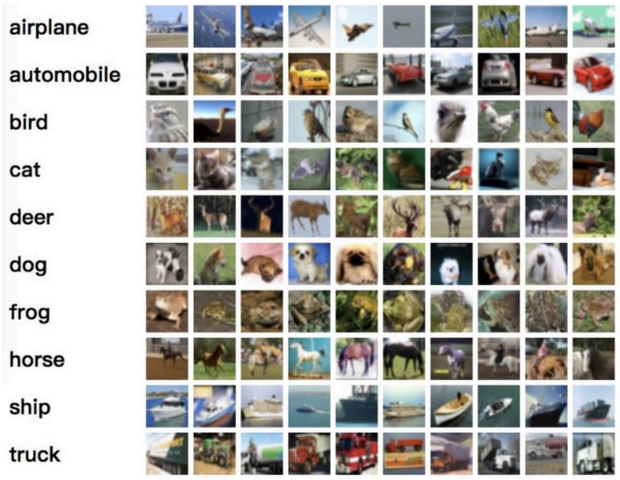
第一次作业

- KNN， SVM/softmax分类器， 2-layer NN， high-level feature
- Deadline: 10月18日
- Submission：
  - ✓生成5个pdf
  - ✓https://workspace.jianguoyun.com/inbox/collect/37884464eac74579a9da4be98d532b0b/submit

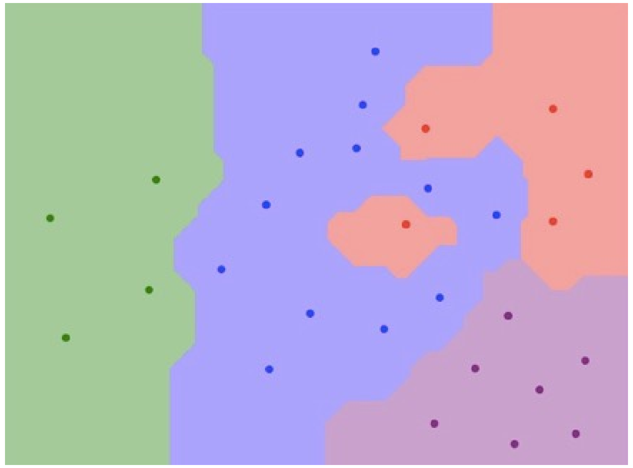
L02-图像分类



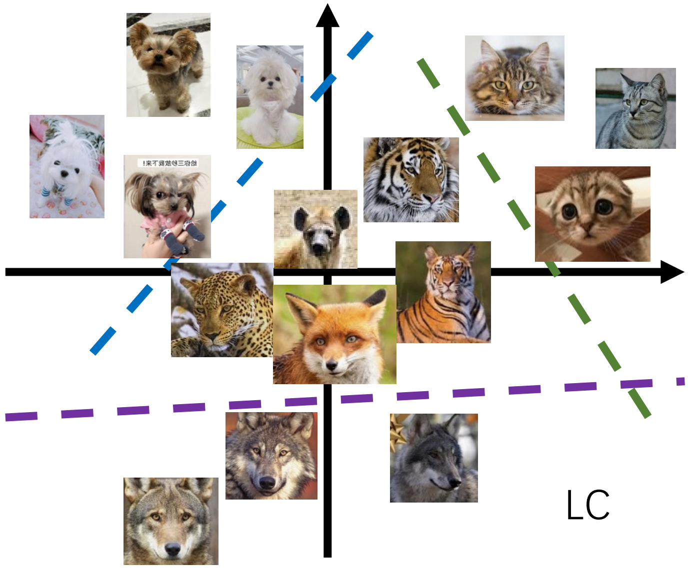




CIFAR-10



kNN



线性分类器的参数

$$\underbrace{\overset{?}{\begin{bmatrix} 1.5 & 1.3 & 2.1 & 0.0 \\ 0.2 & -0.5 & 0.1 & 2.0 \\ 0.0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}}_{\mathbf{W}} \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}_{\mathbf{x}} + \underbrace{\overset{?}{\begin{bmatrix} 3.2 \\ 1.1 \\ -1.2 \end{bmatrix}}}_{\mathbf{W}_0} = \begin{bmatrix} 437.9 \\ -96.8 \\ 61.95 \end{bmatrix} \begin{matrix} \text{狗分数} \\ \text{猫分数} \\ \text{狼分数} \end{matrix}$$

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

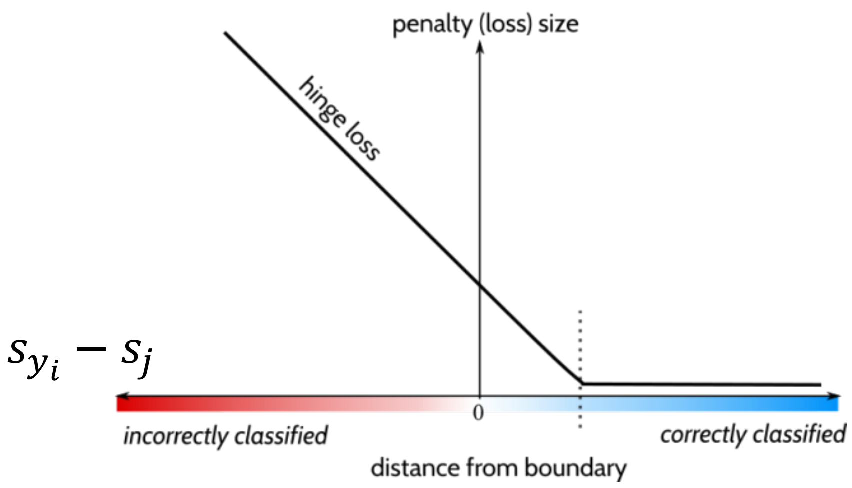
损失函数

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

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

- ✓ 训练集 $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$
  - ➢ $\boldsymbol{x}_i$为第i个图片， $y_i \in Z$为它的类别标签
  - ➢ $f(\boldsymbol{W}, \boldsymbol{x}_i)$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$L_i = l(f(\boldsymbol{W}, \boldsymbol{x}_i), y_i)$
  - ➢ 训练样本总体损失$L = \frac{1}{N}\sum_{i=1}^{N} L_i$

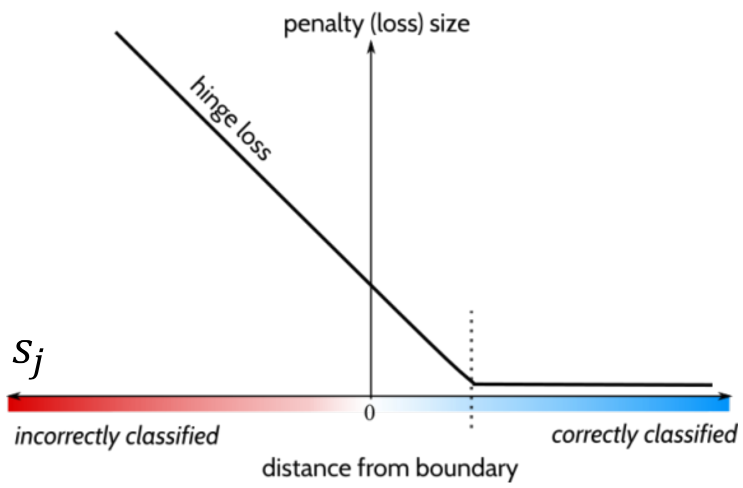
线性分类器常用损失函数



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

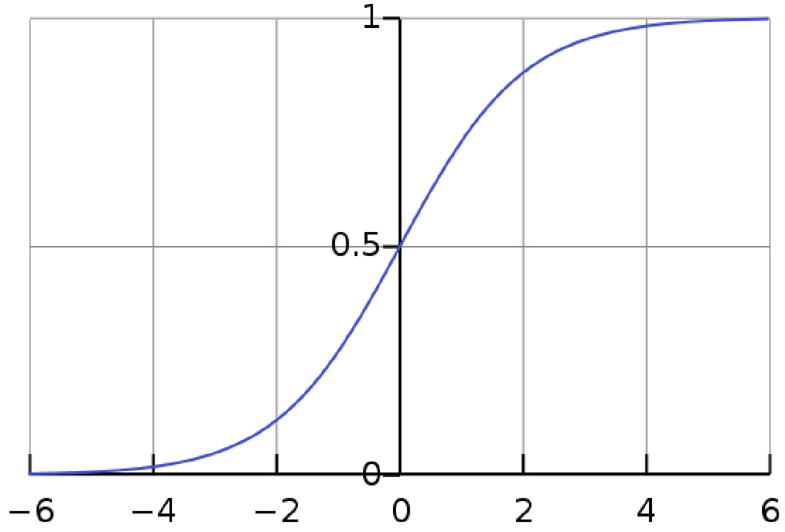
线性分类器常用损失函数

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

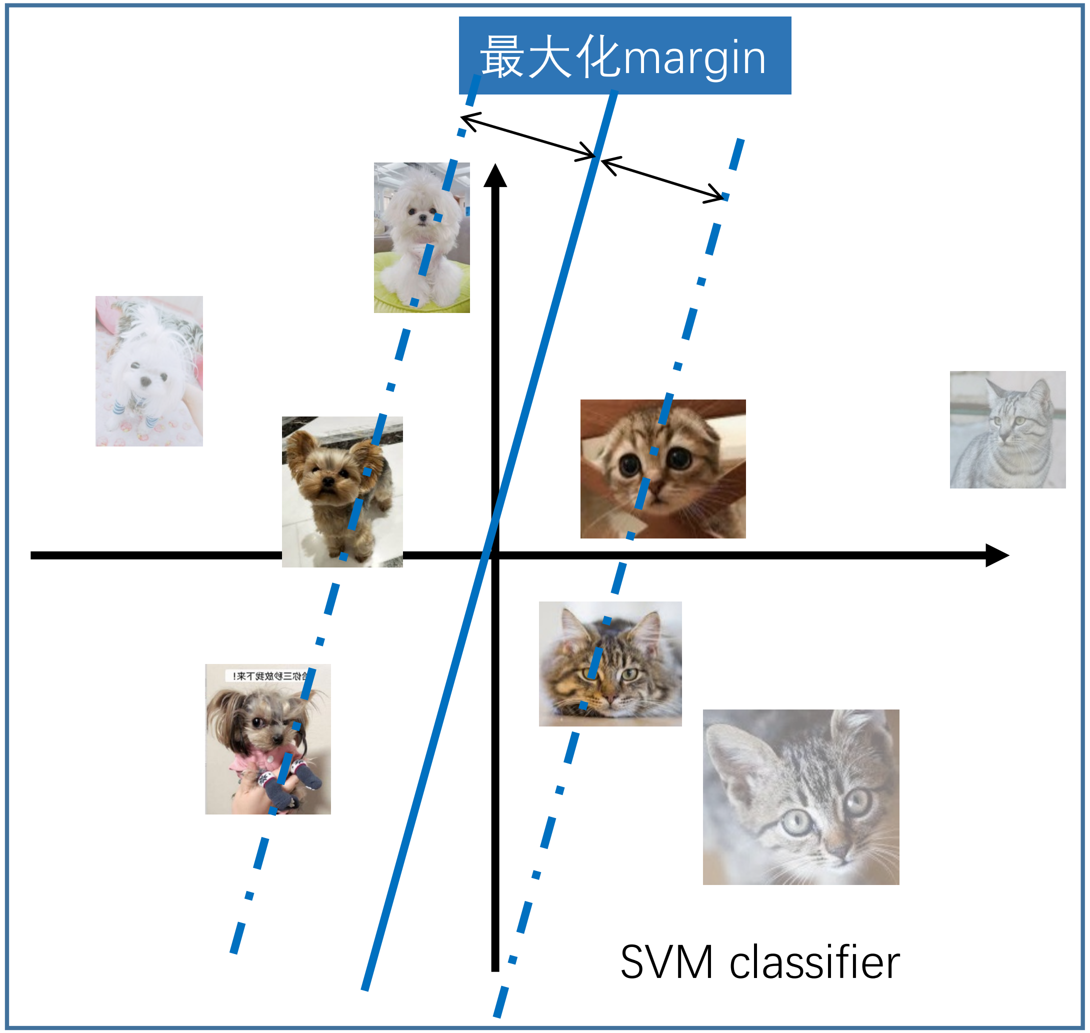


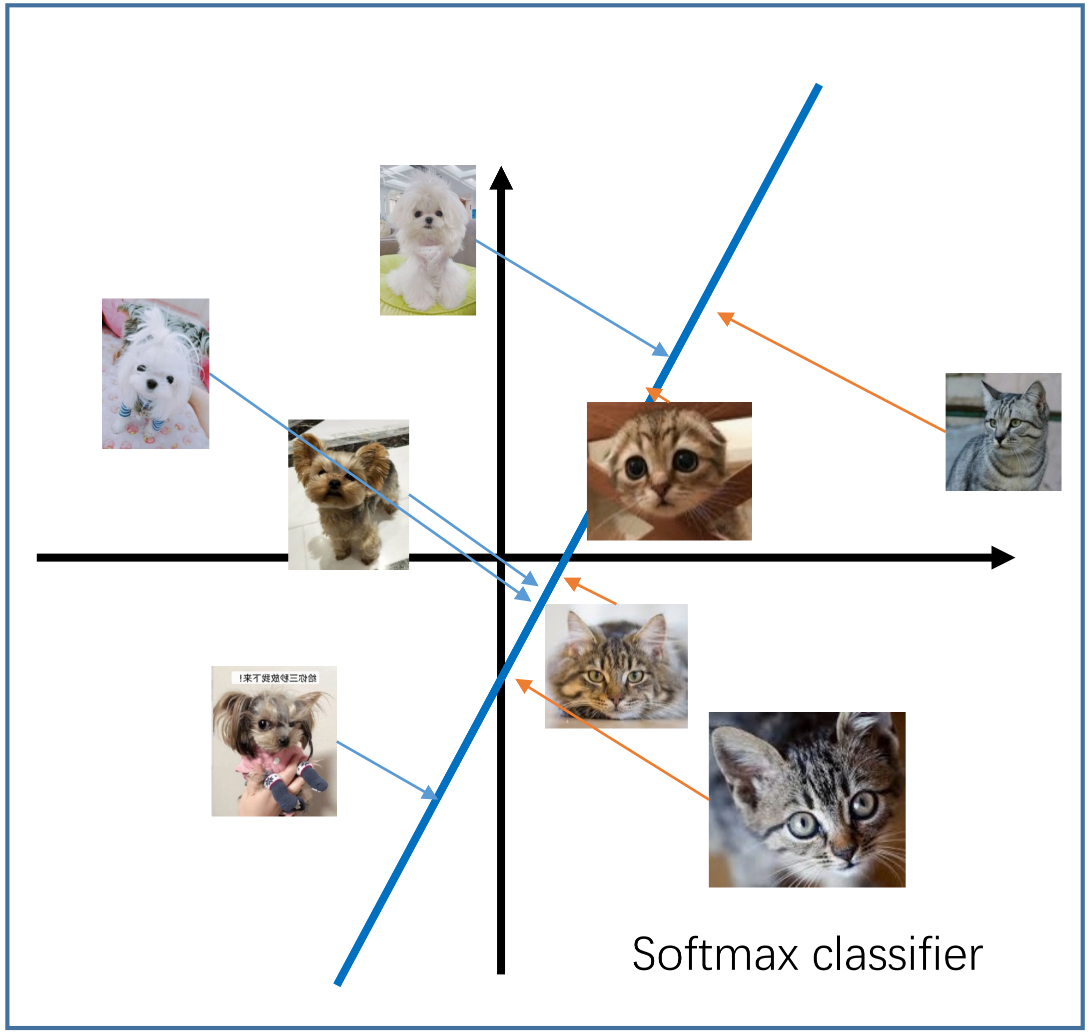
- multiclass SVM loss (Hinge loss)： $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$
  ✓SVM分类器

- cross-entropy loss： $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}})$
  ✓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:

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

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:

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

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:

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

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:

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

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$\boldsymbol{W}$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$\max(0, s_j - s_{y_i} + 2)$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参数的作用。

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$\boldsymbol{W}$使得$L=0$（完美），请问$\boldsymbol{W}$是否唯一？

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

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:

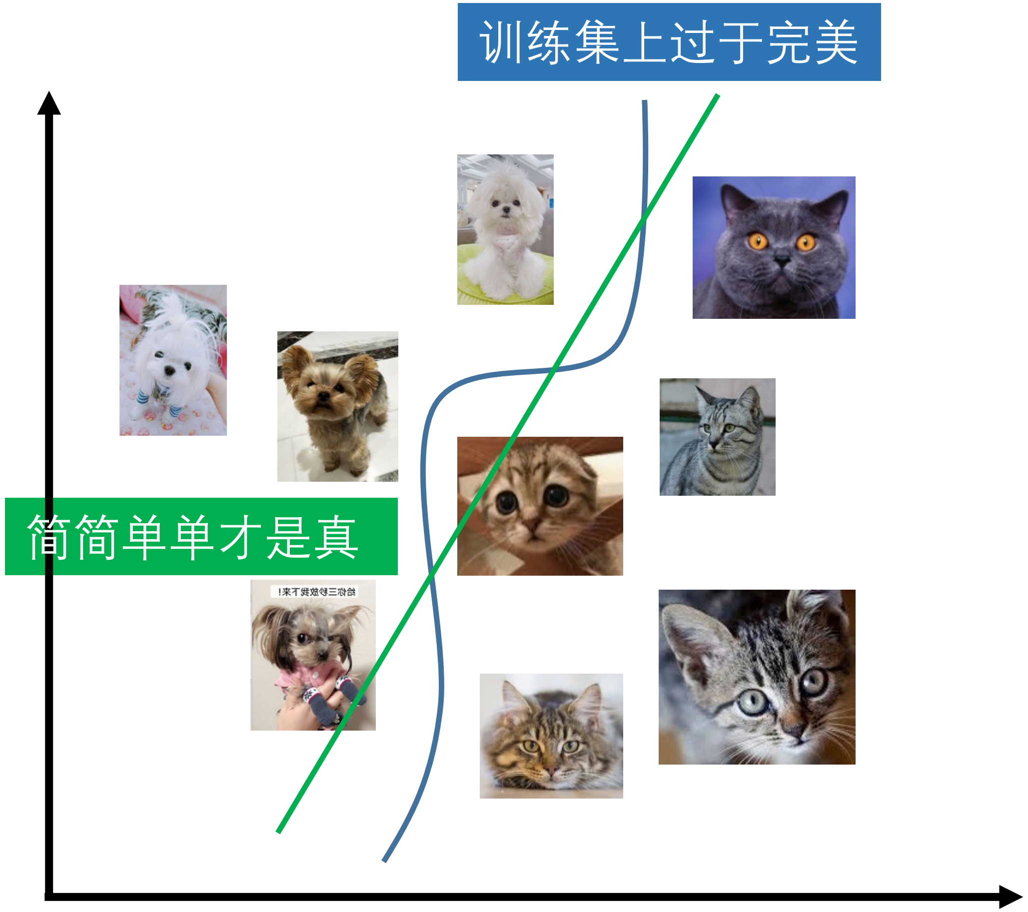
$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

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

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

选取更好的$\boldsymbol{W}$：添加正则项！

正则化



将复杂模型简单化

正则化

正则项
✓ 防止模型过度拟合训练集

$$L(\boldsymbol{W}) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} l(f(\boldsymbol{W}, \boldsymbol{x}_i), y_i)} + \lambda \overbrace{\boxed{R(\boldsymbol{W})}}$$

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

超参数，正则化强度

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

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

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

超参数，正则化强度

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

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

数据损失
✓ 使得模型尽可能拟合训练集

添加正则项的意义：
✓ 缩小参数空间
✓ 调整参数偏好的分布
✓ 提高模型泛化能力

正则化：L1 Vs. L2

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

$$\boldsymbol{W_1} = [1, 0, 0]$$

$$\boldsymbol{W_2} = [0.3, -0.1, 0.8]$$

$$\boldsymbol{W_1^T}x = \boldsymbol{W_2^T}x = 1$$

数据损失也相同

正则项

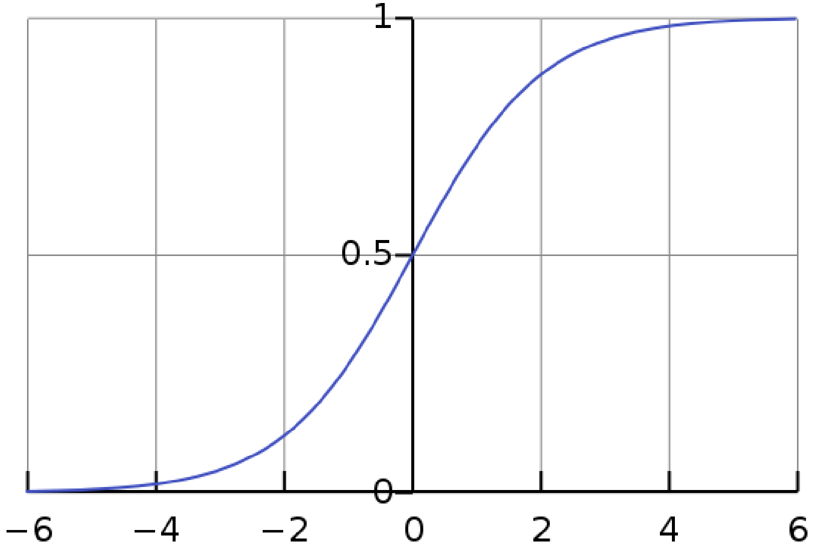
L1: $\sum_k \sum_l |W_1| = 1, \sum_k \sum_l |W_2| = 1.2$

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

- ✓ L1偏向于使参数集中在少数输入像素上
- ✓ L2偏向于使参数分布在所有像素上

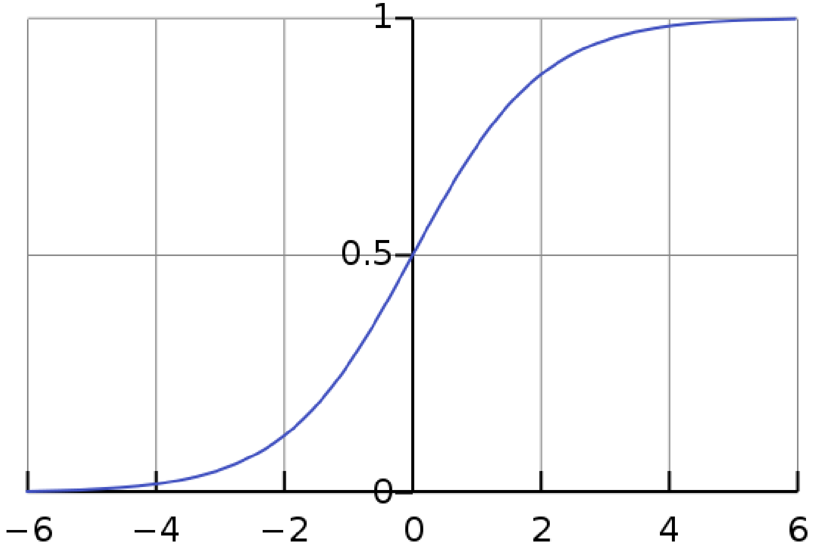
Softmax分类器

- 将分数通过softmax函数转化为概率
  - ✓ $\boldsymbol{s} = f(\boldsymbol{W}, \boldsymbol{x}_i)$
  - ✓ $P(y = k|x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$

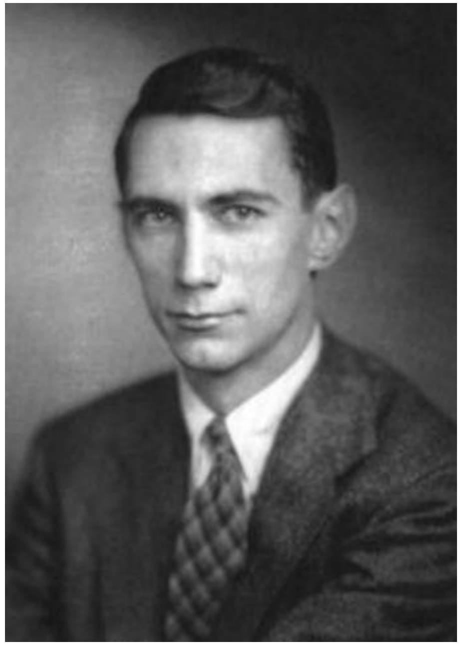


Softmax分类器

- 将分数通过softmax函数转化为概率
  - ✓$\boldsymbol{s} = f(\boldsymbol{W}, \boldsymbol{x}_i)$
  - ✓$P(y = k | x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$



- 利用交叉熵（cross-entropy）计算第i个图片的损失
  - ✓$L_i = -\log P(y = y_i | x_i) = -\log(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}})$
  - ✓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%

- Entropy：衡量概率分布Q的不确定性
  - ✓ $H(Q) = -\sum_i q_i \log q_i$

H(真实)=-(0.5×log(0.5)+ 0.3×log(0.3)+ 0.2×log(0.2))
=0.45

交叉熵（cross-entropy）

真实天气： 50% 30% 20%

H(真实)=-(0.5×log(0.5)+ 0.3×log(0.3)+ 0.2×log(0.2))
=0.45

- Entropy：衡量概率分布Q的不确定性
  - ✓ $H(Q) = -\sum_i q_i \log q_i$

天气预报： 40% 30% 30%

H(真实, 预报)=-(0.5×log(0.4)+ 0.3×log(0.3)+ 0.2×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
  - ✓ $H(Q,P) = -\sum_i q_i \log p_i$

交叉熵（cross-entropy）

真实天气：50% 30% 20%

- Entropy：衡量概率分布Q的不确定性
  - ✓ $H(Q) = -\sum_i q_i \log q_i$

H(真实)=-(0.5×log(0.5)+ 0.3×log(0.3)+ 0.2×log(0.2))
=0.45

天气预报：40% 30% 30%

- Cross-entropy：衡量概率分布P服从概率分布Q的不确定性
  - ✓ $\boxed{H(Q,P) = -\sum_i q_i \log p_i}$
  - ✓ $D_{KL}(Q||P) = H(Q,P) - H(Q)$ KL散度（相对熵）：衡量分布Q和P的差异性

H(真实, 预报)=-(0.5×log(0.4)+ 0.3×log(0.3)+ 0.2×log(0.3))
=0.46

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	0.89	
猫分数	5.8	$\xrightarrow{\text{exp}}$ 330.3	$\xrightarrow{\text{归一化}}$ 0.11	$H(Q,P)$
狼分数	-1.9	0.15	0.00	

预测分布P

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		0.89
猫分数	5.8	$\xrightarrow{\text{exp}}$	330.3	$\xrightarrow{\text{归一化}}$	0.11
狼分数	-1.9		0.15		0.00

$H(Q,P)$

预测分布P	真实分布Q
0.89	1
0.11	0
0.00	0

H(Q, P)=-

(1×log(0.89)

+

0×log(0.11)

+

0×log(0.00))

=**-log(0.89)**

Softmax分类器

第i个图片的损失为：

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

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

狗分数	7.9		2697.3		0.89	1
猫分数	5.8	$\xrightarrow{\text{exp}}$	330.3	$\xrightarrow{\text{归一化}}$	0.11	0
狼分数	-1.9		0.15		0.00	0
					预测分布P	真实分布Q

$H(Q, P)$

H(Q, P)=-

(1×log(0.89)

+

0×log(0.11)

+

0×log(0.00))

=**-log(0.89)**

Softmax分类器

第i个图片的损失为：

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

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

狗分数	7.9		2697.3		0.89	1
猫分数	5.8	$\xrightarrow{\text{exp}}$	330.3	$\xrightarrow{\text{归一化}}$	0.11	0
狼分数	-1.9		0.15		0.00	0
					预测分布P	真实分布Q

$H(Q, P)$

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，损失函数越小

注意和SVM loss的区别

Softmax分类器

第i个图片的损失为:

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

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

		exp		归一化	预测分布P	$H(Q,P)$	真实分布Q
狗分数	7.9		2697.3		0.89		1
猫分数	5.8	→	330.3	→	0.11		0
狼分数	-1.9		0.15		0.00		0

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

第i个图片的损失为：

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

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

		exp		归一化		$H(Q,P)$	
狗分数	7.9		2697.3		0.89		1
猫分数	5.8	→	330.3	→	0.11		0
狼分数	-1.9		0.15		0.00		0
					预测分布P		真实分布Q

Q：假如初始化$\boldsymbol{W}$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log3≈ 0.477

可用于代码正确性检查

损失函数计算流程

✓ 训练集 $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$

- $\boldsymbol{x}_i$为第i个图片， $y_i \in Z$为它的类别标签
- $\boldsymbol{s} = f(\boldsymbol{W}, \boldsymbol{x}_i)$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： $L_i = -\log P(y_i | x_i) = -\log(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}})$

Softmax分类器

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

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

7.9	3.3	-1.1	2.3
5.8	-0.8	3.4	1.5
-1.9	6.5	2.5	4.4

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

Total Loss

损失函数计算流程

✓ 训练集 $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$
- $\boldsymbol{x}_i$为第i个图片， $y_i \in Z$为它的类别标签
- $\boldsymbol{s} = f(\boldsymbol{W}, \boldsymbol{x}_i)$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： $L_i = -\log P(y_i|x_i) = -\log(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}})$

Softmax分类器

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

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

7.9	3.3	-1.1	2.3
5.8	-0.8	3.4	1.5
-1.9	6.5	2.5	4.4

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

Total Loss

需要最小化

优化$\boldsymbol{W}$

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

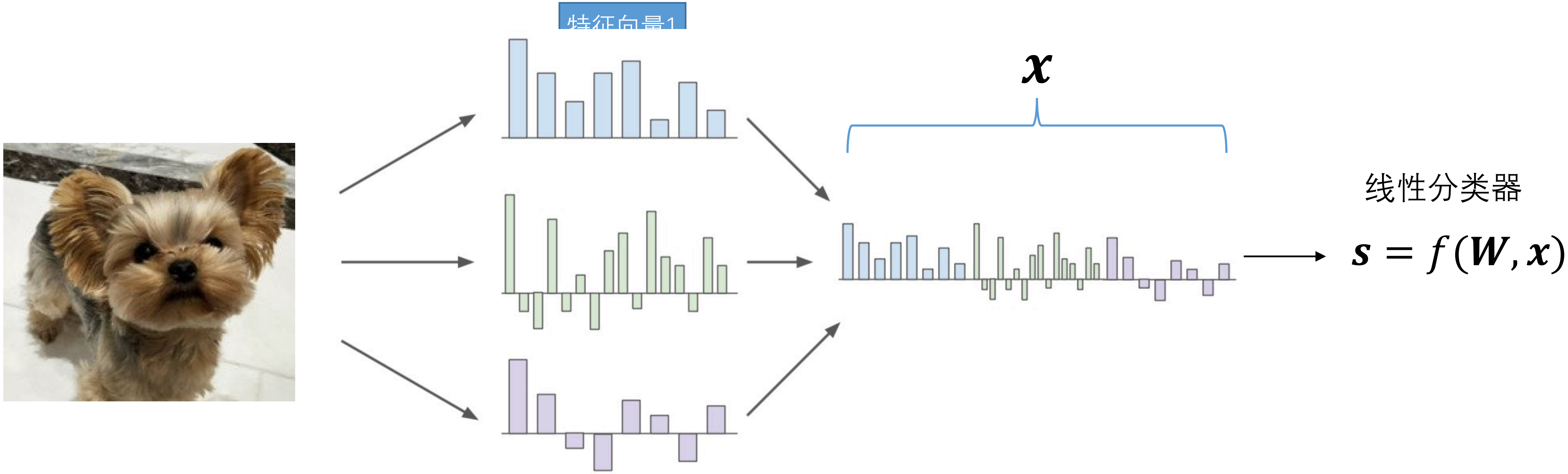
PatientId	Age	Gender	X	ASA	RF
1	45	1	True	True	True
2	50	2	False	True	False
3	45	1	False	True	True
4	59	2	False	True	False
5	22	2	True	False	True
...					

抽取特定的特征，生成结构化数据

feature engineering

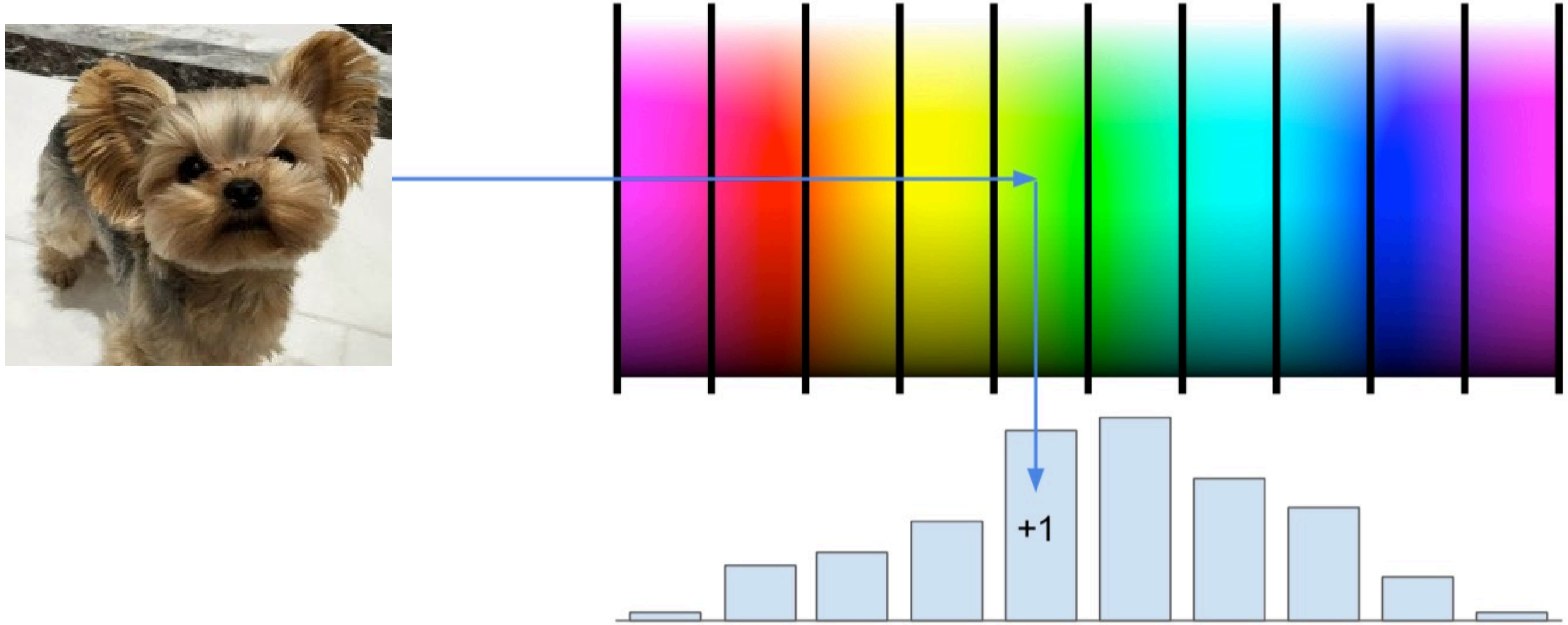
线性分类器应用于图像，往往也需要对原始像素做特征抽取，利用抽取的特征训练模型，提高预测性能

图像特征抽取



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

Color Histogram（Hue Histogram）

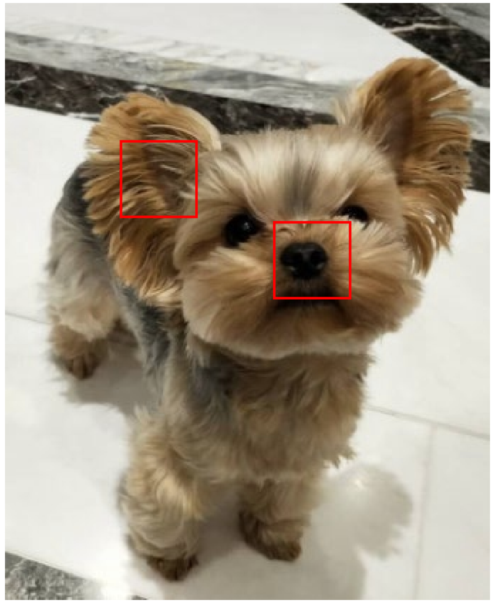


1、建立色相哈希表

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 $\longrightarrow$

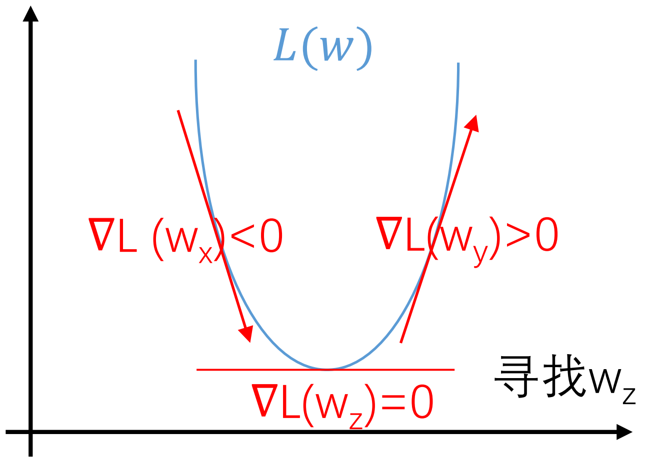
$\text{len}(\boldsymbol{x})=1152$ $\longrightarrow$ $f(\boldsymbol{W},\boldsymbol{x})$

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16×8=128个区域，生成的feature长度为128×9=1152

优化

- 目标：最小化损失函数$L(\boldsymbol{W})$
- 假设只有一个参数w
  - ✓ 导数$\frac{dL(w)}{dw} = \lim_{h\to 0}\frac{L(w+h)-L(w)}{h}$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$L(w)$

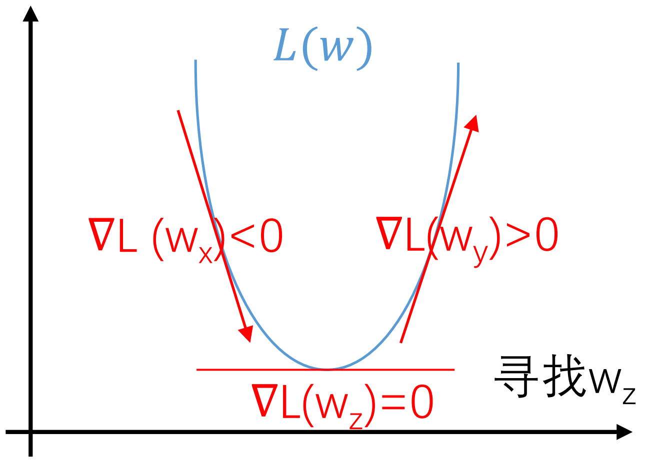


优化

- 目标：最小化损失函数$L(\boldsymbol{W})$

- 假设只有一个参数w
  - ✓ 导数$\frac{dL(w)}{dw} = \lim_{h\to 0}\frac{L(w+h)-L(w)}{h}$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$\boldsymbol{W}$为向量
  - ✓ 偏导数$[\frac{\partial L(\boldsymbol{W})}{\partial w_1}, \frac{\partial L(\boldsymbol{W})}{\partial w_2}, \ldots, \frac{\partial L(\boldsymbol{W})}{\partial w_n}]$代表$L$在$\boldsymbol{W}$处沿每个维度的变化速率和方向，称为梯度(gradient)，记为$\nabla_{\boldsymbol{W}} L$或$grad(L(\boldsymbol{W}))$
  - ✓ $\nabla_{\boldsymbol{W}} L$ 和方向向量 $\boldsymbol{v}$ 的点积即为该方向的斜率（方向导数）



$L(\boldsymbol{W})$沿任意方向$\boldsymbol{v}$的斜率：

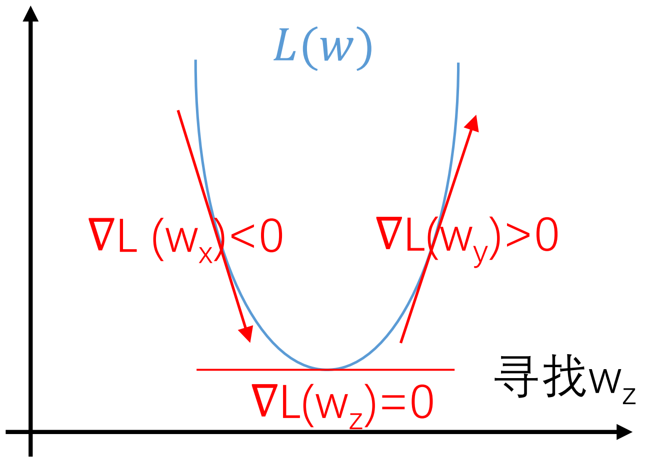
$$\nabla_{\boldsymbol{W}} L \cdot v = |\nabla_{\boldsymbol{W}} L||v| \cos\theta$$

优化

- 目标：最小化损失函数$L(\boldsymbol{W})$

- 假设只有一个参数w
  - ✓ 导数$\frac{dL(w)}{dw} = \lim_{h\to 0}\frac{L(w+h)-L(w)}{h}$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$\boldsymbol{W}$为向量
  - ✓ 偏导数$[\frac{\partial L(\boldsymbol{W})}{\partial w_1}, \frac{\partial L(\boldsymbol{W})}{\partial w_2}, \dots, \frac{\partial L(\boldsymbol{W})}{\partial w_n}]$代表$L$在$\boldsymbol{W}$处沿每个维度的变化速率和方向，称为梯度(gradient)，记为$\nabla_{\boldsymbol{W}} L$或$grad(L(\boldsymbol{W}))$
  - ✓ $\nabla_{\boldsymbol{W}} L$ 和方向向量 $\boldsymbol{v}$ 的点积即为该方向的斜率（方向导数）
  - ✓ 负梯度$-\nabla_{\boldsymbol{W}} L$的方向即为$L$在$\boldsymbol{W}$处下降最快的方向



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

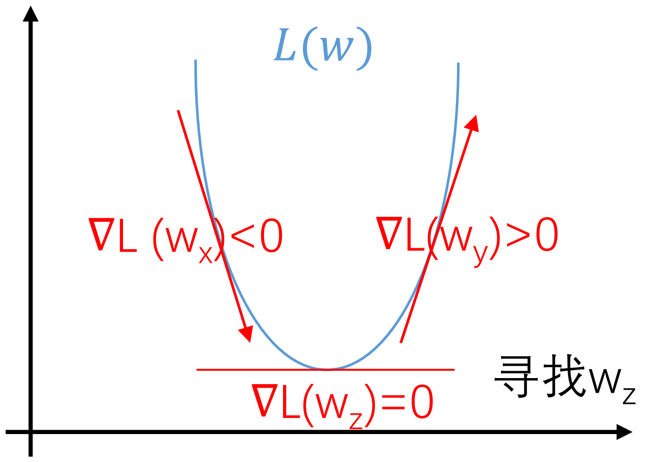
当$\cos\theta = 1$的时候达到最大值，即方向$\boldsymbol{v}$和$\nabla_{\boldsymbol{W}} L$同方向，所以负梯度$-\nabla_{\boldsymbol{W}} L$代表$L$下降最快的方向（最陡峭）

优化

- 目标：最小化损失函数$L(\boldsymbol{W})$

- 假设只有一个参数w
  - ✓ 导数$\frac{dL(w)}{dw} = \lim_{h\to 0} \frac{L(w+h)-L(w)}{h}$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$\boldsymbol{W}$为向量
  - ✓ 偏导数$[\frac{\partial L(\boldsymbol{W})}{\partial w_1}, \frac{\partial L(\boldsymbol{W})}{\partial w_2}, \dots, \frac{\partial L(\boldsymbol{W})}{\partial w_n}]$代表$L$在$\boldsymbol{W}$处沿每个维度的变化速率和方向，称为梯度(gradient)，记为$\nabla_{\boldsymbol{W}} L$或$grad(L(\boldsymbol{W}))$
  - ✓ $\nabla_{\boldsymbol{W}} L$ 和方向向量 $\boldsymbol{v}$ 的点积即为该方向的斜率（方向导数）
  - ✓ 负梯度$-\nabla_{\boldsymbol{W}} L$的方向即为$L$在$\boldsymbol{W}$处下降最快的方向
  - ✓ 沿$-\nabla_{\boldsymbol{W}} L$方向微调$\boldsymbol{W}$即可快速减小$L(\boldsymbol{W})$（梯度下降）



$L(\boldsymbol{W})$沿任意方向$\boldsymbol{v}$的斜率：

$$\nabla_{\boldsymbol{W}} L \cdot v = |\nabla_{\boldsymbol{W}} L||v| \cos\theta$$

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

梯度下降：

$$W_{new} = W - \lambda \nabla_{\boldsymbol{W}} L$$

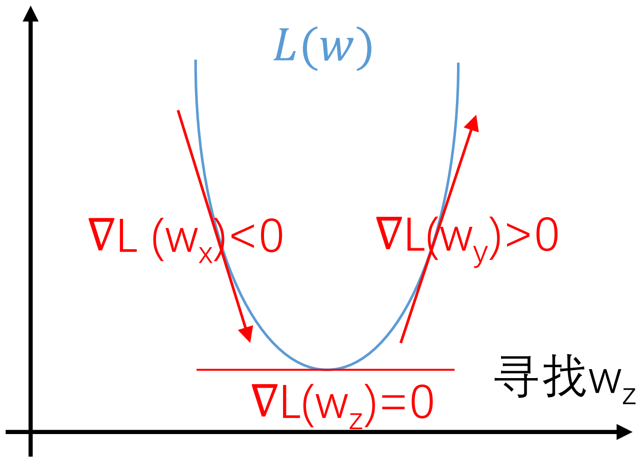
超参数λ为step size或learning rate

优化

- 目标：最小化损失函数$L(\boldsymbol{W})$

- 假设只有一个参数w
  - ✓ 导数$\frac{dL(w)}{dw} = \lim_{h\to 0}\frac{L(w+h)-L(w)}{h}$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$\boldsymbol{W}$为向量
  - ✓ 偏导数$[\frac{\partial L(\boldsymbol{W})}{\partial w_1}, \frac{\partial L(\boldsymbol{W})}{\partial w_2}, \dots, \frac{\partial L(\boldsymbol{W})}{\partial w_n}]$代表$L$在$\boldsymbol{W}$处沿每个维度的变化速率和方向，称为梯度(gradient)，记为$\nabla_{\boldsymbol{W}} L$或$grad(L(\boldsymbol{W}))$
  - ✓ $\nabla_{\boldsymbol{W}} L$ 和方向向量 $\boldsymbol{v}$ 的点积即为该方向的斜率（方向导数）
  - ✓ 负梯度$-\nabla_{\boldsymbol{W}} L$的方向即为$L$在$\boldsymbol{W}$处下降最快的方向
  - ✓ 沿$-\nabla_{\boldsymbol{W}} L$方向微调$\boldsymbol{W}$即可快速减小$L(\boldsymbol{W})$（梯度下降）



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

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

计算梯度$\nabla_{\boldsymbol{W}} L$

梯度下降：
$$W_{new} = W - \lambda \nabla_{\boldsymbol{W}} L$$
超参数λ为step size或learning rate

数值梯度（Numerical Gradient）

$\boldsymbol{W}$		$\boldsymbol{W}+h$ (第一维)
1.5	+0.001	1.501
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）

$\boldsymbol{W}$		$\boldsymbol{W}+\boldsymbol{h}$ (第一维)	$\lim_{h\to 0}\frac{L(w+h)-L(w)}{h}$	$\nabla_{\boldsymbol{W}}L$
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）

$\lim_{h \to 0} \frac{L(w+h) - L(w)}{h}$

(1.2298-1.2356)/0.001=-5.8

$\boldsymbol{W}$		$\boldsymbol{W}+\boldsymbol{h}$ (第二维)	$\nabla_{\boldsymbol{W}} L$
1.5		1.5	1.3
3.3	+0.001	3.301	-5.8
2.1		2.1	?
-0.5		-0.5	?
0.2		0.2	?
…		…	…
Loss=1.2356		Loss=1.2298	

数值梯度（Numerical Gradient）

$\boldsymbol{W}$		$\boldsymbol{W}+\boldsymbol{h}$ (第三维)	$\lim_{h\to 0}\frac{L(w+h)-L(w)}{h}$	$\nabla_{\boldsymbol{W}}L$
1.5		1.5	(1.2356-1.2356)/0.001=0	1.3
3.3		3.3		-5.8
2.1	+0.001	2.101		0
-0.5		-0.5		?
0.2		0.2		?
…		…		…

Loss=1.2356

Loss=1.2356

数值梯度（Numerical Gradient）

$\boldsymbol{W}$		$\boldsymbol{W}+\boldsymbol{h}$ (第三维)		$\nabla_{\boldsymbol{W}} L$
1.5		1.5		1.3
3.3		3.3		-5.8
2.1	+0.001	2.101		0
-0.5		-0.5		?
0.2		0.2		?
…		…		…
Loss=1.2356		Loss=1.2356		

$$\lim_{h \to 0} \frac{L(w+h)-L(w)}{h}$$

(1.2356-1.2356)/0.001=0

开销太大：
需要遍历所有参数
并计算损失和梯度

解析梯度（Analytic Gradient）

- 损失函数是关于$\boldsymbol{W}$的函数

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

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

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

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

- 对损失函数求偏导，编写梯度公式，直接计算梯度

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

解析梯度（Analytic Gradient）

- 损失函数是关于$\boldsymbol{W}$的函数

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

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

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

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

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

梯度: $\nabla_{\boldsymbol{W}} L = [\frac{\partial L(\boldsymbol{W})}{\partial w_1}, \frac{\partial L(\boldsymbol{W})}{\partial w_2}, \ldots, \frac{\partial L(\boldsymbol{W})}{\partial w_n}]$

解析梯度（Analytic Gradient）

Hinge loss和Cross-entropy loss梯度的推导

推导公式极其复杂，请自行推导

第一次作业要求编写解析梯度！！！！

解析梯度（Analytic Gradient）

$\boldsymbol{W}$		$\nabla_{\boldsymbol{W}} L$公式		$\nabla_{\boldsymbol{W}} L$值
1.5		$\frac{\partial L(\boldsymbol{W})}{\partial w_1}$		1.3
3.3		$\frac{\partial L(\boldsymbol{W})}{\partial w_2}$		-5.8
2.1	偏导公式 $\longrightarrow$	$\frac{\partial L(\boldsymbol{W})}{\partial w_3}$	代入$\boldsymbol{W}$ $\longrightarrow$	0
-0.5		$\frac{\partial L(\boldsymbol{W})}{\partial w_4}$		3.7
0.2		$\frac{\partial L(\boldsymbol{W})}{\partial w_5}$		-0.2
…		…		…

梯度下降 vs 随机梯度下降

- 梯度下降：每次更新$\boldsymbol{W}$需要遍历所有数据！

- 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
```

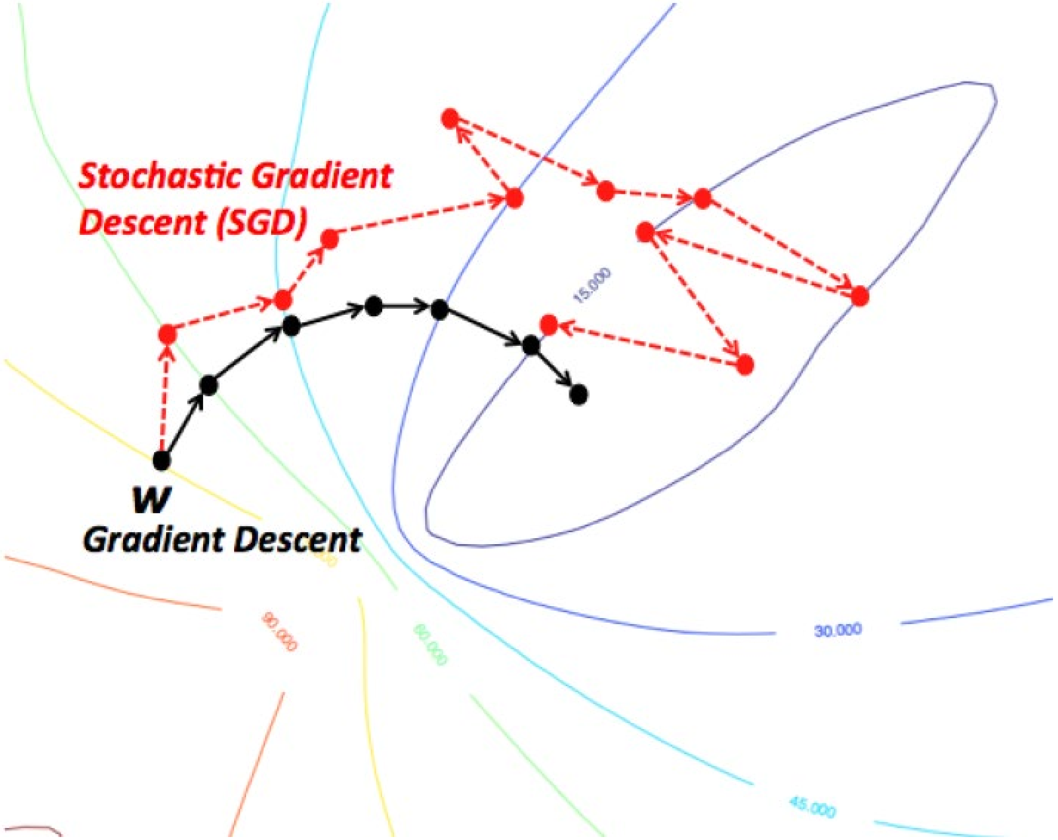
随机选取256个图片，并计算梯度

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