CSE 152: Computer Vision

Hao Su

Lecture 6: Machine Learning Crash Course: Supervised Learning

Agenda

- Introduction to Classification
- K-nearest Neighbor Algorithm
- Bias-Variance Tradeoff
- Logistic Regression

Image Classification: A core task in Computer Vision

This image by Nikita is licensed under CC-BY 2.0

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

An image classifier

```
def classify_image(image):
  # Some magic here?
  return class_label
```

Unlike e.g. sorting a list of numbers,

no obvious way to hard-code the algorithm for recognizing a cat, or other classes.

Attempts have been made

Find edges

Find corners

John Canny, “A Computational Approach to Edge Detection”, IEEE TPAMI 1986

Machine Learning: Data-Driven Approach

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

First classifier: Nearest Neighbor

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

→ Memorize all data and labels

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

→ Predict the label of the most similar training image

Example Dataset: CIFAR10

10 classes
50,000 training images
10,000 testing images

Alex Krizhevsky, “Learning Multiple Layers of Features from Tiny Images”, Technical Report, 2009.

Example Dataset: CIFAR10

10 classes
50,000 training images
10,000 testing images

Test images and nearest neighbors

Alex Krizhevsky, “Learning Multiple Layers of Features from Tiny Images”, Technical Report, 2009.

Nearest Neighbor classifier

Q: With N examples, how fast are training and prediction?

Nearest Neighbor classifier

Q: With N examples, how fast are training and prediction?

A: Train O(1), predict O(N)

Nearest Neighbor classifier

Q: With N examples, how fast are training and prediction?

A: Train O(1), predict O(N)

This is bad: we want classifiers that are **fast** at prediction; **slow** for training is ok

What does this look like?

K-Nearest Neighbors

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

K = 1

K = 3

K = 5

What does this look like?

What does this look like?

K-Nearest Neighbors: Distance Metric

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p \left| I_1^p - I_2^p \right|$$

L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p \right)^2}$$

Hyperparameters

- What is the best value of k to use? What is the best distance to use?

- These are hyperparameters: choices about the algorithm that we set rather than learn

Theory: Bias/variance trade-off

"It takes surprisingly long time to grok bias and variance deeply, but people that understand bias and variance deeply are often able to drive very rapid progress." *--Andrew Ng*

Bias = accuracy
Variance = precision

Scott Fortmann-Roe

Theory: Bias and Variance

Our data-generated model $f_D(X)$ is a **statistical estimate** of the true function $f(X)$.

Bias: if we train models $f_D(X)$ on many training sets D, bias is the expected difference between their predictions and the true y's.

i.e.

$$Bias = \mathrm{E}[f_D(X) - y]$$

E[] is taken over points X and datasets D

Variance: if we train models $f_D(X)$ on many training sets D, variance is the variance of the estimates:

$$Variance = \mathrm{E}\left[\left(f_D(X) - \bar{f}(X)\right)^2\right]$$

Where $\bar{f}(X) = \mathrm{E}[f_D(X)]$ is the average prediction on X.

Choosing k

We have a bias/variance tradeoff:

- Small k → ?
- Large k → ?

Choosing k

- Small k → low bias, high variance
- Large k → high bias, low variance
- Assume the real data follows the blue curve, with some mean-zero additive noise. Red points are a data sample.

Choosing k

- **Small k** → low bias, high variance
- Large k → high bias, low variance

Choosing k

- **Small k** → low bias, high variance
- Large k → high bias, low variance

Choosing k

- Small k → low bias, high variance
- **Large k** → high bias, low variance

Choosing k

- Small k → low bias, high variance
- **Large k** → high bias, low variance

Theory: Bias and Variance

One has to balance bias and variance

Practice: Setting Hyperparameters

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

Practice: Setting Hyperparameters

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

BAD: K = 1 always works perfectly on training data

Your Dataset

Practice: Setting Hyperparameters

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

BAD: K = 1 always works perfectly on training data

Your Dataset

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

train

test

Practice: Setting Hyperparameters

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

BAD: K = 1 always works perfectly 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 algorithm will perform on new data

train

test

Practice: Setting Hyperparameters

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

BAD: K = 1 always works perfectly 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 algorithm 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

Practice: Setting 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 not used too frequently in deep learning

Practice: Setting Hyperparameters

Example of
5-fold cross-validation
for the value of **k.**

Each point: single
outcome.

The line goes
through the mean, bars
indicated standard
deviation

(Seems that k ~= 7 works best
for this data)

Linear Classification

Neural Network

Linear classifiers

This image is CC0 1.0 public domain

Parametric Approach

Image

Array of **32x32x3** numbers
(3072 numbers total)

$f(\mathbf{x}, \mathbf{W})$

10 numbers giving
class scores

W

parameters
or weights

Parametric Approach: Linear Classifier

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

Image

Array of **32x32x3** numbers
(3072 numbers total)

$$f(\mathbf{x},\mathbf{W})$$

10 numbers giving
class scores

W

parameters
or weights

Parametric Approach: Linear Classifier

$$f(x,W) = \underset{10\times d}{\boxed{W}}\ \overset{d\times 1}{\boxed{x}}$$

Image

$$\xrightarrow{\quad} \underset{10\times 1}{f(\mathbf{x},\mathbf{W})} \xrightarrow{\quad}$$

10 numbers giving class scores

W

parameters or weights

Parametric Approach: Linear Classifier

$$f(x,W) = \underbrace{W}_{10\times d}\underbrace{x}_{d\times 1} + \underbrace{b}_{10\times 1}$$

10x1

Image

$\longrightarrow$ $f(\mathbf{x},\mathbf{W})$ $\longrightarrow$ **10** numbers giving class scores

$\mathbf{W}$

parameters or weights

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

Suppose: 3 training examples, 3 classes.
With 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**

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

L=\frac{1}{N}\sum_i \ell(f(x_i,W),y_i)

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

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

Given a dataset of examples

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

Where x_i is image and y_i is (integer) label

Loss over the dataset is a sum of loss over examples:

$$L = \frac{1}{N} \sum_i \ell(f(x_i, W), y_i)$$

So far,

$$f(x_i, W) = \begin{bmatrix} s_{i,1} \\ s_{i,2} \\ \cdots \\ s_{i,C} \end{bmatrix}, \qquad y_i = k$$

We rewrite y_i as a one-hot vector $\hat{y}_i \in \mathbb{R}^C$:

$$\hat{y}_{i,j} = 1 \text{ if } i = j, \hat{y}_{i,j} = 0 \text{ otherwise}$$

Then, we may set $L(f(x_i, W), \hat{y}_i) = \|f(x_i, W) - \hat{y}_i\|_2^2$ **?**

Classification as regression is not too bad,

But it has problems.

Consider the classification of 1-D data:

There is no way to set W so that

f(x, W)=[1.; 0.] for blue points

f(x, W)=[0.; 1.] for red points

Logistic Regression

New criterion

The score of label category is larger than other categories:

$$score_y \geq score_j \text{ for any } j \neq y$$

How to set up a loss for this goal?

Softmax classifier

Let $score_y = \frac{exp(f(x,W)_y)}{\sum_j exp(f(x,W)_j)}$

If we set $\ell(f(x_i,W),y_i) = -\log score_{y_i}$

$$L = \frac{1}{N}\sum_i \ell(f(x_i,W),y_i)$$

$$= -\frac{1}{N}\sum_i \log \frac{exp(f(x_i,W)_{y_i})}{\sum_{j=1}^{C} exp(f(x_i,W)_j)}$$

Softmax Classifier (Multinomial Logistic Regression)

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

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

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

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

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

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

cat	**3.2**	exp →	**24.5**
car	5.1		164.0
frog	-1.7		0.18

unnormalized probabilities

Softmax Classifier (Multinomial Logistic Regression)

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

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

		Probabilities must be >= 0		Probabilities must sum to 1
cat	**3.2**		**24.5**		**0.13**
car	5.1	exp →	164.0	normalize →	0.87
frog	-1.7		0.18		0.00
			unnormalized probabilities		probabilities

Softmax Classifier (Multinomial Logistic Regression)

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

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

cat	**3.2**	**24.5**	**0.13**
car	5.1	164.0	0.87
frog	-1.7	0.18	0.00

exp → normalize →

Unnormalized log-probabilities / logits

unnormalized probabilities

probabilities

Softmax Classifier (Multinomial Logistic Regression)

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

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

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

	Unnormalized log-probabilities / logits	exp →	unnormalized probabilities	normalize →	probabilities	→
cat	**3.2**		**24.5**		**0.13**	L_i = -log(0.13) = **0.89**
car	5.1		164.0		0.87	
frog	-1.7		0.18		0.00	

Softmax Classifier (Multinomial Logistic Regression)

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

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

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

	Unnormalized log-probabilities / logits	exp →	unnormalized probabilities	normalize →	probabilities
cat	**3.2**		**24.5**		**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

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

Maximum Likelihood Estimation
Choose probabilities to maximize the likelihood of the observed data (See CS 229 for details)

Geometric Interpretation

$$f(x,W) = Wx + b$$

Plot created using Wolfram Cloud

Cat image by Nikita is licensed under CC-BY 2.0

Compare the bias/variance of kNN and LR

kNN

Logistic Regression

Which has larger bias?

Compare the bias/variance of kNN and LR

kNN

Logistic Regression

Which has larger variance?