CSC411: Optimization for Machine Learning

University of Toronto

September 20–26, 2018

1

\(^1\text{based on slides by Eleni Triantafillou, Ladislav Rampasek, Jake Snell, Kevin Swersky, Shenlong Wang and other}\)
Convexity
Definition of Convexity

A function $f$ is **convex** if for any two points $\theta_1$ and $\theta_2$ and any $t \in [0, 1],$

$$f(t\theta_1 + (1 - t)\theta_2) \leq tf(\theta_1) + (1 - t)f(\theta_2)$$

We can *compose* convex functions such that the resulting function is also convex:

- If $f$ is convex, then so is $\alpha f$ for $\alpha \geq 0$
- If $f_1$ and $f_2$ are both convex, then so is $f_1 + f_2$
- **etc.**, see http://www.ee.ucla.edu/ee236b/lectures/functions.pdf for more
Why do we care about convexity?

- Any local minimum is a global minimum.
- This makes optimization a lot easier because we don’t have to worry about getting stuck in a local minimum.
Examples of Convex Functions

Quadratics

In [6]:

```python
import matplotlib.pyplot as plt
plt.xkcd()
theta = linspace(-5, 5)
f = theta**2
plt.plot(theta, f)
```

Out[6]: [<matplotlib.lines.Line2D at 0x3ceae90>]

![Graph of a quadratic function]
Examples of Convex Functions

Negative logarithms

In [8]:

```python
import matplotlib.pyplot as plt
plt.xkcd()
theta = linspace(0.1, 5)
f = -np.log(theta)
plt.plot(theta, f)
```

Out[8]:

```
[<matplotlib.lines.Line2D at 0x3ef4a10>]
```
Convexity for logistic regression

**Cross-entropy** objective function for logistic regression is also convex!

\[
f(\theta) = - \sum_n t^{(n)} \log p(y = 1|\mathbf{x}^{(n)}, \theta) + (1 - t^{(n)}) \log p(y = 0|\mathbf{x}^{(n)}, \theta)
\]

Plot of \(-\log \sigma(\theta)\)

```python
In [15]:
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

theta = linspace(-5, 5)
f = -np.log(sigmoid(theta))
plt.plot(theta, f)
```

Out[15]: [<matplotlib.lines.Line2D at 0x4c453d0>]
More on optimization

- **Automatic Differentiation** Modern technique (used in libraries like tensorflow, pytorch, etc) to efficiently compute the gradients required for optimization. A survey of these techniques can be found here: https://arxiv.org/pdf/1502.05767.pdf

- **Convex Optimization** by Boyd & Vandenberghe Book available for free online at http://www.stanford.edu/~boyd/cvxbook/

- **Numerical Optimization** by Nocedal & Wright Electronic version available from UofT Library
Cross-Validation
Cross-Validation: Why Validate?

So far:

Learning as Optimization
Goal: Optimize model complexity (for the task) while minimizing under/overfitting

We want our model to generalize well without overfitting.
We can ensure this by validating the model.
Types of Validation

Hold-Out Validation: Split data into training and validation sets.

• Usually 30% as hold-out set.

Problems:

• Waste of dataset
• Estimation of error rate might be misleading
Types of Validation

- **Cross-Validation**: Random subsampling

  ![Diagram showing cross-validation](image)

  **Problem:**

  - More *computationally expensive* than hold-out validation.

  Figure from Bishop, C.M. (2006). *Pattern Recognition and Machine Learning*. Springer
Variants of Cross-Validation

**Leave-\(p\)-out**: Use \(p\) examples as the validation set, and the rest as training; repeat for all configurations of examples.

\[ \text{Total number of examples} \]

\[ \text{Experiment 1} \]
\[ \text{Experiment 2} \]
\[ \text{Experiment 3} \]
\[ \vdots \]
\[ \text{Experiment N} \]

Single test example

\[ \text{e.g., for } p = 1: \]

Problem:

- **Exhaustive**. We have to train and test \( \binom{N}{p} \) times, where \( N \) is the \# of training examples.
Variants of Cross-Validation

**K-fold**: Partition training data into K equally sized subsamples. For each fold, use the other K-1 subsamples as training data with the last subsample as validation.
K-fold Cross-Validation

• Think of it like leave-$p$-out but without combinatoric amounts of training/testing.

Advantages:
• All observations are used for both training and validation. Each observation is used for validation exactly once.
• **Non-exhaustive:** More tractable than leave-$p$-out
K-fold Cross-Validation

Problems:

• **Expensive** for large $N$, $K$ (since we train/test $K$ models on $N$ examples).
  – But there are some efficient hacks to save time...

• Can still **overfit** if we validate too many models!
  – **Solution**: Hold out an additional test set before doing any model selection, and check that the best model performs well on this additional set (*nested cross-validation*).  =>  Cross-Validception
Practical Tips for Using K-fold Cross-Val

Q: How many folds do we need?
A: With larger $K$, ...

• Error estimation tends to be more accurate
• But, computation time will be greater

In practice:
• Usually use $K \approx 10$
• BUT, larger dataset => choose smaller $K$