Lecture 20: Optical Flow (cont.)
Menu for Today (March 19, 2019)

Topics:

- Optical Flow (cont)
- Classification
- Naive Bayes Classifier
- Bayes' Risk

Readings:

- **Today's** Lecture: Forsyth & Ponce (2nd ed.) 15.1, 15.2
- **Next** Lecture: Forsyth & Ponce (2nd ed.) 16.1.3, 16.1.4, 16.1.9

Reminders:

- **Assignment 4**: Local Invariant Features and RANSAC due today
- **Assignment 5**: Scene Recognition with Bag of Words out soon
Today’s “fun” Example: Visual Microphone

The Visual Microphone: Passive Recovery of Sound from Video

Abe Davis
Michael Rubinstein
Neal Wadhwa
Gautham J. Mysore
Fredo Durand
William T. Freeman

Follow-up work to previous lecture’s example of Eulerian video magnification
Lecture 19: Re-cap

**Optical flow** is the apparent motion of brightness patterns in the image

**Applications**
- image and video stabilization in digital cameras, camcorders
- motion-compensated video compression schemes such as MPEG
- image registration for medical imaging, remote sensing
- action recognition
- motion segmentation
Figure credit: M. Srinivasan
Aperture Problem

In which direction is the line moving?

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)
Aperture Problem

In which direction is the line moving?

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)
Aperture Problem
Aperture Problem

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)
Aperture Problem

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)
Aperture Problem

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)
Aperture Problem

- Without distinct features to track, the true visual motion is ambiguous
- Locally, one can compute only the component of the visual motion in the direction perpendicular to the contour
Aperture Problem

- Without distinct features to track, the true visual motion is ambiguous
- Locally, one can compute only the component of the visual motion in the direction perpendicular to the contour
Visual motion is determined when there are distinct features to track, provided:
— the features can be detected and localized accurately; and
— the features can be correctly matched over time
## Motion as **Matching**

<table>
<thead>
<tr>
<th>Representation</th>
<th>Result is...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point/feature based</td>
<td>(very) sparse</td>
</tr>
<tr>
<td>Contour based</td>
<td>(relatively) sparse</td>
</tr>
<tr>
<td>(Differential) gradient based</td>
<td>dense</td>
</tr>
</tbody>
</table>
Optical Flow **Constraint Equation**

Consider image intensity also to be a function of time, $t$. We write

$$I(x, y, t)$$
Consider image intensity also to be a function of time, $t$. We write

$$I(x, y, t)$$

Applying the **chain rule for differentiation**, we obtain

$$\frac{dI(x, y, t)}{dt} = I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_t$$

where subscripts denote partial differentiation
Optical Flow Constraint Equation

Consider image intensity also to be a function of time, $t$. We write

$$I(x, y, t)$$

Applying the **chain rule for differentiation**, we obtain

$$\frac{dI(x, y, t)}{dt} = I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_t$$

where subscripts denote partial differentiation.

Define $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$. Then $[u, v]$ is the 2-D motion and the space of all such $u$ and $v$ is the **2-D velocity space**.
Optical Flow **Constraint Equation**

Consider image intensity also to be a function of time, $t$. We write

$$I(x, y, t)$$

Applying the **chain rule for differentiation**, we obtain

$$\frac{dI(x, y, t)}{dt} = I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_t$$

where subscripts denote partial differentiation.

Define $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$. Then $[u, v]$ is the 2-D motion and the space of all such $u$ and $v$ is the **2-D velocity space**

Suppose $\frac{dI(x, y, t)}{dt} = 0$. Then we obtain the (classic) **optical flow constraint equation**

$$I_x u + I_y v + I_t = 0$$
Optical Flow **Constraint Equation**

Consider image intensity also to be a function of time, \( t \). We write

\[
I(x, y, t)
\]

Applying the **chain rule for differentiation**, we obtain

\[
\frac{dI(x, y, t)}{dt} = I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_t
\]

where subscripts denote partial differentiation.

Define \( u = \frac{dx}{dt} \) and \( v = \frac{dy}{dt} \). Then \([u, v]\) is the 2-D motion and the space of all such \( u \) and \( v \) is the **2-D velocity space**.

Suppose \( \frac{dI(x, y, t)}{dt} = 0 \). Then we obtain the (classic) **optical flow constraint equation**

\[
I_x u + I_y v + I_t = 0
\]
Consider image intensity also to be a function of time, \( I(x, y, t) \).

We write

\[
\frac{dI(x, y, t)}{dt} = 0
\]

Applying the chain rule for differentiation, we obtain

\[
\frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \frac{\partial I}{\partial t} = 0
\]

Define \( u \) and \( v \). Then \( u \) is the 2-D motion and the space of all such \( u \) and \( v \) is the 2-D velocity space.

Suppose

\[
\frac{dI(x, y, t)}{dt} = 0
\]

Then we obtain the (classic) optical flow constraint equation

\[
I_x u + I_y v + I_t = 0
\]

What does this mean, and why is it reasonable?
Consider image intensity also to be a function of time, $I(x, y, t)$. We write:

Applying the chain rule for differentiation, we obtain:

$$\frac{dI(x, y, t)}{dt} = 0$$

Define $u = u(x, y, t)$ and $v = v(x, y, t)$. Then $u$ is the 2-D motion and the space of all such $u$ and $v$ is the 2-D velocity space.

Suppose:

Then we obtain the (classic) **optical flow constraint equation**

$$I_x u + I_y v + I_t = 0$$

What does this mean, and why is it reasonable?

**Suppose** $\frac{dI(x, y, t)}{dt} = 0$. Then we obtain the (classic) **optical flow constraint equation**

$$I_x u + I_y v + I_t = 0$$

*Image Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Consider image intensity also to be a function of time, . We write

Applying the chain rule for differentiation, we obtain

where subscripts denote partial differentiation. Define and . Then is the 2-D motion and the space of all such and is the 2-D velocity space. Suppose . Then we obtain the (classic) optical flow constraint equation

What does this mean, and why is it reasonable?

Suppose . Then we obtain the (classic) optical flow constraint

The image shows a sequence of frames labeled \( I(x, y, 1) \), \( I(x, y, 2) \), ..., \( I(x, y, k) \) with scene points moving through the sequence. The equation \( I_x u + I_y v + I_t = 0 \) represents the optical flow constraint.
Consider image intensity also to be a function of time, \( . \) We write

\[
\frac{d}{dt} I(x, y, t)
\]

Applying the chain rule for differentiation, we obtain

\[
I_x u + I_y v + I_t = 0
\]

Define \( I_x \) and \( I_y \). Then \( I_x u + I_y v + I_t = 0 \) is the 2-D motion and the space of all such \( I_x \) and \( I_y \) is the 2-D velocity space.

Suppose \( I(x(t), y(t), t) = C \) constant.

**Brightness Constancy Assumption:** Brightness of the point remains the same.

What does this mean, and why is it reasonable?

**Optical Flow Constraint Equation**

\[
I(x(t), y(t), t) = C
\]

constant

Suppose \( \frac{dI(x, y, t)}{dt} = 0 \). Then we obtain the (classic) **optical flow constraint equation**

\[
I_x u + I_y v + I_t = 0
\]
Aside: Derivation of Optical Flow Constraint

\[ I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t) \]

For small space-time step, brightness of a point is the same
Aside: Derivation of Optical Flow Constraint

\[ I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t) \]

For small space-time step, brightness of a point is the same

**Insight:**
If the time step is really small, we can *linearize* the intensity function
Aside: Derivation of Optical Flow Constraint

\[ I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t) \]

Multivariable Taylor Series Expansion
(First order approximation, two variables)

\[ f(x, y) \approx f(a, b) + f_x(a, b)(x - a) - f_y(a, b)(y - b) \]
Aside: Derivation of Optical Flow Constraint

\[ I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t) \]

Multivariable Taylor Series Expansion
(First order approximation, two variables)

\[ f(x, y) \approx f(a, b) + f_x(a, b)(x - a) - f_y(a, b)(y - b) \]

\[ I(x, y, t) + \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t = I(x, y, t) \quad \text{assuming small motion} \]
Aside: Derivation of Optical Flow Constraint

\[ I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t) \]

**Multivariable Taylor Series Expansion**

(First order approximation, two variables)

\[ f(x, y) \approx f(a, b) + f_x(a, b)(x - a) + f_y(a, b)(y - b) \]

Partial derivative

\[ I(x, y, t) + \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t = I(x, y, t) \text{ assuming small motion} \]

Fixed point

Cancel terms

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
Aside: Derivation of Optical Flow Constraint

\[ I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t) \]

Multivariable Taylor Series Expansion
(First order approximation, two variables)

\[ f(x, y) \approx f(a, b) + f_x(a, b)(x - a) - f_y(a, b)(y - b) \]

\[ I(x, y, t) + \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t = I(x, y, t) \]
assuming small motion

\[ \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t = 0 \]
cancel terms
Aside: Derivation of Optical Flow Constraint

\[ I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t) \]

Multivariable Taylor Series Expansion
(First order approximation, two variables)

\[ f(x, y) \approx f(a, b) + f_x(a, b)(x - a) - f_y(a, b)(y - b) \]

\[ I(x, y, t) + \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t = I(x, y, t) \quad \text{assuming small motion} \]

\[ \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t = 0 \quad \text{divide by } \delta t \]

\[ \text{take limit } \delta t \rightarrow 0 \]
Aside: Derivation of Optical Flow Constraint

\[ I(x + ud_t, y + v\delta t, t + \delta t) = I(x, y, t) \]

Multivariable Taylor Series Expansion
(First order approximation, two variables)

\[ f(x, y) \approx f(a, b) + f_x(a, b)(x - a) - f_y(a, b)(y - b) \]

\[ I(x, y, t) + \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t = I(x, y, t) \] assuming small motion

\[ \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t = 0 \]

divide by \( \delta t \)
take limit \( \delta t \to 0 \)

\[ \frac{\partial I}{\partial x} dt + \frac{\partial I}{\partial y} dy + \frac{\partial I}{\partial t} = 0 \]

Brightness Constancy Equation

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
How do we compute ...

\[ I_x u + I_y v + I_t = 0 \]
How do we compute …

$$I_x u + I_y v + I_t = 0$$

$$I_x = \frac{\partial I}{\partial x}, \quad I_y = \frac{\partial I}{\partial y}$$

spatial derivative
How do we compute …

\[ I_x u + I_y v + I_t = 0 \]

\[
I_x = \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y}
\]

**spatial derivative**

Forward difference
   Sobel filter
   Scharr filter
   …
How do we compute …

\[ I_x u + I_y v + I_t = 0 \]

\[ I_x = \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y} \]

spatial derivative

\[ I_t = \frac{\partial I}{\partial t} \]

temporal derivative

Forward difference
Sobel filter
Scharr filter
…

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
How do we compute …

\[ I_x u + I_y v + I_t = 0 \]

\[ I_x = \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y} \]

spatial derivative

Forward difference
Sobel filter
Scharr filter
…

\[ I_t = \frac{\partial I}{\partial t} \]

temporal derivative

Frame differencing

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Frame Differencing: Example

\[ I_t = \frac{\partial I}{\partial t} \]

\( t + 1 \)

\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 10 & 10 \\
1 & 1 & 10 & 10 \\
1 & 1 & 10 & 10 \\
\end{array}
\]

\( t \)

\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 10 & 10 \\
1 & 1 & 10 & 10 \\
1 & 1 & 10 & 10 \\
1 & 1 & 10 & 10 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & -9 & -9 & -9 \\
0 & -9 & 0 & 0 \\
0 & -9 & 0 & 0 \\
0 & -9 & 0 & 0 \\
\end{array}
\]

(example of a forward temporal difference)
\[ I_x = \frac{\partial I}{\partial x} \]

\[ I_y = \frac{\partial I}{\partial y} \]

\[ I_t = \frac{\partial I}{\partial t} \]
How do we compute …

\[ I_x u + I_y v + I_t = 0 \]

**Spatial Derivative**

\[ I_x = \frac{\partial I}{\partial x}, \quad I_y = \frac{\partial I}{\partial y} \]

- Forward difference
- Sobel filter
- Scharr filter

**Optical Flow**

\[ u = \frac{dx}{dt}, \quad v = \frac{dy}{dt} \]

**Temporal Derivative**

\[ I_t = \frac{\partial I}{\partial t} \]

How do you compute this? Frame differencing

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
How do we compute …

\[ I_x u + I_y v + I_t = 0 \]

\[ I_x = \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y} \]

\text{spatial derivative}

\[ u = \frac{dx}{dt} \quad v = \frac{dy}{dt} \]

\text{optical flow}

\[ I_t = \frac{\partial I}{\partial t} \]

\text{temporal derivative}

Forward difference
Sobel filter
Scharr filter
…

\textbf{We need to solve for this!}
(this is the unknown in the optical flow problem)

Frame differencing

\textbf{Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)}
How do we compute …

\[ I_x u + I_y v + I_t = 0 \]

- **Spatial derivative**
  \[ I_x = \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y} \]
  
  - Forward difference
  - Sobel filter
  - Scharr filter
  …

- **Optical flow**
  \[ u = \frac{dx}{dt} \quad v = \frac{dy}{dt} \]
  
  - Solution lies on a line
  - Cannot be found uniquely with a single constraint

- **Temporal derivative**
  \[ I_t = \frac{\partial I}{\partial t} \]
  
  - Frame differencing

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Equation determines a straight line in velocity space

\[ I_x u + I_y v + I_t = 0 \]

many combinations of \( u \) and \( v \) will satisfy the equality
Lucas-Kanade

Observations:
1. The 2-D motion, \([u, v]\), at a given point, \([x, y]\), has two degrees-of-freedom.
2. The partial derivatives, \(I_x, I_y, I_t\), provide one constraint.
3. The 2-D motion, \([u, v]\), cannot be determined locally from \(I_x, I_y, I_t\) alone.
Lucas-Kanade

Observations:
1. The 2-D motion, \([u, v]\), at a given point, \([x, y]\), has two degrees-of-freedom
2. The partial derivatives, \(I_x, I_y, I_t\), provide one constraint
3. The 2-D motion, \([u, v]\), cannot be determined locally from \(I_x, I_y, I_t\) alone

Lucas–Kanade Idea:
Obtain additional local constraint by computing the partial derivatives, \(I_x, I_y, I_t\), in a window centered at the given \([x, y]\)
Lucas-Kanade

Observations:
1. The 2-D motion, \([u, v]\), at a given point, \([x, y]\), has two degrees-of-freedom
2. The partial derivatives, \(I_x, I_y, I_t\), provide one constraint
3. The 2-D motion, \([u, v]\), cannot be determined locally from \(I_x, I_y, I_t\) alone

Lucas–Kanade Idea:
Obtain additional local constraint by computing the partial derivatives, \(I_x, I_y, I_t\), in a window centered at the given \([x, y]\)

Constant Flow Assumption: nearby pixels will likely have same optical flow
Suppose \([x_1, y_1] = [x, y]\) is the (original) center point in the window. Let \([x_2, y_2]\) be any other point in the window. This gives us two equations that we can write

\[
I_{x_1} u + I_{y_1} v = -I_{t_1} \\
I_{x_2} u + I_{y_2} v = -I_{t_2}
\]

and that can be solved locally for \(u\) and \(v\) as

\[
\begin{bmatrix}
  u \\
  v
\end{bmatrix} = - \begin{bmatrix}
  I_{x_1} & I_{y_1} \\
  I_{x_2} & I_{y_2}
\end{bmatrix}^{-1} \begin{bmatrix}
  I_{t_1} \\
  I_{t_2}
\end{bmatrix}
\]

provided that \(u\) and \(v\) are the same in both equations and provided that the required matrix inverse exists.
Considering all \( n \) points in the window, one obtains

\[
I_{x1} u + I_{y1} v = -I_{t1} \\
I_{x2} u + I_{y2} v = -I_{t2} \\
\vdots \\
I_{xn} u + I_{yn} v = -I_{tn}
\]

which can be written as the matrix equation

\[
A \mathbf{v} = \mathbf{b}
\]

where \( \mathbf{v} = [u, v]^T \), \( A = \begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{xn} & I_{yn} \end{bmatrix} \) and \( \mathbf{b} = -\begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \\ I_{tn} \end{bmatrix} \)
The standard least squares solution, $\bar{v}$, to is

$$\bar{v} = (A^T A)^{-1} A^T b$$

again provided that $u$ and $v$ are the same in all equations and provided that the rank of $A^T A$ is 2 (so that the required inverse exists)
Lucas-Kanade

Note that we can explicitly write down an expression for \( A^T A \) as

\[
A^T A = \begin{bmatrix}
\sum I_x^2 & \sum I_x I_y \\
\sum I_x I_y & \sum I_y^2
\end{bmatrix}
\]

which is identical to the matrix \( C \) that we saw in the context of Harris corner detection.
Note that we can explicitly write down an expression for $\mathbf{A}^T \mathbf{A}$ as

$$\mathbf{A}^T \mathbf{A} = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

which is identical to the matrix $\mathbf{C}$ that we saw in the context of Harris corner detection.
Lucas-Kanade Summary

A dense method to compute motion, \([u, v]\) at every location in an image

Key Assumptions:

1. Motion is slow enough and smooth enough that differential methods apply (i.e., that the partial derivatives, \(I_x, I_y, I_t\), are well-defined)

2. The optical flow constraint equation holds (i.e., \(\frac{dI(x, y, t)}{dt} = 0\))

3. A window size is chosen so that motion, \([u, v]\), is constant in the window

4. A window size is chosen so that the rank of \(A^TA\) is 2 for the window
Aside: Optical Flow Smoothness Constraint

Many methods trade off a ‘departure from the optical flow constraint’ cost with a ‘departure from smoothness’ cost.

The optimization objective to minimize becomes

$$ E = \int \int (I_x u + I_y v + I_t)^2 + \lambda (|| \nabla u ||^2 + || \nabla v ||^2) $$

where \( \lambda \) is a weighing parameter.
Horn-Schunck Optical Flow

\[
\min_{u,v} \sum_{i,j} \left\{ E_s(i,j) + \lambda E_d(i,j) \right\}
\]

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Horn-Schunck Optical Flow

Brightness constancy

\[ E_d(i, j) = \left[ I_x u_{ij} + I_y v_{ij} + I_t \right]^2 \]

Smoothness

\[ E_s(i, j) = \frac{1}{4} \left[ (u_{ij} - u_{i+1,j})^2 + (u_{ij} - u_{i,j+1})^2 + (v_{ij} - v_{i+1,j})^2 + (v_{ij} - v_{i,j+1})^2 \right] \]
Summary

Motion, like binocular stereo, can be formulated as a matching problem. That is, given a scene point located at \((x_0, y_0)\) in an image acquired at time \(t_0\), what is its position, \((x_1, y_1)\), in an image acquired at time \(t_1\)?

Assuming image intensity does not change as a consequence of motion, we obtain the (classic) **optical flow constraint equation**

\[
I_x u + I_y v + I_t = 0
\]

where \([u, v]\) is the 2-D motion at a given point, \([x, y]\), and \(I_x, I_y, I_t\) are the partial derivatives of intensity with respect to \(x, y\), and \(t\)

**Lucas–Kanade** is a dense method to compute the motion, \([u, v]\), at every location in an image
CPSC 425: Computer Vision

Lecture 20: Classification
Classification

**Problem:**
Assign new observations into one of a fixed set of categories (classes)

**Key Idea(s):**
Build a model of data in a given category based on observations of instances in that category
Classification

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

---

cat
Classification

Image classification:
- 82% cat
- 15% dog
- 2% hat
- 1% mug

What the computer sees
A **classifier** is a procedure that accepts as input a set of features and outputs a class **label**.

Classifiers can be binary (face vs. not-face) or multi-class (cat, dog, horse, ...).

We build a classifier using a **training set** of labelled examples \( \{ (x_i, y_i) \} \), where each \( x_i \) is a feature vector and each \( y_i \) is a class label.

Given a previously unseen observation, we use the classifier to predict its class label.
Classification

- Collect a database of images with labels
- Use ML to train an image classifier
- Evaluate the classifier on test images

![Example training set](image)

Label → Feature vector computed from the image
Example 1: A Classification Problem

Categorize images of fish — “Atlantic salmon” vs “Pacific salmon”

Use **features** such as length, width, lightness, fin shape & number, mouth position, etc.

Given a previously unobserved image of a salmon, use the learned classifier to guess whether it is an Atlantic or Pacific salmon

**Figure credit:** Duda & Hart
Example 2: Real Classification Problem

SUN Dataset
- 131K images
- 908 scene categories

<table>
<thead>
<tr>
<th>indoor</th>
<th>shopping and dining</th>
<th>auto showroom</th>
</tr>
</thead>
<tbody>
<tr>
<td>outdoor natural</td>
<td>workplace (office building, factory, lab, etc.)</td>
<td>bakery kitchen</td>
</tr>
<tr>
<td>outdoor man-made</td>
<td>home or hotel</td>
<td>bakery shop</td>
</tr>
<tr>
<td>transportation (vehicle interiors, stations, etc.)</td>
<td>bank indoor</td>
<td></td>
</tr>
<tr>
<td>sports and leisure</td>
<td></td>
<td>bank vault</td>
</tr>
<tr>
<td>cultural (art, education, religion, military, law, politics, etc.)</td>
<td>banquet hall</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>bar</td>
</tr>
</tbody>
</table>
**Example 3**: Real Classification Problem

**ImageNet Dataset**
- 14 Million images
- 21K **object** categories
Let $c$ be the class label and let $x$ be the measurement (i.e., evidence).

Bayes Rule (Review and Definitions)

$$P(c|x) = \frac{P(x|c)p(c)}{P(x)}$$

- **P(c|x)**: Posterior probability
- **P(x|c)p(c)**: Class-conditional probability (a.k.a. likelihood)
- **P(x)**: Unconditional probability (a.k.a. marginal likelihood)
- **p(c)**: Prior probability

**Definitions:**
- **Class-conditional probability**: The probability of a class given the evidence.
- **Prior probability**: The probability of a class before observing the evidence.
- **Posterior probability**: The probability of a class after observing the evidence.
Bayes Rule (Review and Definitions)

Let c be the **class label** and let x be the **measurement** (i.e., evidence)

**Simple** case:
- binary classification; i.e., $c \in \{1, 2\}$
- features are 1D; i.e., $x \in \mathbb{R}$

$$P(c|x) = \frac{P(x|c)p(c)}{P(x)}$$

**General** case:
- multi-class; i.e., $c \in \{1, ..., 1000\}$
- features are high-dimensional; i.e., $x \in \mathbb{R}^{2,000+}$
Example: Discrete Bayes Classifier

Assume we have two classes: \( c_1 = \text{male} \quad c_2 = \text{female} \)

We have a person who’s gender we don’t know, who’s name is drew
Example: Discrete Bayes Classifier

Assume we have two classes: $c_1 = \text{male} \quad c_2 = \text{female}$

We have a person who’s gender we don’t know, who’s name is *drew*
Example: Discrete Bayes Classifier

Assume we have two classes: \( c_1 = \text{male} \quad c_2 = \text{female} \)

We have a person who’s gender we don’t know, who’s name is drew

Classifying drew as being male or female is equivalent to asking is it more probable that drew is male or female, i.e. which is greater \( p(\text{male}|\text{drew}) \cdot p(\text{drew}) \) \( \frac{p(\text{female}|\text{drew})}{p(\text{male}|\text{drew})} \)

Example: Discrete Bayes Classifier

Drew Carey

Drew Barrymore

Example from: Eamonn Keogh
Example: Discrete Bayes Classifier

Assume we have two classes: \( c_1 = \text{male} \quad c_2 = \text{female} \)

We have a person who’s gender we don’t know, who’s name is \( drew \)

Classifying \( drew \) as being male or female is equivalent to asking is it more probable that \( drew \) is male or female, i.e. which is greater \( p(\text{male}|drew) \) \( p(\text{female}|drew) \)

\[
p(\text{male}|drew) = \frac{p(drew|\text{male})p(\text{male})}{p(drew)}
\]

Example from: Eamonn Keogh
Example: Discrete Bayes Classifier

\[ p(\text{male}|drew) = \frac{p(drew|\text{male})p(\text{male})}{p(drew)} \]

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drew</td>
<td>Male</td>
</tr>
<tr>
<td>Claudia</td>
<td>Female</td>
</tr>
<tr>
<td>Drew</td>
<td>Female</td>
</tr>
<tr>
<td>Drew</td>
<td>Female</td>
</tr>
<tr>
<td>Alberto</td>
<td>Male</td>
</tr>
<tr>
<td>Karin</td>
<td>Female</td>
</tr>
<tr>
<td>Nina</td>
<td>Female</td>
</tr>
<tr>
<td>Sergio</td>
<td>Male</td>
</tr>
</tbody>
</table>
**Example**: Discrete Bayes Classifier

\[
p(\text{male}) = \\
p(drew | \text{male}) = \\
p(drew) = \\
p(\text{male} | drew) = \frac{p(drew | \text{male}) p(\text{male})}{p(drew)}
\]
Example: Discrete Bayes Classifier

\[ p(\text{male}) = \frac{3}{8} \]

\[ p(\text{drew}|\text{male}) = \]

\[ p(\text{drew}) = \]

\[ p(\text{male}|\text{drew}) = \frac{p(\text{drew}|\text{male})p(\text{male})}{p(\text{drew})} \]

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drew</td>
<td>Male</td>
</tr>
<tr>
<td>Claudia</td>
<td>Female</td>
</tr>
<tr>
<td>Drew</td>
<td>Female</td>
</tr>
<tr>
<td>Drew</td>
<td>Female</td>
</tr>
<tr>
<td>Alberto</td>
<td>Male</td>
</tr>
<tr>
<td>Karin</td>
<td>Female</td>
</tr>
<tr>
<td>Nina</td>
<td>Female</td>
</tr>
<tr>
<td>Sergio</td>
<td>Male</td>
</tr>
</tbody>
</table>

Example from: Eamonn Keogh
Example: Discrete Bayes Classifier

\[ p(\text{male}) = \frac{3}{8} \]

\[ p(\text{drew} | \text{male}) = \frac{1}{3} \]

\[ p(\text{drew}) = \]

\[ p(\text{male} | \text{drew}) = \frac{p(\text{drew} | \text{male}) p(\text{male})}{p(\text{drew})} \]

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drew</td>
<td>Male</td>
</tr>
<tr>
<td>Claudia</td>
<td>Female</td>
</tr>
<tr>
<td>Drew</td>
<td>Female</td>
</tr>
<tr>
<td>Drew</td>
<td>Female</td>
</tr>
<tr>
<td>Alberto</td>
<td>Male</td>
</tr>
<tr>
<td>Karin</td>
<td>Female</td>
</tr>
<tr>
<td>Nina</td>
<td>Female</td>
</tr>
<tr>
<td>Sergio</td>
<td>Male</td>
</tr>
</tbody>
</table>
Example: Discrete Bayes Classifier

\[ p(\text{male}) = \frac{3}{8} \]

\[ p(\text{drew}|\text{male}) = \frac{1}{3} \]

\[ p(\text{drew}) = \frac{3}{8} \]

\[ p(\text{male}|\text{drew}) = \frac{p(\text{drew}|\text{male})p(\text{male})}{p(\text{drew})} \]

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drew</td>
<td>Male</td>
</tr>
<tr>
<td>Claudia</td>
<td>Female</td>
</tr>
<tr>
<td>Drew</td>
<td>Female</td>
</tr>
<tr>
<td>Drew</td>
<td>Female</td>
</tr>
<tr>
<td>Alberto</td>
<td>Male</td>
</tr>
<tr>
<td>Karin</td>
<td>Female</td>
</tr>
<tr>
<td>Nina</td>
<td>Female</td>
</tr>
<tr>
<td>Sergio</td>
<td>Male</td>
</tr>
</tbody>
</table>
Example: Discrete Bayes Classifier

\[ p(\text{male}) = \frac{3}{8} \]
\[ p(drew|\text{male}) = \frac{1}{3} \]
\[ p(drew) = \frac{3}{8} \]
\[ p(\text{male}|drew) = \frac{p(drew|\text{male})p(\text{male})}{p(drew)} = 0.125 \]
Example: Discrete Bayes Classifier

\[
p(\text{male}) = \frac{3}{8} \quad p(\text{female}) = \frac{5}{8}
\]

\[
p(\text{drew}|\text{male}) = \frac{1}{3} \quad p(\text{drew}|\text{female}) = \frac{2}{5}
\]

\[
p(\text{drew}) = \frac{3}{8}
\]

\[
p(\text{male}|\text{drew}) = \frac{p(\text{drew}|\text{male})p(\text{male})}{p(\text{drew})} = 0.125
\]

\[
p(\text{female}|\text{drew}) = \frac{p(\text{drew}|\text{female})p(\text{female})}{p(\text{drew})} = 0.25
\]
Bayes Rule (Review and Definitions)

Let \( c \) be the **class label** and let \( x \) be the **measurement** (i.e., evidence)

**Simple** case:
- binary classification; i.e., \( c \in \{1, 2\} \)
- features are 1D; i.e., \( x \in \mathbb{R} \)

\[
P(c|x) = \frac{P(x|c)p(c)}{P(x)}
\]

**General** case:
- multi-class; i.e., \( c \in \{1, ..., 1000\} \)
- features are high-dimensional; i.e., \( x \in \mathbb{R}^{2,000+} \)
Bayes’ Risk

Some errors may be inevitable: the minimum risk (shaded area) is called the Bayes’ risk

Forsyth & Ponce (2nd ed.) Figure 15.1
Discriminative vs. Generative

Finding a decision boundary is not the same as modeling a conditional density — while a normal density here is a poor fit to $P(1|x)$, the quality of the classifier depends only on how well the boundary is positioned.
Discriminative vs. Generative

Finding a **decision boundary** is not the same as modeling a **conditional density** — while a normal density here is a poor fit to $P(1|x)$, the quality of the classifier depends only on how well the boundary is positioned.