Chapter 7

Prediction

7.1 The Boston Bruins problem

In the 2010-11 National Hockey League (NHL) Finals, my beloved Boston Bruins played a best-of-seven championship series against the despised Vancouver Canucks. Boston lost the first two games 0-1 and 2-3, then won the next two games 8-1 and 4-0. At this point in the series, what is the probability that Boston will win the next game, and what is their probability of winning the championship?

As always, to answer a question like this, we need to make some assumptions. First, it is reasonable to believe that goal scoring in hockey is at least approximately a Poisson process, which means that it is equally likely for a goal to be scored at any time during a game. Second, we can assume that against a particular opponent, each team has some long-term average goals per game, denoted λ.

Given these assumptions, my strategy for answering this question is

1. Use statistics from previous games to choose a prior distribution for λ.

2. Use the score from the first four games to estimate λ for each team.

3. Use the posterior distributions of λ to compute distribution of goals for each team, the distribution of the goal differential, and the probability that each team wins the next game.

4. Compute the probability that each team wins the series.

To choose a prior distribution, I got some statistics from `http://www.nhl.com`, specifically the average goals per game for each team in the 2010-11 season. The distribution is roughly Gaussian with mean 2.8 and standard deviation 0.3.

The Gaussian distribution is continuous, but we'll approximate it with a discrete Pmf. `thinkbayes` provides `MakeGaussianPmf` to do exactly that:

```
def MakeGaussianPmf(mu, sigma, num_sigmas, n=101):
    pmf = Pmf()
    low = mu - num_sigmas*sigma
    high = mu + num_sigmas*sigma

    for x in numpy.linspace(low, high, n):
        p = scipy.stats.norm.pdf(x, mu, sigma)
        pmf.Set(x, p)
    pmf.Normalize()
    return pmf
```

`mu` and `sigma` are the mean and standard deviation of the Gaussian distribution. `num_sigmas` is the number of standard deviations above and below the mean that the Pmf will span, and `n` is the number of values in the Pmf.

Again we use `numpy.linspace` to make an array of `n` equally spaced values between `low` and `high`, including both.

`norm.pdf` evaluates the Gaussian probability density function (PDF).

Getting back to the hockey problem, here's the definition for a suite of hypotheses about the value of λ.

```
class Hockey(thinkbayes.Suite):

    def __init__(self):
        pmf = thinkbayes.MakeGaussianPmf(2.7, 0.3, 4)
        thinkbayes.Suite.__init__(self, pmf)
```

So the prior distribution is Gaussian with mean 2.7, standard deviation 0.3, and it spans 4 sigmas above and below the mean.

As always, we have to decide how to represent each hypothesis; in this case I represent the hypothesis that $\lambda = x$ with the floating-point value `x`.

7.2 Poisson processes

In mathematical statistics, a **process** is a stochastic model of a physical system ("stochastic" means that the model has some kind of randomness in it). For example, a Bernoulli process is a model of a sequence of events, called trials, in which each trial has two possible outcomes, like success and failure. So a Bernoulli process is a natural model for a series of coin flips, or a series of shots on goal.

A Poisson process is the continuous version of a Bernoulli process, where an event can occur at any point in time with equal probability. Poisson processes can be used to model customers arriving in a store, buses arriving at a bus stop, or goals scored in a hockey game.

In many real systems the probability of an event changes over time. Customers are more likely to go to a store at certain times of day, buses are supposed to arrive at fixed intervals, and goals are more or less likely at different times during a game.

But all models are based on simplifications, and in this case modeling a hockey game with a Poisson process is a reasonable choice. Heuer, Müller and Rubner (2010) analyze scoring in a German soccer league and come to the same conclusion; see `http://www.cimat.mx/Eventos/vpec10/img/poisson.pdf`.

The benefit of using this model is that we can compute the distribution of goals per game efficiently, as well as the distribution of time between goals. Specifically, if the average number of goals in a game is `lam`, the distribution of goals per game is given by the Poisson PMF:

```
def EvalPoissonPmf(k, lam):
    return (lam)**k * math.exp(-lam) / math.factorial(k)
```

And the distribution of time between goals is given by the exponential PDF:

```
def EvalExponentialPdf(x, lam):
    return lam * math.exp(-lam * x)
```

I use the variable `lam` because `lambda` is a reserved keyword in Python. Both of these functions are in `thinkbayes.py`.

7.3 The posteriors

Now we can compute the likelihood that a team with a hypothetical value of `lam` scores `k` goals in a game:

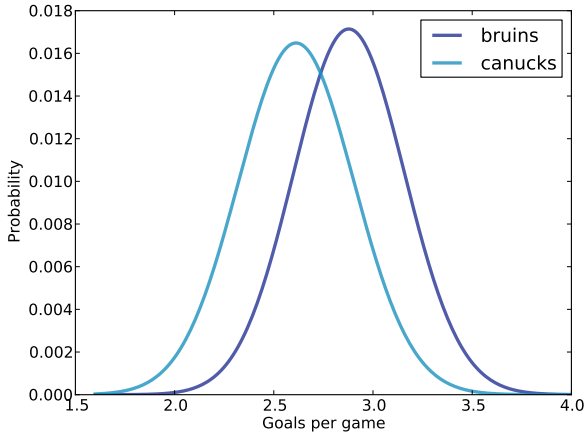

Figure 7.1: Posterior distribution of the number of goals per game.

```
# class Hockey

    def Likelihood(self, data, hypo):
        lam = hypo
        k = data
        like = thinkbayes.EvalPoissonPmf(k, lam)
        return like
```

Each hypothesis is a possible value of λ; `data` is the observed number of goals, `k`.

With the likelihood function in place, we can make a suite for each team and update them with the scores from the first four games.

```
    suite1 = Hockey('bruins')
    suite1.UpdateSet([0, 2, 8, 4])

    suite2 = Hockey('canucks')
    suite2.UpdateSet([1, 3, 1, 0])
```

Figure 7.1 shows the resulting posterior distributions for `lam`. Based on the first four games, the most likely values for `lam` are 2.6 for the Canucks and 2.9 for the Bruins.

7.4 The distribution of goals

To compute the probability that each team wins the next game, we need to compute the distribution of goals for each team.

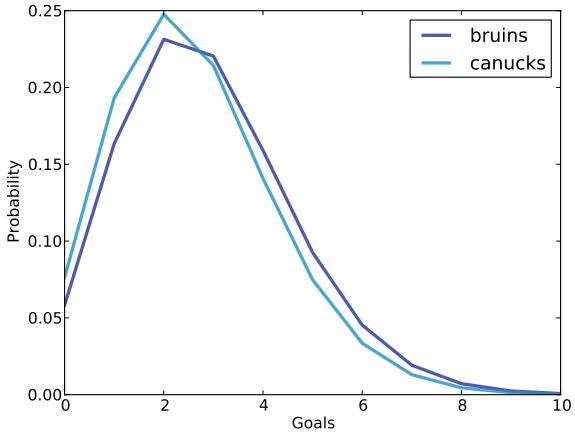

Figure 7.2: Distribution of goals in a single game.

If we knew the value of `lam` exactly, we could use the Poisson distribution again. `thinkbayes` provides a method that computes a truncated approximation of a Poisson distribution:

```
def MakePoissonPmf(lam, high):
    pmf = Pmf()
    for k in xrange(0, high+1):
        p = EvalPoissonPmf(k, lam)
        pmf.Set(k, p)
    pmf.Normalize()
    return pmf
```

The range of values in the computed Pmf is from 0 to `high`. So if the value of `lam` were exactly 3.4, we would compute:

```
lam = 3.4
goal_dist = thinkbayes.MakePoissonPmf(lam, 10)
```

I chose the upper bound, 10, because the probability of scoring more than 10 goals in a game is quite low.

That's simple enough so far; the problem is that we don't know the value of `lam` exactly. Instead, we have a distribution of possible values for `lam`.

For each value of `lam`, the distribution of goals is Poisson. So the overall distribution of goals is a mixture of these Poisson distributions, weighted according to the probabilities in the distribution of `lam`.

Given the posterior distribution of `lam`, here's the code that makes the distribution of goals:

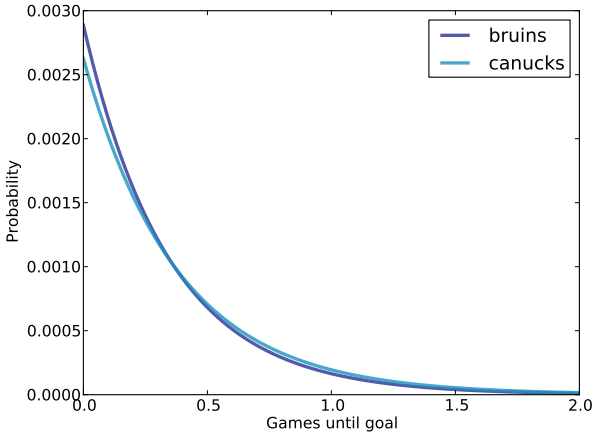

Figure 7.3: Distribution of time between goals.

```
def MakeGoalPmf(suite):
    metapmf = thinkbayes.Pmf()

    for lam, prob in suite.Items():
        pmf = thinkbayes.MakePoissonPmf(lam, 10)
        metapmf.Set(pmf, prob)

    mix = thinkbayes.MakeMixture(metapmf)
    return mix
```

For each value of `lam` we make a Poisson Pmf and add it to the meta-Pmf. I call it a meta-Pmf because it is a Pmf that contains Pmfs as its values.

Then we use `MakeMixture` to compute the mixture (we saw `MakeMixture` in Section 5.6).

Figure 7.2 shows the resulting distribution of goals for the Bruins and Canucks. The Bruins are less likely to score 3 goals or fewer in the next game, and more likely to score 4 or more.

7.5 The probability of winning

To get the probability of winning, first we compute the distribution of the goal differential:

```
goal_dist1 = MakeGoalPmf(suite1)
goal_dist2 = MakeGoalPmf(suite2)
diff = goal_dist1 - goal_dist2
```

The subtraction operator invokes `Pmf.__sub__`, which enumerates pairs of values and computes the difference. Subtracting two distributions is almost the same as adding, which we saw in Section 5.4.

If the goal differential is positive, the Bruins win; if negative, the Canucks win; if 0, it's a tie:

```
p_win = diff.ProbGreater(0)
p_loss = diff.ProbLess(0)
p_tie = diff.Prob(0)
```

With the distributions from the previous section, `p_win` is 46%, `p_loss` is 37%, and `p_tie` is 17%.

In the event of a tie at the end of "regulation play," the teams play overtime periods until one team scores. Since the game ends immediately when the first goal is scored, this overtime format is known as "sudden death."

7.6 Sudden death

To compute the probability of winning in a sudden death overtime, the important statistic is not goals per game, but time until the first goal. The assumption that goal-scoring is a Poisson process implies that the time between goals is exponentially distributed.

Given `lam`, we can compute the time between goals like this:

```
lam = 3.4
time_dist = thinkbayes.MakeExponentialPmf(lam, high=2, n=101)
```

`high` is the upper bound of the distribution. In this case I chose 2, because the probability of going more than two games without scoring is small. `n` is the number of values in the Pmf.

If we know `lam` exactly, that's all there is to it. But we don't; instead we have a posterior distribution of possible values. So as we did with the distribution of goals, we make a meta-Pmf and compute a mixture of Pmfs.

```
def MakeGoalTimePmf(suite):
    metapmf = thinkbayes.Pmf()

    for lam, prob in suite.Items():
        pmf = thinkbayes.MakeExponentialPmf(lam, high=2, n=2001)
        metapmf.Set(pmf, prob)
```

```
    mix = thinkbayes.MakeMixture(metapmf)
    return mix
```

Figure 7.3 shows the resulting distributions. For time values less than one period (one third of a game), the Bruins are more likely to score. The time until the Canucks score is more likely to be longer.

I set the number of values, `n`, fairly high in order to minimize the number of ties, since it is not possible for both teams to score simultaneously.

Now we compute the probability that the Bruins score first:

```
    time_dist1 = MakeGoalTimePmf(suite1)
    time_dist2 = MakeGoalTimePmf(suite2)
    p_overtime = thinkbayes.PmfProbLess(time_dist1, time_dist2)
```

For the Bruins, the probability of winning in overtime is 52%.

Finally, the total probability of winning is the chance of winning at the end of regulation play plus the probability of winning in overtime.

```
    p_tie = diff.Prob(0)
    p_overtime = thinkbayes.PmfProbLess(time_dist1, time_dist2)

    p_win = diff.ProbGreater(0) + p_tie * p_overtime
```

For the Bruins, the overall chance of winning the next game is 55%.

To win the series, the Bruins can either win the next two games or split the next two and win the third. Again, we can compute the total probability:

```
    # win the next two
    p_series = p_win**2

    # split the next two, win the third
    p_series += 2 * p_win * (1-p_win) * p_win
```

The Bruins chance of winning the series is 57%. And in 2011, they did.

7.7 Discussion

As always, the analysis in this chapter is based on modeling decisions, and modeling is almost always an iterative process. In general, you want to start with something simple that yields an approximate answer, identify likely sources of error, and look for opportunities for improvement.

In this example, I would consider these options:

- I chose a prior based on the average goals per game for each team. But this statistic is averaged across all opponents. Against a particular opponent, we might expect more variability. For example, if the team with the best offense plays the team with the worst defense, the expected goals per game might be several standard deviations above the mean.

- For data I used only the first four games of the championship series. If the same teams played each other during the regular season, I could use the results from those games as well. One complication is that the composition of teams changes during the season due to trades and injuries. So it might be best to give more weight to recent games.

- To take advantage of all available information, we could use results from all regular season games to estimate each team's goal scoring rate, possibly adjusted by estimating an additional factor for each pairwise match-up. This approach would be more complicated, but it is still feasible.

For the first option, we could use the results from the regular season to estimate the variability across all pairwise match-ups. Thanks to Dirk Hoag at `http://forechecker.blogspot.com/`, I was able to get the number of goals scored during regulation play (not overtime) for each game in the regular season.

Teams in different conferences only play each other one or two times in the regular season, so I focused on pairs that played each other 4–6 times. For each pair, I computed the average goals per game, which is an estimate of λ, then plotted the distribution of these estimates.

The mean of these estimates is 2.8, again, but the standard deviation is 0.85, substantially higher than what we got computing one estimate for each team.

If we run the analysis again with the higher-variance prior, the probability that the Bruins win the series is 80%, substantially higher than the result with the low-variance prior, 57%.

So it turns out that the results are sensitive to the prior, which makes sense considering how little data we have to work with. Based on the difference between the low-variance model and the high-variable model, it seems worthwhile to put some effort into getting the prior right.

The code and data for this chapter are available from `http://thinkbayes.com/hockey.py` and `http://thinkbayes.com/hockey_data.csv`. For more information see Section 0.3.

7.8 Exercises

Exercise 7.1. *If buses arrive at a bus stop every 20 minutes, and you arrive at the bus stop at a random time, your wait time until the bus arrives is uniformly distributed from 0 to 20 minutes.*

But in reality, there is variability in the time between buses. Suppose you are waiting for a bus, and you know the historical distribution of time between buses. Compute your distribution of wait times.

Hint: Suppose that the time between buses is either 5 or 10 minutes with equal probability. What is the probability that you arrive during one of the 10 minute intervals?

I solve a version of this problem in the next chapter.
Exercise 7.2. *Suppose that passengers arriving at the bus stop are well-modeled by a Poisson process with parameter λ. If you arrive at the stop and find 3 people waiting, what is your posterior distribution for the time since the last bus arrived.*

I solve a version of this problem in the next chapter.
Exercise 7.3. *Suppose that you are an ecologist sampling the insect population in a new environment. You deploy 100 traps in a test area and come back the next day to check on them. You find that 37 traps have been triggered, trapping an insect inside. Once a trap triggers, it cannot trap another insect until it has been reset.*

If you reset the traps and come back in two days, how many traps do you expect to find triggered? Compute a posterior predictive distribution for the number of traps.

Exercise 7.4. *Suppose you are the manager of an apartment building with 100 light bulbs in common areas. It is your responsibility to replace light bulbs when they break.*

On January 1, all 100 bulbs are working. When you inspect them on February 1, you find 3 light bulbs out. If you come back on April 1, how many light bulbs do you expect to find broken?

In the previous exercise, you could reasonably assume that an event is equally likely at any time. For light bulbs, the likelihood of failure depends on the age of the

bulb. Specifically, old bulbs have an increasing failure rate due to evaporation of the filament.

This problem is more open-ended than some; you will have to make modeling decisions. You might want to read about the Weibull distribution (`http://en.wikipedia.org/wiki/Weibull_distribution`*). Or you might want to look around for information about light bulb survival curves.*