Lecture 14:
Linear Classifiers

Administrative

- HW3 due Wednesday, March 4 11:59pm

- TAs will not be checking Piazza over Spring Break. You are **strongly encouraged** to finish the assignment by Friday, February 25

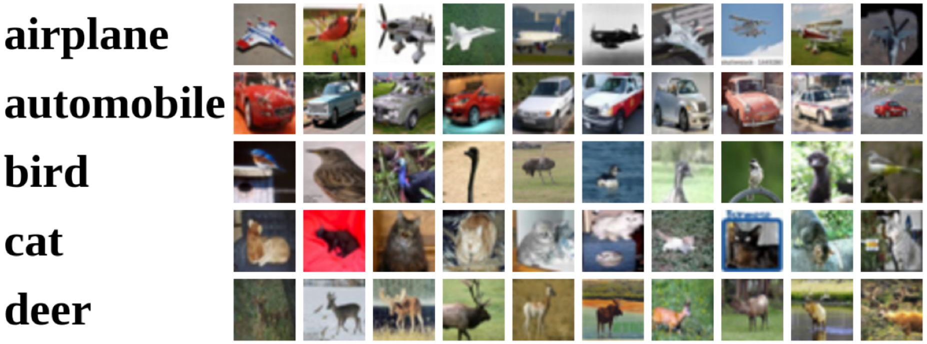
Last Time: Supervised Learning

1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

```
def train(images, labels):
  # Machine learning!
  return model
```

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

Example training set



Last Time: Least Squares

Training ($\mathbf{x}_i$,y_i):

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 \quad \textbf{or}$$

$$\arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} \|\boldsymbol{w}^T \boldsymbol{x}_{\boldsymbol{i}} - y_i\|^2$$

Inference (x):

$$\boldsymbol{w}^T \boldsymbol{x} = w_1 x_1 + \cdots + w_F x_F$$

Testing/Inference:
Given a new output,
what's the prediction?

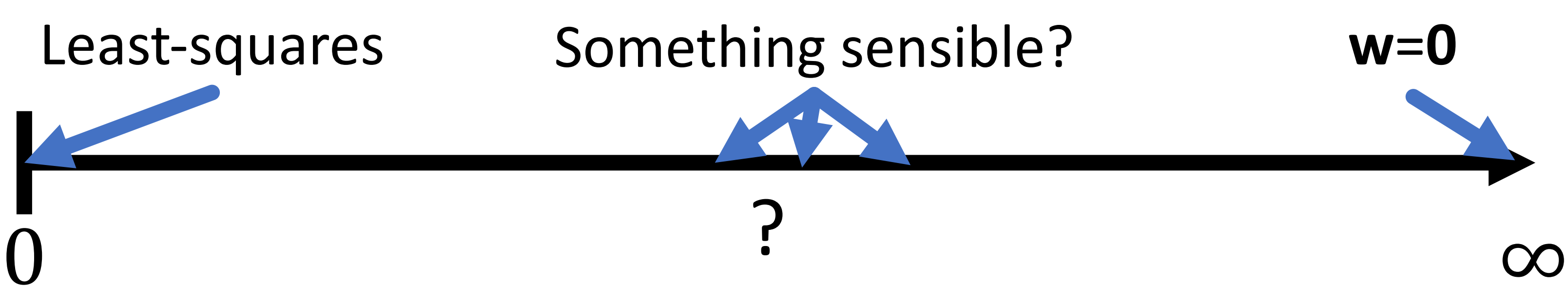
Last Time: Regularization

Objective: $\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{w}\|_2^2$

Loss   Trade-off   Regularization

What happens (and why) if:

- $\lambda = 0$
- $\lambda = \infty$



Hyperparameters

Objective: $\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{w}\|_2^2$

Loss Trade-off Regularization

What happens (and why) if:

- $\lambda=0$
- $\lambda=\infty$

W is a **parameter**, since we optimize for it on the training set

λ is a **hyperparameter**, since we choose it before fitting the training set

Choosing Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: λ =0 always works best on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how we will perform on new data

train	test

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

train	validation	test

Choosing Hyperparameters

Your Dataset

Idea #4: **Cross-Validation**: Split data into **folds**, try each fold as validation and average the results

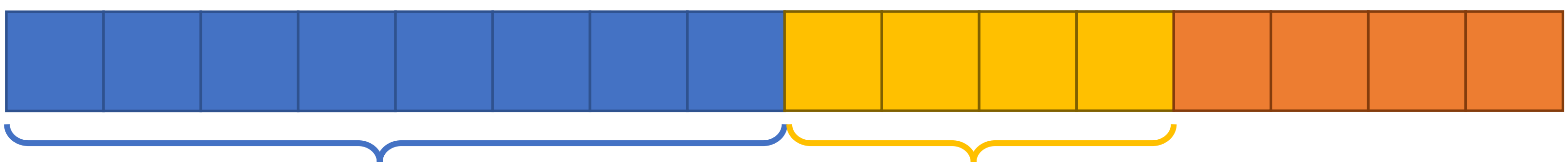
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

Useful for small datasets, but (unfortunately) not used too frequently in deep learning

Training and Testing

Fit model parameters on training set;
find *hyperparameters* by testing on validation set;
evaluate on *entirely unseen* test set.

Training Validation Test

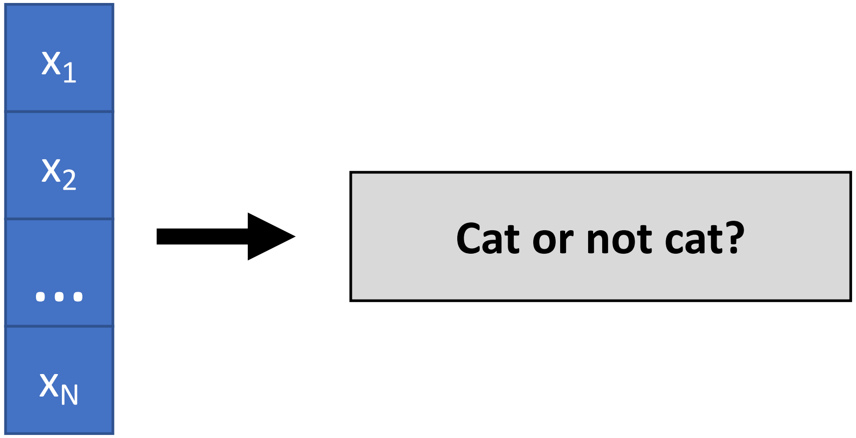


Use these data points to fit $\mathbf{w}^*=(\mathbf{X}^T\mathbf{X}+ \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$

Evaluate on these points for different λ, pick the best

Image Classification

Start with simplest example: binary classification



Actually: a feature vector representing the image

Classification with Least Squares

Treat as regression: x_i is image feature; y_i is 1 if it's a cat, 0 if it's not a cat. Minimize least-squares loss.

Training ($\mathbf{x}_i$,y_i): $$\arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} \|\boldsymbol{w}^T \boldsymbol{x}_{\boldsymbol{i}} - y_i\|^2$$

Inference (x): $$\boldsymbol{w}^T \boldsymbol{x} > t$$

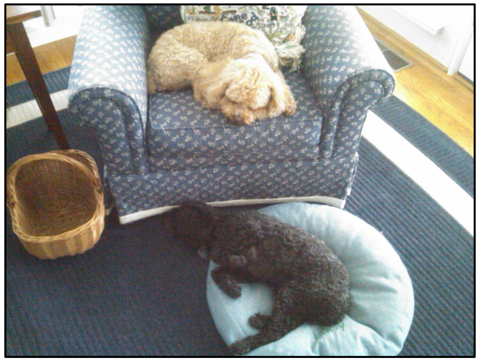
Unprincipled in theory, but often effective in practice
The reverse (regression via discrete bins) is also common

Rifkin, Yeo, Poggio. *Regularized Least Squares Classification* (http://cbcl.mit.edu/publications/ps/rlsc.pdf). 2003
Redmon, Divvala, Girshick, Farhadi. *You Only Look Once: Unified, Real-Time Object Detection.* CVPR 2016.

Classification via Memorization

Just **memorize** (as in a Python dictionary)
Consider cat/dog/hippo classification.

If this:
cat.



If this:
dog.

If this:
hippo.

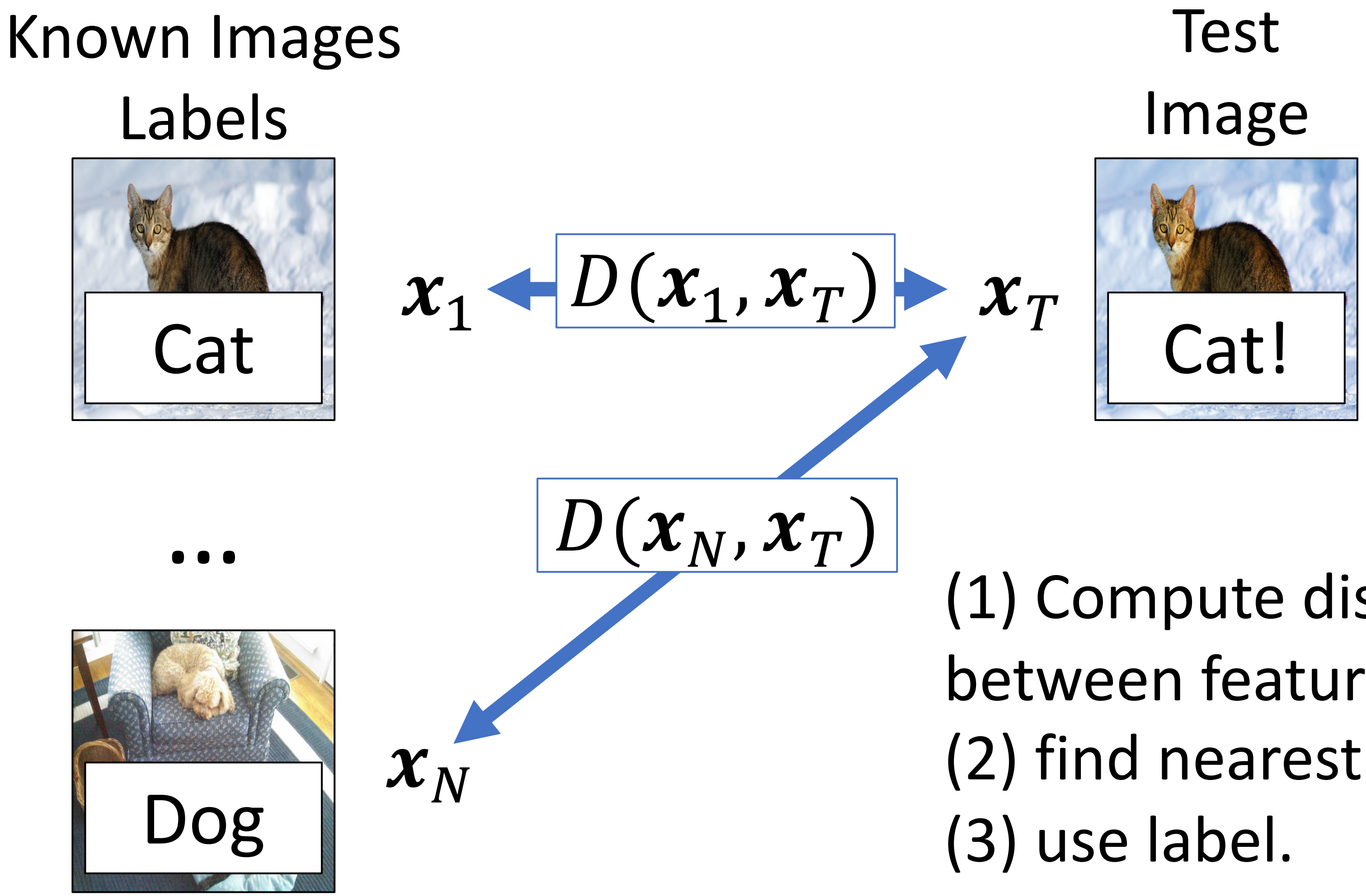
Classification via Memorization

Where does this go wrong?

Rule: if this, then cat

Hmmm. Not quite the same.

Classification via Memorization



(1) Compute distance between feature vectors
(2) find nearest
(3) use label.

Nearest Neighbor

"Algorithm"

Training ($\mathbf{x}_i$,y_i): Memorize training set

Inference (x):

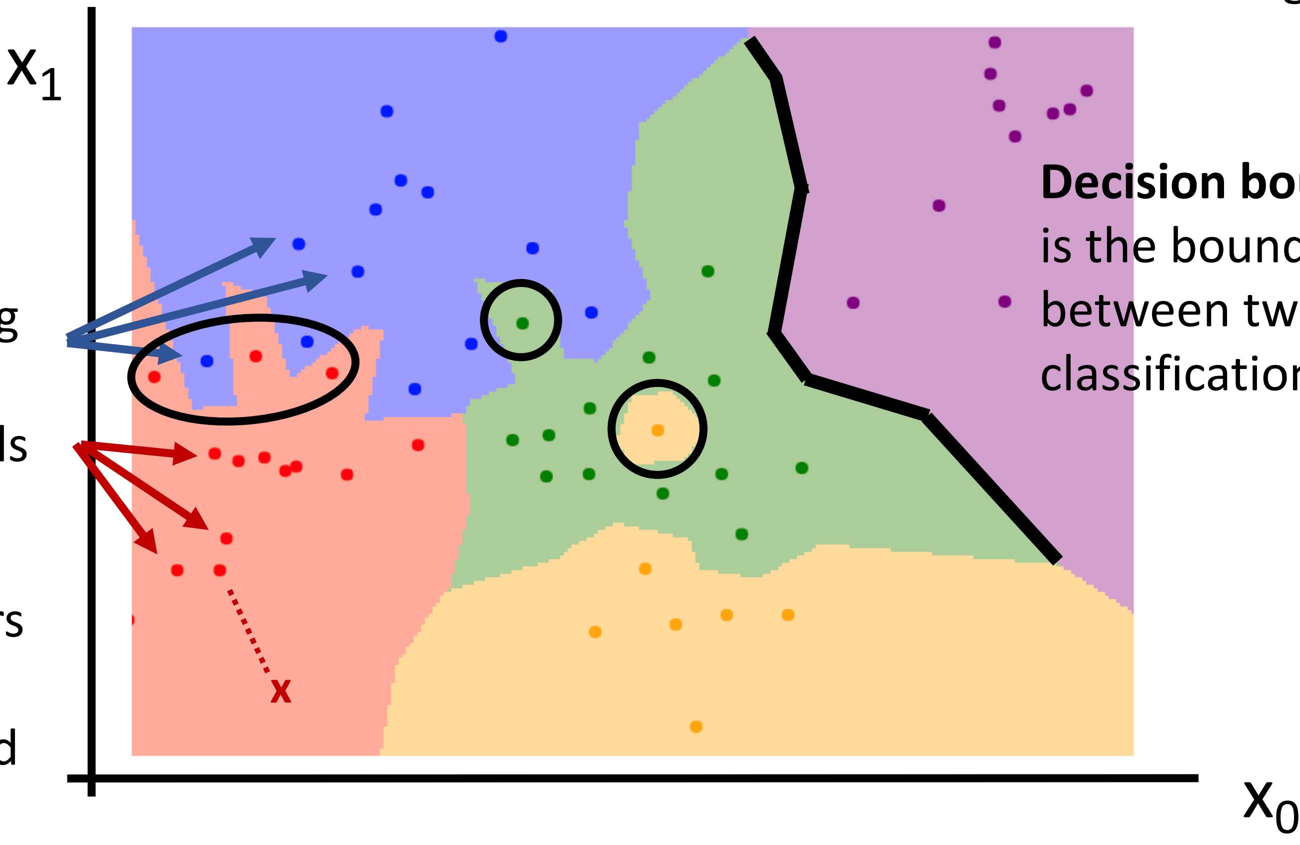
```
bestDist, prediction = Inf, None
for i in range(N):
    if dist(x_i,x) < bestDist:
        bestDist = dist(x_i,x)
        prediction = y_i
```

Nearest Neighbor

Nearest neighbors in two dimensions

Decision boundaries can be noisy; affected by outliers

How to smooth out decision boundaries? Use more neighbors!

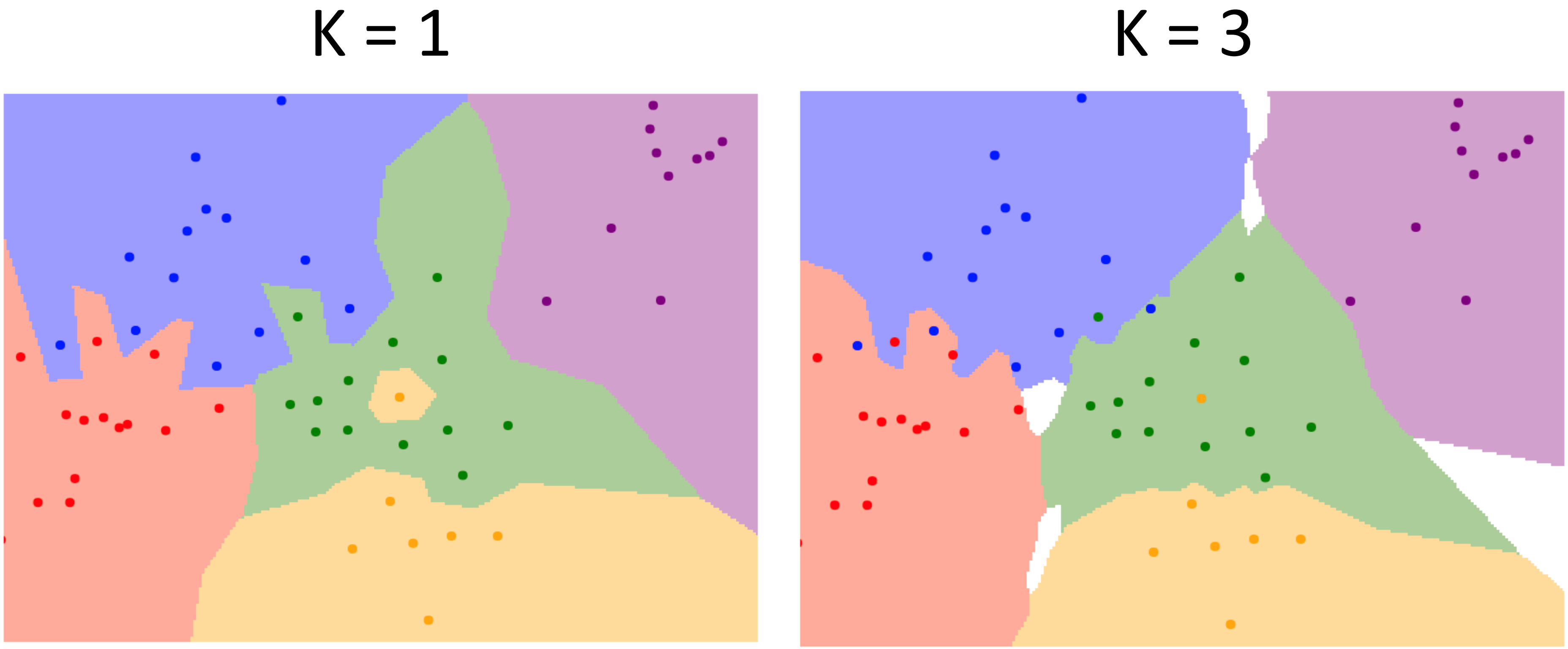


Decision boundary is the boundary between two classification regions

Points are training examples; colors give training labels

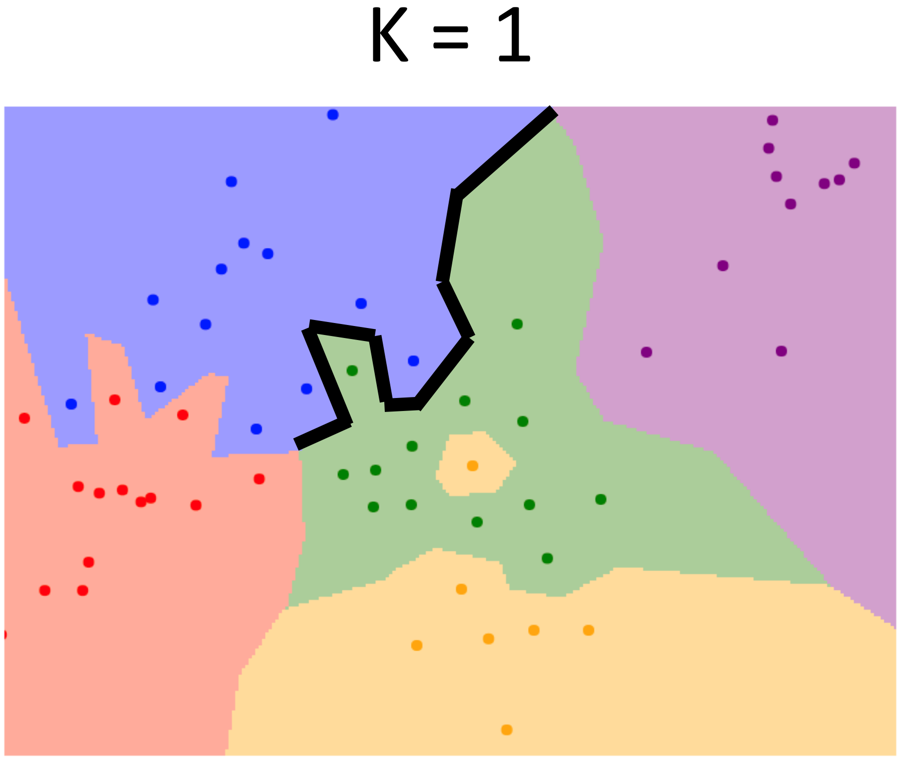
Background colors give the category a test point would be assigned

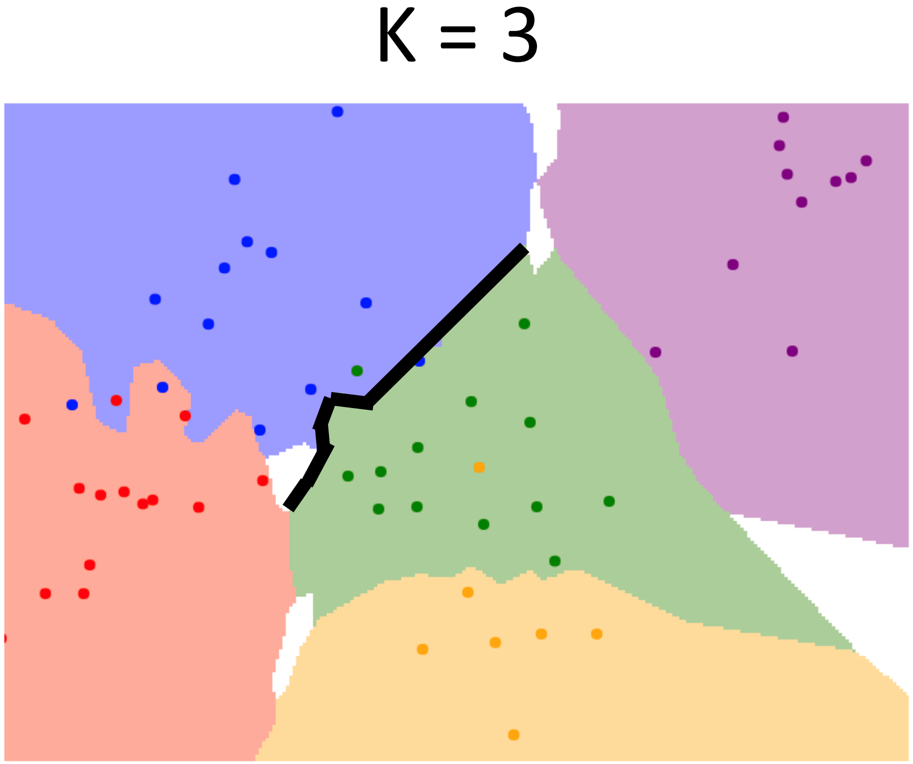
K-Nearest Neighbors



Instead of copying label from nearest neighbor, take **majority vote** from K closest points

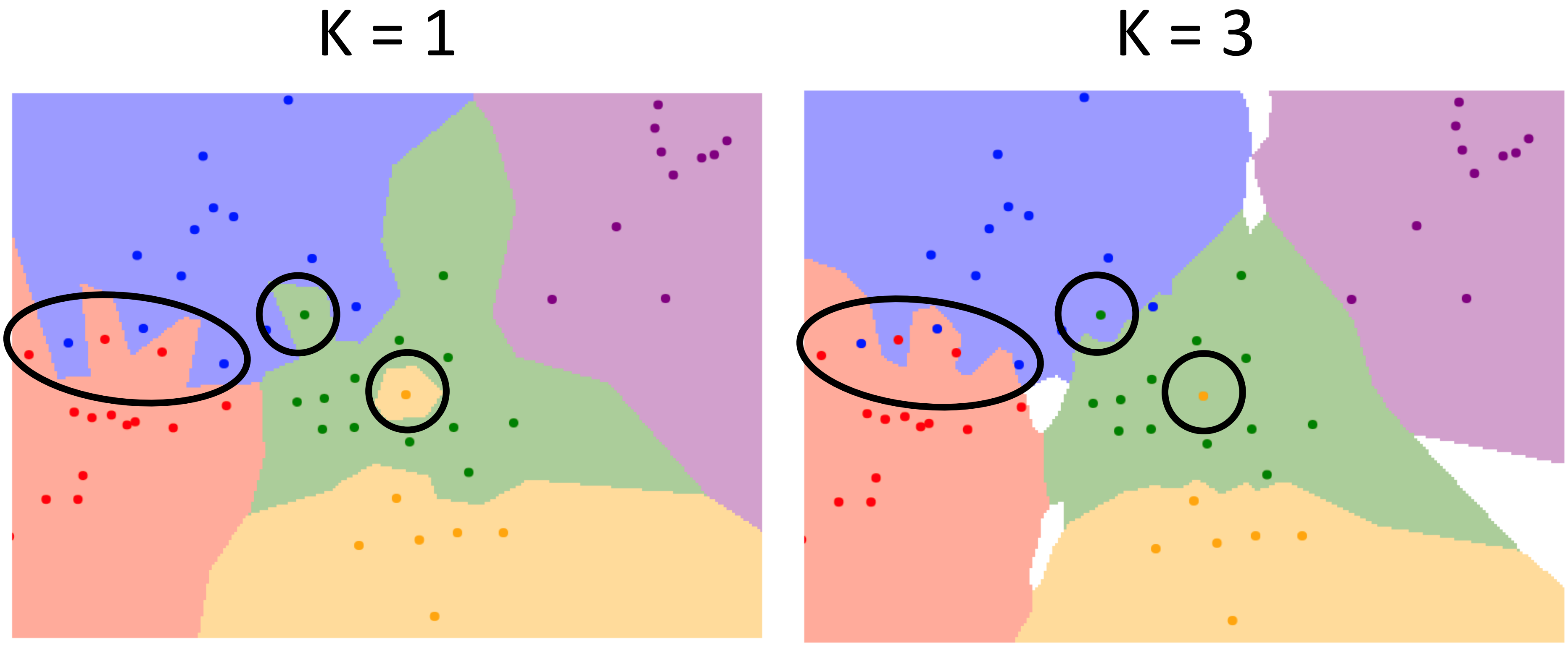
K-Nearest Neighbors





Using more neighbors helps smooth out rough decision boundaries

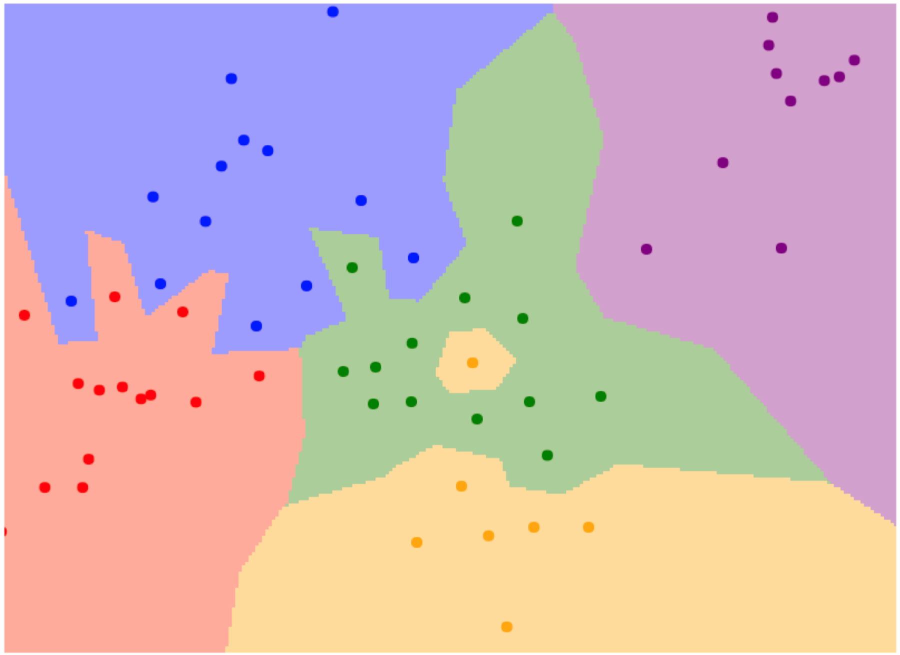
K-Nearest Neighbors



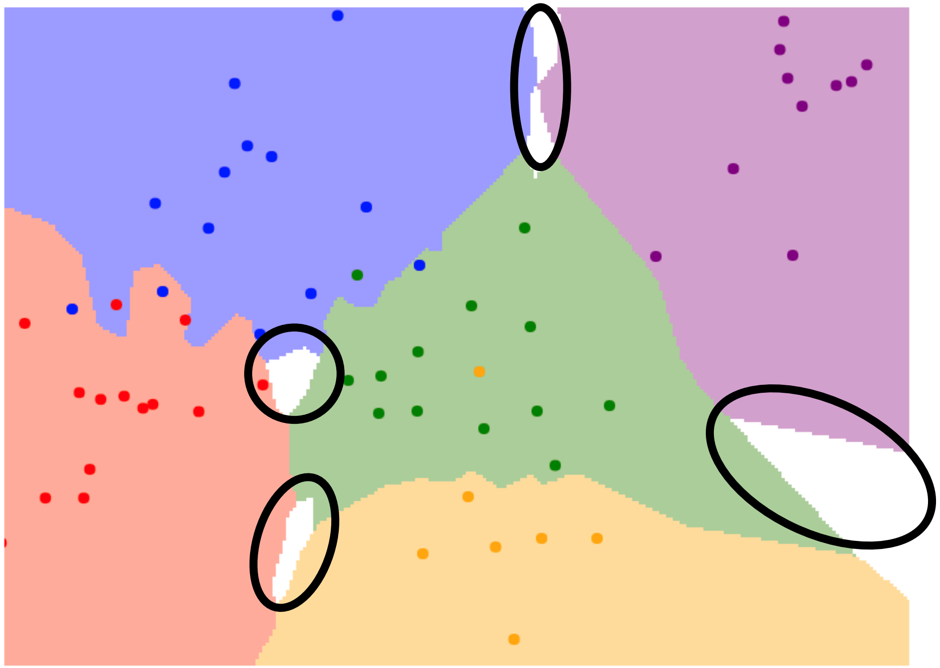
Using more neighbors helps
reduce the effect of outliers

K-Nearest Neighbors

K = 1



K = 3

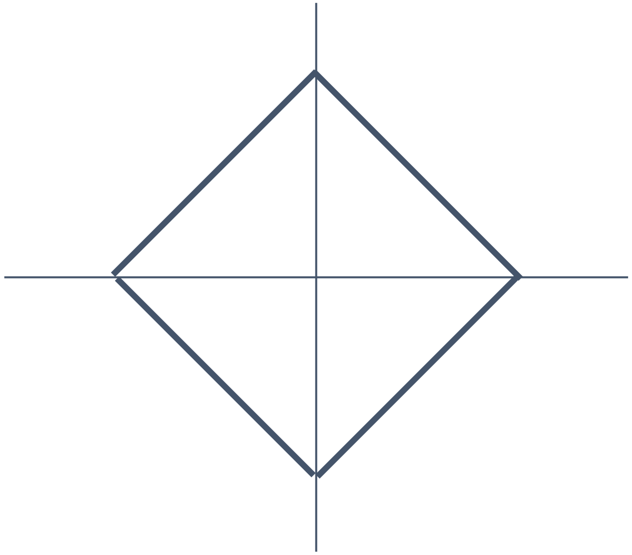


When K > 1 there can be ties!
Need to break them somehow

K-Nearest Neighbors: Distance Metric

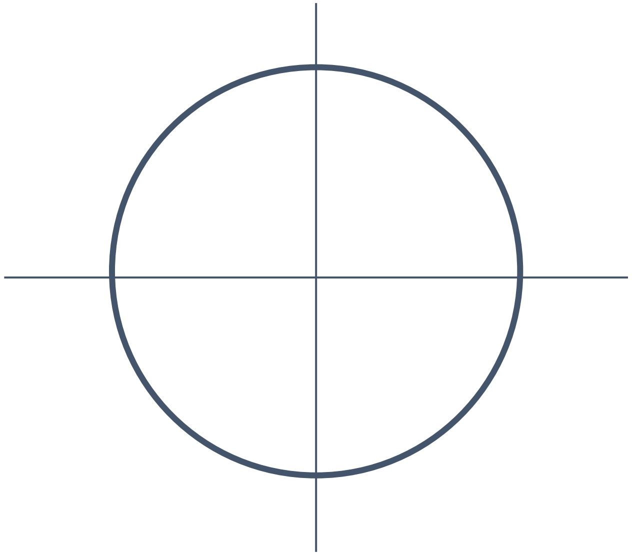
L1 (Manhattan) Distance

$$d(x, y) = \sum_i |x_i - y_i|$$



L2 (Euclidean) Distance

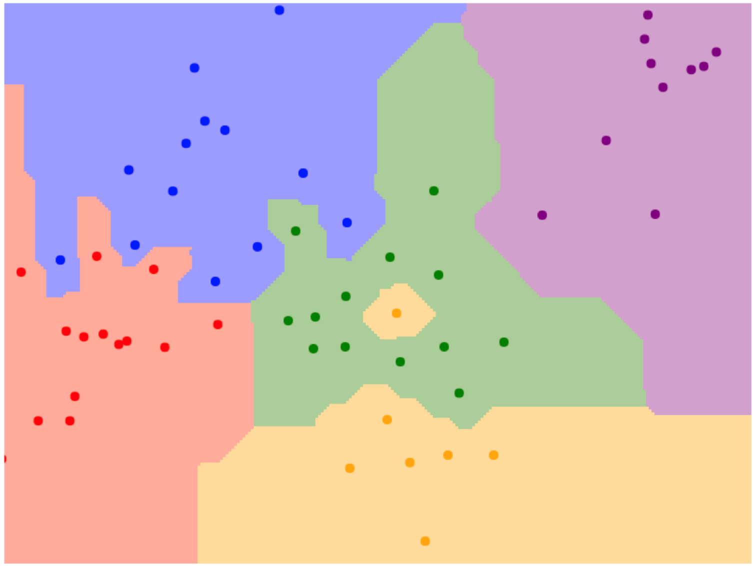
$$d(x, y) = \left(\sum_i (x_i - y_i)^2\right)^{1/2}$$



K-Nearest Neighbors: Distance Metric

L1 (Manhattan) Distance

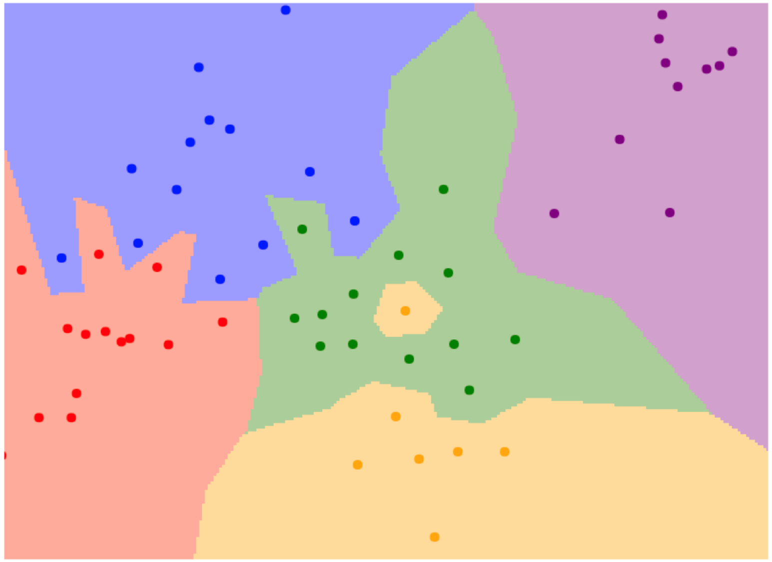
$$d(x,y) = \sum_i |x_i - y_i|$$



K = 1

L2 (Euclidean) Distance

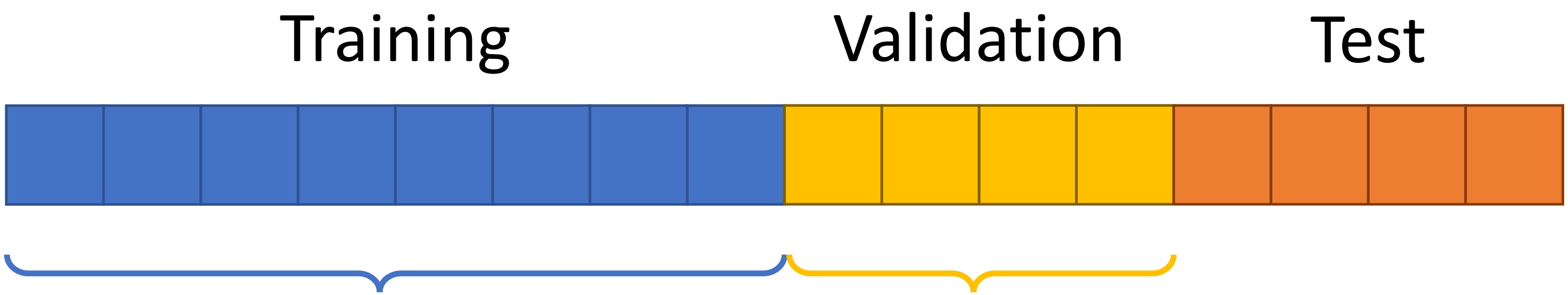
$$d(x,y) = \left(\sum_i (x_i - y_i)^2\right)^{1/2}$$



K = 1

K-Nearest Neighbors

What distance? What value for K?



Use these data points for lookup

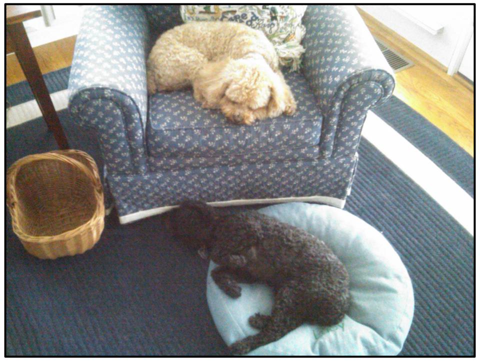
Evaluate on these points for different k, distances

K-Nearest Neighbors

- No learning going on but usually effective
- Same algorithm for every task
- As number of datapoints $\rightarrow \infty$, error rate is guaranteed to be at most 2x worse than optimal you could do on data
- Training is fast, but inference is slow. Opposite of what we want!

Linear Classifiers

Example Setup: 3 classes

Model – one weight per class: $\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2$

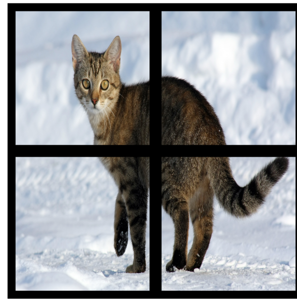
$\boldsymbol{w}_0^T\boldsymbol{x}$ big if cat

$\boldsymbol{w}_1^T\boldsymbol{x}$ big if dog

$\boldsymbol{w}_2^T\boldsymbol{x}$ big if hippo

Stack together: $\boldsymbol{W}_{\boldsymbol{3xF}}$ where **x** is in R^F

Linear Classifiers



Cat weight vector	0.2	-0.5	0.1	2.0	1.1
Dog weight vector	1.5	1.3	2.1	0.0	3.2
Hippo weight vector	0.0	0.3	0.2	-0.3	-1.2

W

56
231
24
2
1

x_i

-96.8	Cat score
437.9	Dog score
61.95	Hippo score

Wx_i

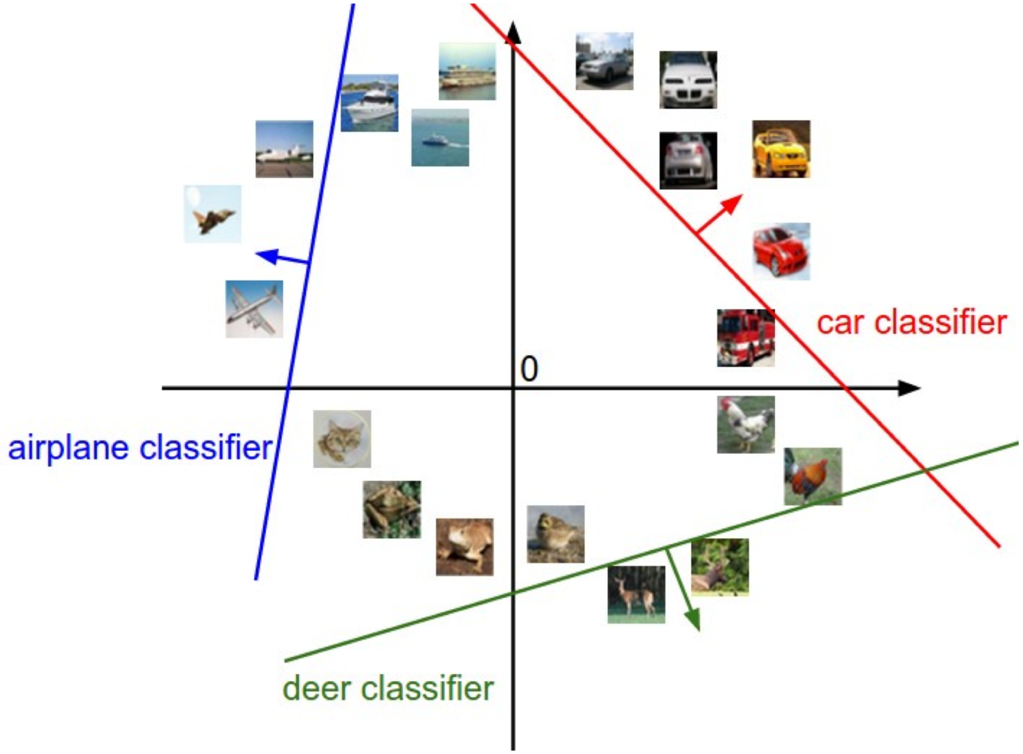
Weight matrix a collection of scoring functions, one per class

Prediction is vector where jth component is "score" for jth class.

Diagram by: Karpathy, Fei-Fei

Linear Classifiers: Geometric Intuition

What does a linear classifier look like in 2D?

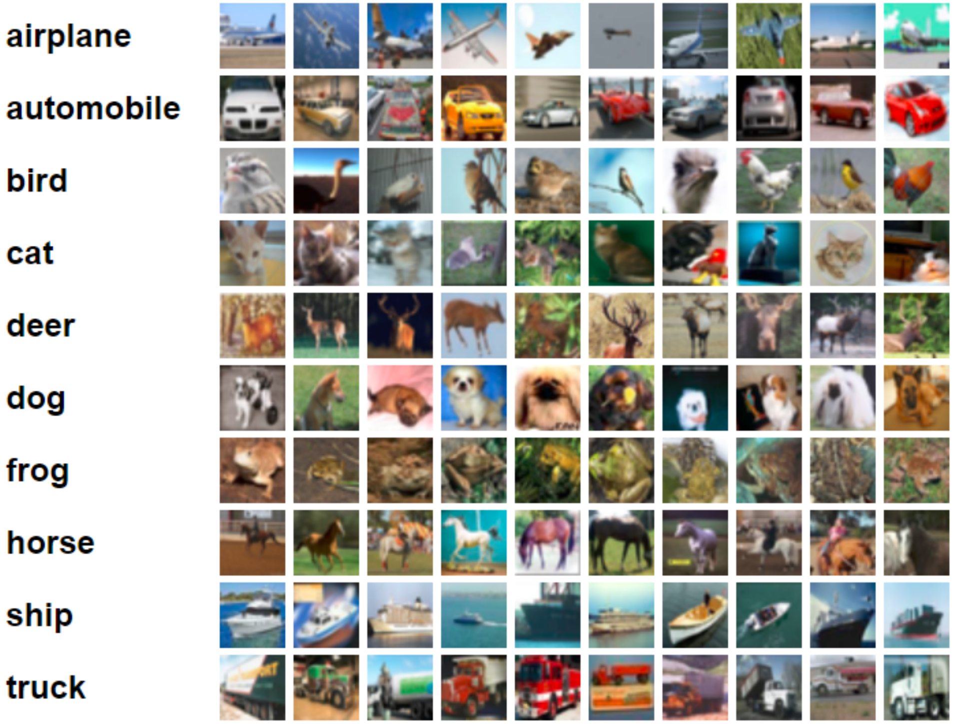


Be aware: Intuition from 2D doesn't always carry over into high-dimensional spaces. See: *On the Surprising Behavior of Distance Metrics in High Dimensional Space.* Charu, Hinneburg, Keim. ICDT 2001

Diagram credit: Karpathy & Fei-Fei

Linear Classifiers: Visual Intuition

CIFAR 10:
32x32x3 Images, 10 Classes

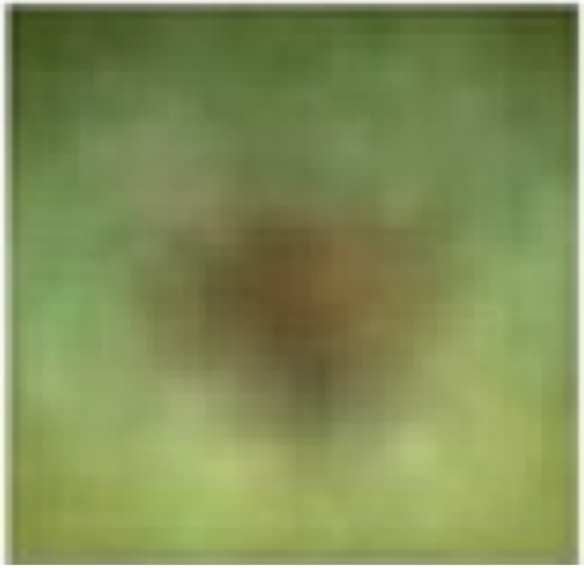


- Turn each image into feature by unrolling all pixels
- Train a linear model to recognize 10 classes

Linear Classifiers: Visual Intuition

Decision rule is $\mathbf{w}^T\mathbf{x}$. If $\mathbf{w}_i$ is big, then big values of x_i are indicative of the class.

Deer or Plane?



Linear Classifiers: Visual Intuition

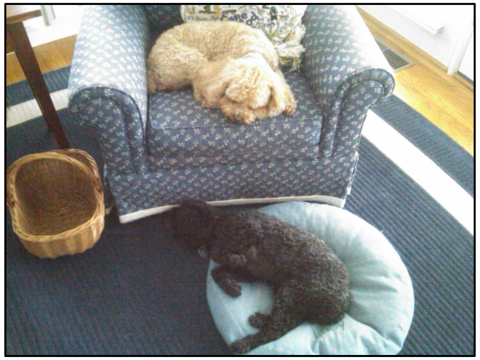
Decision rule is $\mathbf{w}^T\mathbf{x}$. If $\mathbf{w}_i$ is big, then big values of x_i are indicative of the class.

Ship or Dog?

Linear Classifiers: Visual Intuition

Decision rule is $\mathbf{w}^T\mathbf{x}$. If $\mathbf{w}_i$ is big, then big values of x_i are indicative of the class.

So Far: Linear Score Function

Model – one weight per class: $\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2$

$\boldsymbol{w}_0^T\boldsymbol{x}$ big if cat

$\boldsymbol{w}_1^T\boldsymbol{x}$ big if dog

$\boldsymbol{w}_2^T\boldsymbol{x}$ big if hippo

Stack together: $\boldsymbol{W}_{\boldsymbol{3xF}}$ where **x** is in R^F

How do we know which W is best?

Choosing W: Loss Function

A **loss function** tells how good our current classifier is

Low loss = good classifier
High loss = bad classifier

(Also called: **objective function**; **cost function**)

Negative loss function sometimes called **reward function**, **profit function**, **utility function**, **fitness function**, etc

Given a dataset

$$\{(x_i, y_i)\}_{i=1}^{N}$$

of images x_i and labels y_i,

Loss for a single example is:

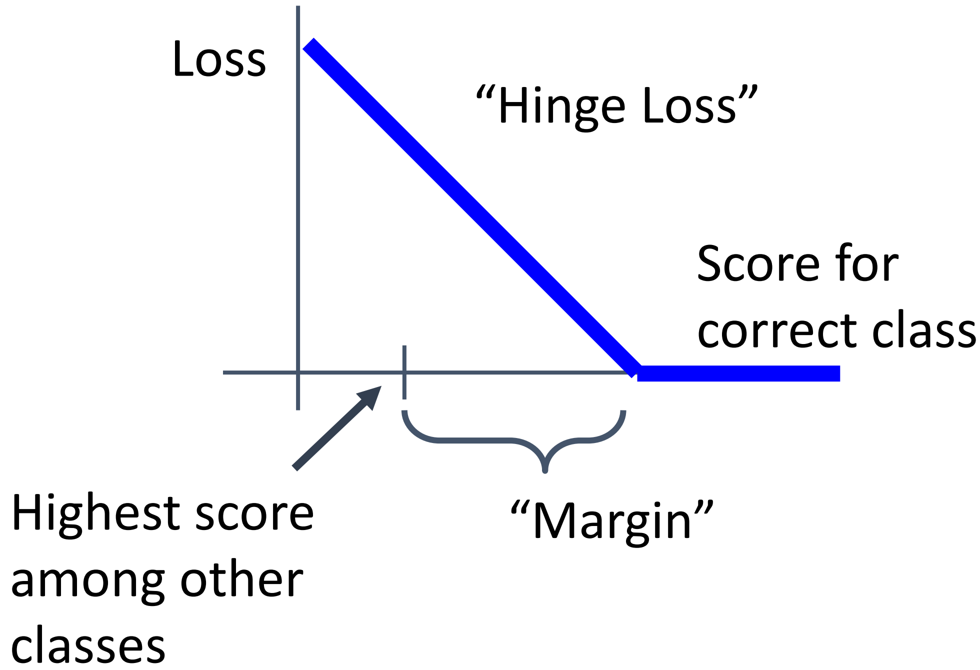
$$L_i(f(x_i, W), y_i)$$

Loss for the dataset is

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i)$$

Multiclass SVM Loss

"The score of the correct class should be higher than all the other scores"



Given an example (x_i, y_i)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

Multiclass SVM Loss

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)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

Multiclass SVM Loss

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)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

Multiclass SVM Loss

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9		

Given an example (x_i, y_i)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

= max(0, 5.1 - 3.2 + 1)
+ max(0, -1.7 - 3.2 + 1)
= max(0, 2.9) + max(0, -3.9)
= 2.9 + 0
= 2.9

Multiclass SVM Loss

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	

Given an example (x_i, y_i)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

= max(0, 1.3 - 4.9 + 1)
+max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

Multiclass SVM Loss



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Given an example (x_i, y_i)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

= max(0, 2.2 - (-3.1) + 1)
+max(0, 2.5 - (-3.1) + 1)
= max(0, 6.3) + max(0, 6.6)
= 6.3 + 6.6
= 12.9

Multiclass SVM Loss

			
cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Given an example (x_i, y_i)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

Loss over the dataset is:

L = (2.9 + 0.0 + 12.9) / 3
= 5.27

Multiclass SVM Loss

Given an example (x_i, y_i)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

			
cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Q: What happens to the loss if the scores for the car image change a bit?

Multiclass SVM Loss

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Given an example (x_i, y_i)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

Q: What are the min and max possible loss?

Multiclass SVM Loss

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Given an example (x_i, y_i)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

Q: If all scores were random, what loss would we expect?

Multiclass SVM Loss

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Given an example (x_i, y_i)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

Q: What would happen if sum were over all classes? (including $j = y_i$)

Multiclass SVM Loss

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Given an example (x_i, y_i)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then 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 loss used mean instead of sum?

Multiclass SVM Loss

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Given an example (x_i, y_i)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

Q: What if we used this loss instead?

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

Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

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

cat	**3.2**
car	5.1
frog	-1.7

Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

Softmax function

Classifier scores

$s = f(x_i, W)$

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

cat	**3.2**
car	5.1
frog	-1.7

Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

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

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

cat	**3.2**
car	5.1
frog	-1.7

Unnormalized log-probabilities / logits

Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

Classifier scores

$s = f(x_i, W)$

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Probabilities must be >= 0

cat	**3.2**		**24.5**
car	5.1	$\xrightarrow{\text{exp}}$	164
frog	-1.7		0.18

Unnormalized log-probabilities / logits

unnormalized probabilities

Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

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

Softmax function
$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Probabilities must be >= 0

Probabilities must sum to 1

			exp		normalize	
cat	**3.2**	$\rightarrow$	**24.5**	$\rightarrow$	**0.13**
car	5.1		164		0.87
frog	-1.7		0.18		0.00

Unnormalized log-probabilities / logits

unnormalized probabilities

probabilities

Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

Classifier scores

$s = f(x_i, W)$

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss

$$L_i = -\log(p_{y_i})$$

Probabilities must be >= 0

Probabilities must sum to 1

	Unnormalized log-probabilities / logits	$\xrightarrow{\text{exp}}$	unnormalized probabilities	$\xrightarrow{\text{normalize}}$	probabilities
cat	**3.2**		**24.5**		**0.13**
car	5.1		164		0.87
frog	-1.7		0.18		0.00

$L_i = -\log(0.13)$
$= \mathbf{2.04}$

Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

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

Softmax function
$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss
$$L_i = -\log(p_{y_i})$$

Probabilities must be >= 0

Probabilities must sum to 1

	Unnormalized log-probabilities / logits	exp →	unnormalized probabilities	normalize →	probabilities
cat	**3.2**		**24.5**		**0.13**
car	5.1		164		0.87
frog	-1.7		0.18		0.00

$L_i = -\log(0.13)$
$= \mathbf{2.04}$

Maximum Likelihood Estimation
Choose weights to maximize the likelihood of the observed data
(See EECS 445 or EECS 545)

Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

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

Softmax function
$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss
$$L_i = -\log(p_{y_i})$$

Probabilities must be >= 0

Probabilities must sum to 1

	Unnormalized log-probabilities / logits	$\xrightarrow{\text{exp}}$ unnormalized probabilities	$\xrightarrow{\text{normalize}}$ probabilities	→ Compare ←	Correct probs
cat	**3.2**	**24.5**	**0.13**		**1.00**
car	5.1	164	0.87		0.00
frog	-1.7	0.18	0.00		0.00

Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

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

Softmax function
$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss
$$L_i = -\log(p_{y_i})$$

Probabilities must be >= 0

Probabilities must sum to 1

	Unnormalized log-probabilities / logits		unnormalized probabilities		probabilities		Correct probs
cat	**3.2**	exp →	**24.5**	normalize →	**0.13**	→ Compare ←	**1.00**
car	5.1		164		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Kullback-Leibler Divergence
$$D_{KL}(\boldsymbol{P} \,||\, \boldsymbol{Q}) = \sum_y \boldsymbol{P}(y) \log \frac{\boldsymbol{P}(y)}{\boldsymbol{Q}(y)}$$

Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

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

Softmax function
$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss
$$L_i = -\log(p_{y_i})$$

Probabilities must be >= 0

Probabilities must sum to 1

	Unnormalized log-probabilities / logits	exp →	unnormalized probabilities	normalize →	probabilities	→ Compare ←	Correct probs
cat	**3.2**		**24.5**		**0.13**		**1.00**
car	5.1		164		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Cross-Entropy:
$$H(P, Q) = H(P) + D_{KL}(P \,||\, Q)$$

Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

cat	**3.2**
car	5.1
frog	-1.7

Classifier scores

$s = f(x_i, W)$

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss

$$L_i = -\log(p_{y_i})$$

Putting it all together:

$$L_i = -\log \frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}$$

Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

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

Softmax function
$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss
$$L_i = -\log(p_{y_i})$$

cat	**3.2**
car	5.1
frog	-1.7

Putting it all together:

$$L_i = -\log \frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}$$

Q: What is the min / max possible loss L_i?

Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

cat	**3.2**
car	5.1
frog	-1.7

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

Softmax function
$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss
$$L_i = -\log(p_{y_i})$$

Putting it all together:
$$L_i = -\log \frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}$$

Q: If all scores are small random values, what is the loss?

Cross-Entropy vs SVM Loss

$$L_i = -\log \frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}$$

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

Assume scores:
[10, -2, 3]
[10, 9, 9]
[10, -100, -100]
and $y_i = 0$

Q: What is cross-entropy loss? What is SVM loss?

A: Cross-entropy loss > 0
SVM loss = 0

Cross-Entropy vs SVM Loss

$$L_i = -\log \frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}$$

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

Assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and $y_i = 0$

Q: What happens to each loss if I slightly change the scores of the last datapoint?

A: Cross-entropy loss will change; SVM loss will stay the same

Cross-Entropy vs SVM Loss

$$L_i = -\log \frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}$$

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

Assume scores:
[10, -2, 3]
[10, 9, 9]
[10, -100, -100]
and $y_i = 0$

Q: What happens to each loss if I double the score of the correct class from 10 to 20?

A: Cross-entropy loss will decrease, SVM loss still 0

Next Time:
How to choose W?
Optimization!