Computer Vision Systems Programming VO

Models vs. Algorithms

Christopher Pramerdorfer

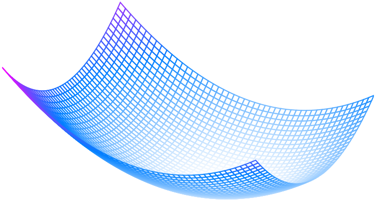
Computer Vision Lab, Vienna University of Technology

Topics

Models vs. algorithms

Approaching computer vision problems

Numerical optimization



Solving CV Problems

CV is about infering some world state $\mathbf{w}$ from measurements $\mathbf{x}$

$\mathbf{x}$ is a feature vector extracted from images

$\mathbf{w} \in \mathbb{R}^m$: **regression** problem (e.g. stereo)

$\mathbf{w} \in \mathbb{N}^m$: **classification** problem (e.g. face recognition)

Solving CV Problems
How Not To Do It

Don't think in terms of **algorithms**

- First I'll convert my image to grayscale
- Then I'll threshold it at some value
- Then I'll clean up holes via morphology

Such solutions

- Rely on many magic numbers that all interact
- Do not encode uncertainty about $\mathbf{w}$

Solving CV Problems

How To Do It

Think about how your data came into being

Use this information to **model** the relationship between $\mathbf{x}$ and $\mathbf{w}$

Ideally use probabilistic models as inference is ill-posed

“Say **what** you want to do, not **how** you are going to to it”
$\sim$ A. Fitzgibbon

On this basis, CV is about

- Defining a **model** that relates $\mathbf{x}$ and $\mathbf{w}$
- **Learning** its parameters $\boldsymbol{\theta}$ from **training samples**
- Using an **inference algorithm** that returns $\Pr(\mathbf{w}|\mathbf{x}, \boldsymbol{\theta})$

Solving CV Problems
Example: Background Subtraction

We want to segment images in foreground and background

- Based on the perceived brightness of objects
- In scenes with a constant background
- On a per-pixel basis

Image from Prince 2012

Therefore we have $w \in \{0, 1\}$ (classification problem)

Given our problem, we model

$$\begin{aligned}
\Pr(x|w=0) &= \underset{x}{\mathrm{Norm}}(\mu, \sigma^2) \\
\Pr(x|w=1) &= \underset{x}{\mathrm{Uniform}}(0, 255) \\
\Pr(w) &= \underset{w}{\mathrm{Bern}}(\lambda)
\end{aligned}$$

We assume $\lambda = 0.5$

And learn $\boldsymbol{\theta} = (\mu, \sigma)$ from training samples $\{x_i\}_{i=1}^n$

- Using the maximum likelihood method
- Assuming that the training samples are independent

We thus learn

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} \Pr(x_1, \ldots, x_n | \boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} \Pr(x_i | \boldsymbol{\theta}) \end{aligned}$$

Continuing from before

$$
\begin{aligned}
\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\arg\max} \prod_{i=1}^{n} \underset{x_i}{\mathrm{Norm}}(\mu, \sigma^2) \\
&= \underset{\boldsymbol{\theta}}{\arg\max} \sum_{i=1}^{n} \log \underset{x_i}{\mathrm{Norm}}(\mu, \sigma^2) \\
&= \underset{\boldsymbol{\theta}}{\arg\max} \left(-0.5n \log \sigma^2 - 0.5n \log(2\pi) - 0.5 \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{\sigma^2} \right)
\end{aligned}
$$

If we differentiate and equate to zero, we obtain

$$\begin{aligned}\hat{\mu} &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ \hat{\sigma}^2 &= \frac{1}{n}\sum_{i=1}^{n} (x_i - \hat{\mu})^2\end{aligned}$$

We have derived **algorithms** for finding our model parameters

Models vs. Algorithms

Models describe the relationship between $\mathbf{x}$ and $\mathbf{w}$

Algorithms are recipes for finding $\boldsymbol{\theta}$

Model	Algorithm
Gaussian	Mean & variance
Linear models*	Linear least squares
Structure from motion*	Bundle adjustment
Rigid motion between point sets*	ICP

* Assuming zero-mean Gaussian noise

Inference

In the example case we have modeled $\Pr(\mathbf{x}|\mathbf{w})$

- Such models are called **generative**
- In this case, we perform inference using Bayes' rule

$$\Pr(\mathbf{w}|\mathbf{x}) = \frac{\Pr(\mathbf{x}|\mathbf{w})\Pr(\mathbf{w})}{\Pr(\mathbf{x})}$$

We can also model $\Pr(\mathbf{w}|\mathbf{x})$ directly

- Such models are called **discriminative**
- In this case, we can perform inference directly

Assume $x = 100, \mu = 95, \sigma = 2$

In this case, we obtain

- $\Pr(w = 0) = \Pr(w = 1) = 0.5$
- $\Pr(x = 100|w = 0) = \mathrm{Norm}_{100}(90, 2^2) \approx 0.009$
- $\Pr(x = 100) = \sum_{i \in \{0,1\}} \Pr(x = 100|w = i)\Pr(w = i) \approx .007$
- $\Pr(w|x = 100) \approx (0.69, 0.31)$

Therefore we classify the pixel as background

Complex Models

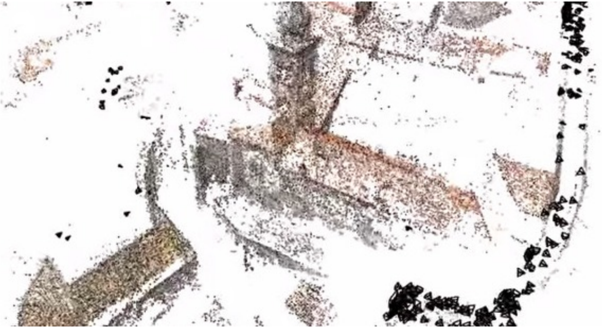


Image from https://www.youtube.com/watch?v=sQegEro5Bfo

Optimization

Learning $\boldsymbol{\theta}$ is an **optimization problem**

- We seek $\hat{\boldsymbol{\theta}}$ that maximizes some objective function $f(\boldsymbol{\theta})$
- E.g. before we sought $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \Pr(x_1, \ldots, x_n | \boldsymbol{\theta})$

There are algorithms available for finding $\boldsymbol{\theta}$ of many models

- But this is not always the case

Therefore, a brief intro on optimization ...

Optimization

We seek $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} -f(\boldsymbol{\theta})$

- Sometimes $\hat{\boldsymbol{\theta}}$ can be found analytically (like before)
- Often we have to resort to iterative methods

Iterative methods

- Start with an estimate $\boldsymbol{\theta}^{[0]}$
- Improve this estimate iteratively using local information
- Until no more improvement can be made

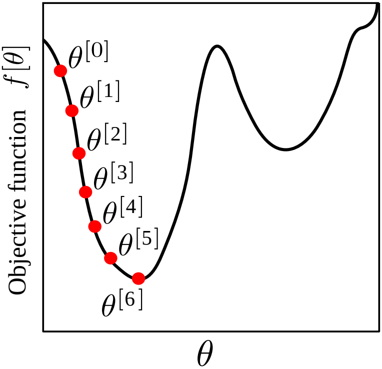


Image adapted from Prince 2012

This means we need a good initial estimate $\boldsymbol{\theta}^{[0]}$

- Otherwise we might get stuck in a **local minimum**
- Unless f is **convex**

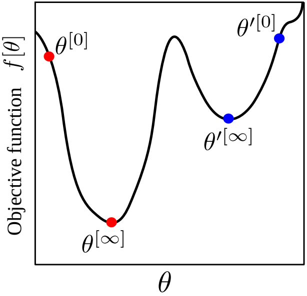


Image from Prince 2012

Iterative methods work by alternatively

- Choosing a search direction using local information
- Searching the minimum along that direction in some interval
- Updating $\boldsymbol{\theta}$ according to the found minimum

Local information from first and second order derivatives

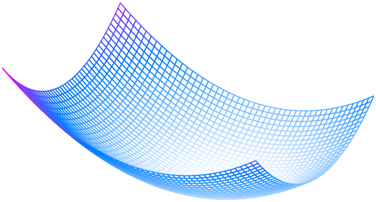
Many different optimization algorithms

- Steepest decent, Newton, Levenberg-Marquardt, BFGS ...
- Some can be tested at `https://awful.codeplex.com/`

Optimization

Iterative Methods – Simple Example

We seek the minimum of $f(\boldsymbol{\theta}) = (\theta_1 - 2)^2 + (\theta_2 - 1)^2$



Finding the minimum with Python and SciPy

```
def f(x):
  return np.power(x[0]-2,2) + np.power(x[1]-1,2)

scipy.optimize.minimize(f, [5, 5]) # returns [2, 1]
```

Finding the minimum with Matlab

```
fminunc(@(x)(x(1)-2)^2 + (x(2)-1)^2, [5, 5]) % returns [2, 1]
```

Summary

We have seen how (not) to approach CV problems

- See Prince 2012 for more information on CV models

Optimization is fundamental in vision problems

- We briefly discussed why, and how optimization works

Literature

Books for further information on these topics

- Prince 2012 (`http://computervisionmodels.com/`)
- Bishop et al. 2006
- Boyd and Vandenberghe 2009
- Wright and Nocedal 1999

Bibliography

Bishop, Christopher et al. (2006). **Pattern Recognition and Machine Learning**. Springer.

Boyd, Stephen and Lieven Vandenberghe (2009). **Convex Optimization**. Cambridge University Press.

Prince, S.J.D. (2012). **Computer Vision: Models Learning and Inference**. Cambridge University Press.

Wright, SJ and J Nocedal (1999). **Numerical optimization**. Springer.