Machine Learning and Data Mining

Lecture 4.1: Naïve Bayes (cont.) and Multiclass Classification

CS 434

© Stefan Lee

RECAP

From Last Lecture

© Stefan Lee

What is a Bayes Classifier?

Optimal Bayes Classifier:

Suppose we know the true distribution $P^*(Y|\boldsymbol{X})$ and for each $\boldsymbol{x}$ we encounter we predict:

$$\hat{y} = argmax_y \, P^*(Y = y|\boldsymbol{X} = \boldsymbol{x})$$

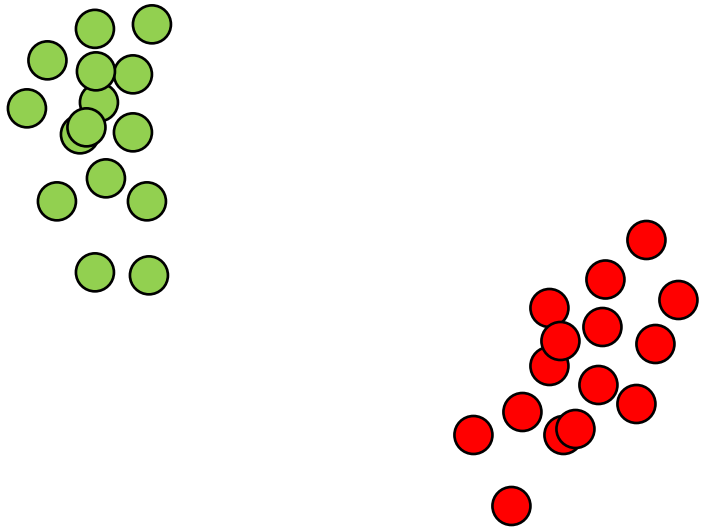
- If we know the true $P^*(Y|\boldsymbol{X})$, this is optimal.
  https://en.wikipedia.org/wiki/Bayes_classifier

Problem: We don't know the true P(Y|X). How to learn it?

© Stefan Lee

Problem: We don't know the true $P(y|x)$. How to learn it?

Consider the following binary classification problem (colors for labels):



In logistic regression, we directly tried to model $P(y|\boldsymbol{x})$ -- assuming it was $\sigma(w^T\boldsymbol{x})$ and learning $\boldsymbol{w}$ with MLE.

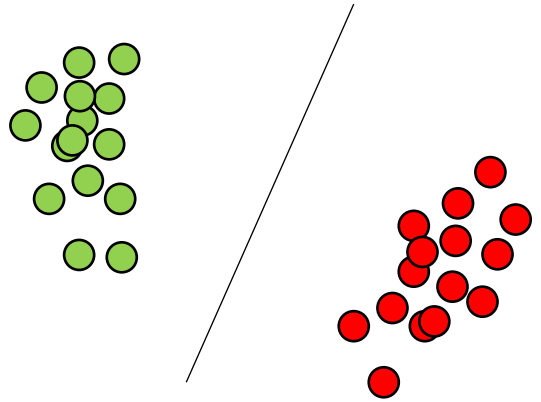
An alternative approach would be to model each cluster separately – i.e. modelling $P(\boldsymbol{x}|y)$.

© Stefan Lee

Generative vs. Discriminative Models

Discriminative Classifiers:

- Learn $P(y|\boldsymbol{x})$ directly
- Logistic regression is one example
- *Nomenclature note -- people will also refer to algorithms that model no distribution as discriminative (such as kNN).*

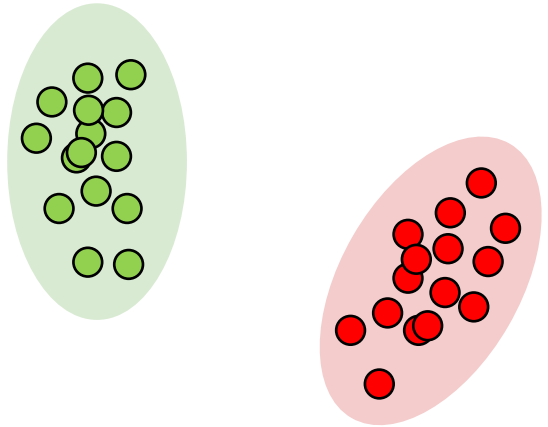


Generative Classifiers:

- Learn $P(\boldsymbol{x}|y)$ and $P(y)$
- Compute $P(y|\boldsymbol{x})$ using Bayes Rule

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})} = \frac{P(\boldsymbol{x}|y)P(y)}{\sum_y P(\boldsymbol{x}|y)P(y)}$$

- Naïve Bayes is one example (today)



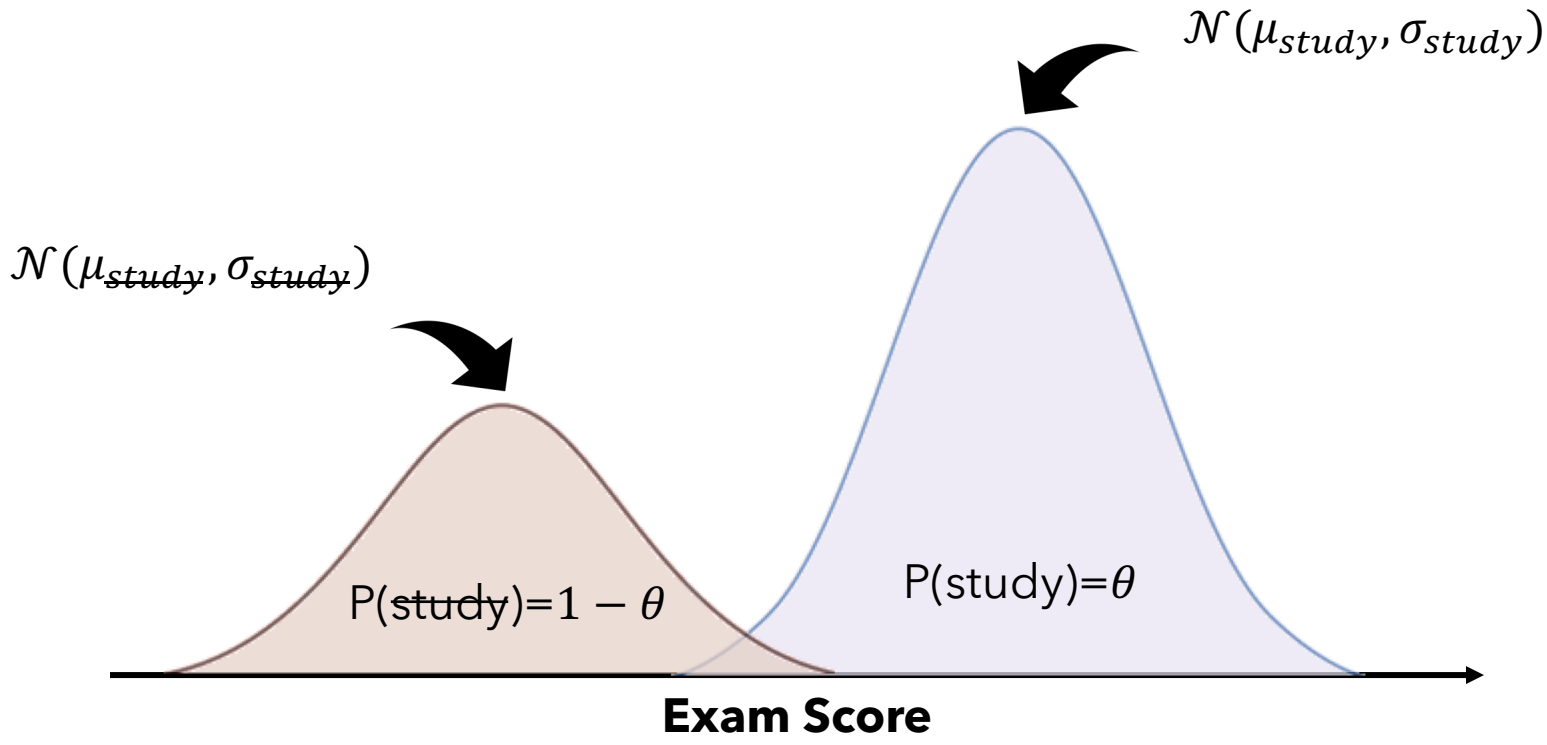
Both classify according to argmax $P(y|\boldsymbol{x})$. Just learn and represent it differently.

© Stefan Lee

Generative Classifiers – Example 1: Studying Students

A simple generative story for this exam scores:

1. Flip a coin (P(heads)=θ) to decide if someone studies (heads=study, tails=no)
2. If **study**, sample the exam score from a Gaussian distribution $\mathcal{N}(\mu_{study}, \sigma_{study})$
3. If **no study**, sample the exam score from a Gaussian distribution $\mathcal{N}(\mu_{\cancel{study}}, \sigma_{\cancel{study}})$



© Stefan Lee

Generative Classifiers – Example 1: Studying Students

Given an observed exam score, how would this story help us predict whether the student studied? Bayes rule to the rescue again.

$$P(study|score) \propto P(score|study)P(study)$$
$$= \mathcal{N}(score;\ \mu_{study}, \sigma_{study})\theta$$

$$P(no\ study|score) \propto P(score|no\ study)P(no\ study)$$
$$= \mathcal{N}(score;\ \mu_{\cancel{study}}, \sigma_{\cancel{study}})(1-\theta)$$

If $P(study|score) > P(no\ study|score)$, then predict study. Otherwise predict no study.

© Stefan Lee

Generative Classifiers – Example 1: Studying Students

Study?	Score
Yes	87
Yes	89
Yes	87
Yes	88
Yes	91
Yes	97
Yes	91
Yes	93
No	80
No	77
No	64
No	75
No	91

$$P(study) = \frac{8}{14} \qquad P(no\ study) = \frac{6}{14}$$

$$P(score|study) = \mathcal{N}(score;\ \mu_{study} = 90.375,\ \ \sigma_{study} = 3.19)$$

$$P(score|\cancel{study}) = \mathcal{N}(score;\ \mu_{\cancel{study}} = 77.4,\ \ \sigma_{\cancel{study}} = 8.68)$$

See a new score of 82, did the student study?

$$P(study|82) \propto P(82|study)P(study)$$

$$= \mathcal{N}\left(82;\ \mu_{study},\ \ \sigma_{study}\right)\frac{8}{14} = 0.00227$$

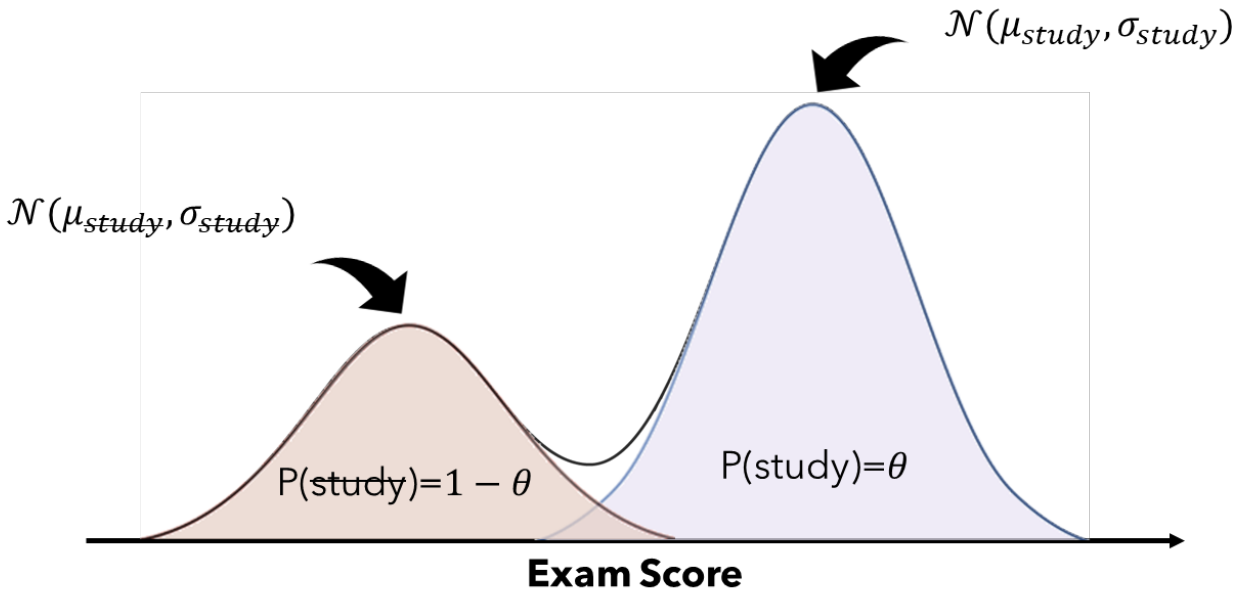
$$P(no\ study|82) \propto P(82|no\ study)P(no\ study)$$

$$= \mathcal{N}\left(82;\ \mu_{\cancel{study}},\ \ \sigma_{\cancel{study}}\right)\frac{6}{14} = 0.0171$$

© Stefan Lee

Generative Classifiers – Example 1: Studying Students

This example had binary labels (y=study / no study) and only a single continuous feature (x=exam score). But we can do similar things with many more features or classes.



© Stefan Lee

Generative Classifiers:

- Learn $P(\boldsymbol{x}|y)$ and $P(y)$
- Compute $P(y|\boldsymbol{x})$ using Bayes Rule

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})} = \frac{P(\boldsymbol{x}|y)P(y)}{\sum_y P(\boldsymbol{x}|y)P(y)}$$

- Decision rule is then:

$$\hat{y} = argmax_c \; P(\boldsymbol{x}|c)P(c)$$

This conditional over x can be EXPENSIVE depending on dimensionality and model choice

© Stefan Lee

$$\hat{y} = argmax_c \; P(\boldsymbol{x}|c)P(c)$$

This conditional over x can be EXPENSIVE depending on dimensionality and model choice

---

Need to estimate the probability of each value the d-dimensional vector $\boldsymbol{x}$ might take on:

- If **x** is a d-dimensional binary vector, then $\boldsymbol{2^d}$ parameters to estimate for each class.
- If **x** is d-dimensional and each dimension can take on m values, need $\boldsymbol{m^d}$ for each class.
- If **x** is real-valued and we choose to model with a multidimensional Gaussian, need to learn $\boldsymbol{d}$ parameters for the mean and $\boldsymbol{d^2}$ for the covariance for each class.

Cost comes from modelling all the dimensions of x jointly.

© Stefan Lee

$$\hat{y} = argmax_c \; P(\boldsymbol{x}|c)P(c)$$

This conditional over x can be EXPENSIVE depending on dimensionality and model choice

---

Modelling all dimensions jointly can also lead to sparse data and zeros in our estimated distributions:

Our Data:

Likes Math	Reads ML News	At Least Junior
0	0	1
1	0	1
1	1	1
0	0	1
1	1	1
0	1	1
0	1	1
1	1	1
1	1	0
1	0	0

Out Joint Distribution:

Likes Math	Reads ML News	At Least Junior	Prob
0	0	0	**0**
0	0	1	**0.2**
0	1	0	**0**
0	1	1	**0.2**
1	0	0	**0.1**
1	0	1	**0.1**
1	1	0	**0.1**
1	1	1	**0.3**

© Stefan Lee

The Naïve Bayes Assumption:

Each feature is **conditionally independent** given the class label.

$$P(\boldsymbol{x}|y) = \prod_{i=1}^{d} P(x_i|y)$$

This has significant impacts on the cost of parameter estimation!

© Stefan Lee

Bayes Model (without naïve assumption)

Our Data:

Color	Type	Stolen
Red	Sport	Yes
Red	SUV	Yes
Blue	Truck	Yes
Red	Sedan	Yes
Blue	Sedan	Yes
Tan	Truck	No
Silver	Truck	No
Tan	Sport	No
Tan	Truck	Yes
Tan	SUV	No

$P(x|y = yes)$

Color	Type	P(x|yes)
Red	Sport	
Red	SUV	
Red	Truck	
Red	Sedan	
Blue	Sport	
Blue	SUV	
Blue	Truck	
Blue	Sedan	
Silver	Sport	
Silver	SUV	
Silver	Truck	
Silver	Sedan	
Tan	Sport	
Tan	SUV	
Tan	Truck	
Tan	Sedan	

$P(x|y = no)$

Color	Type	P(x|yes)
Red	Sport	
Red	SUV	
Red	Truck	
Red	Sedan	
Blue	Sport	
Blue	SUV	
Blue	Truck	
Blue	Sedan	
Silver	Sport	
Silver	SUV	
Silver	Truck	
Silver	Sedan	
Tan	Sport	
Tan	SUV	
Tan	Truck	
Tan	Sedan	

© Stefan Lee

Bayes Model (without naïve assumption)

Our Data:

Color	Type	Stolen
Red	Sport	Yes
Red	SUV	Yes
Blue	Truck	Yes
Red	Sedan	Yes
Blue	Sedan	Yes
Tan	Truck	No
Silver	Truck	No
Tan	Sport	No
Tan	Truck	Yes
Tan	SUV	No

$P(x|y = yes)$

Color	Type	P(x|yes)
Red	Sport	1/7
Red	SUV	1/7
Red	Truck	0
Red	Sedan	1/7
Blue	Sport	0
Blue	SUV	0
Blue	Truck	1/7
Blue	Sedan	1/7
Silver	Sport	0
Silver	SUV	0
Silver	Truck	1/7
Silver	Sedan	0
Tan	Sport	0
Tan	SUV	0
Tan	Truck	1/7
Tan	Sedan	0

$P(x|y = no)$

Color	Type	P(x|yes)
Red	Sport	
Red	SUV	
Red	Truck	
Red	Sedan	
Blue	Sport	
Blue	SUV	
Blue	Truck	
Blue	Sedan	
Silver	Sport	
Silver	SUV	
Silver	Truck	
Silver	Sedan	
Tan	Sport	
Tan	SUV	
Tan	Truck	
Tan	Sedan	

© Stefan Lee

Bayes Model (without naïve assumption)

Our Data:

Color	Type	Stolen
Red	Sport	Yes
Red	SUV	Yes
Blue	Truck	Yes
Red	Sedan	Yes
Blue	Sedan	Yes
Tan	Truck	No
Silver	Truck	No
Tan	Sport	No
Tan	Truck	Yes
Tan	SUV	No

$P(x|y = yes)$

Color	Type	P(x|yes)
Red	Sport	1/7
Red	SUV	1/7
Red	Truck	0
Red	Sedan	1/7
Blue	Sport	0
Blue	SUV	0
Blue	Truck	1/7
Blue	Sedan	1/7
Silver	Sport	0
Silver	SUV	0
Silver	Truck	1/7
Silver	Sedan	0
Tan	Sport	0
Tan	SUV	0
Tan	Truck	1/7
Tan	Sedan	0

$P(x|y = no)$

Color	Type	P(x|no)
Red	Sport	0
Red	SUV	0
Red	Truck	0
Red	Sedan	0
Blue	Sport	0
Blue	SUV	0
Blue	Truck	0
Blue	Sedan	0
Silver	Sport	0
Silver	SUV	0
Silver	Truck	1/4
Silver	Sedan	0
Tan	Sport	1/4
Tan	SUV	1/4
Tan	Truck	1/4
Tan	Sedan	0

© Stefan Lee

Naïve Bayes

Our Data:

Color	Type	Stolen
Red	Sport	Yes
Red	SUV	Yes
Blue	Truck	Yes
Red	Sedan	Yes
Blue	Sedan	Yes
Tan	Truck	No
Silver	Truck	No
Tan	Sport	No
Tan	Truck	Yes
Tan	SUV	No

$P(x|y = yes)$

Color	P(color|yes)
Red	3/6
Blue	2/6
Silver	0
Tan	1/6

Type	P(type|yes)
Sport	1/6
SUV	1/6
Truck	2/6
Sedan	2/6

$P(x|y = no)$

Color	P(color|no)
Red	0
Blue	0
Silver	1/4
Tan	3/4

Type	P(type|no)
Sport	1/4
SUV	1/4
Truck	2/4
Sedan	0

Naïve Bayes results in fewer parameters and fewer zeros in the tables.

© Stefan Lee

Zero-Probability Problem – Laplace Smoothing!

Probably a bad idea to encode **"red cars are always stolen"** into our models…

$P(x|y = yes)$

Color	P(color|yes)
Red	3/6
Blue	2/6
Silver	0
Tan	1/6

Type	P(type|yes)
Sport	1/6
SUV	1/6
Truck	2/6
Sedan	2/6

$P(x|y = no)$

Color	P(color|no)
Red	0
Blue	0
Silver	1/4
Tan	3/4

Type	P(type|no)
Sport	1/4
SUV	1/4
Truck	2/4
Sedan	0

And in general, saying something has "zero probability" is very strong statement.

© Stefan Lee

Zero-Probability Problem – Laplace Smoothing!

Laplace smoothing adds a "prior" to our learned categorical or Bernoulli conditional. As we saw in MAP for Bernoulli-Beta, this amounts to adding fake counts. For example, Laplace smoothing with a value of 1 adds one fake count to each possibility.

$P(x|y = yes)$

Color	P(color|yes)
Red	3/6
Blue	2/6
Silver	0
Tan	1/6

Type	P(type|yes)
Sport	1/6
SUV	1/6
Truck	2/6
Sedan	2/6

$P(x|y = yes)$

Color	P(color|yes)
Red	(3+1)/(6+4)
Blue	(2+1)/(6+4)
Silver	(0+1)/(6+4)
Tan	(1+1)/(6+4)

Type	P(type|yes)
Sport	(1+1)/(6+4)
SUV	(1+1)/(6+4)
Truck	(2+1)/(6+4)
Sedan	(2+1)/(6+4)

$P(x|y = no)$

Color	P(color|no)
Red	0
Blue	0
Silver	1/4
Tan	3/4

Type	P(type|no)
Sport	1/4
SUV	1/4
Truck	2/4
Sedan	0

$P(x|y = no)$

Color	P(color|no)
Red	(0+1)/(4+4)
Blue	(0+1)/(4+4)
Silver	(1+1)/(4+4)
Tan	(3+1)/(4+4)

Type	P(type|no)
Sport	(1+1)/(4+4)
SUV	(1+1)/(4+4)
Truck	(2+1)/(4+4)
Sedan	(0+1)/(4+4)

© Stefan Lee

Zero-Probability Problem – Laplace Smoothing!

Laplace smoothing adds a "prior" to our learned categorical or Bernoulli conditional. As we saw in MAP for Bernoulli-Beta, this amounts to adding fake counts. For example, Laplace smoothing with a value of α adds α fake counts to each possibility. (α need not be an integer)

$P(x|y = yes)$

Color	P(color|yes)
Red	3/6
Blue	2/6
Silver	0
Tan	1/6

Type	P(type|yes)
Sport	1/6
SUV	1/6
Truck	2/6
Sedan	2/6

$P(x|y = yes)$

Color	P(color|yes)
Red	(3+α)/(6+4α)
Blue	(2+α)/(6+4α)
Silver	(0+α)/(6+4α)
Tan	(1+α)/(6+4α)

Type	P(type|yes)
Sport	(1+α)/(6+4α)
SUV	(1+α)/(6+4α)
Truck	(2+α)/(6+4α)
Sedan	(2+α)/(6+4α)

$P(x|y = no)$

Color	P(color|no)
Red	0
Blue	0
Silver	1/4
Tan	3/4

Type	P(type|no)
Sport	1/4
SUV	1/4
Truck	2/4
Sedan	0

$P(x|y = no)$

Color	P(color|no)
Red	(0+α)/(4+4α)
Blue	(0+α)/(4+4α)
Silver	(1+α)/(4+4α)
Tan	(3+α)/(4+4α)

Type	P(type|no)
Sport	(1+α)/(4+4α)
SUV	(1+α)/(4+4α)
Truck	(2+α)/(4+4α)
Sedan	(0+α)/(4+4α)

© Stefan Lee

Questions Break!

If in the previous example, thieves were only stealing “red sportscars”, why would a Naïve Bayes model be a poor choice?

© Stefan Lee

Questions Break!

You are hired to predict whether a machine in a factory needs maintenance using Naïve Bayes. Each machine is represented by:

- The average temperature of the machine in the last hour (continuous)
- The number of outputs with defects in the last hour (non-negative integer)
- Whether the machine has undergone maintenance in the last 6 months (binary)

What are class labels here? What conditionals need to be learned?

How would you choose to model these conditionals? (i.e., what distributions would you assume for each feature?)

© Stefan Lee

Summary of Naïve Bayes Classifier

- **Generative Model:** Estimate $P(y|\boldsymbol{x})$ by learning $P(\boldsymbol{x}|y)$ and $P(y)$. However $P(\boldsymbol{x}|y)$ can have way too many parameters to be fit effectively.

- **Naïve Bayes Assumption:** Assume features are conditionally independent given the class labels: $P(\boldsymbol{x}|y) = \prod P(x_i|y)$

- Training a Naïve Bayes classifier comes down to fitting distributions $P(x_i|y)$ and $P(y)$ either with MLE or MAP (MAP is more robust to data sparsity)

- Naïve Bayes is cheap and survives tens of thousands of attributes easily. Also does okay even when conditional independent doesn't hold.

- Any density estimator can be plugged in to estimate $p(x_i|y)$ for Naïve Bayes

    - Real valued attributes can be discretized or directly modeled using simple continuous distributions such as Gaussian (Normal) distribution

© Stefan Lee

Today's Learning Objectives

Be able to answer:

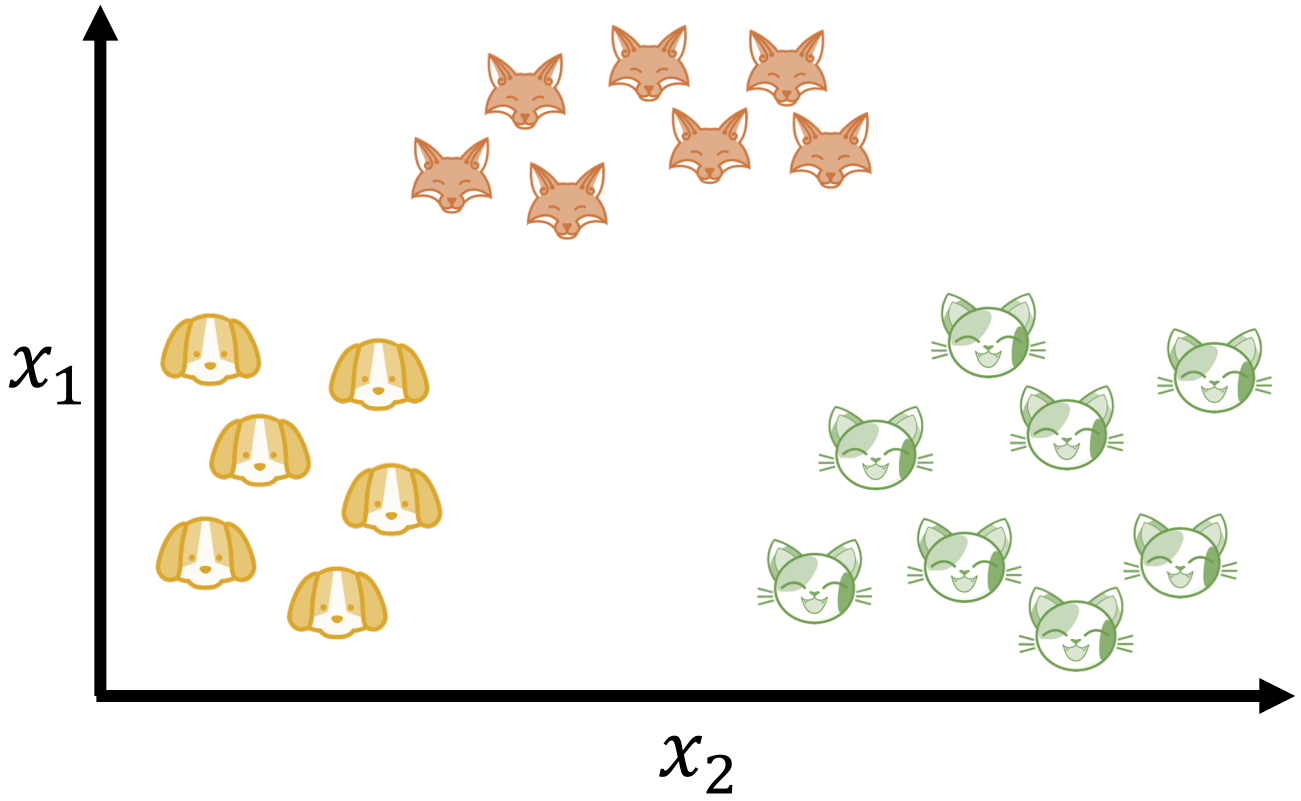
- What is multi-class classification?
- How can we do multi-class classification?
  - How do 1-vs-all / all-vs-all / tree- classifiers work?
- How does multiclass logistic regression work?
  - What is the softmax function?
- How to evaluate multi-class classifiers?
  - What is a multiclass confusion matrix?
  - How can recall and precision adapt to multiclass?

© Stefan Lee

Multiclass Classification

Multiclass Classification describes classification algorithms capable of predicting between more than two possible output labels.

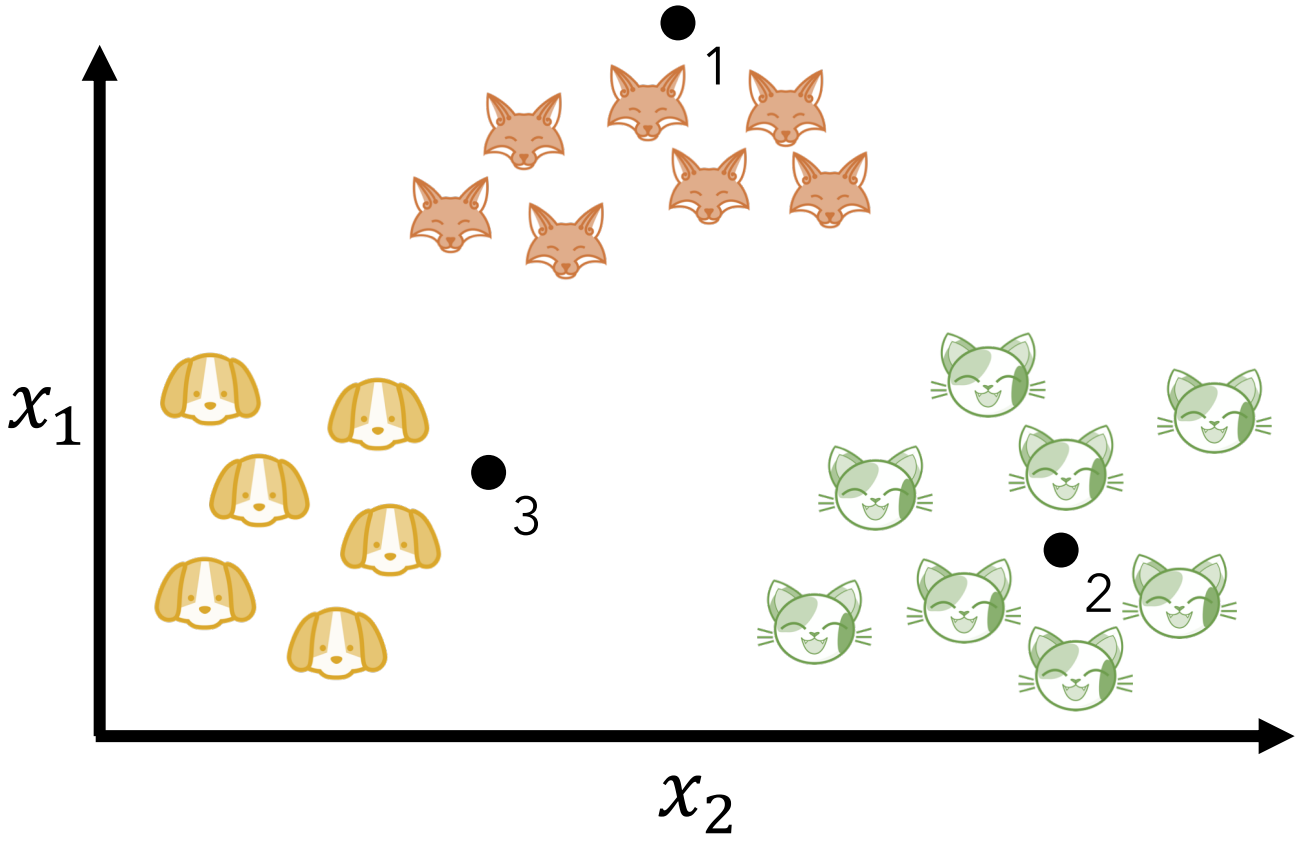
Example below would have 3 class – dog, cat, and fox.



© Stefan Lee

Multiclass Classification

Some of the algorithm's we've already covered already allow for this sort of problem:



k Nearest Neighbors basic decision rule:

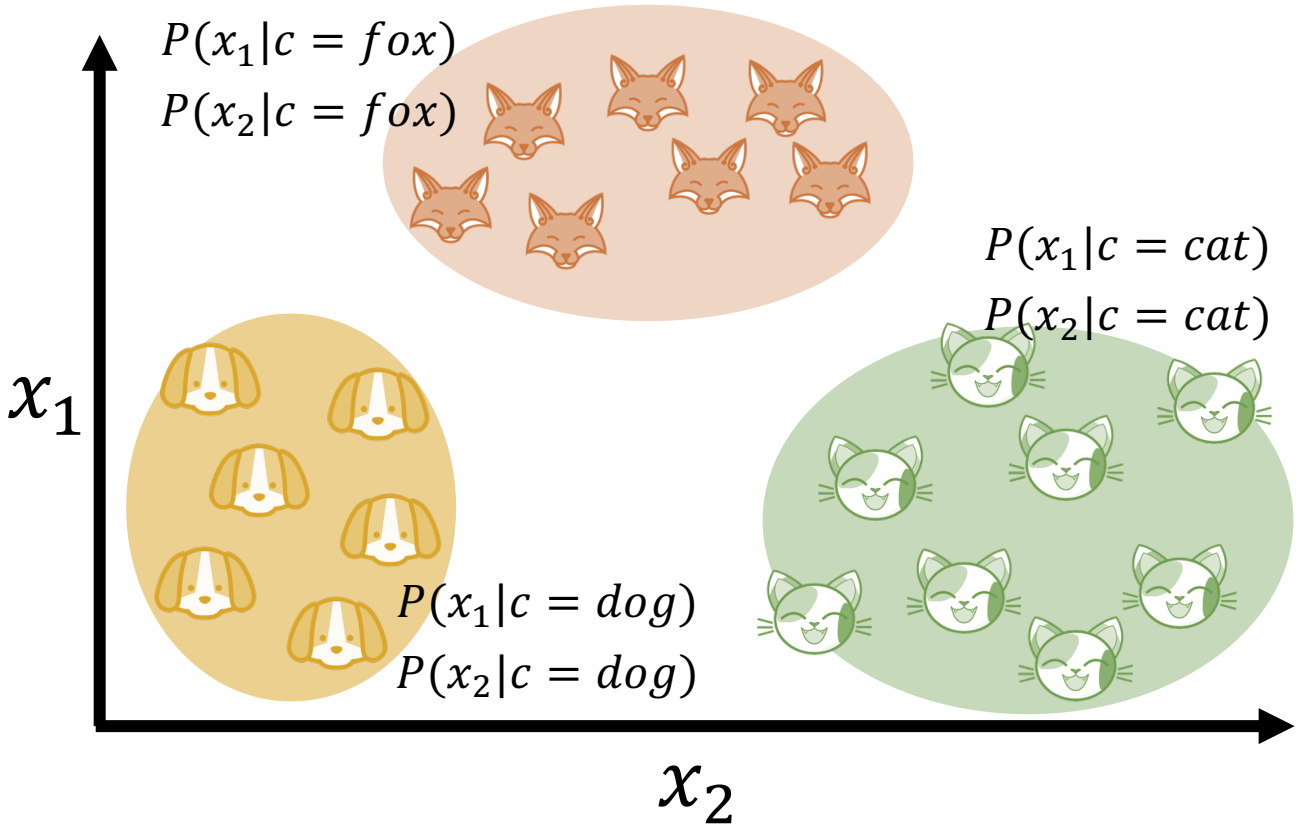
- Find the labels of closest k points
- Label the current point as the majority

Doesn't matter if there are 2 or 2000 classes, nothing changes in the algorithm!

© Stefan Lee

Multiclass Classification

Some of the algorithm's we've already covered already allow for this sort of problem:



Naïve Bayes basic decision rule:

$$argmax_c \; P(c) \prod_{i=1}^{d} P(x_i|c)$$

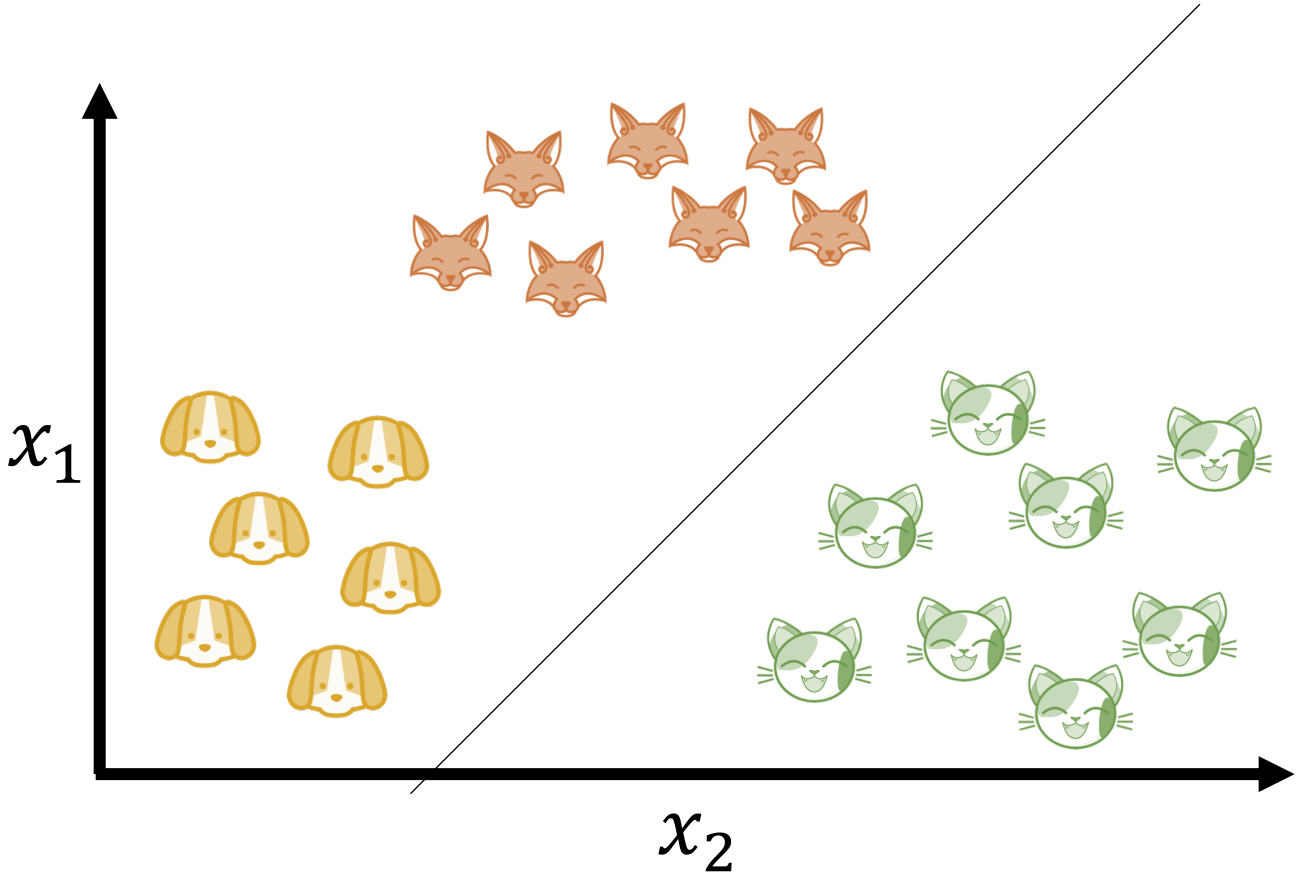
More classes means more entries in $P(c)$ and need to learn more conditionals $P(x_i|c)$

Still flexible with the number of classes!

© Stefan Lee

Multiclass Classification

Others **don't** allow for this sort of problem by default:



Logistic Regression basic decision rule:

$$w^T x > 0 \quad \Rightarrow \quad y = 1$$

A line can only divide a space into two halves.

© Stefan Lee

Multiclass Classification

Many of the classifiers we've discussed so far in class have been **binary classifiers** – i.e., algorithms that can predict whether an input is one of two possible classes.

However; some could handle multiple possible classes and are considered **multiclass classifiers.**

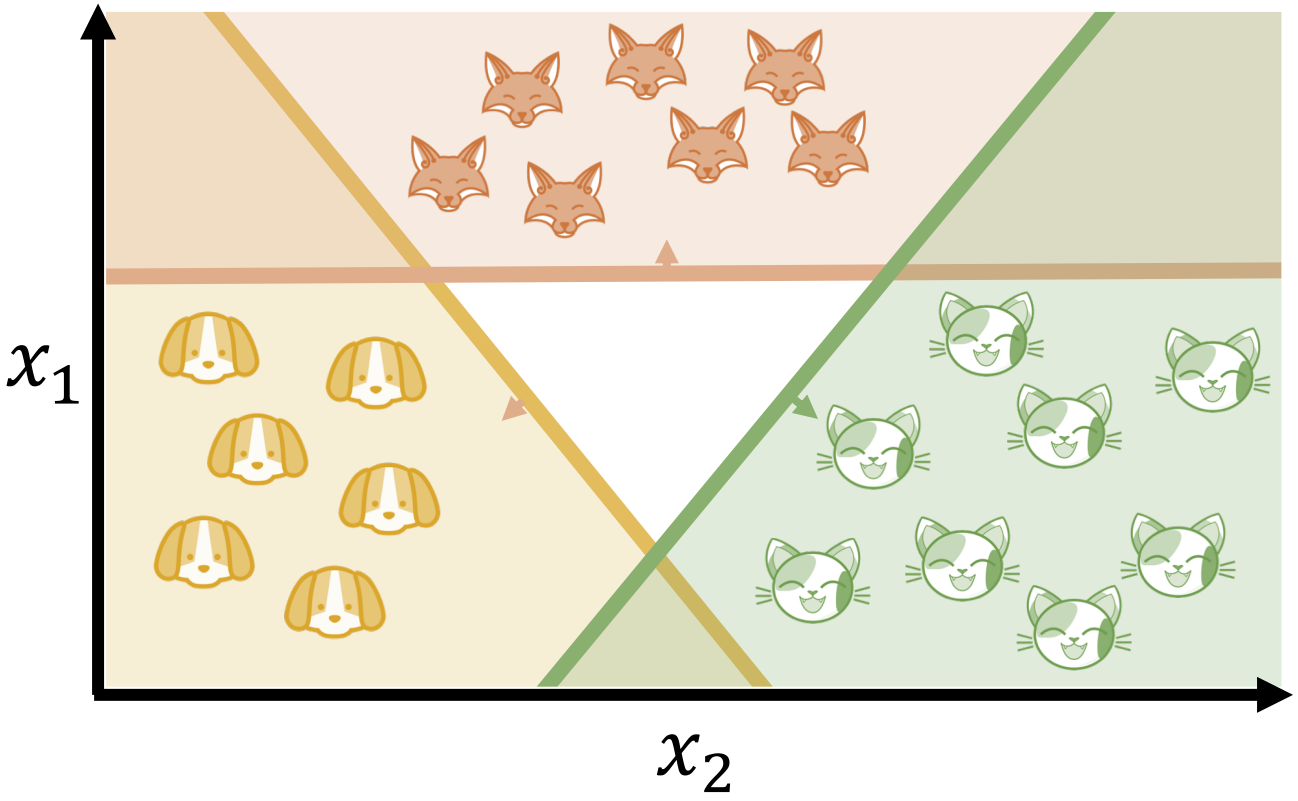
Binary Classifiers $y \in \{0, 1\}$	**Multiclass Classifiers** $y \in \{0, 1, 2, \ldots C\}$
Logistic Regression	Naïve Bayes
Perceptron	k Nearest Neighbors

Suppose I want to use Logistic Regression for a multiclass setting, how could I do that?

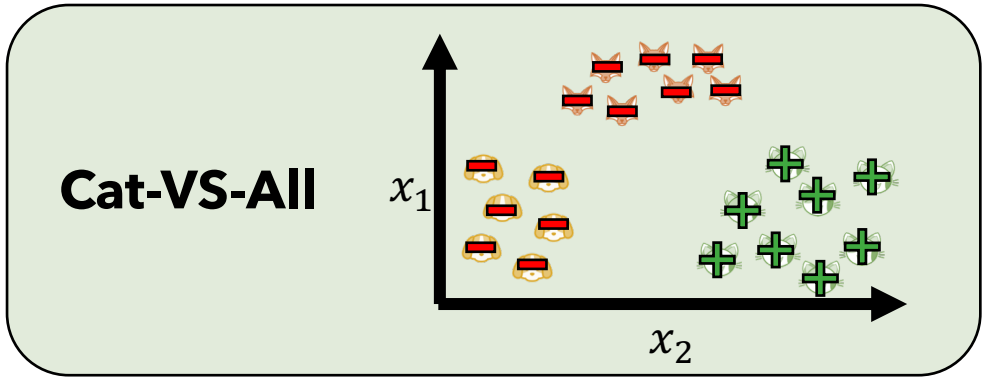
© Stefan Lee

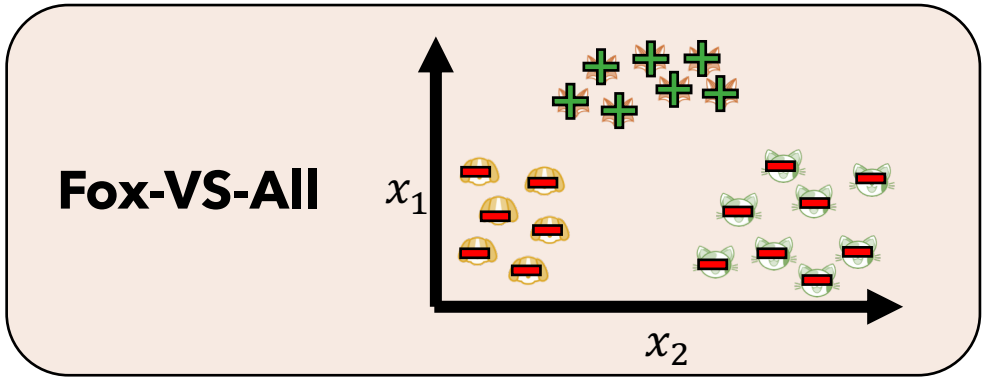
One-VS-All Multiclass Classifier

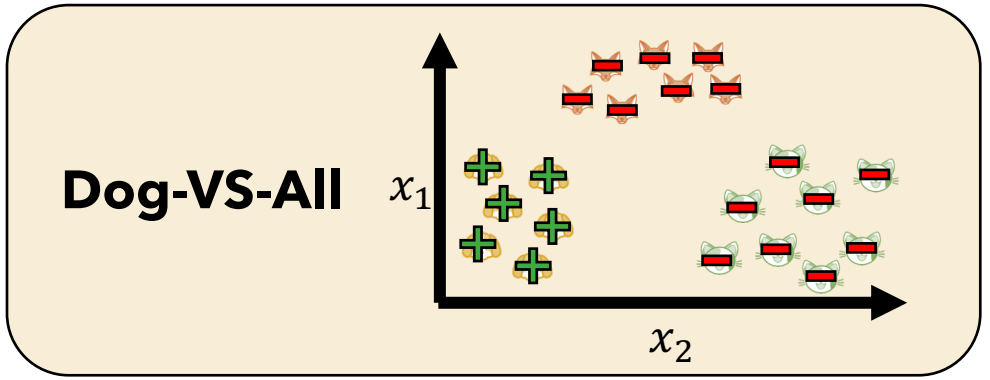
Consider the following multiclass classification problem. How can we use binary classifiers to figure out if a new point is a fox, dog, or cat?



Train Three One-Vs-All Classifiers







© Stefan Lee

One-VS-All Multiclass Classifier

One-VS-All Training(D , TrainBinaryClassifier**):**

For each class c in 0,1,2,…, C:

D_{bin} ← relabel dataset D so class c is positive and all other classes negative

f_c ← TrainBinaryClassifier(D_{bin})

Return $f_0, f_1, \ldots, f_C$

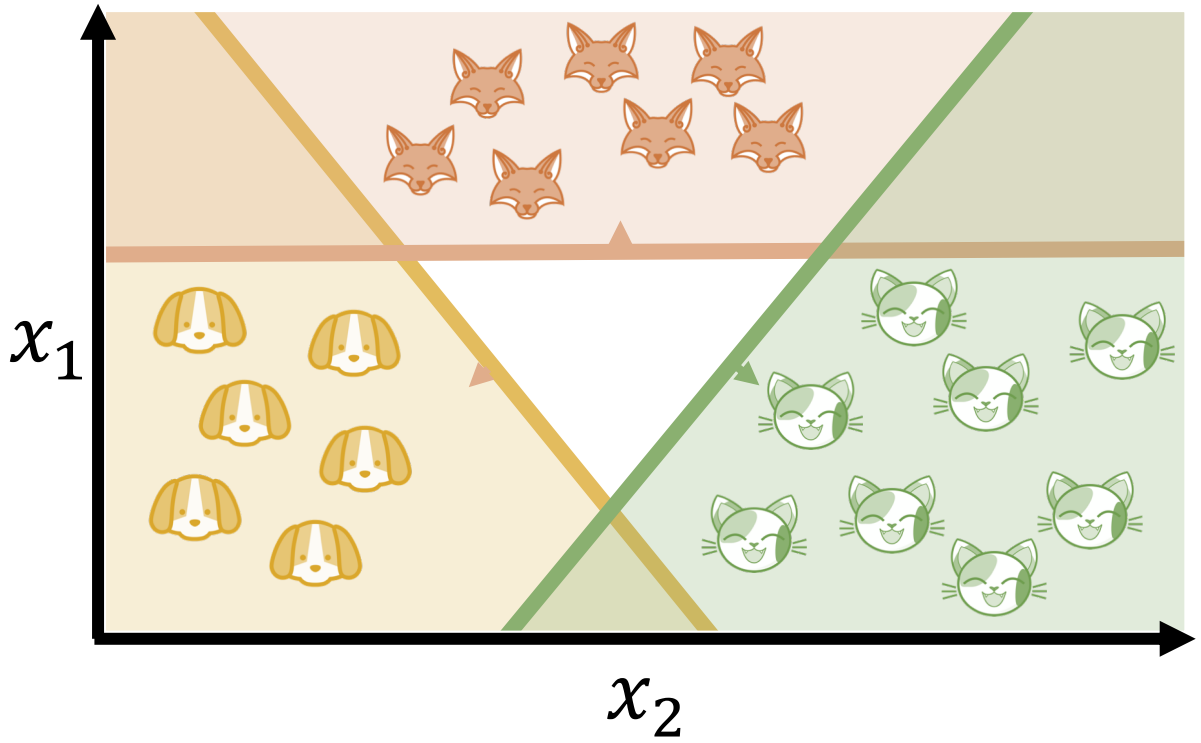
One-VS-All Testing($f_0, f_1, \ldots, f_C$, x**):**

scores = zeros(C)

For each class c in 0,1,2, …, C:

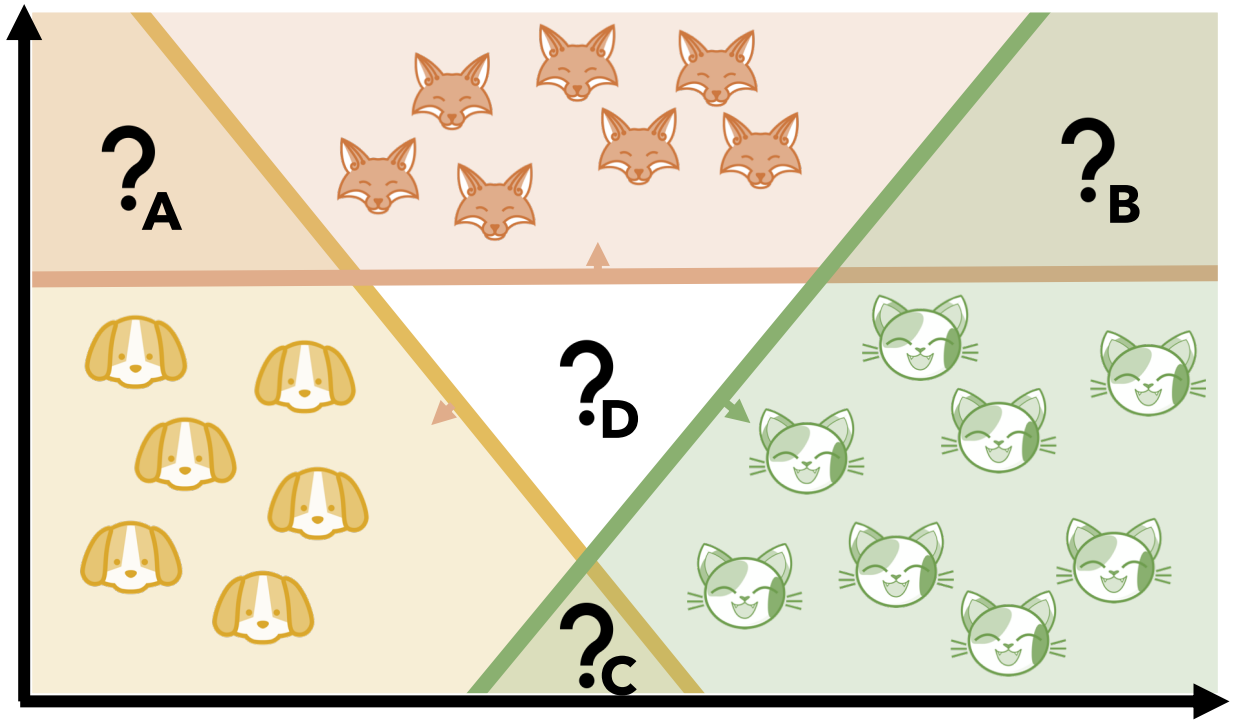
$scores_c$ ← $\text{scores}_c + f_c(\boldsymbol{x})$

Return $argmax_c \ scores_c$



© Stefan Lee

One-VS-All Multiclass Classifier



Problem: What happens in these areas?

Case 1: If the binary classifiers output comparable continuous scores (e.g., like a probability in logistic regression), can just take the argmax.

Case 2: If the binary classifiers just make 0/1 decisions (SVM / Perceptron), will need to choose randomly in those areas among relevant classes.

- **Cat/Fox in A, Fox/Cat in B, Dog/Cat in C, Cat/Fox/Dog in D**

Error bound in case 2: Suppose the average error is ϵ, then the error rate of the One-VS-All classifier for multiclass classification is at most $(K-1)\epsilon$ [Proof in this week's reading]

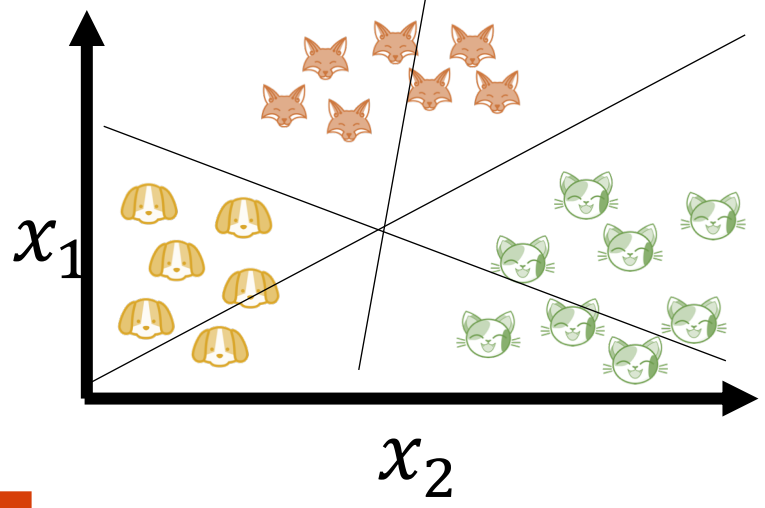
© Stefan Lee

Other Options for Doing Multiclass with Binary Classifiers

All-VS-All Classifier

Train $\binom{C}{2}$ one-vs-one classifiers to separate each pair of classes. Run all models on a new input and check which class was predicted the most.

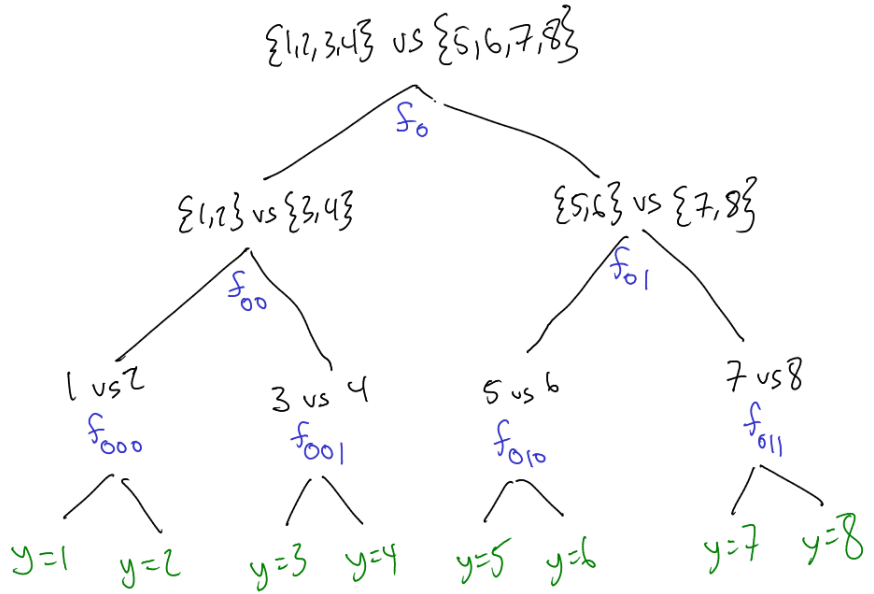
- Simple to implement
- Rarely have ties among classes
- Weaker error bound than one-vs-all
- Have to learn and run lots of classifiers



Tree Classifier

Train half-vs-half classifiers to form a binary tree of decision boundaries. Follow the decision path to classify a new input.

- Strong error bound $[\log_2 C]\epsilon$
- Not too many classifiers to learn
- Have to choose splits somehow (good or bad)



© Stefan Lee

Multiclass Classification

The methods we've just discussed are general ways to turn *any* binary classification method into a multiclass classification approach. However, they are a bit inelegant and expensive.

Would prefer to simply modify our classifiers to be multiclass to start with!

Support Vector Machines → Structured Support Vector Machines. (See SVMs soon!)
Generalizes the notion of a margin to multiclass setting – adds some additional constraints. (Has some other significant new abilities for arbitrarily complex predictions beyond classes). See original 2001 paper.

Perceptron → Multiclass Perceptron.
Considers classification over an augmented feature $\phi(x_i, y)$ that encodes both a datapoint and potential label – lets the perceptron learn a linear "compatibility" function between x and y. 17.1 in the textbook.

Logistic Regression → Multiclass Logistic Regression.
Today in class! Yay!

© Stefan Lee

Today's Learning Objectives

Be able to answer:

- ~~What is multi-class classification?~~
- ~~How can we do multi-class classification?~~
  - ~~How do 1-vs-all / all-vs-all / tree- classifiers work?~~
- How does multiclass logistic regression work?
  - What is the softmax function?
- How to evaluate multi-class classifiers?
  - What is a multiclass confusion matrix?
  - How can recall and precision adapt to multiclass?

© Stefan Lee

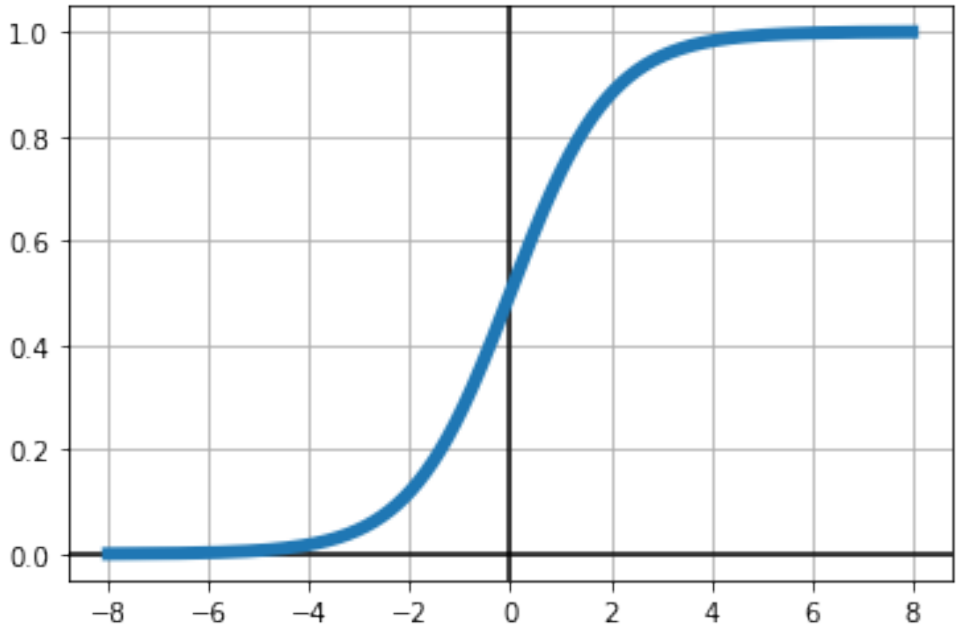
Introducing the logistic function:

- May also see it referred to as a sigmoid function or "logit" function

Logistic Function

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Maps $(-\infty, \infty)$ to $(0,1)$



© Stefan Lee

Logistic Regression Model Assumption:

$$P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w}) = \sigma(\boldsymbol{w^T x_i}) = \frac{1}{1 + e^{-\boldsymbol{w^T x_i}}}$$

$$P(y_i = 0|\boldsymbol{x_i}; \boldsymbol{w}) = 1 - \sigma(\boldsymbol{w^T x_i}) = \frac{e^{-\boldsymbol{w^T x_i}}}{1 + e^{-\boldsymbol{w^T x_i}}}$$

© Stefan Lee

Recall Logistic Regression for Binary Classification

Model Assumption:

$$P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w}) = \frac{1}{1 + e^{-\boldsymbol{w^T x_i}}}$$

$$P(y_i = 0|\boldsymbol{x_i}; \boldsymbol{w}) = 1 - P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w}) = \frac{e^{-\boldsymbol{w^T x_i}}}{1 + e^{-\boldsymbol{w^T x_i}}}$$

Negative Log-Likelihood under a Bernoulli Model:

$$-logP(D \mid w) = -\sum_{i=1}^{N} y_i \log \sigma\left(\boldsymbol{w^T x_i}\right) + (1 - y_i) \log\left(1 - \sigma\left(\boldsymbol{w^T x_i}\right)\right)$$

Optimized parameter via gradient descent.

$$\nabla - logP(D \mid w) = \sum_{i=1}^{N} \nabla l_i = \sum_{i=1}^{N} \left(\sigma\left(\boldsymbol{w^T x_i}\right) - y_i\right)\mathbf{x_i}$$

© Stefan Lee

So we have an expression for the gradient but can't find the optimal analytically…

$$\nabla - logP(D \mid w) = \sum_{i=1}^{N} \nabla l_i = \sum_{i=1}^{N} \left(\sigma\left(\boldsymbol{w^T x_i}\right) - y_i\right)\mathbf{x_i}$$

Gradient Descent Algorithm for Logistic Regression

w = random(d) // randomly initialize weight vector

Repeat:

for each example $x_i, y_i \in D$: // compute gradient

$\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \frac{1}{1+e^{-\boldsymbol{w^T x_i}}}$ // compute the prediction for this point

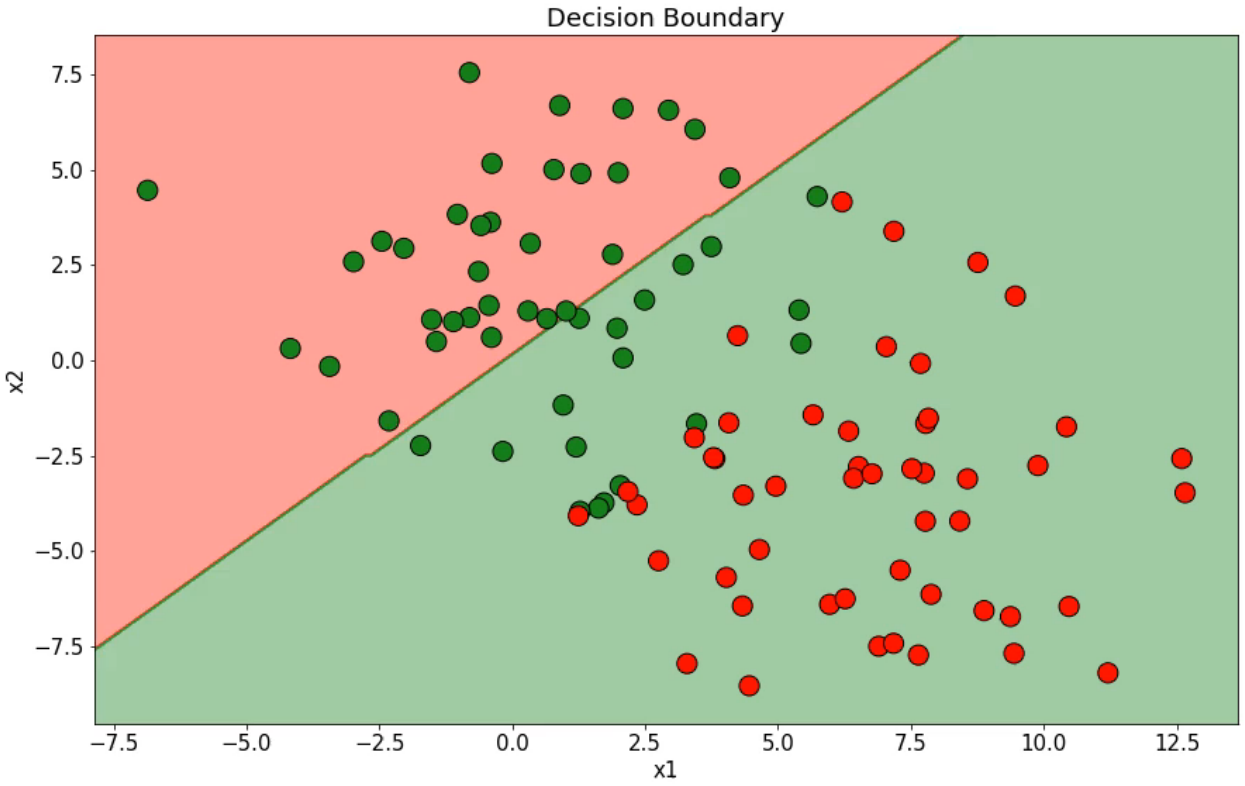
$\boldsymbol{\nabla} \mathrel{+}= (\hat{y}_i - y_i)\boldsymbol{x_i}$ // compute gradient of loss of point and sum

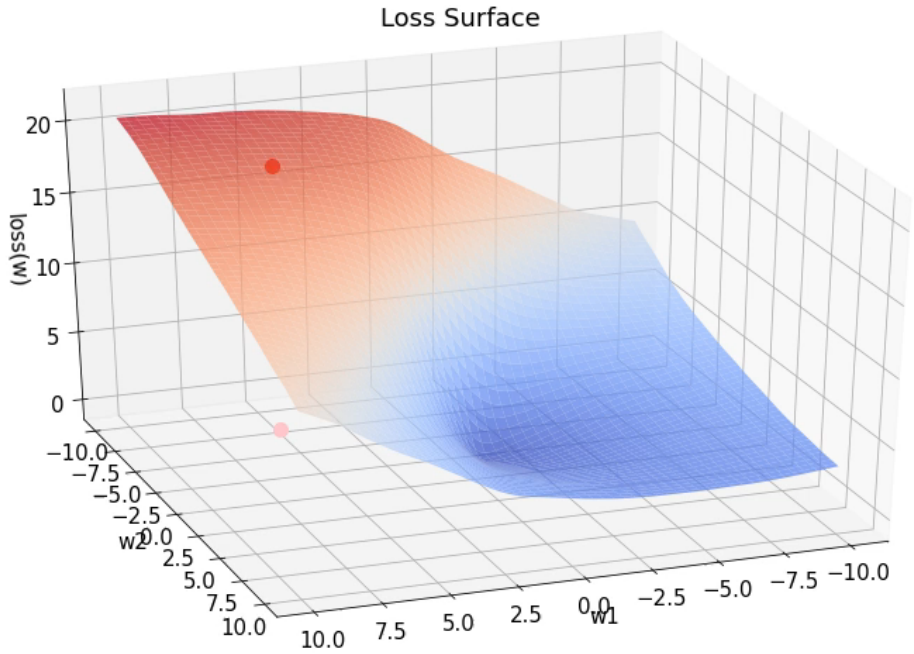
$\mathbf{w} = \mathbf{w} - \alpha\boldsymbol{\nabla}$ // take a step in the opposite direction of gradient

Until $|\nabla| \leq \epsilon$ // keep taking steps until convergence

© Stefan Lee

An example of Logistic Regression with Gradient Descent





© Stefan Lee

Recall Logistic Regression for Binary Classification

Model Assumption:

$$P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w}) = \frac{1}{1 + e^{-\boldsymbol{w^T x_i}}} \qquad P(y_i = 0|\boldsymbol{x_i}; \boldsymbol{w}) = \frac{e^{-\boldsymbol{w^T x_i}}}{1 + e^{-\boldsymbol{w^T x_i}}}$$

I want to start playing with this a little. Let's start by replacing 1 with e^0

$$P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w}) = \frac{e^0}{e^0 + e^{-\boldsymbol{w^T x_i}}} \qquad P(y_i = 0|\boldsymbol{x_i}; \boldsymbol{w}) = \frac{e^{-\boldsymbol{w^T x_i}}}{e^0 + e^{-\boldsymbol{w^T x_i}}}$$

Let $w_1 = \vec{0}$ and $w_0 = w$, could write this as:

$$P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w_0}, \boldsymbol{w_1}) = \frac{e^{-\boldsymbol{w_1^T x_i}}}{e^{-\boldsymbol{w_1^T x_i}} + e^{-\boldsymbol{w_0^T x_i}}} \qquad P(y_i = 0|\boldsymbol{x_i}; \boldsymbol{w}_0, \boldsymbol{w_1}) = \frac{e^{-\boldsymbol{w_0^T x_i}}}{e^{-\boldsymbol{w_1^T x_i}} + e^{-\boldsymbol{w_0^T x_i}}}$$

© Stefan Lee

Recall Logistic Regression for Binary Classification

Tweaked Model Assumption:

Let $w_1 = \vec{0}$ and $w_0 = w$, could write this as:

$$P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w_0}, \boldsymbol{w_1}) = \frac{e^{-\boldsymbol{w_1^T x_i}}}{e^{-\boldsymbol{w_1^T x_i}} + e^{-\boldsymbol{w_0^T x_i}}} \qquad P(y_i = 0|\boldsymbol{x_i}; \boldsymbol{w_0}, \boldsymbol{w_1}) = \frac{e^{-\boldsymbol{w_0^T x_i}}}{e^{-\boldsymbol{w_1^T x_i}} + e^{-\boldsymbol{w_0^T x_i}}}$$

If we relax this restriction and let w_1 be anything, does it change the problem? Let's see.

Ask if $P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w_0}, \boldsymbol{w_1}) \stackrel{?}{=} P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w})$ for some $\boldsymbol{w'}$

$$P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w_0}, \boldsymbol{w_1}) = \frac{e^{-\boldsymbol{w_1^T x_i}}}{e^{-\boldsymbol{w_1^T x_i}} + e^{-\boldsymbol{w_0^T x_i}}} * \boldsymbol{1}$$

$$= \frac{e^{-\boldsymbol{w_1^T x_i}}}{e^{-\boldsymbol{w_1^T x_i}} + e^{-\boldsymbol{w_0^T x_i}}} * \frac{e^{\boldsymbol{w_1^T x_i}}}{e^{\boldsymbol{w_1^T x_i}}} = \frac{e^{-\boldsymbol{w_1^T x_i} + \boldsymbol{w_1^T x_i}}}{e^{-\boldsymbol{w_1^T x_i} + \boldsymbol{w_1^T x_i}} + e^{-\boldsymbol{w_0^T x_i} + \boldsymbol{w_1^T x_i}}} = \frac{e^{\boldsymbol{0}}}{e^{\boldsymbol{0}} + e^{-\boldsymbol{w_0^T x_i} + \boldsymbol{w_1^T x_i}}}$$

$$= \frac{\boldsymbol{1}}{\boldsymbol{1} + \boldsymbol{e}^{-(\boldsymbol{w_0} - \boldsymbol{w_1})^T \boldsymbol{x_i}}} = P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w'}) \quad \blacksquare$$

Nope. Same model, just more parameters.

© Stefan Lee

Moving To Multiclass Logistic Regression

Binary Model Assumption:

$$P(y_i = 0|\boldsymbol{x_i}; \boldsymbol{w}_0, \boldsymbol{w_1}) = \frac{e^{-\boldsymbol{w_0^T x_i}}}{e^{-\boldsymbol{w_0^T x_i}} + e^{-\boldsymbol{w_1^T x_i}}}$$

$$P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w_0}, \boldsymbol{w_1}) = \frac{e^{-\boldsymbol{w_1^T x_i}}}{e^{-\boldsymbol{w_0^T x_i}} + e^{-\boldsymbol{w_1^T x_i}}}$$

Can we generalize this form to three classes?

$$P(y_i = 0|\boldsymbol{x_i}; \boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2) = \frac{e^{-\boldsymbol{w_0^T x_i}}}{e^{-\boldsymbol{w_0^T x_i}} + e^{-\boldsymbol{w_1^T x_i}} + e^{-\boldsymbol{w_2^T x_i}}}$$

$$P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2) = \frac{e^{-\boldsymbol{w_1^T x_i}}}{e^{-\boldsymbol{w_0^T x_i}} + e^{-\boldsymbol{w_1^T x_i}} + e^{-\boldsymbol{w_2^T x_i}}}$$

$$P(y_i = 2|\boldsymbol{x_i}; \boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2) = \frac{e^{-\boldsymbol{w_2^T x_i}}}{e^{-\boldsymbol{w_0^T x_i}} + e^{-\boldsymbol{w_1^T x_i}} + e^{-\boldsymbol{w_2^T x_i}}}$$

Sanity checks:

$$P(y_i = c|\boldsymbol{x_i}; \boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2) \geq 0$$

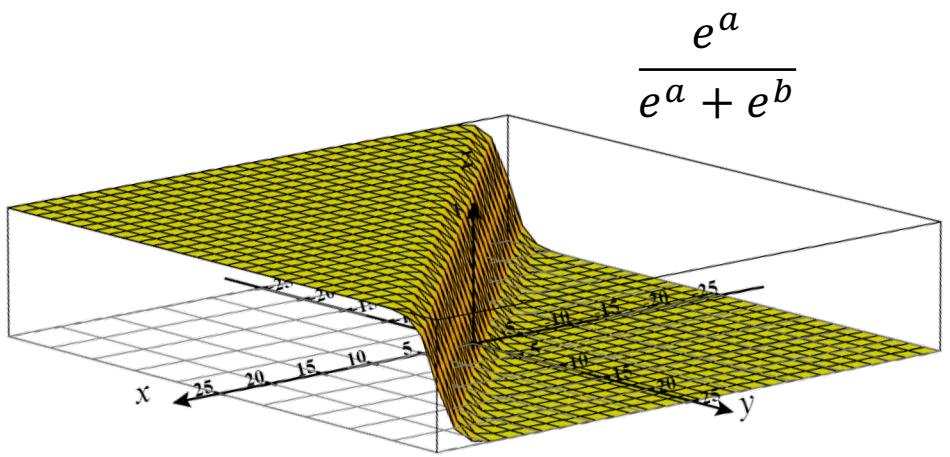
$$\sum P(y_i = c|\boldsymbol{x_i}; \boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2) = 1$$

© Stefan Lee

A Note About The Softmax Function

This functional form has a name – the softmax function. It turns a real-valued vector $\mathbb{R}^d$ to a categoric distribution -- i.e. each element in $(0,1)$ and it sums to 1.

$$Softmax\left(\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_d \end{bmatrix}\right) = \begin{bmatrix} e^{z_1} \\ e^{z_2} \\ \vdots \\ e^{z_d} \end{bmatrix} \frac{1}{\sum_{i=1}^{d} e^{z_i}}$$



"**max**" – amplifies probability of largest x_i
"**soft**" – differentiable and still assigns non-zero probability to other entries

© Stefan Lee

Multiclass Logistic Regression

Dataset: Given a dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$ where $y_i \in \{0,1,2, \dots, C\}$

Model assumptions:

$$y_i \sim Categorical\left(\theta_c = \frac{e^{-w_c^T x_i}}{\sum_j e^{-w_j^T x_i}}\right)$$

$$\Rightarrow P(y_i | x_i, \boldsymbol{w_0}, \dots \boldsymbol{w_c}) = \prod_{c=1}^{C} \theta_c{}^{\mathbb{I}[y_i == c]} = \theta_{y_i}$$

Write the negative log-likelihood of the training data as a function of parameters:

$$-logP(D \mid \boldsymbol{w_0}, \dots \boldsymbol{w_c}) = -\sum_{i=1}^{N} logP(y_i | \boldsymbol{x_i}, \boldsymbol{w_0}, \dots \boldsymbol{w_c}) = -\sum_{i=1}^{N} \log \frac{e^{-w_{y_i}^T x_i}}{\sum_j e^{-w_j^T x_i}}$$

© Stefan Lee

Multiclass Logistic Regression

$$-logP(D \mid \boldsymbol{w_0}, \ldots \boldsymbol{w_c}) = -\sum_{i=1}^{N} \log \frac{e^{-w_{y_i}^T x_i}}{\sum_j e^{-w_j^T x_i}} = \sum_{i=1}^{N} \boldsymbol{w}_{y_i}^T \boldsymbol{x_i} + \log\left(\sum_j e^{-\boldsymbol{w}_j^T \boldsymbol{x}_i}\right)$$

$$\frac{d(-logP(D \mid \boldsymbol{w_0}, \ldots \boldsymbol{w_c}))}{dw_c} = \sum_{i=1}^{N} \mathbb{I}[y_i == c]\boldsymbol{x_i} - \left[\frac{1}{\sum_j e^{-\boldsymbol{w}_j^T \boldsymbol{x}_i}} e^{-\boldsymbol{w}_c^T \boldsymbol{x}_i}\right] \boldsymbol{x_i}$$

$$= \sum_{i=1}^{N} \left(\mathbb{I}[y_i == c] - \frac{e^{-\boldsymbol{w}_c^T \boldsymbol{x}_i}}{\sum_j e^{-\boldsymbol{w}_j^T \boldsymbol{x}_i}} \right) \boldsymbol{x_i}$$

$$= \sum_{i=1}^{N} \left(\mathbb{I}[y_i == c] - P(c|\boldsymbol{x_i}, \boldsymbol{w_0}, \ldots \boldsymbol{w_c}) \right) \boldsymbol{x_i}$$

© Stefan Lee

Questions Break!

Math can be hard. How might we "sanity check" this result to make sure it aligns with our intuitions?

Specifically, we just derived this expression for multiclass logistic regression:

$$\frac{d(-logP(D \mid \boldsymbol{w_0}, \ldots \boldsymbol{w_c}))}{dw_c} = \sum_{i=1}^{N} \left(\mathbb{I}[y_i == c] - P(c|\boldsymbol{x_i}, \boldsymbol{w_0}, \ldots \boldsymbol{w_c})\right) \boldsymbol{x_i}$$

Is there something we could check to make sure we didn't break anything?

© Stefan Lee

Multiclass Logistic Regression (Sanity Check)

Derivation showing this gets back to the binary logistic regression gradient if we only consider two classes (only one weight vector w_0):

$$\frac{d(-logP(D \mid \boldsymbol{w_0}))}{dw_0} = \sum_{i=1}^{N} \left(\mathbb{I}[y_i == 0] - P(0|\boldsymbol{x_i}, \boldsymbol{w_0})\right) \boldsymbol{x_i}$$

$$= \sum_{i=1}^{N} (P(1|\boldsymbol{x_i}, \boldsymbol{w_0}) + \mathbb{I}[y_i == 0] - 1)\, \boldsymbol{x_i}$$

$(\mathbb{I}[y_i == 0] - 1) = -1 \; if \; y_i = 1,$
$(\mathbb{I}[y_i == 0] - 1) = 0 \; if \; y_i = 0$

➡

$$= \sum_{i=1}^{N} (P(1|\boldsymbol{x_i}, \boldsymbol{w_0}) - y_i)\, \boldsymbol{x_i}$$

Gradient Descent. We want t find optimal weights $\mathbf{w}^* = argmin_{\mathbf{w}} - logP(D|\mathbf{w})$. However, taking the gradient of the negative log-likelihood yields the expression below which does not offer a closed-form solution.

$$\nabla_{\mathbf{w}}(-logP(D|\mathbf{w})) = \sum_{i=1}^{n}(\sigma(\mathbf{w}^T\mathbf{x}_i) - y_i)x_i \tag{14}$$

© Stefan Lee

Gradient Descent Algorithm for Multiclass Logistic Regression

w_0 = random(d) // randomly initialize weight vectors

…

w_C = random(d) // randomly initialize weight vectors

Repeat:

$\nabla w_0, \nabla w_1, \ldots, \nabla w_c = \vec{0}$

for each example $x_i, y_i \in D$: // loop over dataset to aggregate gradient

for c in 0,1,2,…,C:

$$p_c = \frac{e^{-w_c^T x_i}}{\sum_j e^{-w_j^T x_i}}$$ // compute probability of each class

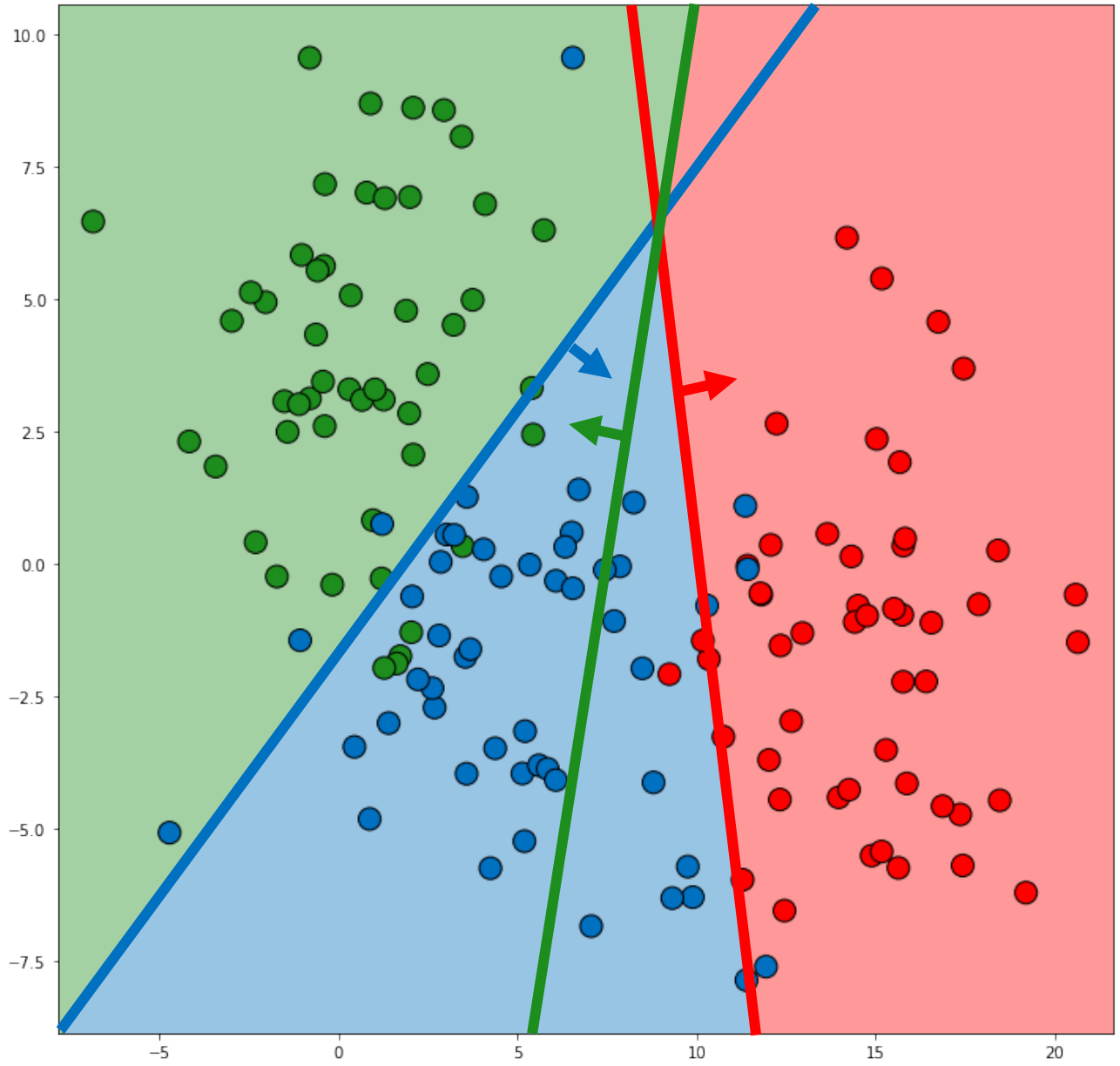
$\nabla \mathbf{w_c} \mathrel{+}= (\mathbb{I}[y_i == c] - p_c)\boldsymbol{x_i}$ // compute gradient of loss of point and sum

$\mathbf{w_c} = \mathbf{w_c} - \alpha \nabla \mathbf{w_c}$ for c in 0, 1, 2, … C // take a step in the opposite direction of gradient

Until $\max\_iters$ // keep taking steps until convergence

© Stefan Lee

Multiclass Logistic Regression Example



Multiclass Logistic Regression learns multiple linear decision boundaries, the max probability is taken as the class label.

$$y = argmax_c \frac{e^{-w_c^T x_i}}{\sum_j e^{-w_j^T x_i}}$$

© Stefan Lee

Summarizing Multiclass Logistic Regression

Multiclass Logistic Regression (also called Multinomial Logistic Regression) is a straight-forward extension of Logistic Regression to more possible output classes.

- **Learn a weight vector for each possible output class $w_0, w_1, \dots, w_C$**
- **Apply the softmax function to turn $-w_0^T x_i, -w_1^T x_i, \dots, -w_C^T x_i$ into probabilities**
- **During training, Maximize Categorical Negative Log-Likelihood with gradient descent**
- **During testing, just take the argmax probability.**

© Stefan Lee

Today's Learning Objectives

Be able to answer:

- ~~What is multi-class classification?~~
- ~~How can we do multi-class classification?~~
  - ~~How do 1-vs-all / all-vs-all / tree- classifiers work?~~
- ~~How does multiclass logistic regression work?~~
  - ~~What is the softmax function?~~
- How to evaluate multi-class classifiers?
  - What is a multiclass confusion matrix?
  - How can recall and precision adapt to multiclass?

© Stefan Lee

Evaluating Multiclass Classifiers

How can we evaluate performance of a multiclass classifier?

- Accuracy still works fine:

$$\frac{\#\ correct}{total} = \frac{\sum_{i=1}^{n} \mathbb{I}[y_i == \hat{y}_i]}{n}$$

- Recall? Precision? Confusion matrices?

© Stefan Lee

Evaluating Multiclass Classifiers

Remember for binary classification we considered a confusion matrix:

		Predicted	
		Negative (**N**) **-**	Positive (**P**) **+**
Actual	Negative **-**	True Negatives (T**N**)	False Positives (F**P**)
	Positive **+**	False Negatives (F**N**)	True Positives (T**P**)

© Stefan Lee

Evaluating Multiclass Classifiers

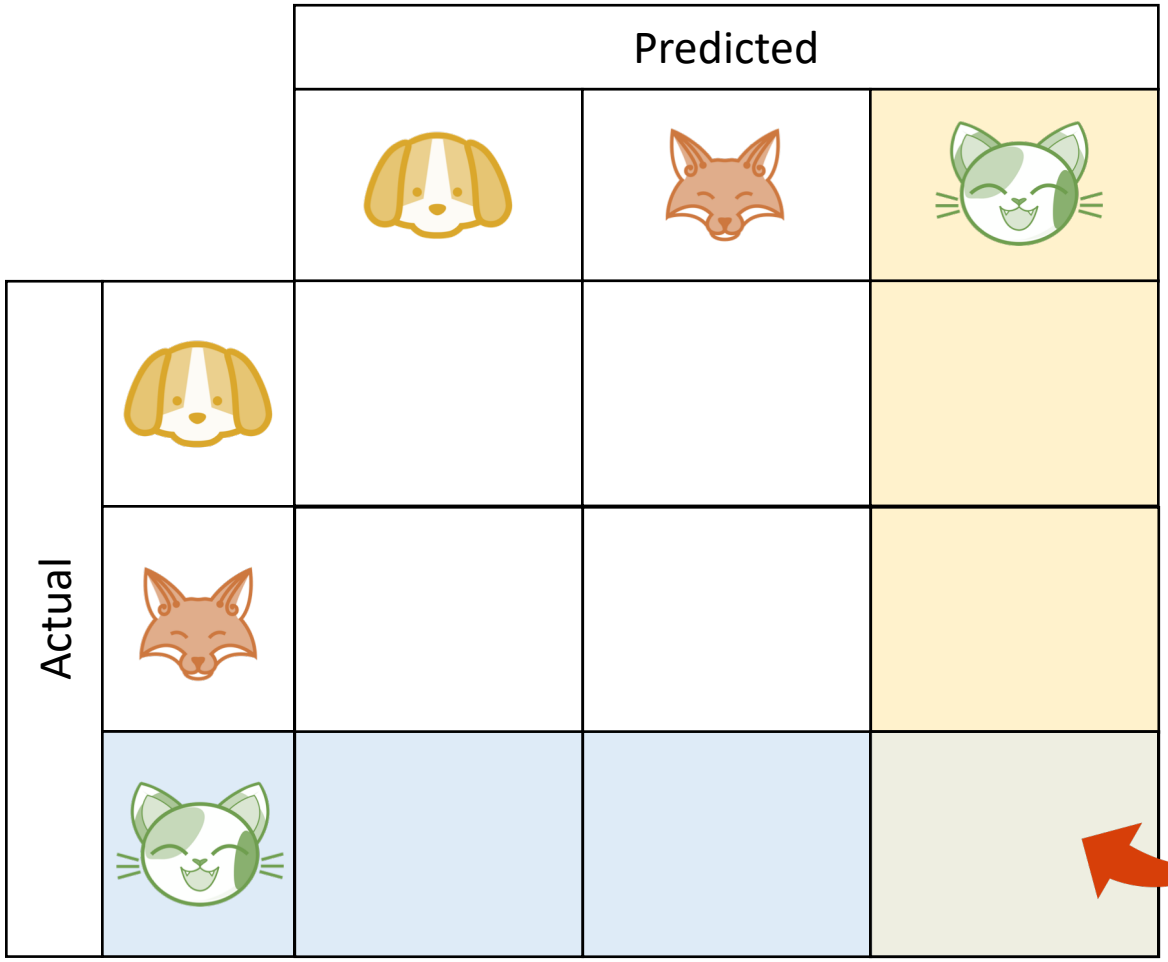
For multiclass case, still consider a matrix where rows are true labels and columns are predictions:

		Predicted		
				
Actual				
				
				

© Stefan Lee

Evaluating Multiclass Classifiers

For multiclass case, still consider a matrix where rows are true labels and columns are predictions:

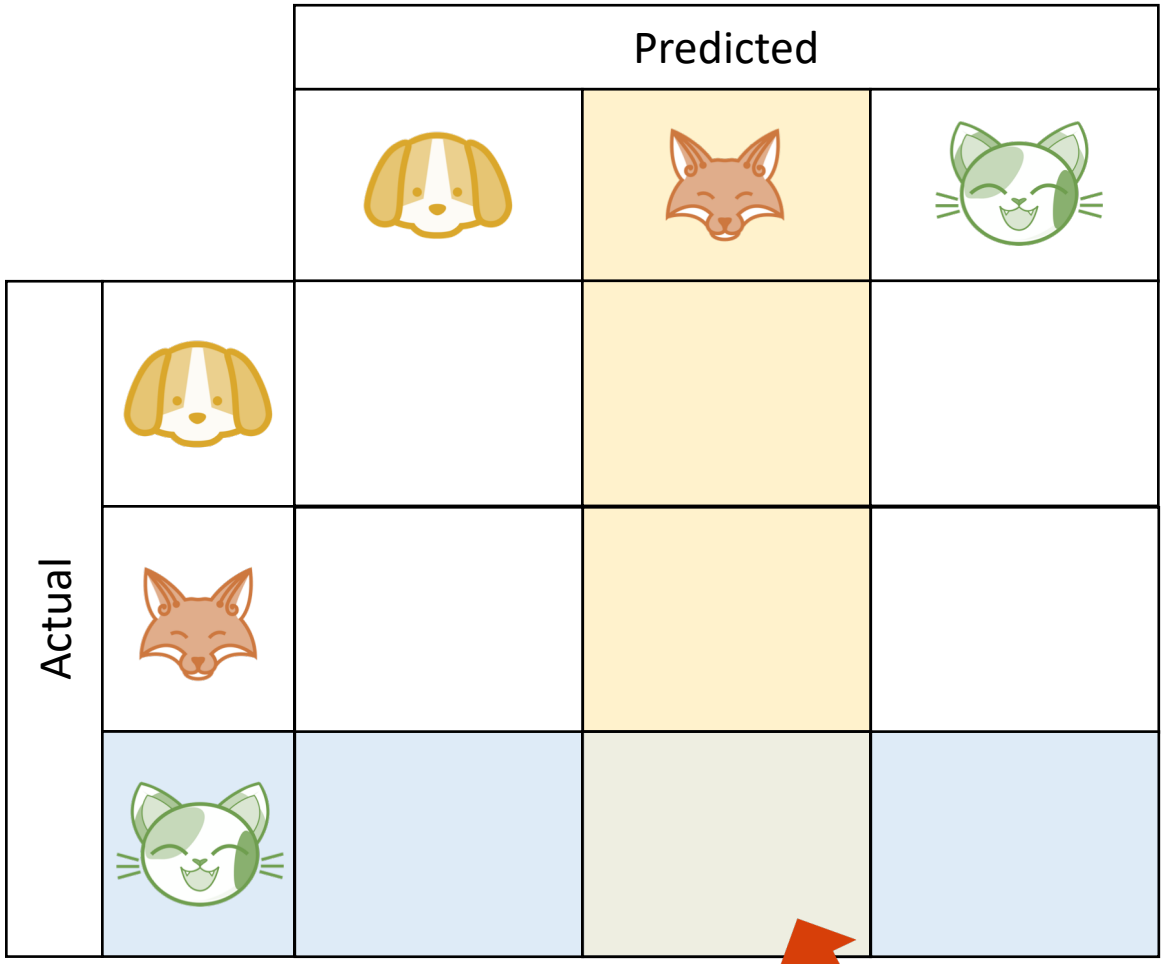


Number of instances that:
Actually Cat & Predicted Cat

© Stefan Lee

Evaluating Multiclass Classifiers

For multiclass case, still consider a matrix where rows are true labels and columns are predictions:

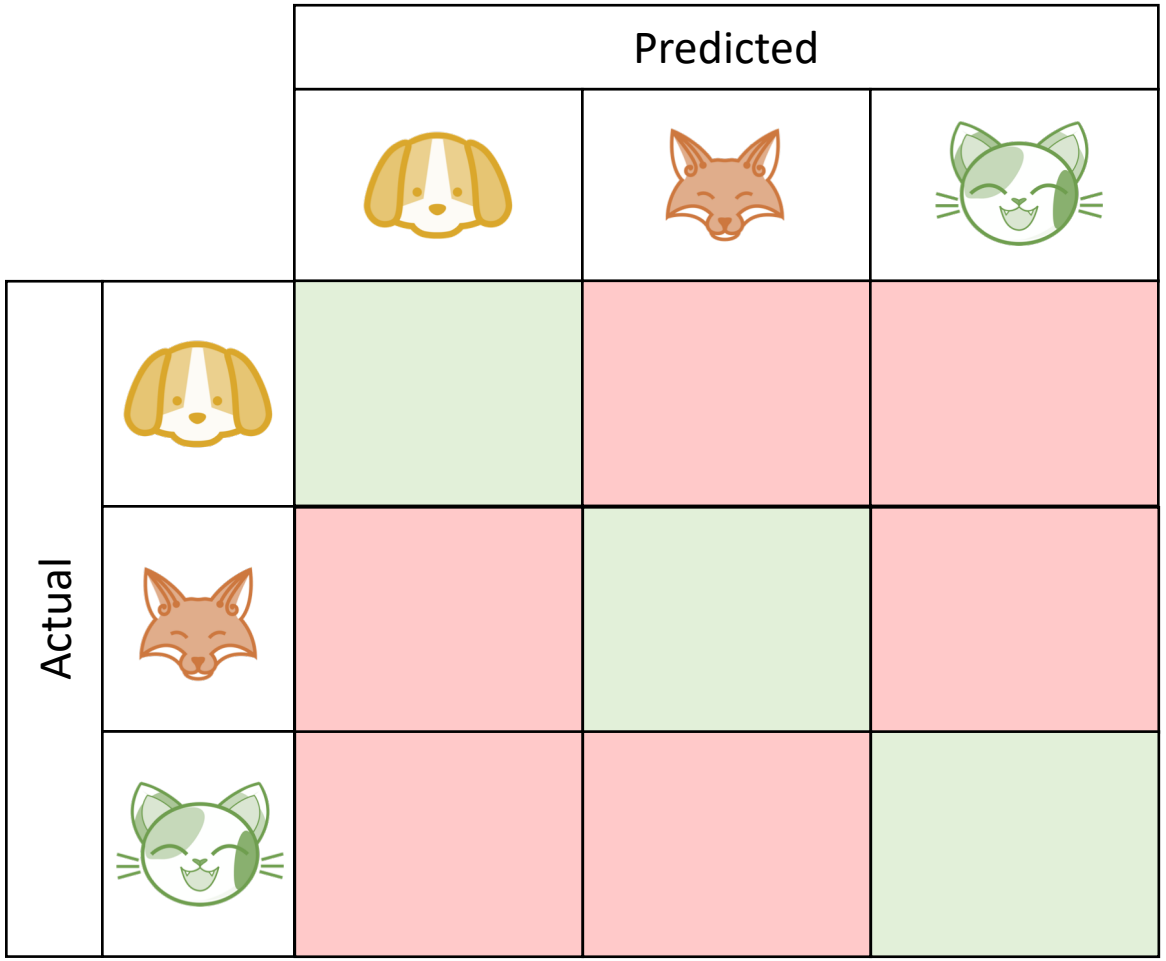


Number of instances that:
Actually Cat & Predicted Fox

© Stefan Lee

Evaluating Multiclass Classifiers

For multiclass case, still consider a matrix where rows are true labels and columns are predictions:



Diagonal contains correct

Off-diagonal errors.

© Stefan Lee

Evaluating Multiclass Classifiers

Example what this could look like:

y	$\hat{y}$
Cat	Dog
Cat	Cat
Cat	Cat
Cat	Cat
Dog	Fox
Dog	Dog
Fox	Dog
Fox	Fox
Dog	Cat
Dog	Dog

		Predicted		
				
Actual		15	3	2
		2	22	10
		4	15	45

© Stefan Lee

Questions Break!

Given the confusion matrix below, what would you say is the classifier's biggest problem?

		Predicted		
				
Actual		15	3	2
		2	22	10
		4	15	45

© Stefan Lee

Back when talking about binary classifiers, defined:

Recall: What fraction of positive does your model predict as positives:

$$Recall = \frac{\#TP}{\#TP + \#FN}$$

Precision: Of things your model predicts as positives, what fraction are correct?

$$Precision = \frac{\#TP}{\#TP + \#FP}$$

© Stefan Lee

Evaluating Multiclass Classifiers

What about recall and precision? **Can compute for each class as if one-vs-all**:

Actual \ Predicted			
	15	3	2
	2	22	10
	4	15	45

$$Recall(Dog) = \frac{\#TP}{\#TP + \#FN} = \frac{15}{15 + 3 + 2}$$

© Stefan Lee

Evaluating Multiclass Classifiers

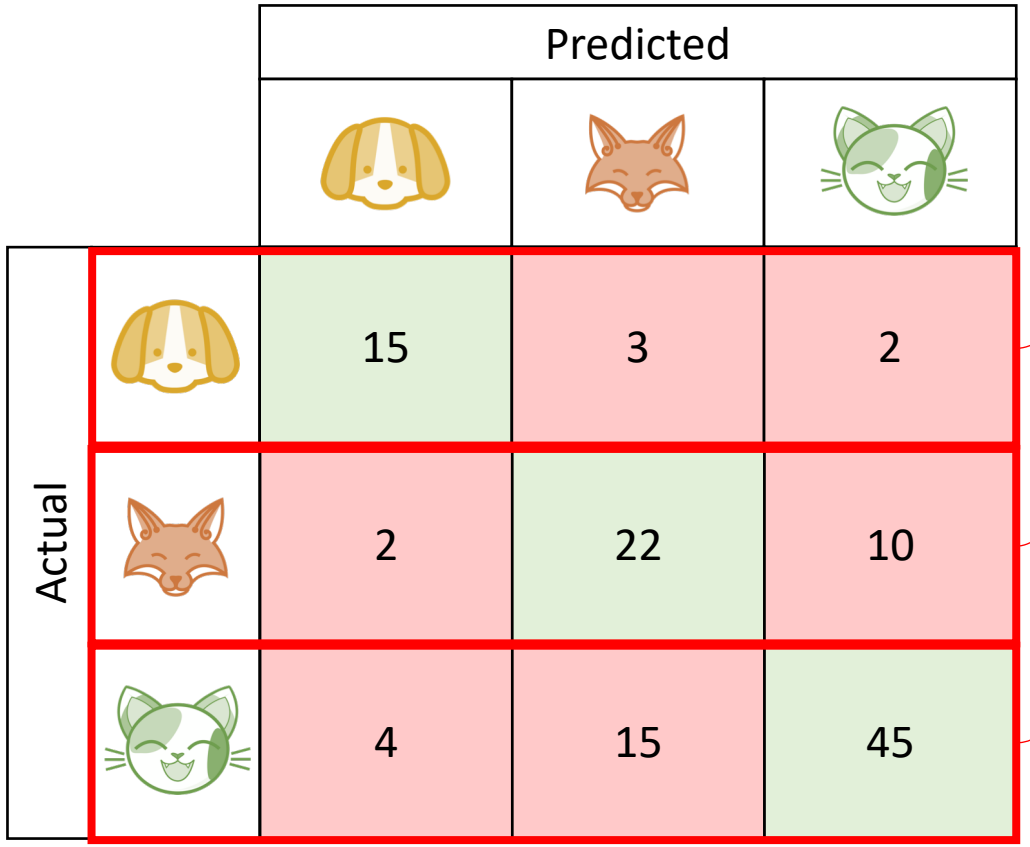
What about recall and precision? **Can compute for each class as if one-vs-all**:

		Predicted		
				
Actual		15	3	2
		2	22	10
		4	15	45

$$Precision(Dog) = \frac{\#TP}{\#TP + \#FP} = \frac{15}{15 + 2 + 4}$$

© Stefan Lee

Evaluating Multiclass Classifiers



$$Recall(Dog) = \frac{15}{15+3+2} = 0.75$$

$$Recall(Fox) = \frac{22}{2+22+10} \approx 0.64$$

$$Recall(Cat) = \frac{45}{4+15+45} \approx 0.70$$

$$Avg.\,Recall = 0.69$$

Could average recall or precision across classes, but…

- Not particularly useful when recall varies significantly between classes
- Doesn't account for class imbalances (weighted version could fix this one)

© Stefan Lee

Today's Learning Objectives

Be able to answer:

- ~~What is multi-class classification?~~
- ~~How can we do multi-class classification?~~
  - ~~How do 1-vs-all / all-vs-all / tree- classifiers work?~~
- ~~How does multiclass logistic regression work?~~
  - ~~What is the softmax function?~~
- ~~How to evaluate multi-class classifiers?~~
  - ~~What is a multiclass confusion matrix?~~
  - ~~How can recall and precision adapt to multiclass?~~

© Stefan Lee

Next time

Next Time: We'll talk about Support Vector Machines!

© Stefan Lee