

Designing, Visualizing and Understanding Deep Neural Networks

Lecture 3: Machine Learning Background II

CS 194/294-129 Spring 2018
John Canny

Last Time

- Generative vs. Discriminative models:
 - Deep nets are discriminative models.
 - May learn more slowly at first, but better asymptotic accuracy.

Generative:

Noisy linear functions

Naïve Bayes

Hidden Markov Models

Gaussian mixture models

Latent Dirichlet Allocation

Discriminative:

Linear Least Squares

Logistic Regression

Conditional Random Fields

Support Vector Machines (SVM)

Decision trees + Random Forests

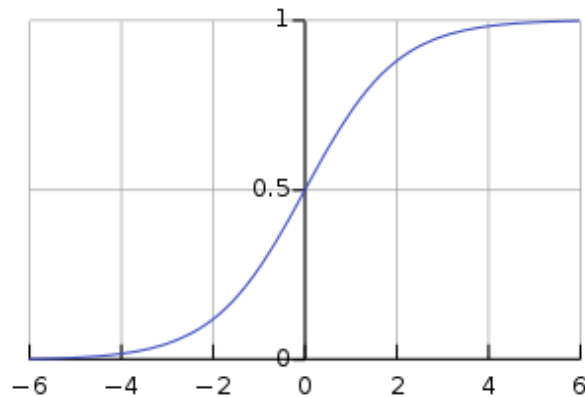
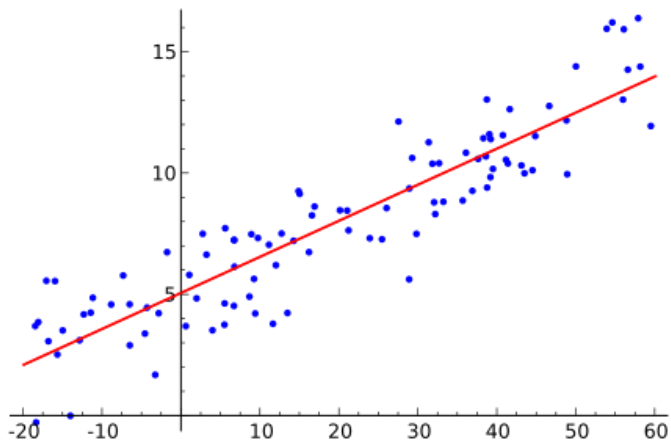
Neural Networks

Last Time

- Loss: a measure of difference between predictions \hat{y} and actual targets y .
- A good model should minimize loss on its predictions.
- Risk:
 - Risk is expected loss (so applies to new data)
 - Empirical risk is the loss on a finite subset of data.
- Machine learning models seek to minimize risk, but actually minimize empirical risk.

Last Time

- Prediction functions $\hat{y} = f(x)$:
 - Linear regression: predict a real value, loss = squared loss.
 - Logistic regression: predict a binary target, loss = cross-entropy loss.



Updates - Discussion

Discussion sections are scheduled (and on bCourses page):

Time	Location
Mon 3-4 pm	310 Soda
Weds 12-1 pm	405 Soda
Thurs 1-2 pm	320 Soda
Thurs 2-3 pm	310 Soda

Discussions start tomorrow 1/25: Thursday (both sessions).

Material presented based on most recent MW lectures, so each topic presented on Thursday → Monday → Weds. Thursday 1-2 is first presentation, others are repeats.

Updates - Discussion

Discussion section load balancing:

Please fill out assignment 0.1 on bCourses. This is another poll with just the scheduled section slots. Please do this:

- Fill out the poll immediately if you can make only one section, and if you plan to attend regularly.
- If you can attend multiple sections, wait until there are say > 40 responses, then pick the least crowded section that you can make.
- This poll is not anonymous (so you can see what the other responses are), but you can use a pseudonym or firstname+initial etc.
- In any case, please pick only one option.

Updates – Assignment 1

Assignment 1 is out, due on Feb 12th, 11pm.

- Uses python + ipython. Please check right away that you have a working installation of python 2.7 and ipython2, and that you can load and execute the assignment notebooks.
- Python virtualenv is your best bet, but there are tricks to doing it on a Mac.
- Don't assume that just because python/ipython used to work on a given machine, that it still does. e.g. I had to uninstall and reinstall pyzmq on a machine that used to work...
- The assignment itself closely tracks the lecture material over the next couple of weeks, so you'll get best value by doing it in stages.

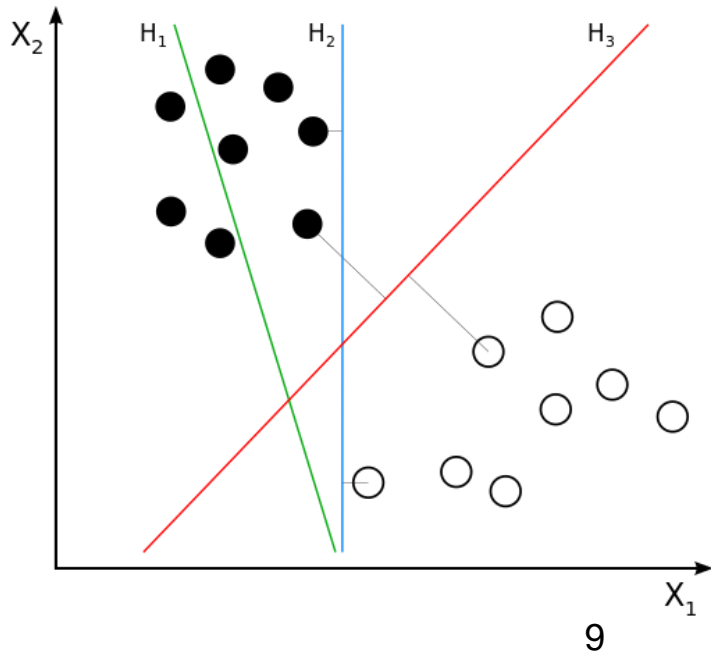
This Time

- Support Vector Machines (SVMs).
- Multi-class SVMs.
- Multi-class Logistic Regression and Softmax.
- Cross-validation.

Support Vector Machines

Suppose we have some two-dimensional observations, i.e. pairs (x_1, x_2) that belong to two classes.

We can plot them in the plane as black and white points.



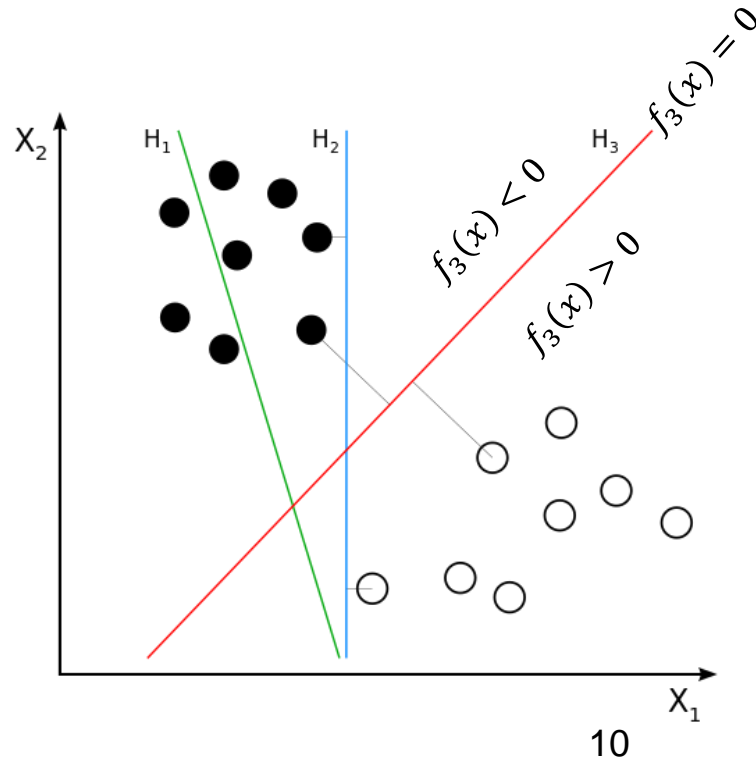
Support Vector Machines

We can construct a linear classifier with a weight vector $w = (w_0, w_1, w_2)$ and the decision function:

$$f(x) = w_0 + w_1x_1 + w_2x_2 > 0$$

The lines on the right show the decision boundaries for 3 different weight vectors.

i.e. they show where $f_j(x) = 0$ where $j \in \{1,2,3\}$ corresponding to H_1, H_2, H_3

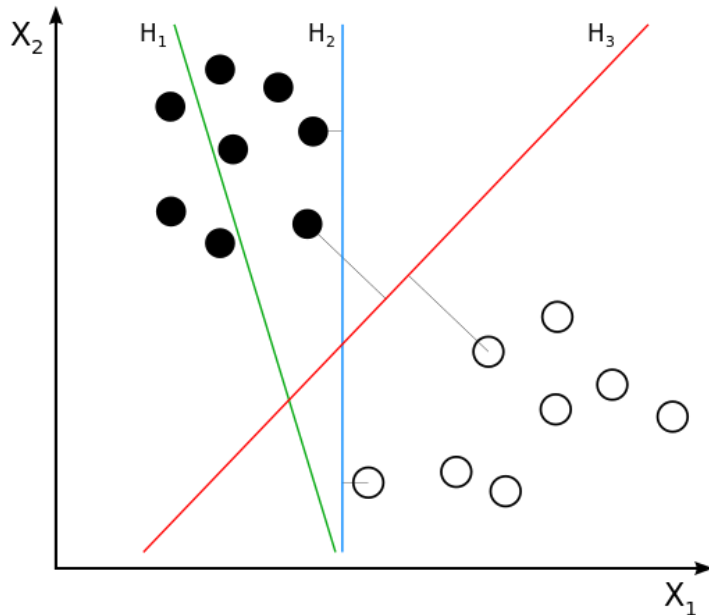


Support Vector Machines

H_1 does not correctly classify the data, while both H_2 and H_3 do.

But H_2 and H_3 do not define equally good classifiers.

Which (H_2 or H_3) boundary gives a better classifier and why?



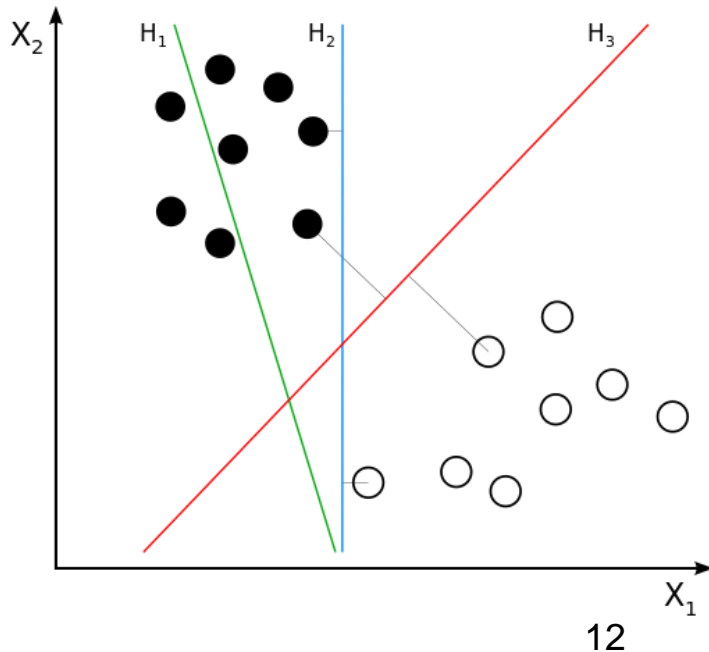
Support Vector Machines

H_1 does not correctly classify the data, while both H_2 and H_3 do.

But H_2 and H_3 do not define equally good classifiers.

Which (H_2 or H_3) boundary gives a better classifier and why?

H_3 has a larger **margin** (gray lines) and is more likely to correctly classify new data.

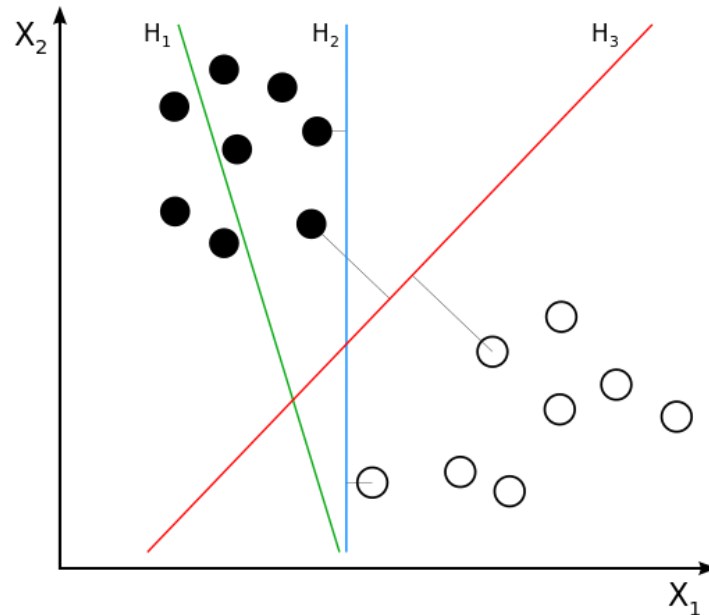


Support Vector Machines

H_3 has a larger **margin** (gray lines) and is more likely to correctly classify new data.

New data points have to be **at least as far away as the margin** in order to be misclassified.

A large margin implies these points have to be far from current data (unlikely).

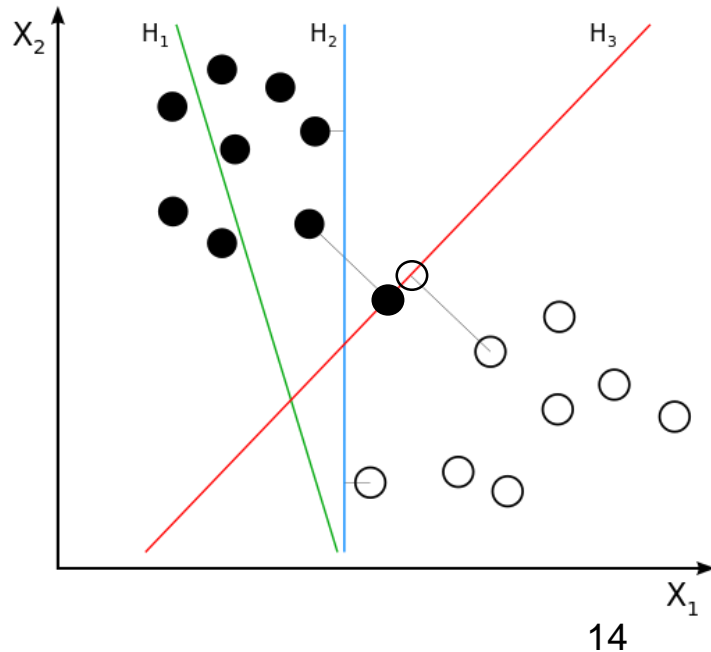


Support Vector Machines

H_3 has a larger **margin** (gray lines) and is more likely to correctly classify new data.

New data points have to be **at least as far away as the margin** in order to be misclassified.

A large margin implies these points have to be far from current data (unlikely).

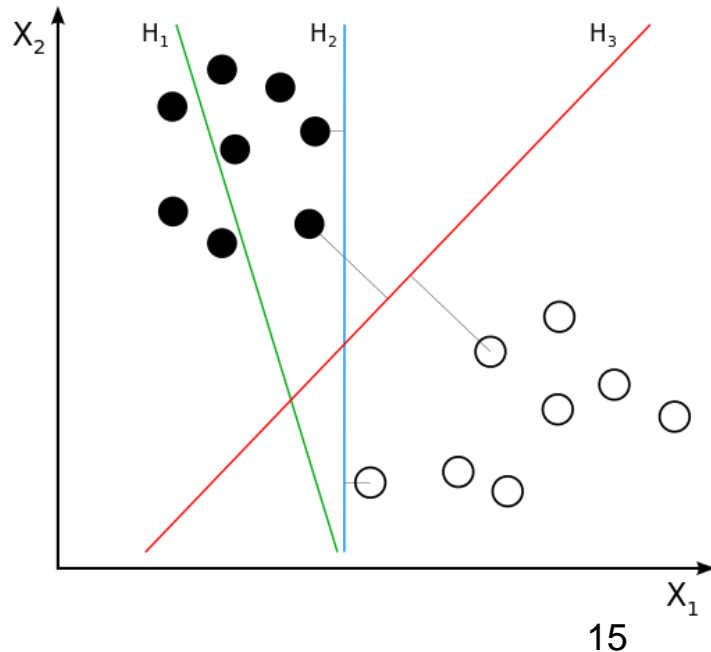


Support Vector Machines

H_3 has a larger **margin** (gray lines) and is more likely to correctly classify new data.

New data points have to be **at least as far away as the margin** in order to be misclassified.

A small margin implies these points can be near current data (likely).

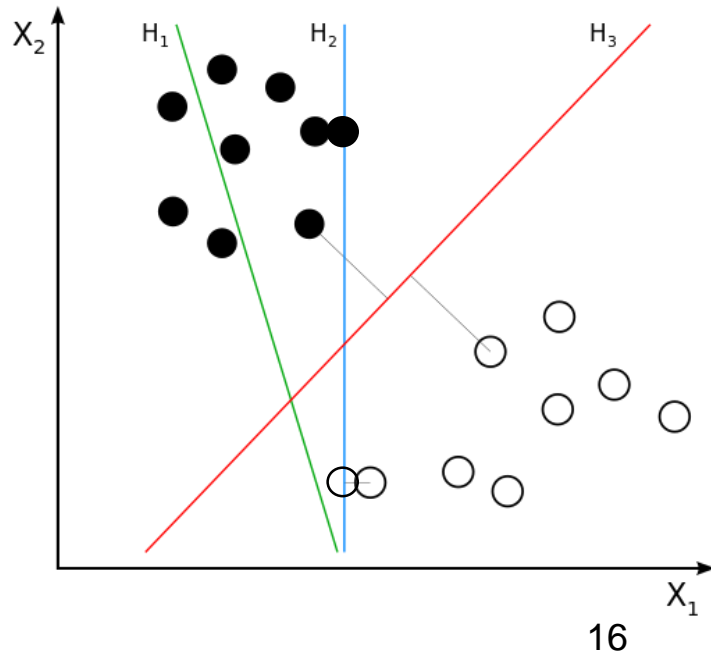


Support Vector Machines

H_3 has a larger **margin** (gray lines) and is more likely to correctly classify new data.

New data points have to be **at least as far away as the margin** in order to be misclassified.

A small margin implies these points can be near current data (likely).



SVM Loss

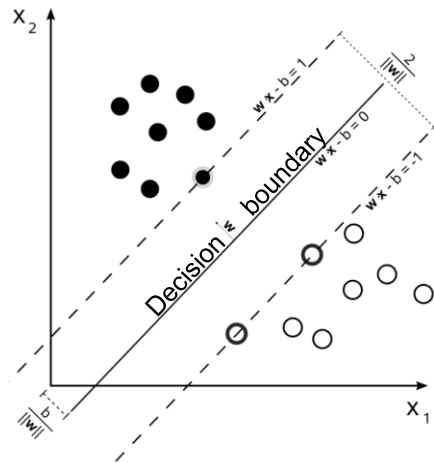
We would like to maximize the classifier's margin, which we do in several stages.

If the 2-norm (length) of the weight w vector $\|w\|_2 = 1$, then $f(x) = w^T x + b$ is a signed measure of the distance from x to the line $f(x) = 0$.

We can create a unit-length margin using the lines $f(x) = 1$ and $f(x) = -1$ (see figure on the right), and setting:

$$f(x) \geq 1 \text{ for } x \in \mathcal{C}$$

$$f(x) \leq -1 \text{ for } x \notin \mathcal{C}$$



Hinge Loss

Next we use a common labeling trick $y = \begin{cases} -1 & \text{if } x \notin \mathcal{C} \\ 1 & \text{if } x \in \mathcal{C} \end{cases}$

And then the two inequalities

$$f(x) \geq 1 \text{ for } x \in \mathcal{C}$$

$$f(x) \leq -1 \text{ for } x \notin \mathcal{C}$$

Simplify to one:

$$yf(x) \geq 1$$

Then we want a loss that measures how much this constraint is **violated**. This value does it:

$$\max(0, 1 - yf(x))$$

This is called a **hinge loss**.

Hinge Loss

For a set of data points, the hinge loss is just the sum of per-point losses

$$l = \sum_{i=1}^n \max(0, 1 - y_i f(x_i))$$

We assumed so far that $f(x) = w^T x + b$ with $\|w\|_2 = 1$.

As $\|w\|$ increases, the hinge loss decreases. Why?

Minimizing l creates a “pressure” on $\|w\|$ to increase. Instead of forcing $\|w\| = 1$ (expensive), we add a loss term $\lambda \|w\|^2$ which tends to decrease $\|w\|$:

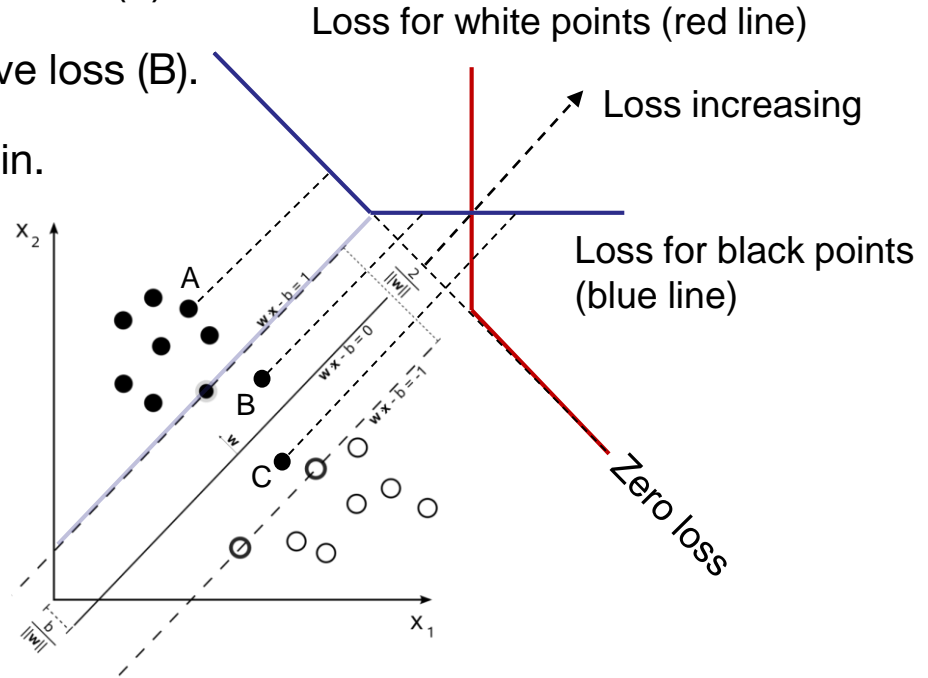
$$l = \sum_{i=1}^n \max(0, 1 - y_i f(x_i)) + \lambda \|w\|^2$$

And λ can be adjusted to get $\|w\|$ close to 1. i.e. this classifier learns the margin as well as the boundary. This is a **soft-margin** SVM.

Hinge Loss

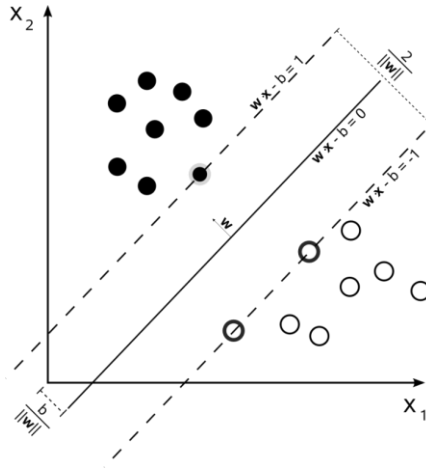
We can show the hinge losses for positive and negative instances on the figure:

- Correct points outside the margin have 0 loss (A).
- Correct points within margin have positive loss (B).
- Loss grows with distance from the margin.
- Incorrect points have positive loss (C).



Compare with Logistic Regression

How does the logistic regression (cross-entropy) loss vary for these points?

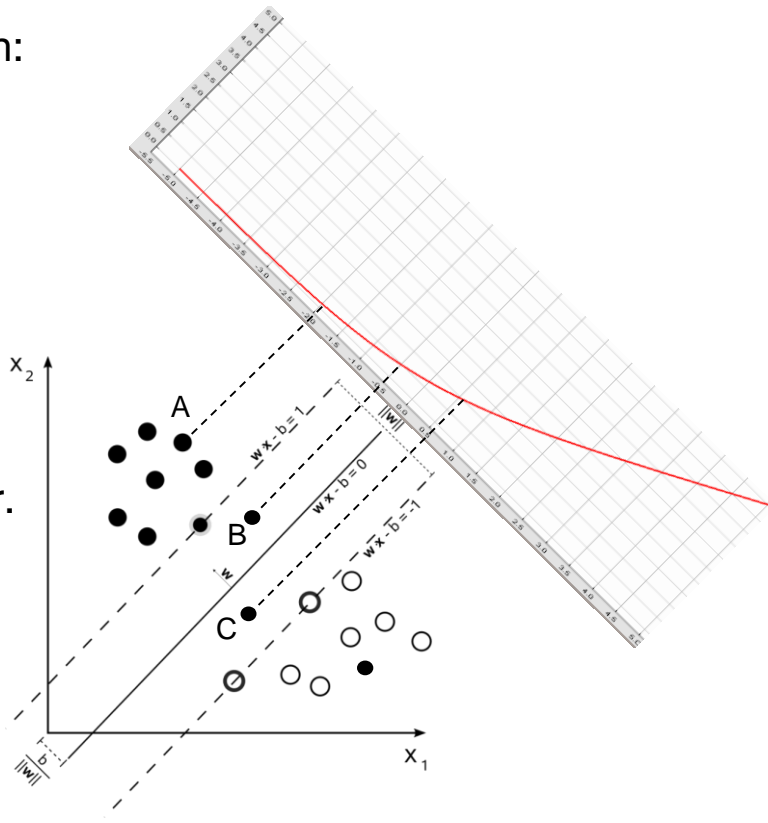


Compare with Logistic Regression

Logistic regression loss is the log of the logistic function:

- Small loss to correct points outside the margin (A)
- Larger loss to correct points in the margin (B)
- Still larger loss to incorrect points in the margin (C)

So logistic regression is a kind of ***soft-margin*** classifier.



Multi-Class Classification

One-versus-rest: For k classes, we construct k functions $f_i(x)$ for $i = 1, \dots, k$, designed to estimate the probability that x is in class i .

We train $f_i(x)$ with labeled data: $y = 1$ if $x \in C$ and $y = 0$ if x is in any other class.

The predicted class label for x is

$$\operatorname{argmax}_i f_i(x)$$

One-versus-one: For k classes we construct $\binom{k}{2}$ functions $f_{ij}(x)$ with $i < j \in \{1, \dots, k\}$.

Each $f_{ij}(x)$ is a binary classifier for class i and class j , and is trained on those classes.

$f_{ij}(x) > 0$ is a vote for class i , while $f_{ij}(x) < 0$ is a vote for class j .

For a new instance x , tally the votes from all $\binom{k}{2}$ classifiers. Output the class with highest vote.

Multi-Class SVM loss

We consider the one-versus-rest design because its scalable (only k classifiers).

Want a notion of **margin**: in this case we take the difference between the correct label class and the current class. We want

$$f_y(x) - f_j(x) \geq 1$$

where y is the correct label class corresponding to x , and j is any other class.

Then the loss for class j is

$$\max\left(0, 1 - \left(f_y(x) - f_j(x)\right)\right)$$

And we sum this over all the classes j not equal to y :

$$l = \sum_{j \neq y} \max\left(0, 1 - f_y(x) + f_j(x)\right)$$

Simplify Notation

Since every $f_j(x)$ is a linear function with a weight vector w^j , we can write it as $f_j(x, w^j)$.

We can compactly describe the full predictor (all classes) as:

$$f(x, W) = \begin{bmatrix} f_1(x, w^1) \\ f_2(x, w^2) \\ \vdots \\ f_k(x, w^k) \end{bmatrix}$$

And then $f(x, W) = Wx$ where

$$W = \begin{bmatrix} (w^1)^T \\ (w^2)^T \\ \vdots \\ (w^k)^T \end{bmatrix}$$

Loss functions

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1

Bold values are scores for the correct class.

Multiclass SVM loss:

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,
and using the shorthand for
the scores vector:

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

the SVM loss has the form:

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

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9		

Multiclass SVM loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,
and using the shorthand for
the scores vector:

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

the SVM loss has the form:

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

$$\begin{aligned} &= \max(0, 5.1 - 3.2 + 1) \\ &\quad + \max(0, -1.7 - 3.2 + 1) \\ &= \max(0, 2.9) + \max(0, -3.9) \\ &= 2.9 + 0 \\ &= 2.9 \end{aligned}$$

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	

Multiclass SVM loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,
and using the shorthand for
the scores vector:

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

the SVM loss has the form:

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

$$\begin{aligned} &= \max(0, 1.3 - 4.9 + 1) \\ &\quad + \max(0, 2.0 - 4.9 + 1) \\ &= \max(0, -2.6) + \max(0, -1.9) \\ &= 0 + 0 \\ &= 0 \end{aligned}$$

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	10.9

Multiclass SVM loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,
and using the shorthand for
the scores vector:

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

the SVM loss has the form:

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

$$\begin{aligned} &= \max(0, 2.2 - (-3.1) + 1) \\ &\quad + \max(0, 2.5 - (-3.1) + 1) \\ &= \max(0, 5.3) + \max(0, 5.6) \\ &= 5.3 + 5.6 \\ &= 10.9 \end{aligned}$$

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	10.9

Multiclass SVM loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,
and using the shorthand for
the scores vector:

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

the SVM loss has the form:

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

and the full training loss is the mean over all
the examples:

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

$$L = (2.9 + 0 + 10.9)/3 = 4.6$$

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	10.9

Multiclass SVM loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,
and using the shorthand for
the scores vector:

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

the SVM loss has the form:

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

Q: what if the sum was instead
over all classes?
(including $j = y_i$)

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	10.9

Multiclass SVM loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,
and using the shorthand for
the scores vector:

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

the SVM loss has the form:

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

Q2: what if we used a mean
instead of a sum here?

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	10.9

Multiclass SVM loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,
and using the shorthand for
the scores vector:

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

the SVM loss has the form:

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

Q3: what if we used a
squared loss:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)^2$$

Now compute loss across all N samples x_i in a dataset:

$$f(x, W) = Wx$$
$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + 1)$$

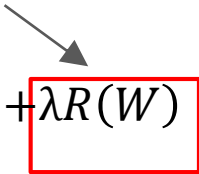
Remember we assumed $\|w^j\|_2 = 1$, but this loss has a linear dependence on $\|w^j\|_2$. This dependence is usually negative, so minimizing risk will tend to make $\|w^j\|_2$ grow.

We can fix this as before with a regularizing term $\lambda \|W\|_2$ where $\|*\|_2 = \text{sum of squares of all matrix elements}$:

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + 1) + \lambda \|W\|_2$$

Weight Regularization

λ = regularization strength
(hyperparameter)

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + 1) + \lambda R(W)$$


In common use:

L2 regularization

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

L1 regularization

$$R(W) = \sum_k \sum_l |W_{k,l}|$$

Elastic net (L1 + L2)

$$R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$$

Dropout (will see later)

Max norm regularization (might see later)

L2 regularization: motivation

What would L1 regularization do?

Multi-Class Logistic Regression

We again consider the one-versus-rest design because its scalable.

We assume x is an m -dimensional vector. It may be binary or real-valued.

We would like $f_j(x)$ to estimate the probability that x is in class j .

Assume again that we apply the $k \times m$ linear weight vector W to x , and let $s = Wx$.

We define:

$$f_j(x) = \frac{\exp(s_j)}{\exp(s_1) + \exp(s_2) + \cdots + \exp(s_k)}$$

Clearly

$$\sum_{j=1}^k f_j(x) = 1$$

So it acts as a probability

Logistic Regression (Softmax)

The formula:

$$f_j(x) = \frac{\exp(s_j)}{\exp(s_1) + \exp(s_2) + \cdots + \exp(s_k)}$$

Is called a **softmax** of the vector (s_1, \dots, s_k) .

The intuition for the name “softmax” is that if some s_j is somewhat larger than the others, its exponential will dominate the sum, and the output will be close to $(0, \dots, 1, \dots, 0)$ where the 1 is in the j^{th} position.

But that’s not very compelling. Is there a more convincing argument for this particular choice of function?

Relation to (multiclass) naïve Bayes

There is a very good reason: once again multiclass logistic regression (softmax of Wx) is ***the discriminative version of*** multiclass naïve bayes.

In particular, if x is a binary vector, there is a weight matrix W such that the softmax classifier implements the multiclass naïve bayes classifier (so it's the optimal classifier under the assumptions of naïve bayes, that features are independent given the class).

Proof: (check yourself)

Assume x has a constant coordinate x_0 . Then define $W_{j,0}$

$$W_{j,0} = \log(P(X = 0|Y = j)P(Y = j)) = \log(P(X_1 = 0|Y = j) \cdots P(X_m = 0|Y = j)P(Y = j))$$

And

$$W_{j,i} = \log\left(\frac{P(X_i = 1|Y = j)}{P(X_i = 0|Y = j)}\right)$$

Softmax Classifier (Multinomial Logistic Regression)



cat **3.2**

car 5.1

frog -1.7

Softmax Classifier (Multinomial Logistic Regression)



Scores = unnormalized log prob. of the classes

$$s = f(x, W)$$

cat **3.2**

car 5.1

frog -1.7

Softmax Classifier (Multinomial Logistic Regression)



Scores = unnormalized log prob. of the classes

$$P(Y = k|X = x_i) = \frac{e^{s_{yi}}}{\sum_j e^{s_{yj}}} \text{ where } s = f(x, W)$$

cat **3.2**

car 5.1

frog -1.7

Softmax Classifier (Multinomial Logistic Regression)



Scores = unnormalized log prob. of the classes

$$P(Y = k|X = x_i) = \frac{e^{s_{yi}}}{\sum_j e^{s_{yj}}} \text{ where } s = f(x, W)$$

Softmax function

cat **3.2**

car 5.1

frog -1.7

Softmax Classifier (Multinomial Logistic Regression)



Scores = unnormalized log prob. of the classes

$$P(Y = k|X = x_i) = \frac{e^{s_{yi}}}{\sum_j e^{s_{yj}}} \text{ where } s = f(x, W)$$

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

cat **3.2**

car 5.1

frog -1.7

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

Softmax Classifier (Multinomial Logistic Regression)



Scores = unnormalized log prob. of the classes

$$P(Y = k|X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \text{ where } s = f(x, W)$$

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

cat **3.2**

car 5.1

frog -1.7

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

In summary:

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

Softmax Classifier (Multinomial Logistic Regression)



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

cat	3.2
car	5.1
frog	-1.7

unnormalized log probabilities

Softmax Classifier (Multinomial Logistic Regression)



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

unnormalized probabilities

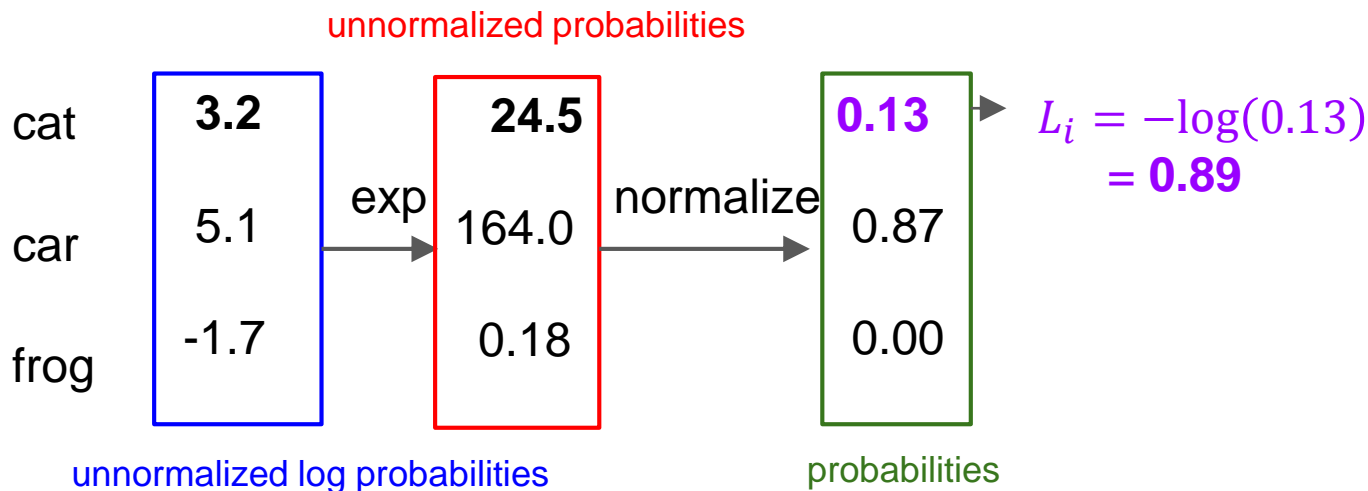
cat	3.2	24.5
car	5.1	164.0
frog	-1.7	0.18

unnormalized log probabilities

Softmax Classifier (Multinomial Logistic Regression)



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

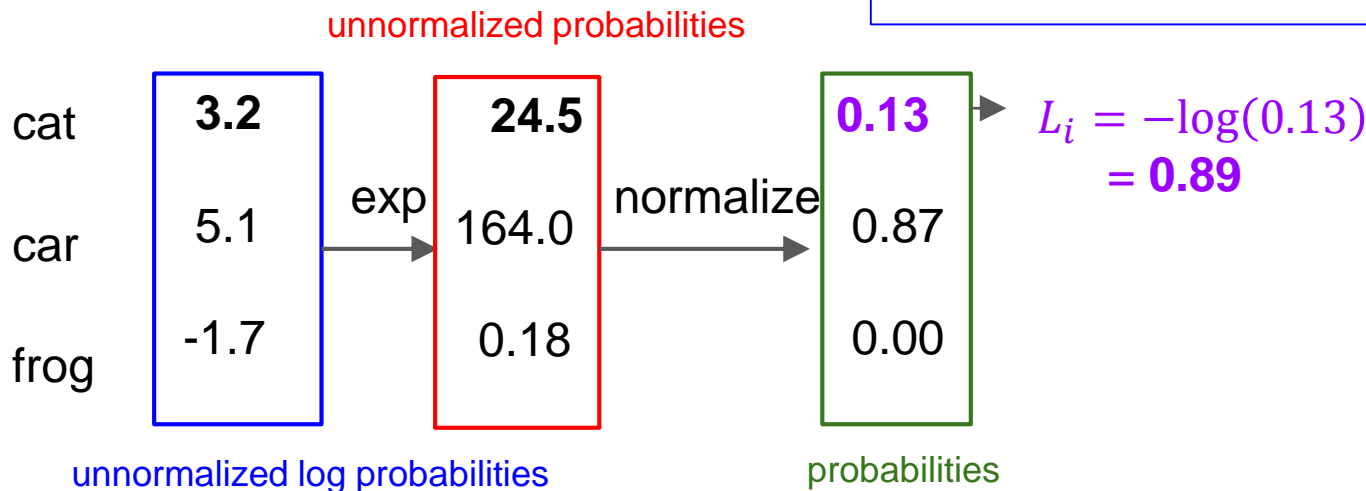


Softmax Classifier (Multinomial Logistic Regression)



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

Q: What is the min/max possible L_i ?

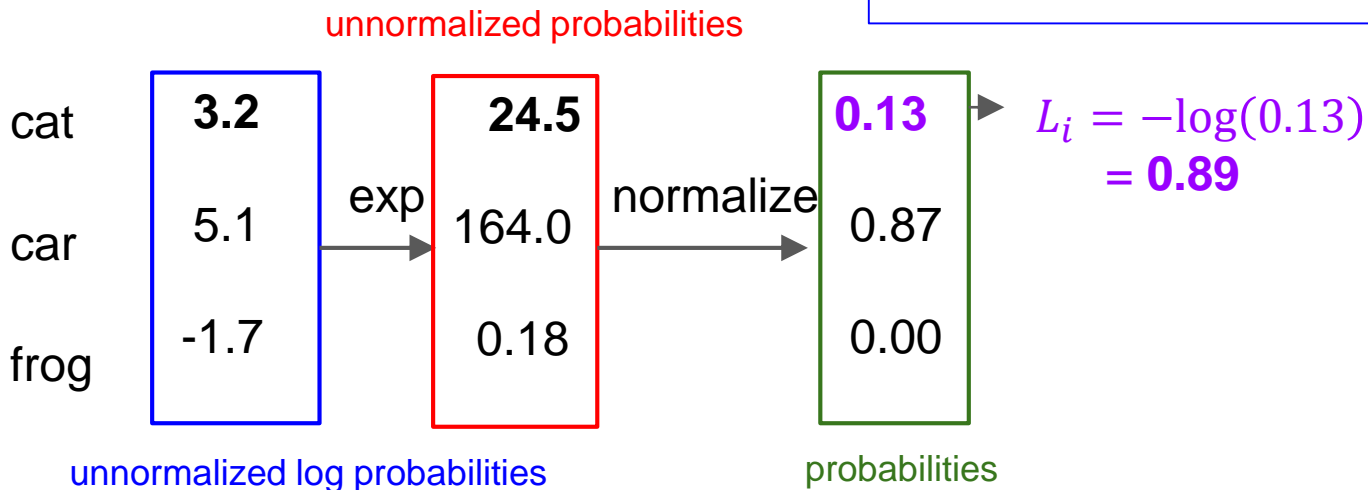


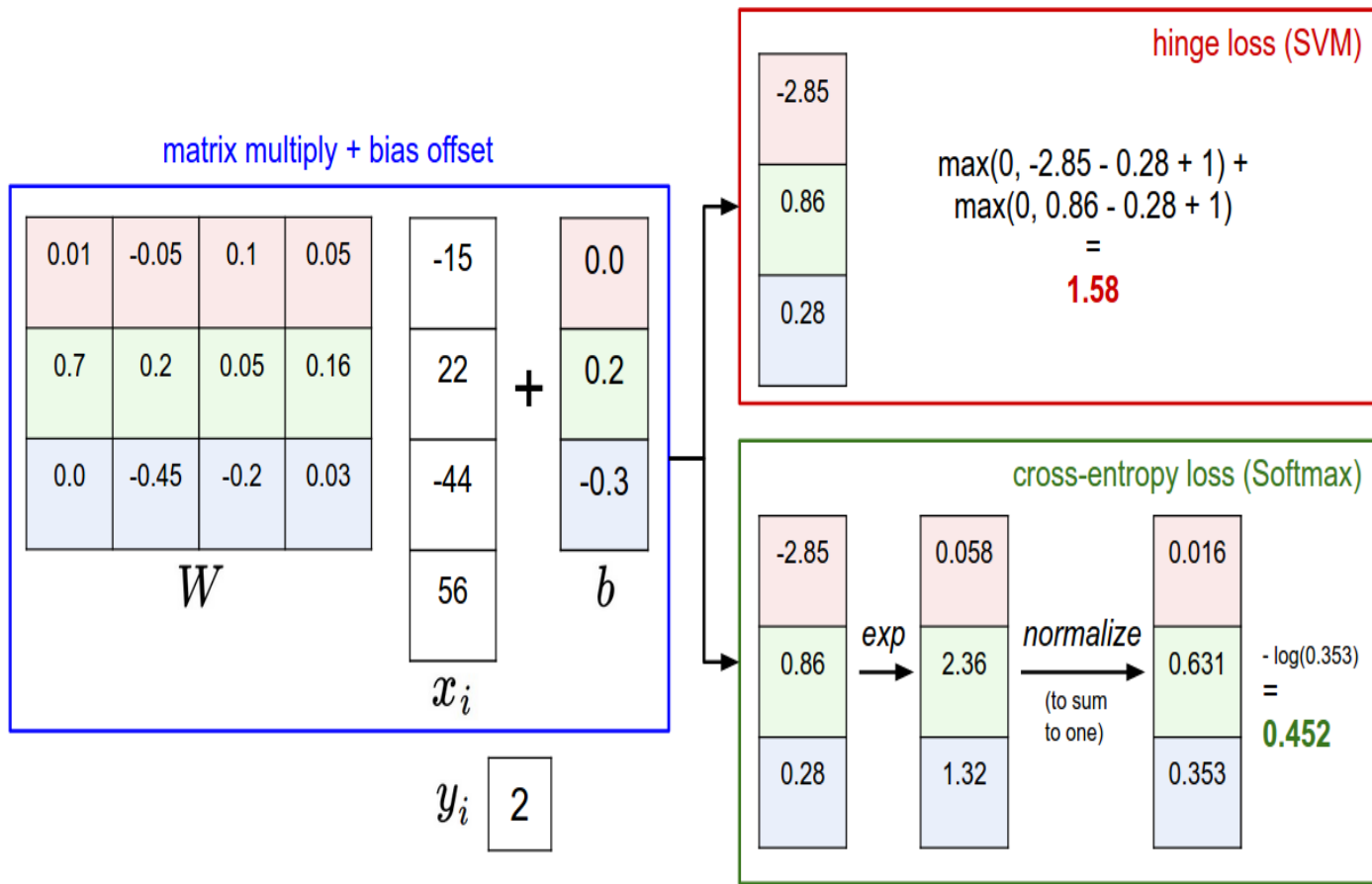
Softmax Classifier (Multinomial Logistic Regression)



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

Q: usually at initialization W are small numbers, so all $s \approx 0$. What is the loss?





Softmax vs. SVM

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

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

Softmax vs. SVM

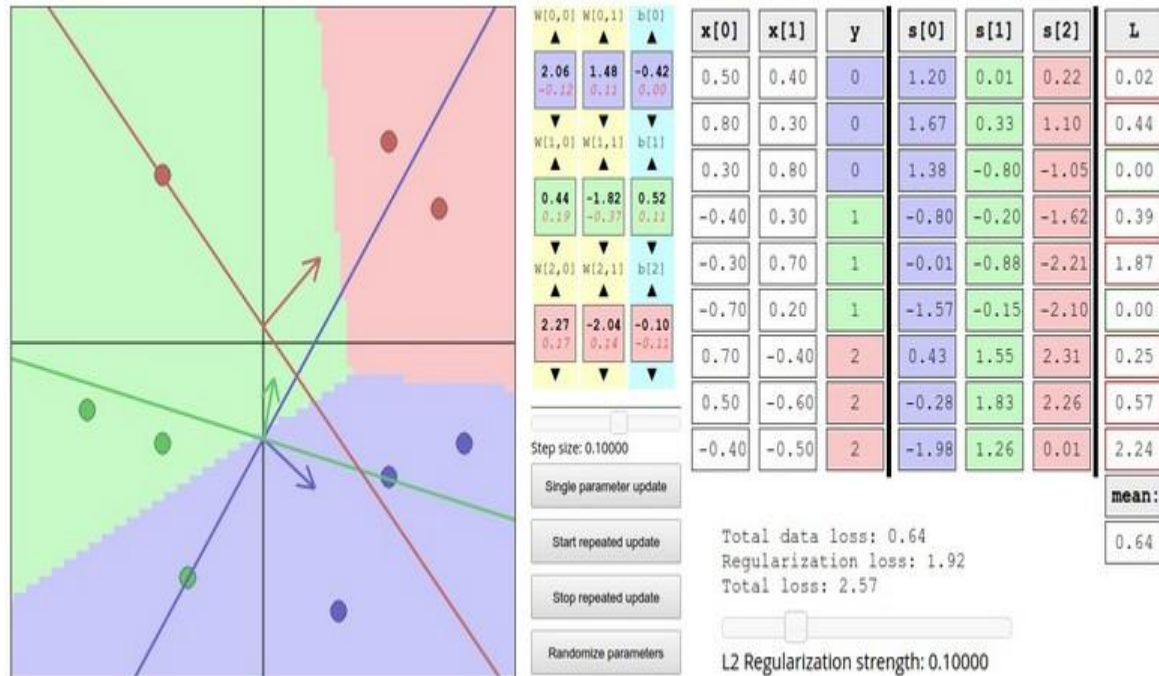
$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right) \qquad L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Multiclass SVM loss is a “hard-margin” loss. Ignores points outside the margin – which is the gap between true class score and any other class score.

Logistic regression has a soft margin – points near the margin have influence that grows as they move toward the wrong side of the decision boundary.

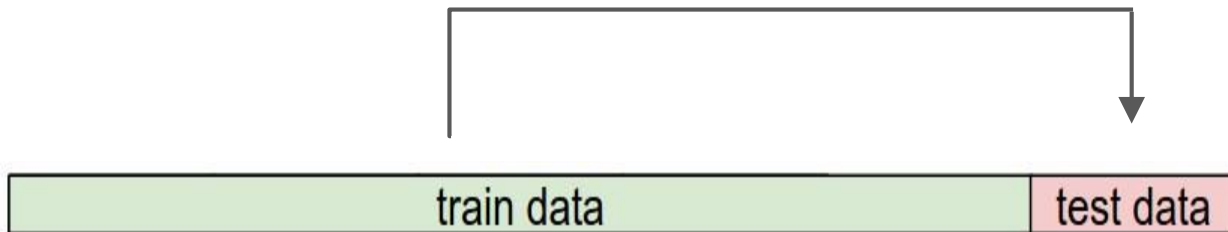
Both losses can be arbitrarily large for poorly-classified points.

Interactive Web Demo time....



<http://vision.stanford.edu/teaching/cs231n/linear-classify-demo/>

Measuring Performance

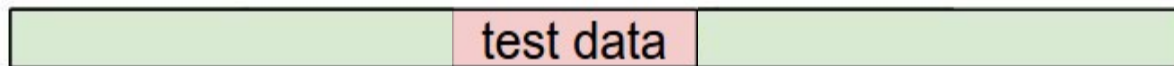


When we measure a classifiers performance, we want to use the most accurate model, so we want to use as much data as possible for training.

But we also want to use as much as possible for testing, in order to get the most accurate measurement.

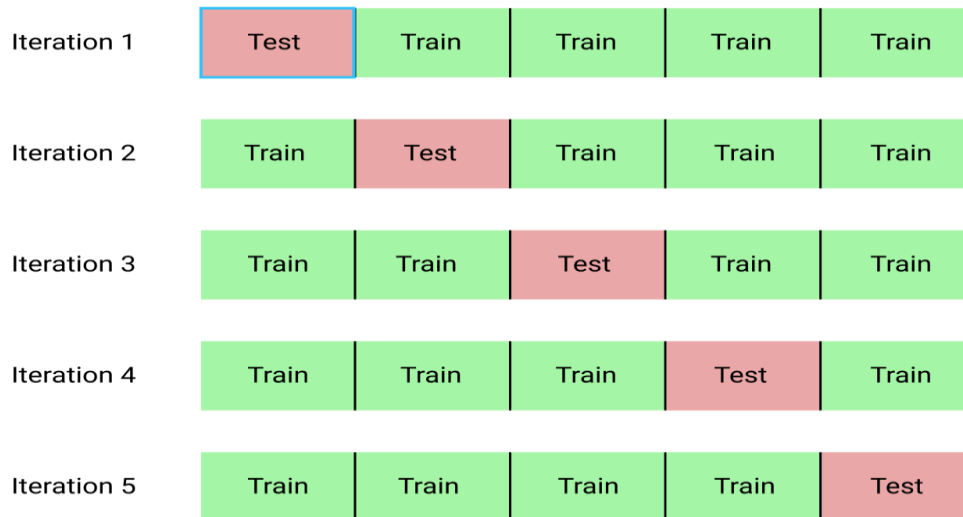
It seems that we can only partition the data as above, but can we do better?

Cross Validation



Yes, we can make other partitionings of the data, like the one above.

If we keep the test samples disjoint, they will give mostly independent measurements. We do this by using a k-fold cross-validation design (k=5 here):



Summary

- We defined a linear classifier's **margin**, and then represented it with a **hinge loss**.
- Maximizing the margin gives a Support Vector Machine (SVM) classifier.
- We generalized from binary SVM classifiers to multi-class.
- We then generalized logistic regression and cross-entropy loss to multiple classes, and derived a **softmax** classifier.
- We explored these methods with an **interactive visualization**.
- We touched on cross-validation as a way to more accurately test a model.
- **Assignment 1** is out, due Feb 12. Please start soon.
- Sections start tomorrow 1pm. Please fill out assignment 0.1 on bCourses to load balance.