3.5. Выяснение стороны и размера ставки

В этом разделе мы обсудим вопрос маркировки примеров для того, чтобы обучающийся алгоритм мог узнать сторону и размер ставки[1]. Мы заинтересованы в том, чтобы знать сторону ставки, когда у нас нет базовой модели для установки знака нашей позиции (длинная или короткая). При таких обстоятельствах мы не можем провести различие между барьером взятия прибыли и барьером остановки убытка, поскольку для этого требуется знать сторону ставки. Выяснение стороны подразумевает, что либо нет горизонтальных барьеров, либо горизонтальные барьеры должны быть симметричными.

Листинг 3.3 реализует функцию `getEvents`, которая находит время первого касания барьера. Данная функция получает следующие ниже аргументы:

- `close`: ценовой ряд библиотеки pandas.
- `tEvents`: индекс DateTimeIndex библиотеки pandas, содержащий временные штампы, которые делают посев каждого тройного барьера. Это те самые временные штампы, которые отбираются процедурами отбора, описанными в разделе 2.5 главы 2.
- `ptSl`: неотрицательное вещественное, задающее ширину двух барьеров. Значение 0 означает, что соответствующий горизонтальный барьер (взятие прибыли и/или остановка убытка) будет отключен.
- `t1`: ряд библиотеки pandas с временными штампами вертикальных барьеров. Мы передаем `False`, когда хотим отключить вертикальные барьеры.
- `trgt`: ряд библиотеки pandas с целями, выраженный в виде абсолютных финансовых возвратов.
- `minRet`: минимальный целевой финансовый возврат, необходимый для выполнения тройного барьерного поиска.
- `numThreads`: число потоков, одновременно используемых функцией.

Листинг 3.3. Получение времени первого касания

```
def getEvents(close,tEvents,ptSl,trgt,minRet,numThreads,t1=False):
    #1) получить цель
    trgt=trgt.loc[tEvents]
    trgt=trgt[trgt>minRet] # minRet
    #2) получить t1 (максимальный период владения)
    if t1 is False: t1=pd.Series(pd.NaT,index=tEvents)
    #3) сформировать объект событий events, применить стоп-лосс на t1
    side_=pd.Series(1.,index=trgt.index)
```

[1] Термин «сторона ставки» (the side of the bet) отражает то, в какую сторону пойдет движение цены — в длинную (вверх) или короткую (вниз). Этот термин имеет синоним «позиция», то есть соответственно длинная или короткая позиция. — *Примеч. науч. ред.*

```
    events=pd.concat({'t1':t1,'trgt':trgt,'side':side_}, \
        axis=1).dropna(subset=['trgt'])
    df0=mpPandasObj(func=applyPtSlOnT1,pdObj=('molecule',events.index), \
        numThreads=numThreads,close=close,events=events,ptSl=[ptSl,ptSl])
    events['t1']=df0.dropna(how='all').min(axis=1) # pd.min игнорирует
                                                   # значения nan
    events=events.drop('side',axis=1)
    return events
```

Предположим, что $I = 1E6$ и $h = 1E3$, тогда число оцениваемых условий на одном инструменте достигает миллиарда. Многие задачи машинного обучения — вычислительно дорогостоящи, если только вы не знакомы с многопоточностью, и эта задача одна из таких. Именно здесь в игру вступают параллельные вычисления. В главе 20 обсуждается несколько функций мультиобработки, которые мы будем использовать во всей книге.

Функция `mpPandasObj` вызывает механизм мультиобработки, который подробно описан в главе 20. На данный момент вам просто нужно знать, что эта функция будет выполнять `applyPtsiOnT1` параллельно. Функция `applyPtsiOnT1` возвращает временные штампы, в которых происходит касание каждого барьера (если вообще оно происходит). Тогда время первого касания — это самое раннее время среди трех, возвращенных функцией `applyPtsiOnT1`. Так как мы должны узнать сторону ставки, мы передали в качестве аргумента `ptSl = [ptSl, ptSl]` и произвольно установили сторону всегда равной длинной (горизонтальные барьеры симметричны, поэтому для определения времени первого касания сторона не имеет значения). Вывод этой функции — кадр данных библиотеки pandas со столбцами:

- `t1`: временной штамп первого касания барьера;
- `trgt`: цель, которая была использована для генерирования горизонтальных барьеров.

Листинг 3.4 показывает один из способов определения вертикального барьера. Для каждого индекса в `tEvents` он находит временной штамп следующего ценового бара в числе дней `numDays` или сразу после. Этот вертикальный барьер может быть передан в функцию `getEvents` как необязательный аргумент `t1`.

Листинг 3.4. Добавление вертикального барьера

```
t1=close.index.searchsorted(tEvents+pd.Timedelta(days=numDays))
t1=t1[t1<close.shape[0]]
t1=pd.Series(close.index[t1],index=tEvents[:t1.shape[0]]) # NaNs в конце
```

Наконец, мы можем промаркировать наблюдения с помощью функции `getBins`, определенной в листинге 3.5. Ее аргументами являются кадр данных событий `events`, который мы только что обсуждали, и ценовой ряд библиотеки pandas `close`. На выходе получается кадр данных со столбцами:

- `ret`: финансовый возврат, реализованный в момент первого касания барьера;

- `bin`: метка, {–1, 0, 1}, в качестве знаковой функции `sign` исхода. Данную функцию можно легко скорректировать, чтобы маркировать как 0 те события, когда вертикальный барьер был задет первым, что мы оставляем в качестве упражнения.

Листинг 3.5. Маркировка стороны и размера ставки

```
def getBins(events,close):
    #1) цены выровнены с событиями events
    events_=events.dropna(subset=['t1'])
    px=events_.index.union(events_['t1'].values).drop_duplicates()
    px=close.reindex(px,method='bfill')
    #2) создать объект
    out=pd.DataFrame(index=events_.index)
    out['ret']=px.loc[events_['t1'].values].values/px.loc[events_.index]-1
    out['bin']=np.sign(out['ret'])
    return out
```

3.6. Метамаркировка

Предположим, у вас есть модель для установки стороны ставки (длинной или короткой). Вам осталось узнать размер этой ставки, который подразумевает возможность отсутствия ставки вообще (нулевой размер). Практики сталкиваются с этой ситуацией регулярно. Мы часто знаем, хотим ли мы купить или продать продукт, и единственный оставшийся вопрос заключается в том, каким количеством денег мы должны рискнуть в такой ставке. Мы не хотим, чтобы алгоритм МО обучался узнавать сторону ставки просто для того, чтобы сообщать нам, какой размер является подходящим. В этой связи, вероятно, вас не удивит, если вы услышите, что ни в одной книге или публикации эта распространенная задача до сих пор не обсуждалась. К счастью, на этом страдания заканчиваются. Я называю эту задачу метамаркировкой, потому что мы хотим построить вторичную модель МО, которая учится использовать первичную экзогенную, внешне обусловленную модель.

Для того чтобы обрабатывать метамаркировку, вместо написания совершенно новой функции `getEvents` мы внесем несколько корректировок в предыдущий исходный код. Во-первых, мы принимаем новый необязательный аргумент `side` (по умолчанию `None`), который содержит сторону наших ставок, как было решено первичной моделью. Когда аргумент `side` равен `None`, данная функция понимает, что в игру вступает метамаркировка. Во-вторых, поскольку теперь мы знаем сторону, мы можем эффективно различать взятие прибыли и остановку убытка. Горизонтальные барьеры не обязательно должны быть симметричными, как в разделе 3.5. Аргумент `ptSl` — это список из двух неотрицательных вещественных значений, где `ptSl[0]` представляет собой сомножитель, который умножает `trgt` для установки ширины верхнего барьера, и `ptSl[1]` — сомножитель, который умножает `trgt` для определения ширины нижнего барьера. Если значение равно 0, то соответствующий барьер отключается. Листинг 3.6 реализует эти улучшения.

Листинг 3.6. Расширение функции getEvents для встраивания метамаркировки

```
def getEvents(close,tEvents,ptSl,trgt,minRet,numThreads,t1=False,side=None):
    #1) получить цель
    trgt=trgt.loc[tEvents]
    trgt=trgt[trgt>minRet] # minRet
    #2) получить t1 (максимальный период владения)
    if t1 is False: t1=pd.Series(pd.NaT,index=tEvents)
    #3) сформировать объект событий events, применить стоп-лосс к t1
    if side is None: side_,ptSl_=pd.Series(1.,index=trgt.
                     index),[ptSl[0],ptSl[0]]
    else: side_,ptSl_=side.loc[trgt.index],ptSl[:2]
    events=pd.concat({'t1':t1,'trgt':trgt,'side':side_}, \
        axis=1).dropna(subset=['trgt'])
    df0=mpPandasObj(func=applyPtSlOnT1,pdObj=('molecule',events.index), \
        numThreads=numThreads,close=inst['Close'],events=events,ptSl=ptSl_)
    events['t1']=df0.dropna(how='all').min(axis=1) # pd.min игнорирует
                                                   # значение nan
    if side is None: events=events.drop('side',axis=1)
    return events
```

Чтобы функция `getBins` обрабатывала метамаркировку, мы должны ее расширить схожим образом. Код из листинга 3.7 реализует необходимые изменения.

Листинг 3.7. Расширение функции getBins для встраивания метамаркировки

```
def getBins(events,close):
    '''
    Вычислить исход события (включая информацию о стороне, если
предоставлена).
    events является кадром данных, где:
    —events.index — время начала события
    —events['t1'] — время окончания события
    —events['trgt'] — цель события
    —events['side'] (необязательно) предполагает сторону позиции алгоритма
    Случай 1: ('side' не находится в events):
        интервал в (-1,1) <— метка по движению цены
    Случай 2: ('side' находится в events):
        интервал в (0,1) <— метка по прибыли и убыткам pnl (метамаркировка)
    '''
    #1) цены выровнены с событиями events
    events_=events.dropna(subset=['t1'])
    px=events_.index.union(events_['t1'].values).drop_duplicates()
    px=close.reindex(px,method='bfill')
    #2) создать объект
    out=pd.DataFrame(index=events_.index)
    out['ret']=px.loc[events_['t1'].values].values/px.loc[events_.index]-1
    if 'side' in events_: out['ret']*=events_['side']    # метамаркировка
    out['bin']=np.sign(out['ret'])
    if 'side' in events_: out.loc[out['ret']<=0,'bin']=0 # метамаркировка
    return out
```

Теперь вероятные значения разметки для `out['bin']` лежат в пределах {0,1}, в противовес предыдущим подходящим величинам {–1,0,1}. Алгоритм МО учится выбирать между решением открыть позицию и бездействием — сугубо бинарное прогнозирование. Если прогнозируемая маркировка имеет значение 1, мы можем использовать вероятность второго прогноза для определения объема средств, причем тип позиции будет заранее установлен основной моделью.

3.7. Как использовать метамаркировку

Задачи бинарной классификации представляют собой компромисс между ошибками 1-го рода (ложные утверждения) и ошибками 2-го рода (ложные отрицания). В общем случае, увеличение частоты истинных утверждений бинарного классификатора будет приводить к увеличению его частоты ложных утверждений. Кривая операционной характеристики приемника (receiver operating characteristic, ROC) бинарного классификатора измеряет стоимость увеличения частоты истинных утверждений с точки зрения принятия более высоких частот ложных утверждений.

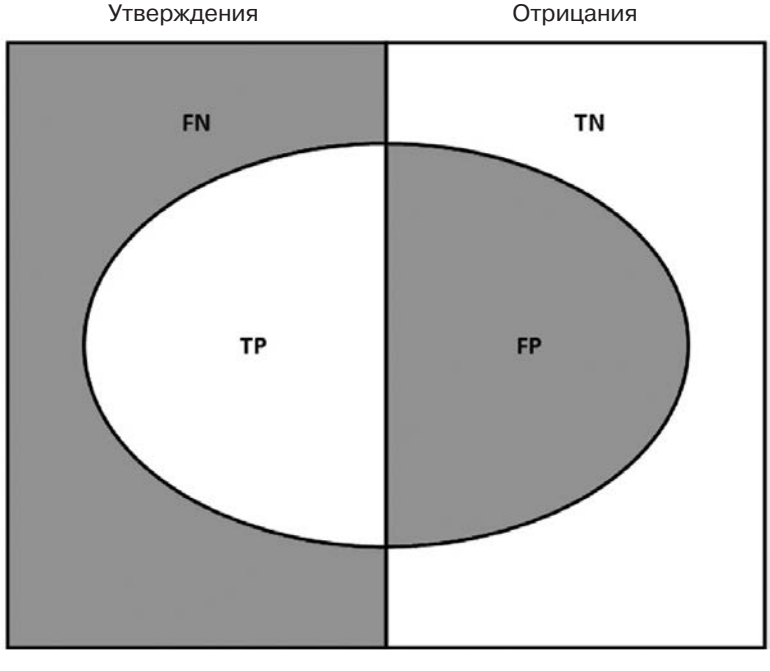

Рис. 3.2. Визуализация «матрицы ошибок». FN (false negatives) — ложные отрицания, TP (true positives) — истинные утверждения, TN (true negatives) — истинные отрицания, FP (false positives) — ложные утверждения

На рис. 3.2 показана так называемая матрица ошибок. На множестве наблюдений есть элементы, которые показывают условие (утверждения, левый прямоугольник), и элементы, которые не показывают условие (отрицания, правый прямоугольник). Бинарный классификатор предсказывает, что некоторые элементы показывают условие (эллипс), где область TP содержит истинные утверждения и область TN содержит истинные отрицания. Это приводит к двум видам ошибок: ложным утверждениям (FP) и ложным отрицаниям (FN). «Точность» — это соотношение между площадью TP и площадью эллипса. «Полнота» — это отношение между областью TP и областью в левом прямоугольнике. Данное понятие полноты (так называемой частоты истинных утверждений) находится в контексте классификационных задач, аналогично «силе» в контексте проверки статистических гипотез. «Правильность» — это сумма областей TP и TN, деленная на совокупное множество элементов (квадрат). В общем случае, уменьшение области FP происходит за счет увеличения области FN, потому что более высокая точность обычно означает меньшее число распознаваний, следовательно, более низкую полноту. Тем не менее существует некое сочетание точности и полноты, которое максимизирует суммарную эффективность классификатора. Балльная оценка F1 служит мерой эффективности классификатора в качестве гармонического среднего между точностью и полнотой (подробнее об этом в главе 14).

Метамаркировка особенно полезна, когда вы хотите достичь более высоких оценок F1. Во-первых, мы строим модель, которая достигает высокой полноты, даже если точность не особенно высока. Во-вторых, мы поправляем низкую точность, применяя метамаркировку к утвердительным исходам, предсказанным первичной моделью.

Метамаркировка увеличит вашу оценку F1, отфильтровав ложные утверждения, где большинство утвердительных исходов уже были идентифицированы первичной моделью. Иными словами, роль вторичного алгоритма МО заключается в определении того, является ли утверждение первичной (экзогенной) модели истинным или ложным. Его цель не в том, чтобы выдавать возможность делать ставки, а в том, чтобы определять, следует ли нам действовать либо следует пропустить предоставленную возможность.

Метамаркировка является очень мощным инструментом, который стоит иметь в своем арсенале по четырем дополнительным причинам. Во-первых, алгоритмы МО часто критикуются как черные ящики (см. главу 1). Метамаркировка позволяет построить систему МО поверх белого ящика (подобно фундаментальной модели, основанной на экономической теории). Эта способность трансформировать фундаментальную модель в модель МО должна сделать метамаркировку особенно полезной для «квантоментальных» фирм. Во-вторых, когда вы применяете метамаркировку, эффекты от переподгонки ограничены, потому что алгоритм будет решать не сторону вашей ставки, а только размер. В-третьих, отделяя предсказание стороны ставки от предсказания ее размера, метамаркировка позволяет создавать сложные стратегические структуры. Например, учтите, что признаки, служащие

драйвером для ралли, могут отличаться от признаков, служащих драйвером для активной распродажи. В этом случае вы, возможно, захотите разработать стратегию МО исключительно для длинных позиций, опираясь на рекомендации первичной модели по покупке, и обучающуюся стратегию исключительно для коротких позиций, опираясь на рекомендации совершенно другой первичной модели по продаже. В-четвертых, достижение высокой точности на малых ставках и низкой точности на больших ставках вас погубит. Правильное определение размера хороших возможностей так же важно, как и их выявление, поэтому имеет смысл разработать алгоритм МО, исключительно сосредоточенный на правильном получении этого критически важного решения (установление размера). Мы пересмотрим этот четвертый пункт в главе 10. По моему опыту, обучающиеся модели маркировки могут обеспечить более робастные и надежные результаты, чем стандартные модели маркировки.

3.8. Квантоментальный способ

Возможно, вам приводилось читать в прессе, что многие хеджевые фонды с восторгом принимают квантоментальный подход. Простой поиск в интернете покажет сообщения о том, что многие хеджевые фонды, в том числе некоторые из самых традиционных, инвестируют десятки миллионов долларов в технологии, предназначенные для объединения человеческого опыта с квантитативными методами. Оказывается, метамаркировка — это именно то, чего эти люди ждали. Давайте посмотрим почему.

Предположим, что у вас есть ряд, состоящий из признаков, которые, по вашему мнению, могут прогнозировать некоторые цены, вы просто не знаете как. Поскольку у вас нет модели, которая определяет сторону каждой ставки, вам нужно узнать ее сторону и размер. Вы применяете то, что вы узнали в разделе 3.5, и производите некие метки, основываясь на тройном барьерном методе с симметричными горизонтальными барьерами. Теперь вы готовы выполнить подгонку своего алгоритма на тренировочном подмножестве и оценить точность своих прогнозов на тестовом подмножестве. В качестве альтернативы можно сделать следующее:

1. Использовать свои прогнозы из первичной модели и генерировать метаметки. Помните, что горизонтальные барьеры не обязательно должны быть симметричными.
2. Снова выполнить подгонку модели на том же тренировочном подмножестве, но на этот раз с использованием только что сгенерированных метаметок.
3. Совместить «стороны» из первой обучающейся модели с «размерами» из второй обучающейся модели.

Вы всегда можете добавить метамаркировочный слой в любую первичную модель, будь то алгоритм МО, эконометрическое уравнение, техническое торговое правило, фундаментальный анализ и т. д. Сюда относятся и прогнозы, генерируемые челове-

ком исключительно на основе его интуиции. В этом случае метамаркировка поможет нам выяснить, когда мы должны исполнять или отклонять ставку дискреционного портфельного менеджера. Признаки, используемые таким обучающимся алгоритмом маркировки, могут варьироваться от рыночной информации до биометрической статистики и психологических оценок. Например, алгоритм МО маркировки может обнаружить, что дискреционные портфельные менеджеры склонны делать особенно хорошие ставки, когда имеется структурный сдвиг (глава 17), поскольку они могут быстрее схватывать изменение рыночного режима. И наоборот, он может обнаружить, что дискреционные портфельные менеджеры, находящиеся в состоянии стресса, что проявляется при уменьшении числа часов для сна, во время усталости, изменении в весе и т. д., склонны делать неточные предсказания[1]. Многие профессии требуют регулярных психологических экзаменов, и алгоритм МО маркировки может обнаруживать, что эти балльные оценки также релевантны для того, чтобы оценивать нашу текущую степень уверенности в предсказаниях портфельного менеджера. Вполне возможно, ни один из этих факторов на дискреционных портфельных менеджеров не влияет, и их мозги работают независимо от эмоционального состояния, как холодные вычислительные машины. Смею предположить, что это не так, и поэтому метамаркировка должна стать существенным методом МО для каждого дискреционного хеджевого фонда. В ближайшем будущем каждый дискреционный хеджевый фонд станет квантоментальной фирмой, и метамаркировка предлагает им четкий путь для этого перехода.

3.9. Исключение ненужных меток

Некоторые классификаторы МО плохо работают, когда классы слишком несбалансированны. В этих условиях предпочтительнее отказаться от крайне редких меток и сосредоточиться на более распространенных исходах. Листинг 3.8 представляет процедуру, которая рекурсивно отбрасывает наблюдения, связанные с чрезвычайно редкими метками. Функция `dropLabels` рекурсивно устраняет те наблюдения, связанные с классами, которые появляются меньше, чем доля `minPct` случаев, если только не осталось всего два класса.

Листинг 3.8. Устранение малозаселенных меток

```
def dropLabels(events,minPtc=.05):
    # применить веса, устранить метки с недостаточным числом примеров
    while True:
        df0=events['bin'].value_counts(normalize=True)
        if df0.min()>minPct or df0.shape[0]<3: break
        print 'dropped label',df0.argmin(),df0.min()
        events=events[events['bin']!=df0.argmin()]
    return events
```

[1] Вы, вероятно, знаете по крайней мере один крупный хедж-фонд, который круглосуточно отслеживает эмоциональное состояние своих аналитиков.

Кстати, еще одна причина, по которой вы, возможно, захотите удалять ненужные метки, — это известный дефект библиотеки sklearn: https://github.com/scikit-learn/scikit-learn/issues/8566. Такого рода дефекты являются следствием очень фундаментальных допущений, принятых в реализации библиотеки sklearn, и их устранение далеко не тривиально. В данном конкретном случае ошибка связана с решением разработчиков библиотеки sklearn работать со стандартными массивами библиотеки numpy, а не со структурированными массивами или объектами библиотеки pandas. Маловероятно, что к моменту прочтения вами этой главы или в ближайшем будущем будет найдено его исправление. В последующих главах мы рассмотрим способы обхода подобных ошибок реализации путем создания собственных классов и расширения функциональности библиотеки sklearn.

Упражнения

3.1. Сформируйте долларовые бары для фьючерсного контракта E-mini S&P 500:

- **(а)** Примените симметричный фильтр CUSUM (глава 2, раздел 2.5.2.1), где порогом является среднеквадратическое отклонение суточных финансовых возвратов (листинг 3.1).
- **(б)** Используйте листинг 3.4 на ряде `t1` библиотеки pandas, где `numDays` = `1`.
- **(в)** На этих отобранных признаках примените тройной барьерный метод, где `ptSl=[1,1]`, а `t1` — это ряд, созданный вами в пункте 3.1.б.
- **(г)** Примените функцию `getBins` для генерирования меток.

3.2. Из упражнения 3.1 используйте листинг 3.8 для устранения редких меток.

3.3. Скорректируйте функцию `getBins` (листинг 3.5) так, чтобы она возвращала 0 всякий раз, когда вертикальный барьер затрагивается первым.

3.4. Разработайте стратегию следования за трендом, опираясь на популярный статистический показатель технического анализа (к примеру, пересечение скользящих средних). Для каждого наблюдения модель предлагает сторону, но не размер ставки.

- **(а)** Получите метаметки для `ptSl=[1,2]` и `t1`, где `numDays=1`. В качестве `trgt` используйте суточное среднеквадратическое отклонение, как вычислено в листинге 3.1.
- **(б)** Натренируйте случайный лес принимать решение, торговать или нет. Важно: решением является торговать или нет, {0, 1}, поскольку лежащая в основе модель (пересекающиеся скользящие средние) приняла решение о стороне ставки {–1, 1}.

3.5. Разработайте стратегию взврата к среднему значению на основе полос Боллинджера. Для каждого наблюдения модель предлагает сторону, но не размер ставки.

(а) Получите метаметки для `ptSl` = `[0,2]` и `t1`, где `numDays` = `1`. В качестве `trgt` используйте суточное среднеквадратическое отклонение, как вычислено в листинге 3.1.

(б) Натренируйте случайный лес принимать решение, торговать или нет. В качестве признаков используйте: волатильность, внутрирядовую корреляцию и пересекающиеся скользящие средние из упражнения 3.2.

(в) Какова точность предсказаний от первичной модели (то есть если вторичная модель не фильтрует ставки)? Каковы точность, полнота и оценка F1?

(г) Какова точность предсказаний от вторичной модели? Каковы точность, полнота и балльная оценка F1?