Linear classifiers: Review and multi-class classification

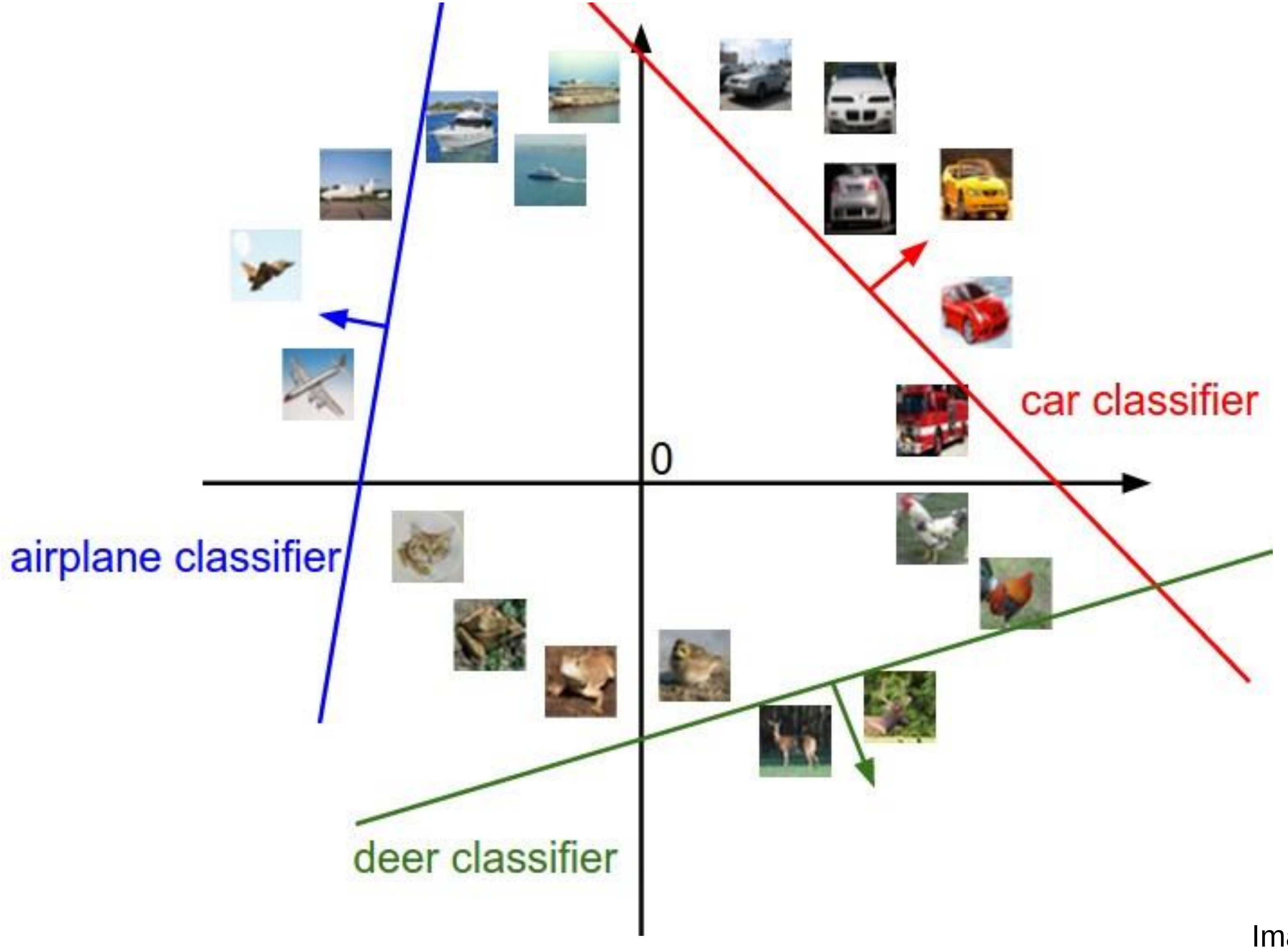


Image source: [Stanford CS 231n]

Review: Linear models

1. Linear regression
2. Logistic regression
3. Perceptron training algorithm
4. Support vector machines

Review: Training linear classifiers

- **Given:** i.i.d. training data $\{(x_i, y_i), i = 1, \dots, n\}$,
  $$y_i \in \{-1,1\}$$

- **Prediction function:** $f_w(x) = \mathrm{sgn}(w^T x)$

- Classification with *bias*, i.e. $f_w(x) = \mathrm{sgn}(w^T x + b)$, can be reduced to the case w/o bias by letting $w' = [w; b]$ and $x' = [x; 1]$

General recipe

- Find parameters w that minimize the sum of a *regularization loss* and a *data loss*:

$$\underbrace{\hat{L}(w)}_{\text{empirical loss}} = \underbrace{\lambda R(w)}_{\text{regularization}} + \frac{1}{n}\sum_{i=1}^{n} \underbrace{l(w, x_i, y_i)}_{\text{data loss}}$$

- Optimize by *stochastic gradient descent* (SGD): At each iteration, sample a single data point (x_i, y_i) and take a step in the direction *opposite* the gradient of the loss for that point:

$$w \leftarrow w - \eta \, \nabla_w \left[\frac{\lambda}{n} R(w) + l(w, x_i, y_i) \right]$$

Model 1: Linear regression

- **Data loss:**

$$l(w, x_i, y_i) = (w^T x_i - y_i)^2$$

- **Regularization:**
  - None
- **Interpretation:**
  - *Negative log likelihood* assuming $y|x$ is normally distributed with mean $w^T x$
- **Pros:** convex loss, easy to optimize
- **Cons:** conceptually inappropriate for classification, sensitive to outliers

Model 2: Logistic regression

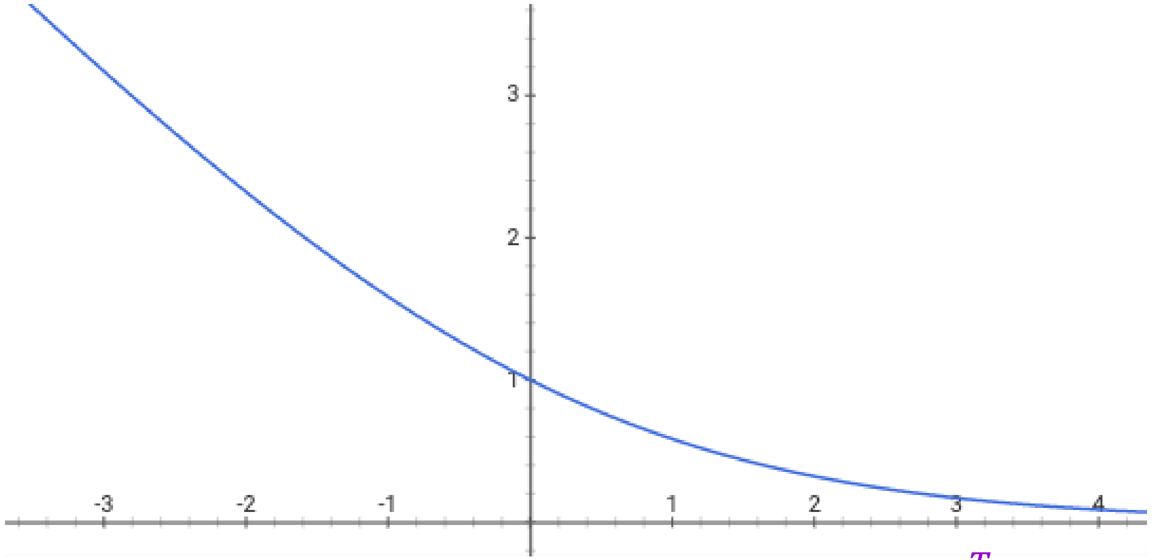
- **Data loss:**

Model 2: Logistic regression

- **Data loss:** *logistic loss*

$$l(w, x_i, y_i) = -\log P_w(y_i|x_i) = -\log \sigma(y_i w^T x_i)$$
$$= -\log \left[\frac{1}{1 + \exp(-y_i w^T x_i)}\right]$$
$$= \log\left[1 + \exp(-y_i w^T x_i)\right]$$

Incorrectly classified



$y_i w^T x_i$

Correctly classified

Model 2: Logistic regression

- **Data loss:** *logistic loss*

$$l(w, x_i, y_i) = -\log P_w(y_i|x_i) = -\log \sigma(y_i w^T x_i)$$
$$= -\log \left[\frac{1}{1 + \exp(-y_i w^T x_i)}\right]$$
$$= \log\left[1 + \exp(-y_i w^T x_i)\right]$$

- **Regularization:**
  - None
- **Interpretation:**
  - Negative log likelihood assuming Gaussian *class-conditional distributions* $P(x|y)$

Model 3: Perceptron training algorithm

Model 3: Perceptron training algorithm

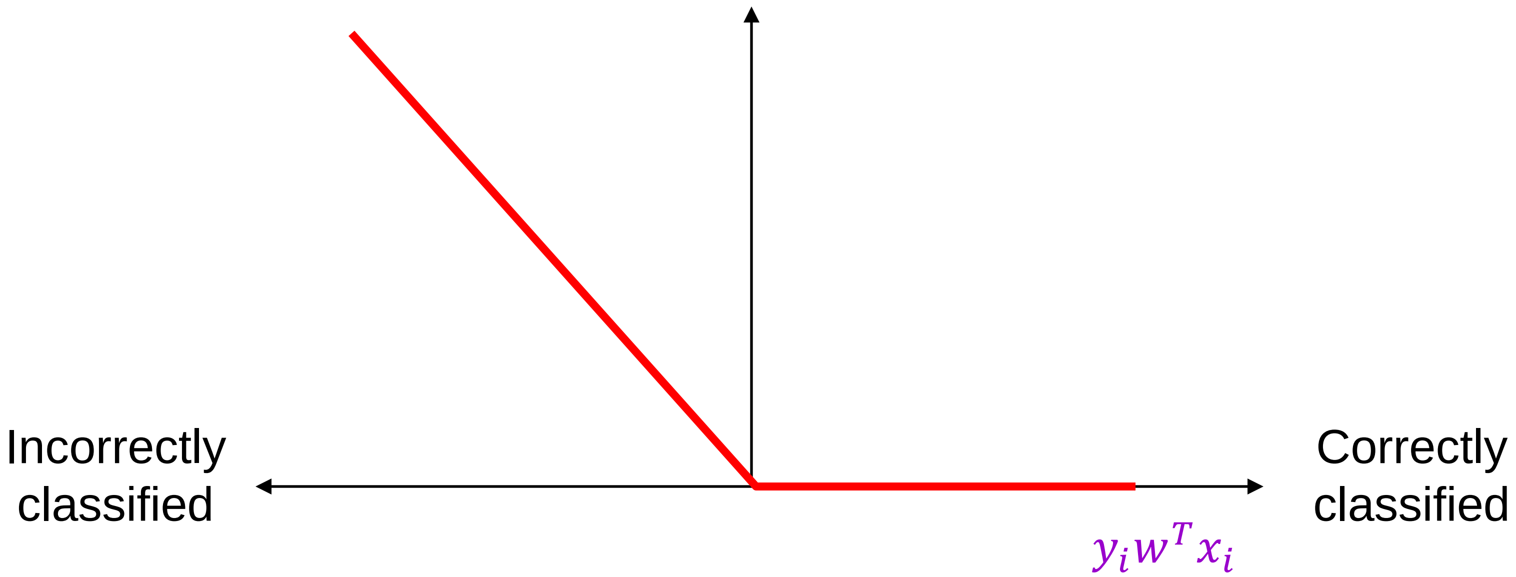
- **Data loss:**

Model 3: Perceptron training algorithm

- **Data loss:** *hinge loss*

$$l(w, x_i, y_i) = \max(0, -y_i w^T x_i)$$

- **Regularization:**
  - None



Model 4: Support vector machines

- **Data loss:**

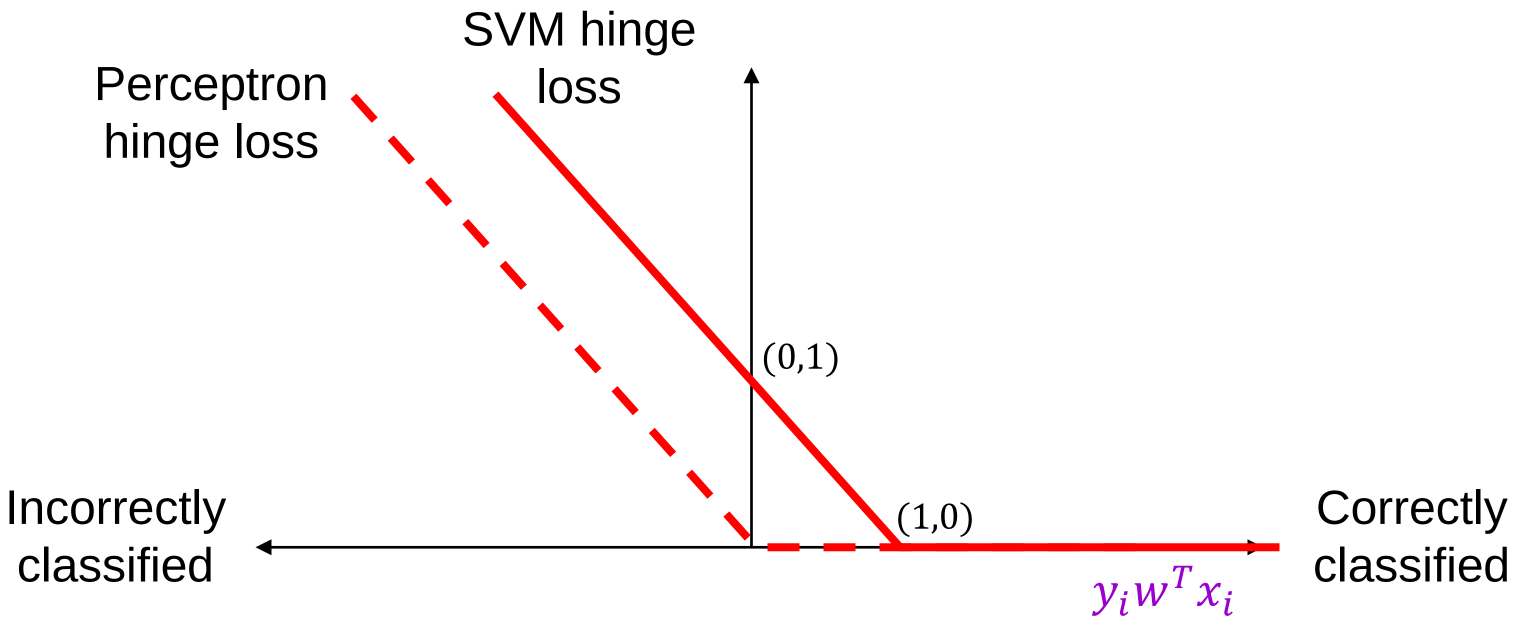
Model 4: Support vector machines

- **Data loss:** *hinge loss*

$$l(w, x_i, y_i) = \max(0, 1 - y_i w^T x_i)$$

- **Regularization:**

$$R(w) = \frac{1}{2}\|w\|^2$$



Model 4: Support vector machines

- **Data loss:** *hinge loss*

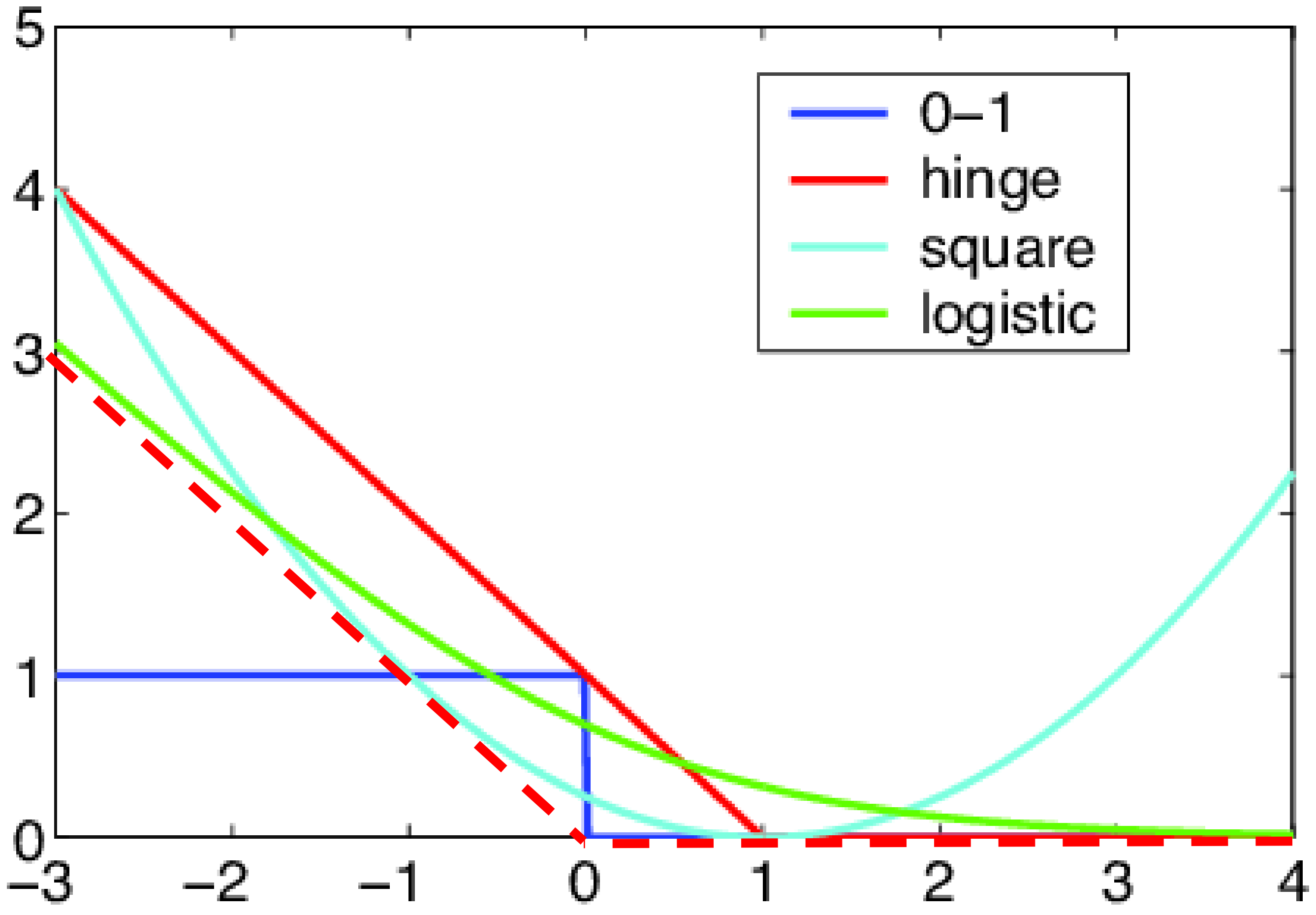
$$l(w, x_i, y_i) = \max(0, 1 - y_i w^T x_i)$$

- **Regularization:**

$$R(w) = \frac{1}{2} ||w||^2$$

- **Interpretation:**
  - Maximize margin while minimizing constraint violations

Summary of data losses



Incorrectly classified

Correctly classified

$y_i w^T x_i$

Image source

Summary of SGD updates

- Linear regression:

$$w \leftarrow w + \eta\,(y_i - w^T x_i)\,x_i$$

- Logistic regression:

$$w \leftarrow w + \eta\,\sigma(-y_i w^T x_i)\,y_i x_i$$

- Perceptron:

$$w \leftarrow w + \eta\,\mathbb{I}[y_i w^T x_i < 0]\,y_i x_i$$

- SVM:

$$w \leftarrow \left(1 - \frac{\eta\lambda}{n}\right) w + \eta\,\mathbb{I}[y_i w^T x_i < 1]\,y_i x_i$$

Multi-class classification

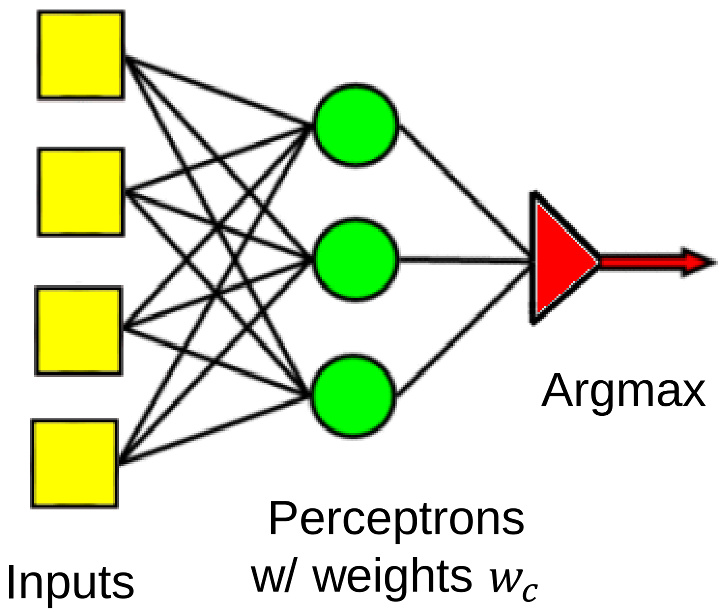
Multi-class classification: Overview

1. Multi-class perceptrons
2. Multi-class SVM
3. Softmax

One-vs-all classification

- Let $y \in \{1, \ldots, C\}$
- Learn C scoring functions $f_1, f_2, \ldots, f_C$
- Classify x to class $\hat{y} = \text{argmax}_c\, f_c(x)$
- Let's start with multi-class perceptrons:

$$f_c(x) = w_c^T x$$



Multi-class perceptrons

- Multi-class perceptrons: $f_c(x) = w_c^T x$
- Let W be the matrix with rows w_c
- What loss should we use for multi-class classification?

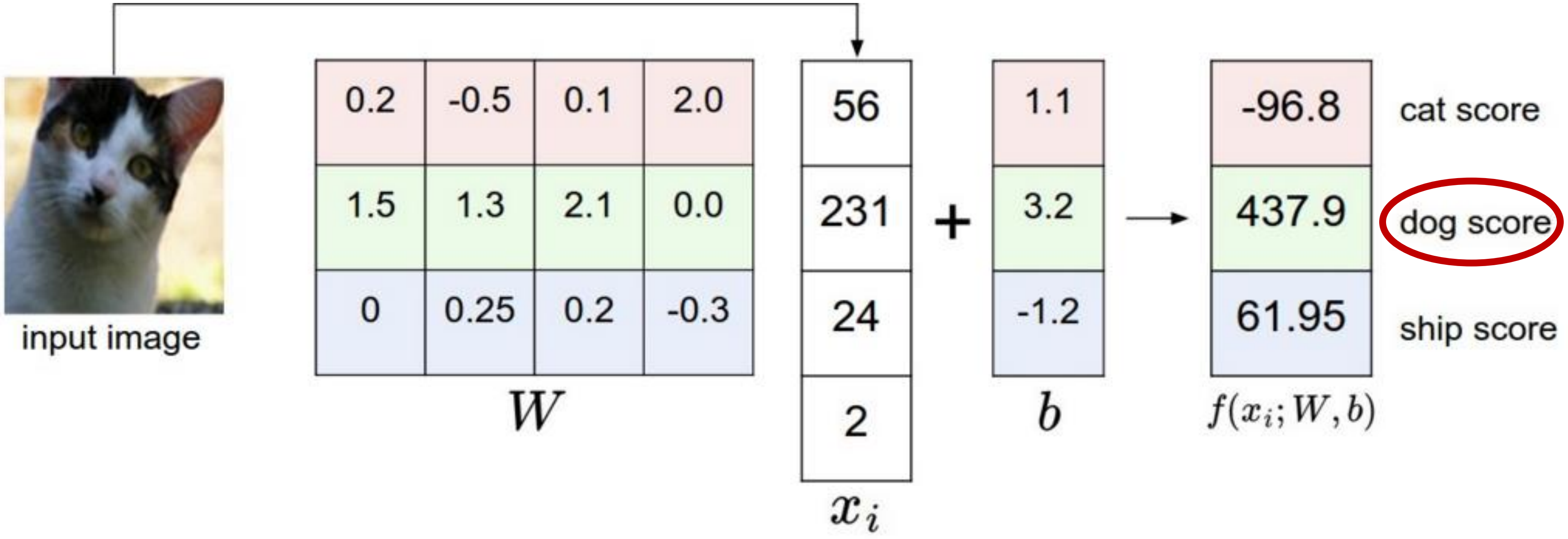


Figure source: [Stanford 231n]

Multi-class perceptrons

- Multi-class perceptrons: $f_c(x) = w_c^T x$
- Let W be the matrix with rows w_c
- What loss should we use for multi-class classification?
- For (x_i, y_i), let the loss be the *sum of hinge losses* associated with predictions for all *incorrect* classes:

$$l(W, x_i, y_i) = \sum_{c \neq y_i} \max[0, w_c^T x_i - w_{y_i}^T x_i]$$

Multi-class perceptrons

$$l(W, x_i, y_i) = \sum_{c \neq y_i} \max[0, w_c^T x_i - w_{y_i}^T x_i]$$

- Gradient w.r.t. w_{y_i}:

$$-\sum_{c \neq y_i} \mathbb{I}\,[w_c^T x_i > w_{y_i}^T x_i] x_i$$

Recall: $\frac{\partial}{\partial a} \max(0, a) = \mathbb{I}[a > 0]$

Multi-class perceptrons

$$l(W, x_i, y_i) = \sum_{c \neq y_i} \max[0, w_c^T x_i - w_{y_i}^T x_i]$$

- Gradient w.r.t. w_{y_i}:

$$-\sum_{c \neq y_i} \mathbb{I}\,[w_c^T x_i > w_{y_i}^T x_i] x_i$$

- Gradient w.r.t. w_c, $c \neq y_i$:

$$\mathbb{I}[w_c^T x_i > w_{y_i}^T x_i] x_i$$

- Update rule: for each c s.t. $w_c^T x_i > w_{y_i}^T x_i$:

$$w_{y_i} \leftarrow w_{y_i} + \eta x_i$$
$$w_c \leftarrow w_c - \eta x_i$$

Multi-class SVM

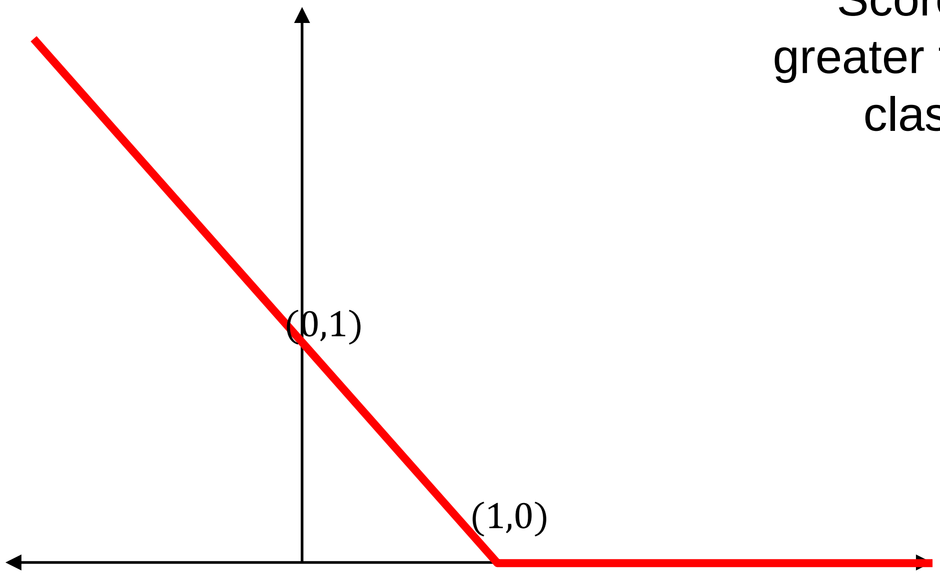
- Recall single-class SVM loss:

$$l(w, x_i, y_i) = \frac{\lambda}{2n}\|w\|^2 + \max[0, 1 - y_i w^T x_i]$$

- Generalization to multi-class:

$$l(W, x_i, y_i) = \frac{\lambda}{2n}\|W\|^2 + \sum_{c \neq y_i} \max[0, 1 - w_{y_i}^T x_i + w_c^T x_i]$$

Score for correct class has to be greater than the score for the incorrect class *by at least a margin of 1*



Score for correct class – score for incorrect class

Source: [Stanford 231n]

Multi-class SVM

$$l(W, x_i, y_i) = \frac{\lambda}{2n}\|W\|^2 + \sum_{c\neq y_i} \max[0, 1 - w_{y_i}^T x_i + w_c^T x_i]$$

- Gradient w.r.t. w_{y_i}:

$$\frac{\lambda}{n} w_{y_i} - \sum_{c\neq y_i} \mathbb{I}[w_{y_i}^T x_i - w_c^T x_i < 1] x_i$$

- Gradient w.r.t. w_c, $c \neq y_i$:

$$\frac{\lambda}{n} w_c + \mathbb{I}[w_{y_i}^T x_i - w_c^T x_i < 1] x_i$$

- Update rule:
  - For $c = 1, \dots, C$: $w_c \leftarrow \left(1 - \eta \frac{\lambda}{n}\right) w_c$
  - For each $c \neq y_i$ s.t. $w_{y_i}^T x_i - w_c^T x_i < 1$: $w_{y_i} \leftarrow w_{y_i} + \eta x_i$, $w_c \leftarrow w_c - \eta x_i$

Softmax

- We want to squash the vector of responses $(f_1, \ldots, f_c)$ into a vector of “probabilities”:

$$\text{softmax}(f_1, \ldots, f_c) = \left(\frac{\exp(f_1)}{\sum_j \exp(f_j)}, \ldots, \frac{\exp(f_c)}{\sum_j \exp(f_j)} \right)$$

- The entries are between 0 and 1 and sum to 1
- If one of the inputs is much larger than the others, then the corresponding softmax value will be close to 1 and others will be close to 0

Note on numerical stability

- Exponentiated classifier responses $\exp(w_c^T x)$ can become very large
- However, adding the same constant to all raw responses does not change the output of the softmax:

$$\frac{\exp(w_c^T x)}{\sum_j \exp(w_j^T x_i)} = \frac{K \exp(w_c^T x)}{\sum_j K \exp(w_j^T x)} = \frac{\exp(w_c^T x + \log K)}{\sum_j \exp(w_j^T x + \log K)}$$

- We can let $\log K = -\max_j w_j^T x$. That is, subtract from each raw response the max response over all the classes

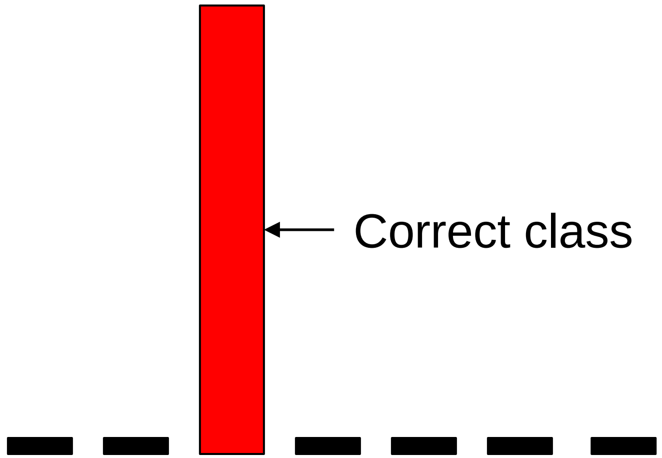
Cross-entropy loss

- It is natural to use negative log likelihood loss with softmax:

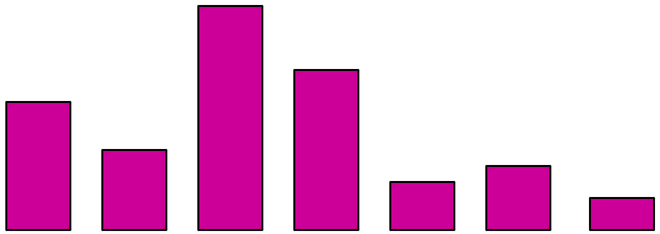
$$l(W, x_i, y_i) = -\log P_W(y_i|x_i) = -\log\left(\frac{\exp(w_{y_i}^T x_i)}{\sum_j \exp(w_j^T x_i)}\right)$$

- This can be viewed as the *cross-entropy* between the "empirical" and "estimated" distributions $\hat{P}(c|x_i) = \mathbb{I}[c = y_i]$ and $P_W(c|x_i)$:

$$-\log P_W(y_i|x_i) = -\sum_c \hat{P}(c|x_i) \log P_W(c|x_i)$$



Empirical distribution $\hat{P}(c|x_i)$



Estimated distribution $P_W(c|x_i)$

Cross-entropy loss

- It is natural to use negative log likelihood loss with softmax:

$$l(W, x_i, y_i) = -\log P_W(y_i|x_i) = -\log\left(\frac{\exp(w_{y_i}^T x_i)}{\sum_j \exp(w_j^T x_i)}\right)$$

- This can be viewed as the *cross-entropy* between the "empirical" and "estimated" distributions $\hat{P}(c|x_i) = \mathbb{I}[c = y_i]$ and $P_W(c|x_i)$:

$$-\log P_W(y_i|x_i) = -\sum_c \hat{P}(c|x_i) \log P_W(c|x_i)$$

- Minimizing cross-entropy is equivalent to minimizing *Kullback-Leibler divergence* between empirical and estimated label distributions

SGD with cross entropy loss

$$l(W, x_i, y_i) = -\log P_W(y_i | x_i) = -\log\left(\frac{\exp(w_{y_i}^T x_i)}{\sum_j \exp(w_j^T x_i)}\right)$$

$$= -w_{y_i}^T x_i + \log\left(\sum_j \exp(w_j^T x_i)\right)$$

- Gradient w.r.t. w_{y_i}:

$$-x_i + \frac{\exp(w_{y_i}^T x_i) x_i}{\sum_j \exp(w_j^T x_i)} = (P_W(y_i | x_i) - 1) x_i$$

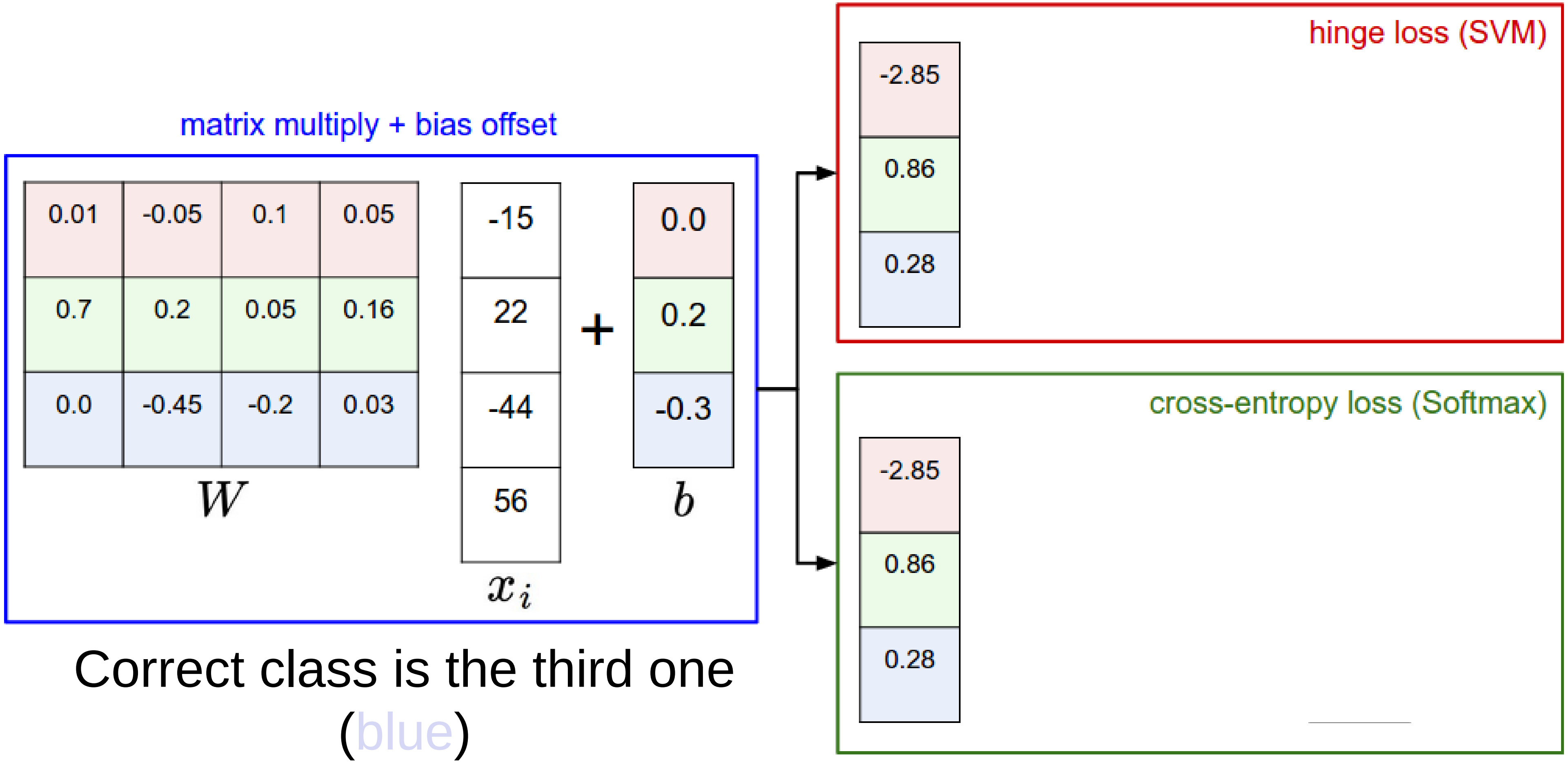
- Gradient w.r.t. w_c, $c \neq y_i$:

$$\frac{\exp(w_c^T x_i) x_i}{\sum_j \exp(w_j^T x_i)} = P_W(c | x_i) x_i$$

SGD with cross-entropy loss

- Gradient w.r.t. w_{y_i}: $(P_W(y_i|x_i) - 1)x_i$
- Gradient w.r.t. w_c, $c \neq y_i$: $P_W(c|x_i)x_i$
- Update rule:
  - For y_i:
    $$w_{y_i} \leftarrow w_{y_i} + \eta(1 - P_W(y_i|x_i))x_i$$
  - For $c \neq y_i$:
    $$w_c \leftarrow w_c - \eta P_W(c|x_i)x_i$$

SVM vs. softmax



Correct class is the third one (blue)

Source: [Stanford 231n]

Acknowledgement

- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Convolutional Neural Networks for Visual Recognition, Stanford University
- Natural Language Processing with Deep Learning, Stanford University