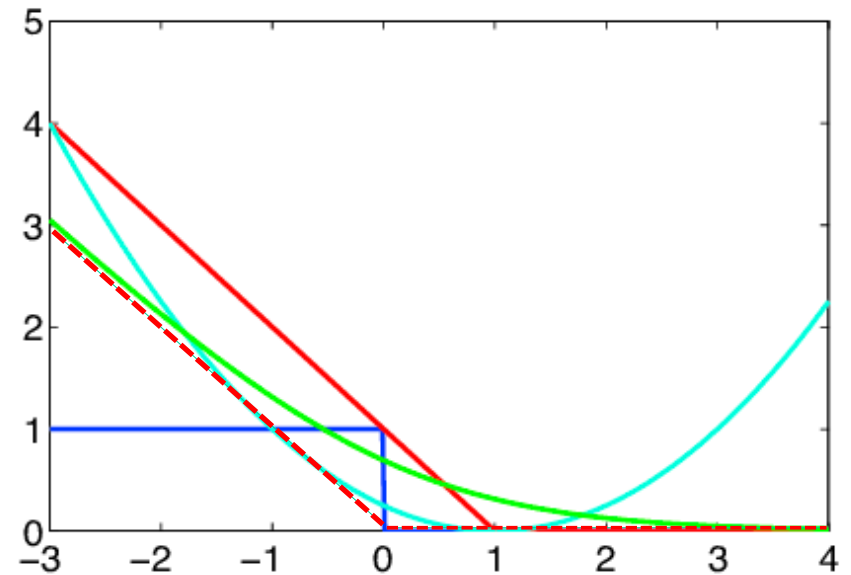


Review: General recipe for loss functions

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

$$\hat{L}(w) = \lambda R(w) + \frac{1}{n} \sum_{i=1}^n l(w, x_i, y_i)$$

empirical loss regularization data loss



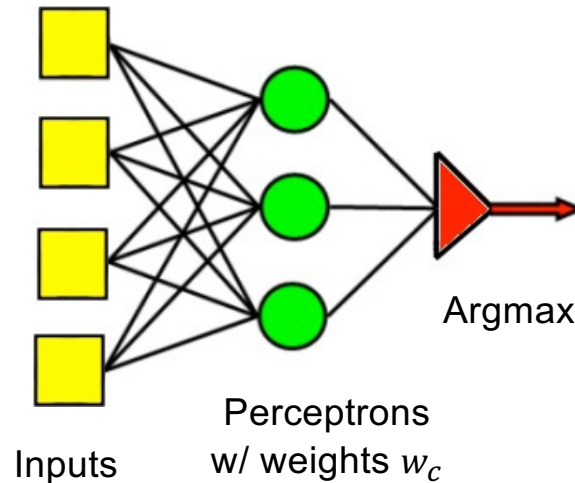
Linear classifiers: Outline

- Empirical loss minimization framework
- Linear classification models
 1. Linear regression
 2. Logistic regression
 3. Perceptron training algorithm
 4. Support vector machines
- **Multi-class classification**
 1. Multi-class perceptron
 2. Multi-class SVM
 3. Softmax

One-vs-all classification

- Let $y \in \{1, \dots, C\}$
- Learn C scoring functions f_1, f_2, \dots, f_C
- Classify x to class $\hat{y} = \operatorname{argmax}_c f_c(x)$
- 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?

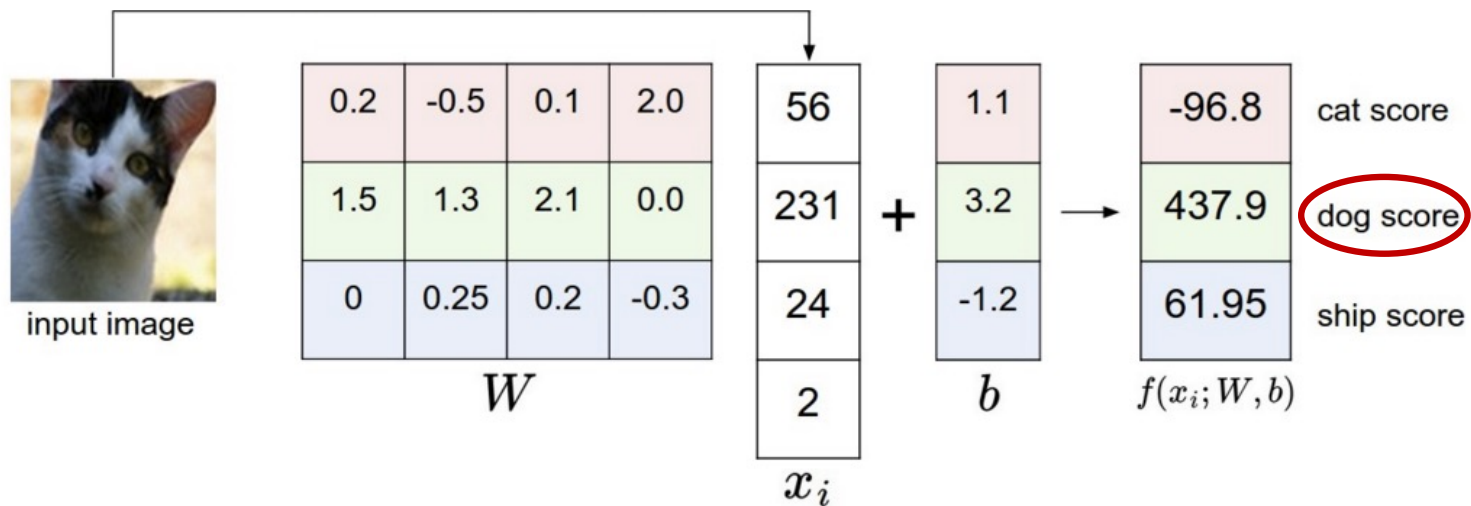


Figure source: [Stanford 231n](#)

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?
- Let the loss be the *sum of hinge losses* associated with predictions for all *incorrect* classes:

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

Score for correct class (y_i)
has to be greater than the
score for the incorrect class (c)

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} :

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

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 perceptrons

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

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

$$w_c \leftarrow w_c - \eta x_i$$

- Is this equivalent to training C independent one-vs-all classifiers?



input image

	Independent	Multi-class
Cat score: 65.1	Do nothing	Increase
Dog score: 101.4	Decrease	Decrease
Ship score: 24.9	Decrease	Do nothing

Multi-class SVM

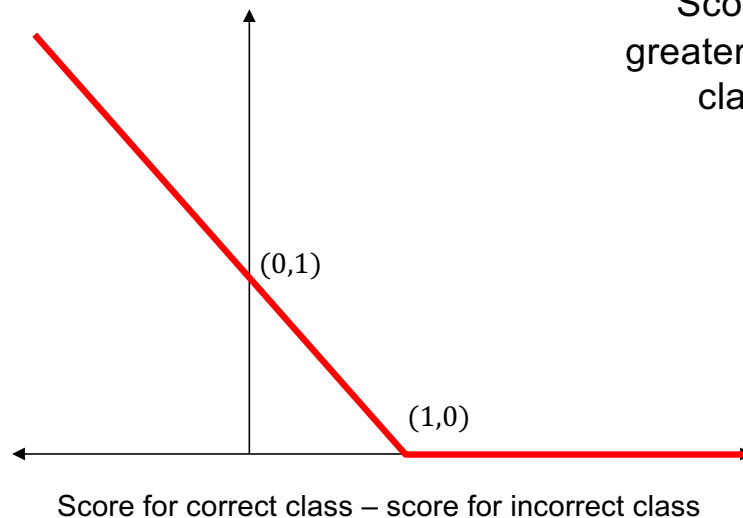
- Recall single-class SVM loss:

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

- 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*



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 (almost* equivalent to above):

- For each $c \neq y_i$ s.t. $w_{y_i}^T x_i - w_c^T x_i < 1$: $w_{y_i} \leftarrow w_{y_i} + \eta x_i$, $w_c \leftarrow w_c - \eta x_i$

- For $c = 1, \dots, C$: $w_c \leftarrow \left(1 - \eta \frac{\lambda}{n}\right) w_c$

Linear classifiers: Outline

- Empirical loss minimization framework
- Linear classification models
 1. Linear regression
 2. Logistic regression
 3. Perceptron training algorithm
 4. Support vector machines
- Multi-class classification
 - Multi-class perceptrons
 - Multi-class SVM
 - **Softmax**

Softmax

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

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

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

Softmax and sigmoid

- For two classes:

$$\begin{aligned}\text{softmax}(f, -f) &= \left(\frac{\exp(f)}{\exp(f) + \exp(-f)}, \frac{\exp(-f)}{\exp(f) + \exp(-f)} \right) \\ &= \left(\frac{1}{1 + \exp(-2f)}, \frac{1}{\exp(2f) + 1} \right) \\ &= (\sigma(2f), \sigma(-2f))\end{aligned}$$

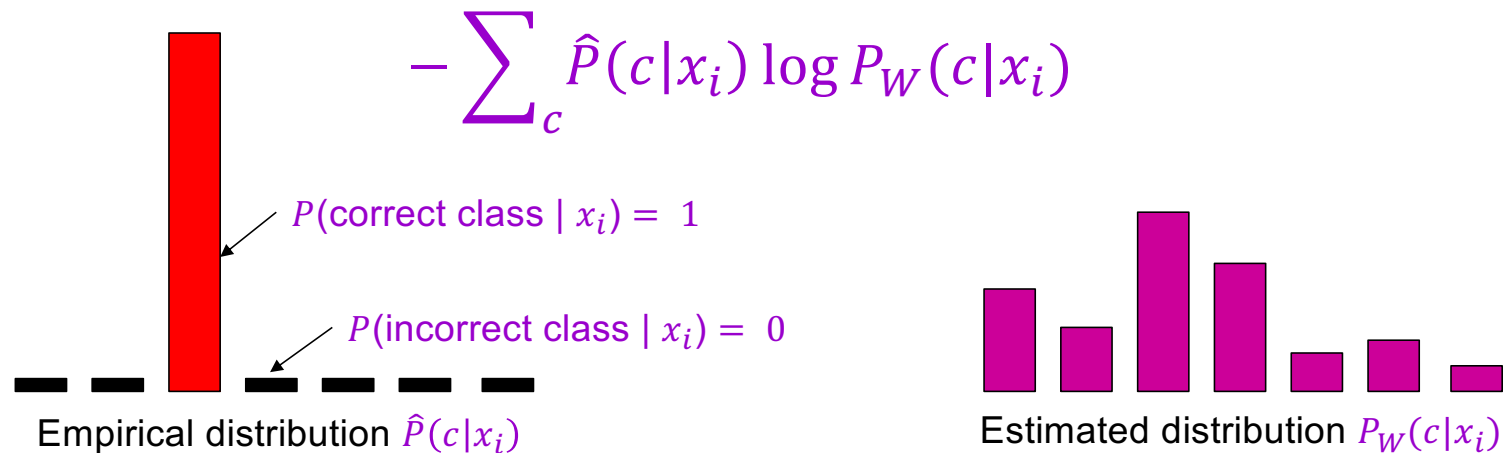
- Thus, softmax is the generalization of sigmoid for more than two classes

Cross-entropy loss

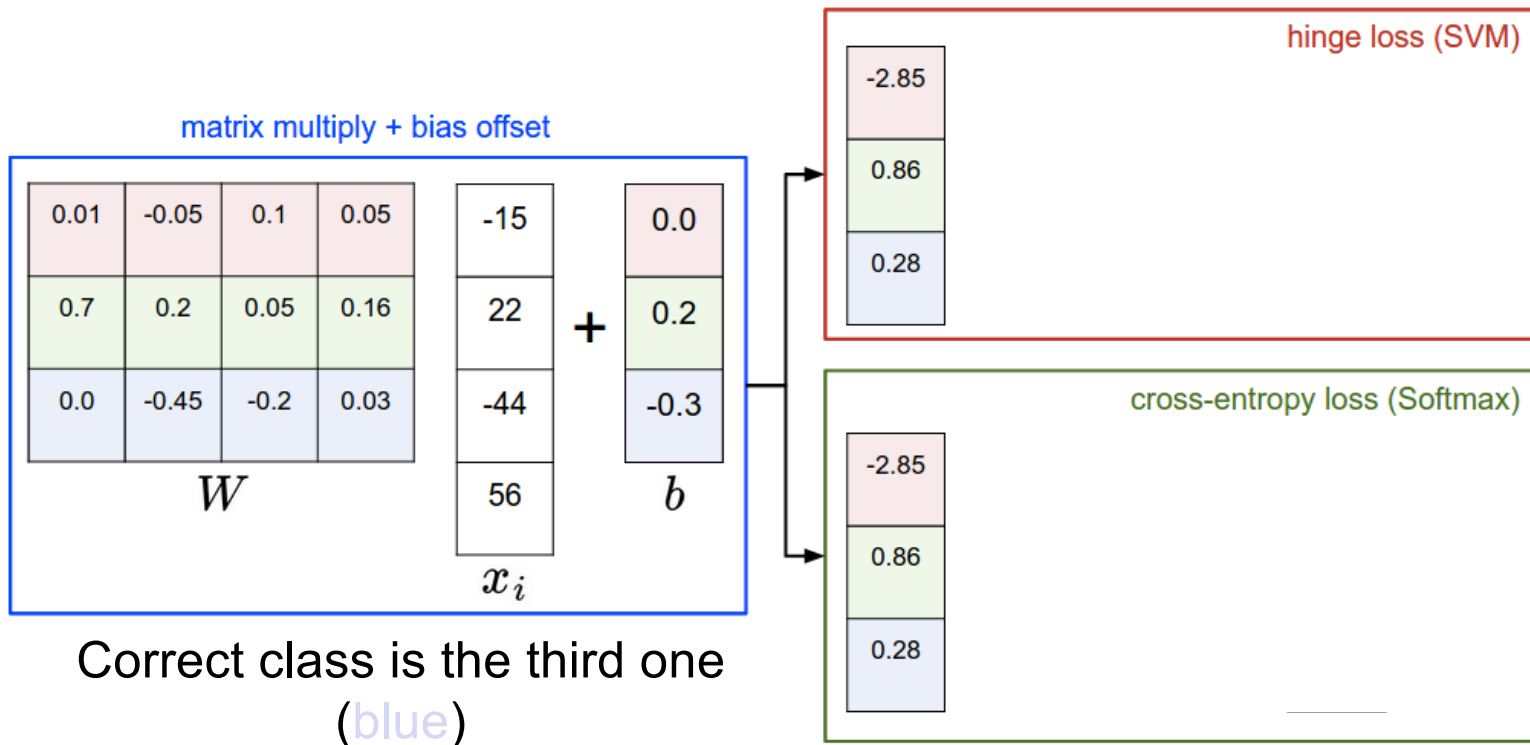
- It is natural to use negative log likelihood loss with softmax:

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

- This happens to be the *cross-entropy* between the “empirical” distribution $\hat{P}(c|x_i) = \mathbb{I}[c = y_i]$ and the “estimated” distribution $P_W(c|x_i)$:



SVM loss vs. cross-entropy loss



SGD with cross-entropy loss

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

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

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

SGD with cross-entropy loss

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

- 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 $c \neq y_i$:
$$w_c \leftarrow w_c - \eta P_W(c|x_i)x_i$$

Softmax trick: Avoiding overflow

- Exponentiated values $\exp(f_c)$ can become very large and cause overflow
- What happens if we add the same constant to all softmax inputs (*logits*)?

$$\frac{\exp(f_c + K)}{\sum_j \exp(f_j + K)} = \frac{\exp(f_c) \exp(K)}{\sum_j \exp(f_j) \exp(K)} = \frac{\exp(f_c)}{\sum_j \exp(f_j)}$$

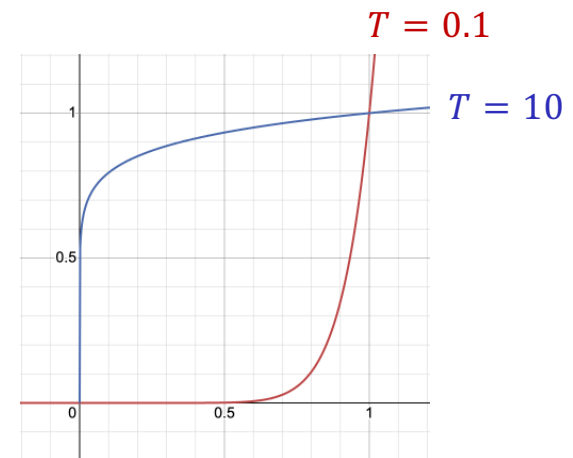
- Then we can let $K = -\max_j f_j$ (i.e., make largest input to softmax be 0)

Softmax trick: Temperature scaling

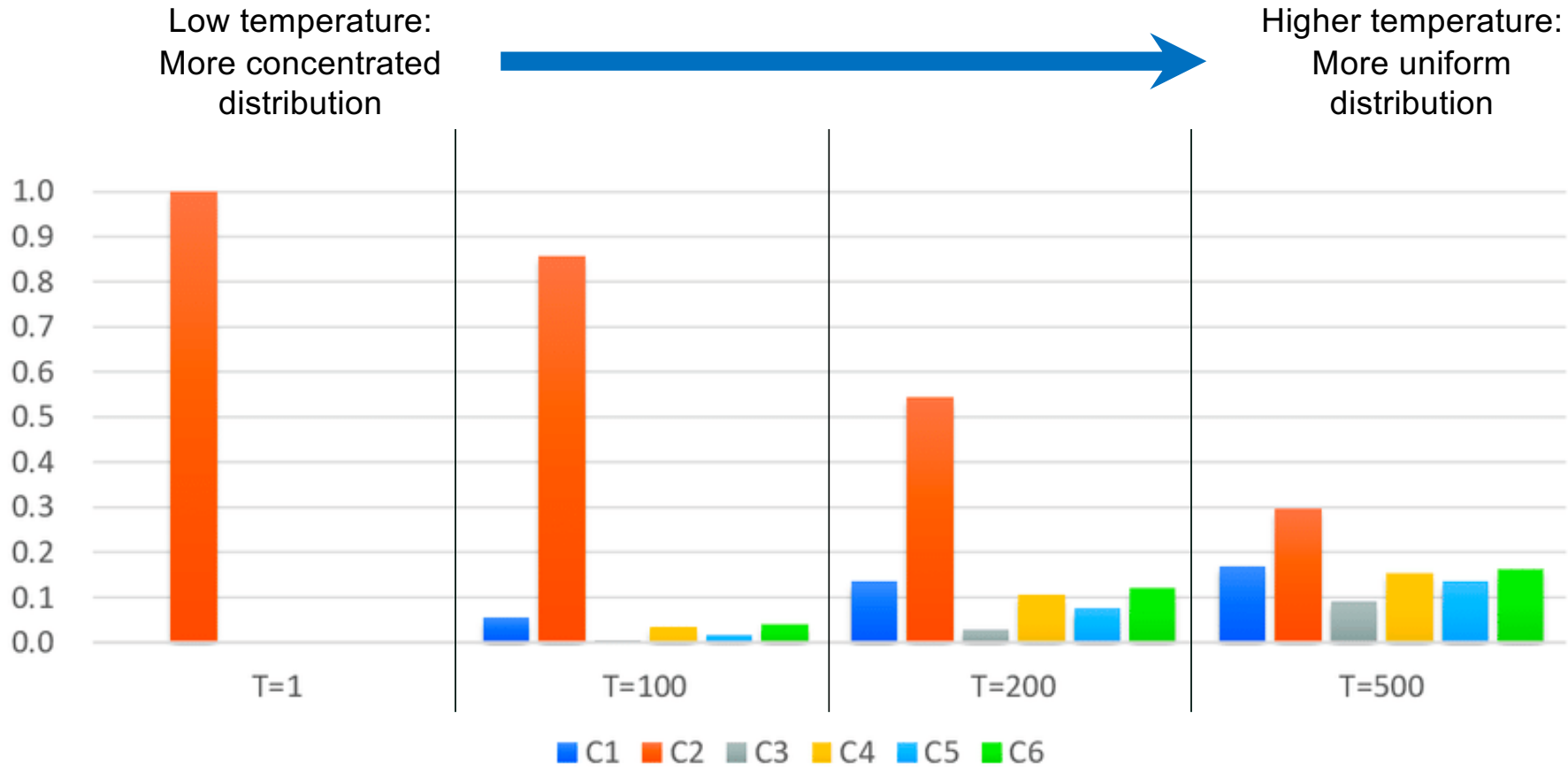
- Suppose we divide every input to the softmax by the same constant T :

$$\text{softmax}(f_1, \dots, f_c; T) = \left(\frac{\exp(f_1/T)}{\sum_j \exp(f_j/T)}, \dots, \frac{\exp(f_c/T)}{\sum_j \exp(f_j/T)} \right)$$

- Prior to normalization, each entry $\exp(f_1)$ is raised to the power $1/T$
- If T is close to 0, the largest entry will dominate and output distribution will approach *one-hot*
- If T is high, output distribution will approach uniform



Softmax trick: Temperature scaling



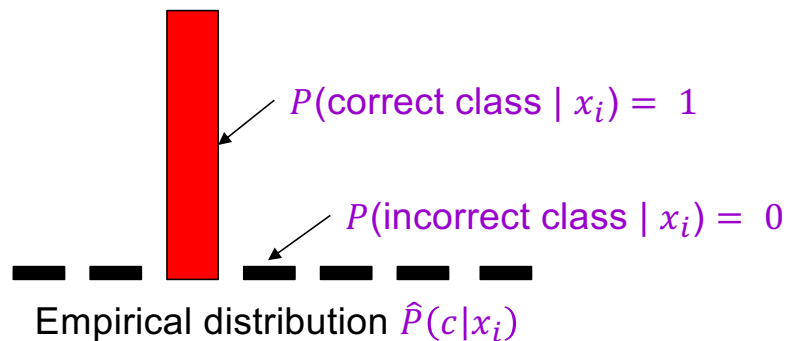
[Figure source](#)

Softmax trick: Label smoothing

- Recall: cross-entropy loss measures the difference between the “observed” label distribution $\hat{P}(c|x_i)$ and “estimated” distribution $P_W(c|x_i)$:

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

“Hard” prediction targets

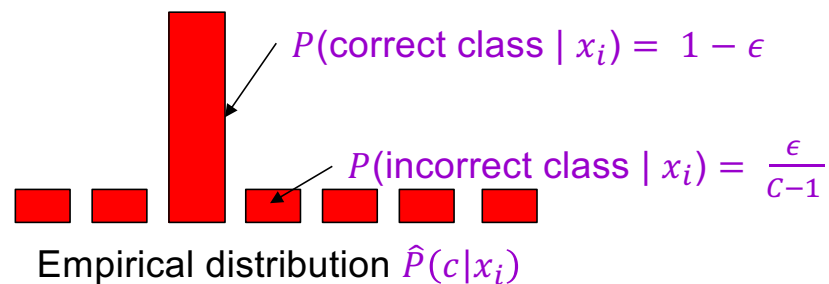


Softmax trick: Label smoothing

- Recall: cross-entropy loss measures the difference between the “observed” label distribution $\hat{P}(c|x_i)$ and “estimated” distribution $P_W(c|x_i)$:

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

“Soft” prediction targets



Softmax trick: Label smoothing

- When using softmax loss, replace hard 1 and 0 prediction targets with “soft” targets of $1 - \epsilon$ and $\frac{\epsilon}{C-1}$
- Why is this a good idea?
 - Helps to avoid overly confident predictions, account for label noise