Lecture 21

Utility/Risk and Model Comparison

Previously

- glms, contd
- oceanic tools example and centering
- model comparison
- oceanic tools and other models model comparison
- poisson over-dispersion and hierarchical modeling
- prosocial chimps in lab

Today

- Decision theory
- point estimates for decisions
- classification risk
- probabilistic estimates for estimation
- which model to use
- probabilistic estimates for model comparison

Decision Theory

Predictions (or actions based on predictions) are described by a utility or loss function, whose values can be computed given the observed data.

Indeed one can consider prediction itself as actions to be undertaken with respect to a particular utility.

Components of Decision problem

1. $a \in A$, available actions
2. $\omega \in \Omega$, a state in the set of states of the world.
3. $p(\omega|D)$ which tells us out current beliefs about the world.
4. A utility function $u(a, \omega) : A \times \Omega \rightarrow R$ that awards a score/ utility/profit to each action a when the state of the universe is ω. This can be also formulated as a risk/loss.

2 Bayes distributions: posterior and posterior predictive

1. a parameter value or prediction, or action based on prediction
2. If $\Omega = \{y^*\}$, then $\omega = y^*$, a future y. If Ω is the posterior, then $\omega = \theta$ is a value of a parameter(s)
3. This is either the posterior distribution (for θ) or the posterior predictive distribution (for y^*)
4. A utility for an action based on θ, eg point prediction of median, or on y^*, same idea.

Process

First define the distribution-averaged utility:

$$\bar{u}(a) = \int d\omega \, u(a, \omega) \, p(\omega|D)$$

We then find the a that maximizes this utility:

$$\hat{a} = \arg\max_a \bar{u}(a)$$

This action is called the **bayes action**.

The resulting maximized expected utility is given by:

$$\bar{u}(\hat{a}, p) = \bar{u}(\hat{a}) = \int d\omega \, u(\hat{a}, \omega) \, p(\omega|D),$$

sometimes referred to as the entropy function, and an associate **divergence** can be defined:

$$d(a, p) = \bar{u}(p, p) - \bar{u}(a, p)$$

Then one can think of minimizing $d(a, p)$ with respect to a to get $\hat{a}$, so that this discrepancy can be thought of as a loss function.

Example: Bayes action for posterior predictive

$$\bar{u}(a) = \int dy^* \, u(a, y^*) \, p(y^* | D, M) \text{ OR}$$

$$\bar{u}(a(x)) = \int dy^* \, u(a(x), y^*) \, p(y^* | x^*, D, M) \text{ (supervised)}$$

$$\bar{u}(\hat{a}(x^*)) = \int dy^* \, u(\hat{a}, y^*) \, p(y^* | x^*, D, M)$$

$$\hat{a}(x^*) = \arg\max_a \bar{u}(a(x^*))$$

Point Predictions: squared loss

Sometimes we want to make point predictions. In this case a is a single number.

squared error loss/utility: $l(a, y^*) = (a - y^*)^2$

The optimal point prediction that minimizes the expected loss (negative expected utility):

$$\bar{l}\,(a) = \int dy^*\,(a - y^*)^2\,p(y^*|D, M),$$

is the posterior predictive mean:

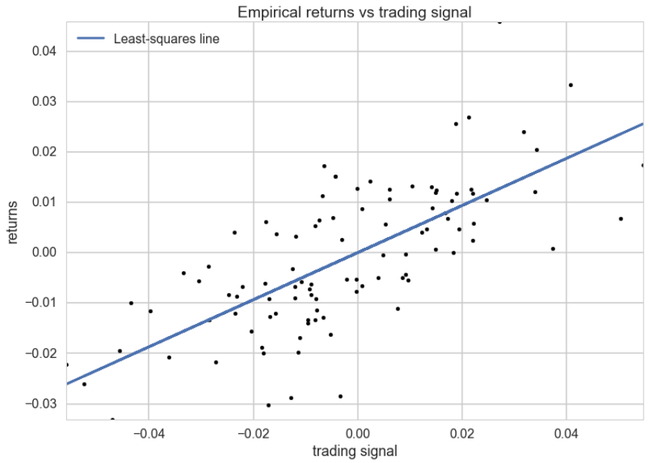
$$\hat{a} = E_p[y^*].$$

The expected loss then becomes:

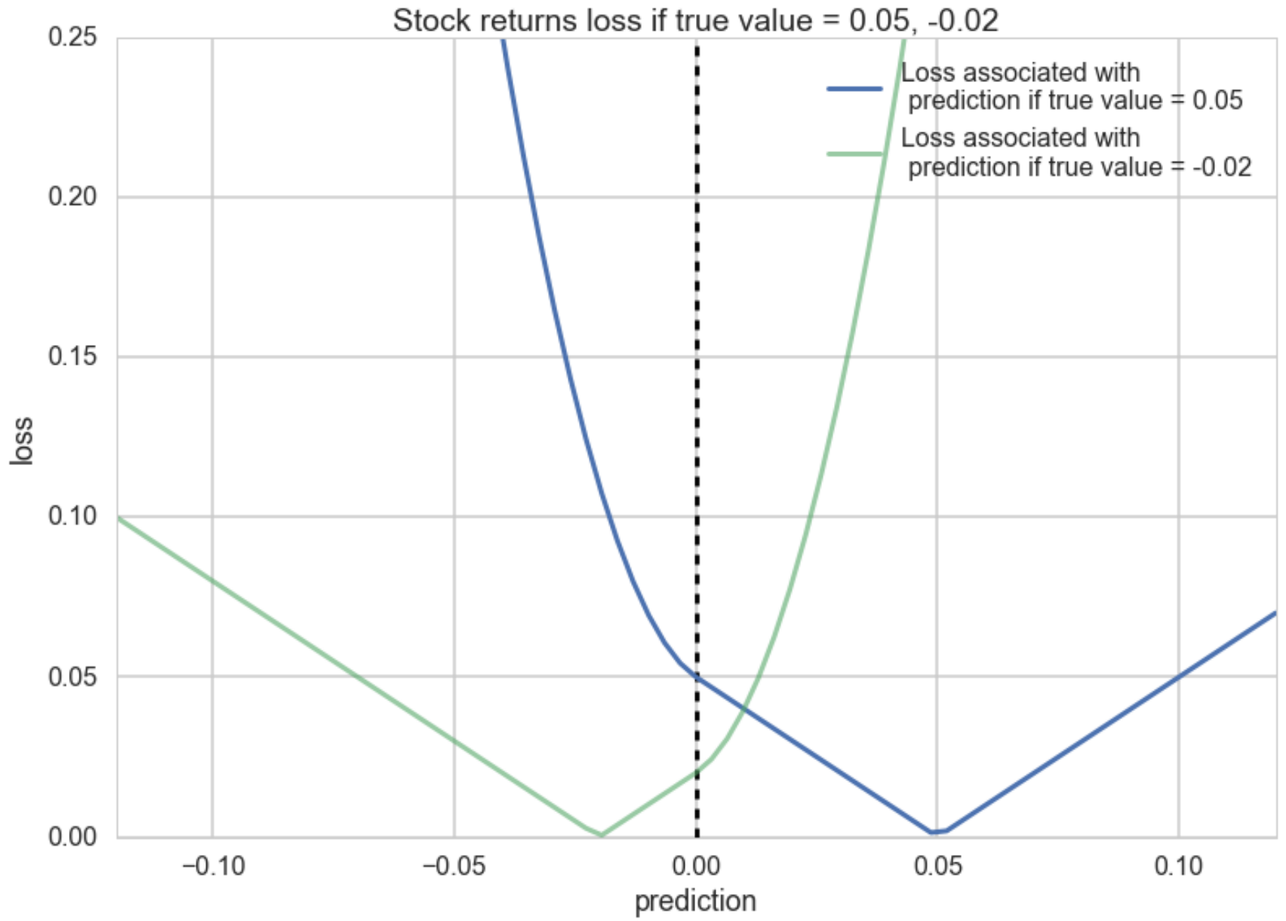
$$\bar{l}(\hat{a}) = \int dy^* \, (\hat{a} - y^*)^2 \, p(y^*|D, M) = \int dy^* \, (E_p[y^*] - y^*)^2 \, p(y^*|D, M) = Var_p[y^*]$$

Squared loss $\implies$ we dont care about skewness or kurtosis

Custom Loss: Stock Market Returns

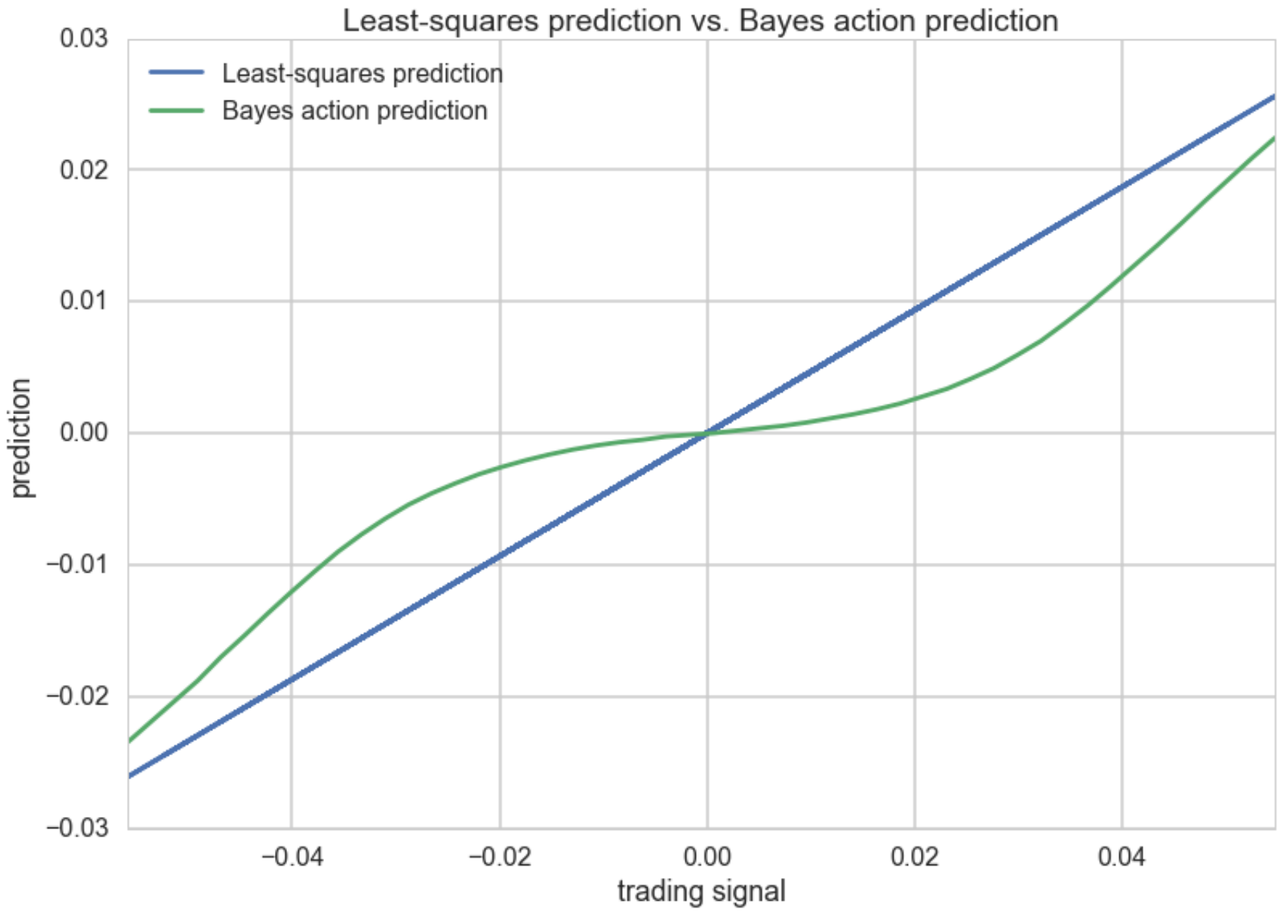



```
def stock_loss(stock_return, pred, alpha = 100.):
    if stock_return * pred < 0:
        #opposite signs, not good
        return alpha*pred**2 - np.sign(stock_return)*pred \
                        + abs(stock_return)
    else:
        return abs(stock_return - pred)
```



Loss at very x

```
with pm.Model() as model:
    std = pm.Uniform("std", 0, 100)

    beta = pm.Normal("beta", mu=0, sd=100)
    alpha = pm.Normal("alpha", mu=0, sd=100)

    mean = pm.Deterministic("mean", alpha + beta*X)

    obs = pm.Normal("obs", mu=mean, sd=std, observed=Y)

    trace = pm.sample(100000, step=pm.Metropolis())
    burned_trace = trace[20000:]
    ...
noise = std_samples*np.random.randn(N)

#posterior predictive samples at every x
possible_outcomes = lambda signal: alpha_samples + beta_samples*signal + noise


opt_predictions = np.zeros(50)
trading_signals =  np.linspace(X.min(), X.max(), 50)
for i, _signal in enumerate(trading_signals):
        _possible_outcomes = possible_outcomes(_signal)
        #expected loss over posterior predictive
        tomin = lambda pred: stock_loss(_possible_outcomes, pred).mean()
        #bayes action minimizes expected loss
        opt_predictions[i] = fmin(tomin, 0, disp = False)
```



The two risks

There are *two risks in learning* that we must consider, one to *estimate probabilities*, which we call **estimation risk**, and one to *make decisions*, which we call **decision risk**.

The **decision loss** $l(y, a)$ or **utility** $u(l, a)$ (profit, or benefit) in making a decision a when the predicted variable has value y. For example, we must provide all of the losses l(no-cancer, biopsy), l (cancer, biopsy), $l\$4(no - cancer, no - biopsy), and l\$\$$(cancer, no-biopsy). One set of choices for these losses may be 20, 0, 0, 200

Classification Risk

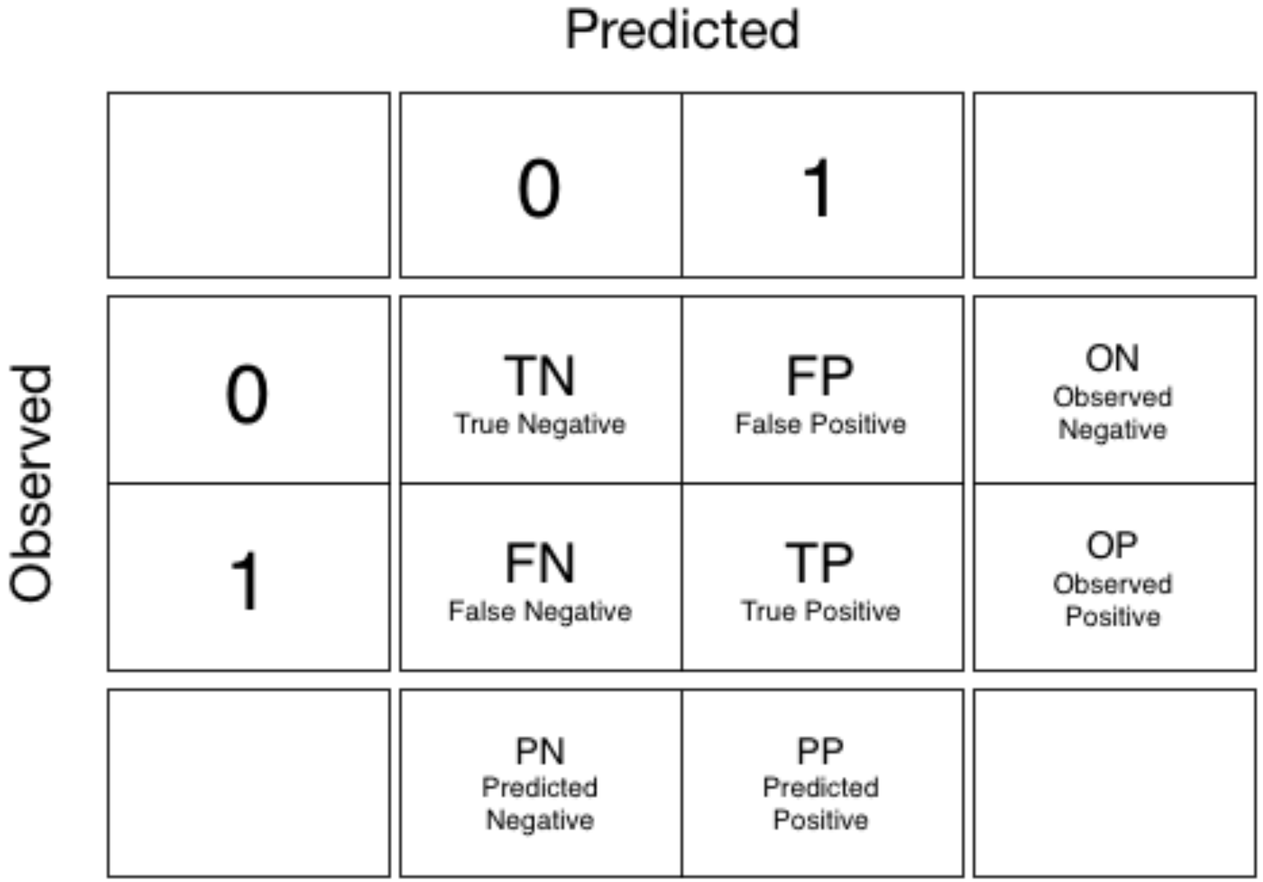
$$R_a(x) = \sum_y l(y, a(x))p(y|x)$$

That is, we calculate the **predictive averaged risk** over all choices y, of making choice a for a given data point.

Overall risk, given all the data points in our set:

$$R(a) = \int dx p(x) R_a(x)$$

Two class Classification



$$R_a(x) = l(1,g)p(1|x) + l(0,g)p(0|x).$$

Then for the "decision" $a = 1$ we have:

$$R_1(x) = l(1,1)p(1|x) + l(0,1)p(0|x),$$

and for the "decision" $a = 0$ we have:

$$R_0(x) = l(1,0)p(1|x) + l(0,0)p(0|x).$$

Now, we'd choose 1 for the data point at x if:

$$R_1(x) < R_0(x).$$

$$P(1|x)(l(1,1) - l(1,0)) < p(0|x)(l(0,0) - l(0,1))$$

So, to choose '1', the Bayes risk can be obtained by setting:

$$p(1|x) > rP(0|x) \implies r = \frac{l(0,1) - l(0,0)}{l(1,0) - l(1,1)}$$

$$P(1|x) > t = \frac{r}{1+r}.$$

One can use the prediction cost matrix corresponding to the consufion matrix

$$r == \frac{c_{FP} - c_{TN}}{c_{FN} - c_{TP}}$$

If you assume that True positives and True negatives have no cost, and the cost of a false positive is equal to that of a false positive, then $r = 1$ and the threshold is the usual intuitive $t = 0.5$.

		Predicted	
		0	1
Observed	0	TNC True Negative Cost	FPC False Positive Cost
	1	FNC False Negative Cost	TPC True Positive Cost

Log score: probabilistic prediction (estimation risk)

Here we want to find a distribution a.

The utility is defined as:

$$u(a, y^*) = log\, a(y^*),$$

The expected utility then is

$$\bar{u}(a) = \int dy^*\, log(a(y^*))\, p(y^*|D, M).$$

The a that maximizes this utility is the posterior-predictive itself!

$$\hat{a}(y^*) = p(y^*|D, M)$$

The maximized utility then is:

$$\bar{u}(a) = \int dy^* \, log(p(y^*|D, M)) \, p(y^*|D, M).$$

This is just the negative entropy of the posterior predictive distribution, and the associated divergence is our old friend the KL-divergence.

- we used this log score in model comparison
- its positive and local
- there we started from the K-L divergence but can generalize
- decision theory uses utility to make both point and distributional predictions

Risk from the posterior: point estimates

ω, unknown state of the world is some θ parameter $\in \Theta$.

Utility function is of the form $u(a, \theta)$ and our belief about the unknown state of the world: posterior distribution $p(\theta|D, M)$.

The optimal prediction is : $\bar{u}(a) = \int d\theta u(a, \theta) p(\theta|D, M)$

$$\hat{a} = \arg\max_a \bar{u}(a)$$

and then the optimal utility is $\bar{u}(\hat{a}) = \int d\theta u(\hat{a}, \theta) p(\theta|D, M)$

Indeed, bayesian decision theory is often formulated with respect to the posterior to summarize posterior with point estimates.

Identify the θ space utility as an average over the sampling distribution: $u(a, \theta) = \int u(a, y^*) p(y^*|\theta, M) dy^*$

the two approaches are equivalent and we have merely changed the order of integration.

Which Model?

- In model comparison scenario we might use the "true" distribution:

$$\bar{u}_t(\hat{a}) = \int dy^* u(\hat{a}, y^*) p_t(y^*)$$

Notice that we use $u(\hat{a}, y^*)$. The $\hat{a}$ has already been found by optimizing over our posterior predictive.

True-belief distribution

- model M_{tb} that has undergone posterior predictive checks and is very expressive, a model we can use as a reference model.
- often non-parametric or found via bayesian model averaging.
- if the true generating process is outside the hypothesis set of the models you are using, true belied never = true. This is called misfit or bias. Problem in cross-validation
- overfitting: too expressive model, use strong regularizing priors.

Bayesian Model Averaging

$$p_{BMA}(y^*|x^*, D) = \sum_k p(y^*|x^*, D, M_k)p(M_k|D)$$

where the averaging is with repect to weights $w_k = p(M_k|D)$, the posterior probabilities of the models M_k.

Last time we calculated these "Akaike" weights from the WAIC.

Model comparison

The key idea in model comparison is that we will sort our average utilities in some order. The exact values are not important, and may be computed with respect to some true distribution or true-belief distribution M_{tb}.

Utility is maximized with respect to some model M_k whereas the average of the utility is computed with respect to either the true, or true belief distribution.

$$\bar{u}(M_k, \hat{a}_k) = \int dy^* u(\hat{a}_k, y^*) p(y^* | D, M_{tb})$$

where a_k is the optimal prediction under the model M_k. Now we compare the actions, that is, we want:

$$\hat{M} = \arg\max_k \bar{u}(M_k, \hat{a}_k)$$

No calibration, but calculating the standard error of the difference can be used to see if the difference is significant, as we did with the WAIC score

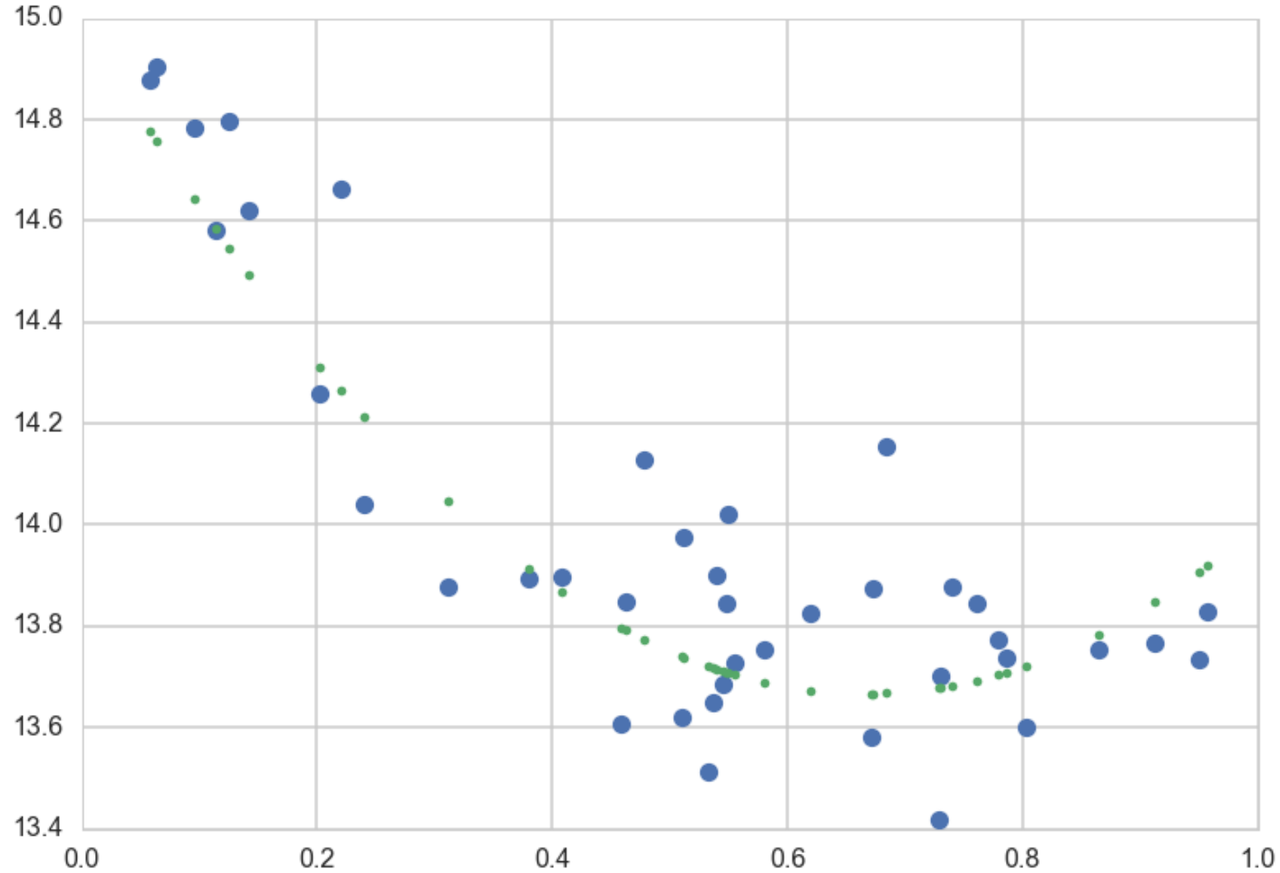
We now maximize this over M_k . This is equivalent to minimizing the KL-divergence as it is the negative KL divergence upto a M_k independent constant.

For the squared loss the first step gives us $\hat{a}_k = E_{p(y^*|D,M_k)}[y^*]$.
Then:

$$\bar{l}(\hat{a_k}) = \int dy^* \, (\hat{a}_k - y^*)^2 \, p(y^*|D, M_{tb})$$

$$= \int dy^* \, (E_{p_k}[y^*] - y^*)^2 \, p(y^*|D, M_{tb}) = Var_{p_{tb}}[y^*] + (E_{p_{tb}}[y^*] - E_{p_k}[y^*])^2$$

Dataset



- 20 data points
- keep train and test, we'll see these later
- center x for speed of sampling
- fit multiple order polynomials
- The `logp` method of a node gives us the conditional (log) probability of the node given its parents.
- Conditional probability of the observed node, which is also the model node, ...

model.logp({paramdict})

gives us likelihood of data $p(y|etc)$

```
ridge=3
sigma=0.2
pf=PolynomialFeatures(msize, include_bias=False).fit_transform(xtrain.reshape(-1,1))
print(pf.shape)
with pm.Model() as m:
    alpha = pm.Normal('alpha', 0, 100)
    beta = pm.Normal('beta', mu=0, sd=ridge, shape=msize)
    mu = alpha + pm.math.dot(pf, beta)
    o = pm.Normal('o', mu, sigma, observed=ytrain)
    trace=pm.sample(5000, init='MAP')
```

```
meanpoint={'alpha': trace['alpha'].mean(), 'beta': trace['beta'].mean(0)}
{'alpha': 13.840500632702547, 'beta': array([-0.99418043,  2.31483997])}
m.logp(meanpoint), mtest.logp(meanpoint)
(array(-1.699085634529018), array(-5.251454815218393))
```

Information criteria

- simulate an ensemble of polynomials
- use information criteria to decide between them
- these come from the deviance

$$D_{KL}(p,q) = E_p[log(p) - log(q)] = E_p[log(p/q)] = \sum_i p_i log(\frac{p_i}{q_i}) \; or \; \int dP log(\frac{p}{q})$$

Use **law or large numbers** to replace the true distribution by

its empirical estimate, then we have:

$$D_{KL}(p,q) = E_p[log(p/q)] = \frac{1}{N}\sum_i (log(p_i) - log(q_i)$$

Thus minimizing the KL-divergence involves maximizing $\sum_i log(q_i)$,

justifies the maximum likelihood principle.

$$D_{KL}(p,q) - D_{KL}(p,r) = E_p[log(r) - log(q)] = E_p[log(\frac{r}{q})]$$

Deviance

$$D(q) = -2 \sum_i log(q_i),$$

then

$$D_{KL}(p, q) - D_{KL}(p, r) = \frac{2}{N}(D(q) - D(r))$$

More generally: $D(q) = -\frac{N}{2} E_p[log(q)]$

Bayesian deviance

$D(q) = -\frac{N}{2} E_p[log(pp(y))]$ posterior predictive for points y on the test set or future data

replace joint posterior predictive over new points y by product of marginals:

ELPD: $\sum_i E_p[log(pp(y_i))]$

Since we do not know the true distribution p,

replace elpd: $\sum_i E_p[log(pp(y_i))]$

by the computed "log pointwise predictive density" (lppd) **in-sample**

$$\sum_j log\left\langle p(y_j|\theta)\right\rangle = \sum_j log\left(\frac{1}{S}\sum_s p(y_j|\theta_s)\right)$$

WAIC

$$WAIC = lppd + 2p_W$$

where

$$p_W = 2\sum_i (log(E_{post}[p(y_i|\theta)] - E_{post}[log(p(y_i|\theta))])$$

Once again this can be estimated by

$$\sum_i Var_{post}[log(p(y_i|\theta))]$$

```
For Model with 1 slope, waic is WAIC_r(WAIC=18.086154924759676, WAIC_se=10.162063571524614, p_WAIC=4.6685686490640128)
For Model with 2 slope, waic is WAIC_r(WAIC=-7.9746057666052668, WAIC_se=4.8913389425902176, p_WAIC=2.8290346198804275)
For Model with 4 slope, waic is WAIC_r(WAIC=-5.1476015964310298, WAIC_se=4.5248859948089732, p_WAIC=5.0163702157862842)
For Model with 10 slope, waic is WAIC_r(WAIC=91.825697802482608, WAIC_se=34.336737894534828, p_WAIC=53.519631187786722)
For Model with 19 slope, waic is WAIC_r(WAIC=61.178349788662352, WAIC_se=25.002444963302214, p_WAIC=38.140875766495071)
```

it is tempting to use information criteria to compare models with different likelihood functions. Is a Gaussian or binomial better? Can't we just let WAIC sort it out?
Unfortunately, WAIC (or any other information criterion) cannot sort it out. The problem is that deviance is part normalizing constant. The constant affects the absolute magnitude of the deviance, but it doesn't affect fit to data.

— *McElreath*

cross-validation

- estimate the out-of-sample risk as an average, thus gaining robustness to odd validation sets
- providing some measure of uncertainty on the out-of-sample performance.
- less data to fit so biased models
- we are not talking here about cross-validation to do hyperparameter optimization

hyperparameter fitting

- part of the prior specification, uses entire data set
- or we can use empirical bayes, and use entire data set.
- faster than cross-val but prone to model mis-specification
- but EB is not a model selection procedure

LOOCV

- The idea here is that you fit a model on N-1 data points, and use the Nth point as a validation point. Clearly this can be done in N ways.
- the N-point and N-1 point posteriors are likely to be quite similar, and one can sample one from the other by using importance sampling.

$$E_f[h] = \frac{\sum_s w_s h_s}{\sum_s w_s} \text{ where } w_s = f_s/g_s.$$

Fit the full posterior once. Then we have

$$w_s = \frac{p(\theta_s|y_{-i})}{p(\theta_s|y)} \propto \frac{1}{p(y_i|\theta_s, y_{-i})}$$

- the importance sampling weights can be unstable out in the tails.
- importance weights have a long right tail, pymc (`pm.loo`) fits a generalized pareto to the tail (largest 20% importance ratios) for each held out data point i (a MLE fit). This smooths out any large variations.

$$elpd_{loo} = \sum_i log(p(y_i|y_{-i}))$$

$$= \sum_i log\left(\frac{\sum_s w_{is} p(y_i|\theta_s)}{\sum_s w_{is}}\right)$$

over the training sample.

```
For Model with 1 slope, loo is LOO_r(LOO=19.345559544708721, LOO_se=10.448387931742174, p_LOO=5.2982709590385335)
For Model with 2 slope, loo is LOO_r(LOO=-6.4907292356212007, LOO_se=5.6639101289730434, p_LOO=3.5709728853724583)
For Model with 4 slope, loo is LOO_r(LOO=-7.8411331189326319, LOO_se=4.9343760037143145, p_LOO=3.6696044545354827)
For Model with 10 slope, loo is LOO_r(LOO=-1.3783161338355767, LOO_se=6.6512996064759902, p_LOO=6.917624219627621)
For Model with 19 slope, loo is LOO_r(LOO=1.1788924202325761, LOO_se=7.3342281925207269, p_LOO=8.1411470822801828)
```

What should you use?

1. LOOCV and WAIC are fine. The former can be used for models not having the same likelihood, the latter can be used with models having the same likelihood.

2. WAIC is fast and computationally less intensive, so for same-likelihood models (especially nested models where you are really performing feature selection), it is the first line of attack

3. One does not always have to do model selection. Sometimes just do posterior predictive checks to see how the predictions are, and you might deem it fine.

4. For hierarchical models, WAIC is best for predictive performance within an existing cluster or group. Cross validation is best for new observations from new groups