Axioms of prob
P(A) = 1, P(~A) = 0
sum(P(A)) = 1
P(AUB) = P(A) + P(B) – P(A^B)

P(A^B) = P(A|B)P(B)

Disjunction: Union
Conjunction: Intersection

$$\mathbf{P}(\mathbf{Y}) = \sum_{\mathbf{z} \in \mathbf{Z}} \mathbf{P}(\mathbf{Y}, \mathbf{z})$$

full joint dist
O(m^n) m = num of values
n = num of RVs

$$P(h_i \mid \mathbf{d}) = \alpha P(\mathbf{d} \mid h_i) P(h_i)$$

Bayesian Learning
The key quantities in the Bayesian approach are the Hypothesis prior, P (hi), and the likelihood of the data under each Hypothesis, P (d | h i).

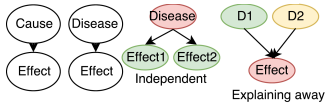
$$P(\mathbf{d} \mid h_i) = \prod_j P(d_j \mid h_i)$$

Example:

$$P(b \mid a) = \frac{P(a \mid b) P(b)}{P(a)}$$

$$\mathbf{P}(Y \mid X, \mathbf{e}) = \frac{\mathbf{P}(X \mid Y, \mathbf{e}) \mathbf{P}(Y \mid \mathbf{e})}{\mathbf{P}(X \mid \mathbf{e})}$$

$$P(cause \mid effect) = \frac{P(effect \mid cause) P(cause)}{P(effect)}$$



Independent

Explaining away
When effect is given

given A->C and B->C
P(A|B,C) = P(A|C)
P(B|A,C) = P(B|C)
P(A,B|C) = P(A|C) x P(B|C)
proof of marginalization
1. use propositional logic.
A = [A and B] U [A and ~B]
2. Apply prob on both sides
3. apply inclu-exclu principle
proof of product rule
Supose B has occurred, then P(A) ~= P(A)
Since A->B. Hence its P(A,B). But we dont
Know if B occurred, so divide by P(B)
P(A|B)=P(A,B)/P(B)

$$\text{beta}[a, b](\theta) = \alpha\, \theta^{a-1}(1-\theta)^{b-1}$$

Entropy: $H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k)$

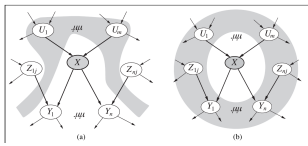
$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p+n} B(\frac{p_k}{p_k + n_k})$$

$$Gain(A) = B(\frac{p}{p+n}) - Remainder(A)$$

Example:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} \theta(x_i \mid parents(X_i))$$

$$\begin{aligned} P(j, m, a, \neg b, \neg e) &= P(j \mid a) P(m \mid a) P(a \mid \neg b \wedge \neg e) P(\neg b) P(\neg e) \\ &= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.000628 \end{aligned}$$



Linear Gaussian for X and Y
Which are continuous

$$P(y \mid x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-(\theta_1 x + \theta_2))^2}{2\sigma^2}}$$

Minimizing the sum squared error in linear regression
Gives the maximum likelihood straight line model,
Given the data are generated with gaussian noise
Of fixed variance. This is proved by above formula
Where X and Y are continuous.

Noisy-OR
First, it assumes that all the possible causes are listed. Second, it assumes that inhibition of each parent is independent of inhibition of any other parents. For example, whatever inhibits Malaria from causing a fever is independent of whatever inhibits Flu from causing a fever. Given these assumptions, Fever is false if and only if all its true parents are inhibited, and the probability of this is the product of the inhibition probabilities q for each parent

$$q_{\text{cold}} = P(\neg fever \mid cold, \neg flu, \neg malaria) = 0.6\,,$$
$$q_{\text{flu}} = P(\neg fever \mid \neg cold, flu, \neg malaria) = 0.2\,,$$
$$q_{\text{malaria}} = P(\neg fever \mid \neg cold, \neg flu, malaria) = 0.1$$

$$P(x_i \mid parents(X_i)) = 1 - \prod_{\{j: X_j = true\}} q_j$$

Every linear regression problem with L2 loss function
Is Convex. There are no local minima.

Gradient Descent
$\mathbf{w} \leftarrow$ any point in the parameter space
loop until convergence **do**
for each w_i **in w do**

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$$

Cold	*Flu*	*Malaria*	$P(Fever)$	$P(\neg Fever)$
F	F	F	0.0	1.0
F	F	T	0.9	**0.1**
F	T	F	0.8	**0.2**
F	T	T	0.98	$0.02 = 0.2 \times 0.1$
T	F	F	0.4	**0.6**

Add dummy variable
In linear regression to
Allow an intercept term
W0. Then the form of
Linear regression changes
To:

$$h_{sw}(\mathbf{x}_j) = \mathbf{w} \cdot \mathbf{x}_j = \mathbf{w}^\top \mathbf{x}_j = \sum_i w_i x_{j,i}$$

For multi-variate case, each
Weight vector will be updated
As follows: (where xji =
Feature i of training example j

$$w_i \leftarrow w_i + \alpha \sum_j x_{j,i} (y_j - h_{\mathbf{w}}(\mathbf{x}_j))$$

Analytical without regularization

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

i.i.d $\mathbf{P}(E_j | E_{j-1}, E_{j-2}, \ldots) = \mathbf{P}(E_j)$,
ıch example has an identical prior prol

$$\mathbf{P}(E_j) = \mathbf{P}(E_{j-1}) = \mathbf{P}(E_{j-2}) = \cdots$$

There are four reasons why $\hat{h}_*$ may differ from the true function, f : unrealizability, variance, noise, and computational complexity. First, f may not be realizable—may not be in H—or may be present in such a way that other hypotheses are preferred. Second, a learning algorithm will return different hypotheses for different sets of examples, even if those sets are drawn from the same true function f , and those hypotheses will make different predictions on new examples. The higher the variance among the predictions, the higher the probability of significant error. Note that even when the problem is realizable, there will still be random variance, but that variance decreases towards zero as the number of training examples increases. Third, f may be nondeterministic or noisy—it may return different values for f (x) each time x occurs. By definition, noise cannot be predicted; in many cases, it arises because the observed labels y are the result of attributes of the environment not listed in x. And finally, when H is complex, it can be computationally intractable to systematically search the whole hypothesis space. The best we can do is a local search (hill climbing or greedy search) that explores only part of the space. That gives us an approximation error. Combining the sources of error, we're left with an estimation of an approximation of the true function f .

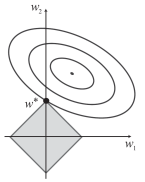
Learning Bayes Net Structure

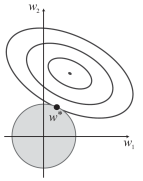
1. Start with no structure, ie. a set of nodes with no edges.
Keep adding edges, estimating parameters, and evaluating the model on the validation data to find out the best structure. This is a search method.
2. Start with an initial estimate or guess, and use hill-climbing methods to improve the structure iteratively.

1. Using the learnt structure, try to assert
that the conditional independence assumptions in the data set are still valid, ie.
`P(Fri/Sat, Bar | WillWait) = P(Fri/Sat | WillWait)P(Bar | WillWait)`
However the uncertainty and noise in the data set will never allow the assumption to hold true accurately, so some threshold or confidence value has to be chosen. The more strict the threshold is (that ie. the values should match very closely), the more complete the network becomes with added links, and more danger of overfitting.
2. Another way is to assess the degree (or probability) by which the structure explains the data, using something like maximum likelihood or MAP estimates.
However just using maximum likelihood hypothesis we may end up with a fully connected network, because adding new parents to a node does not decrease the likelihood.
MAP subtracts a penalty from the likelihood of each structure before comparing different structures. The bayesian approach places a joint prior over structures and parameters. There are usually far too many structures to sum over, so most practitioners use MCMC to sample over strucutres. With tabular distributions, the complexity penalty for a node's distribution grows exponentially with the number of parents, but with, say, noisy-OR distributions, it grows only linearly. This means that learning with noisy-OR (or other compactly parameterized) models tends to produce learned structures with more parents than does learning with tabular distributions.

$$Cost(h) = EmpLoss(h) + \lambda\, Complexity(h)$$

L1 regularization
has an important advantage:
It tends to produce a sparse model.
L2 is preferred if predictive features
Are correlated





Minimizing Loss(w) + λComplexity (w
is equivalent to minimizing Loss(w)
subject to the constraint that Comple:
some constant c that is related to λ.

$$w_i \leftarrow w_i + \alpha\,(y - h_{\mathbf{w}}(\mathbf{x})) \times x_i$$ perceptron weight update

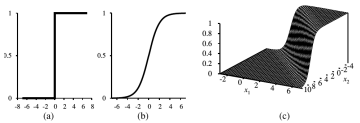
The perceptron algorithm is again not guaranteed to converge to the minimum, but if the learning rate is decayed then it can be shown to converge to minimum error solution _when examples are shown in random order_.

The result is that the
number of examples required
to find a good h is linear in the
number of irrelevant features for
L2 regularization, but only
Logarithmic with L1 regularization.

L0 Norm: Number
Of non-zero
Elements

Batch Gradient Descene
(where weights are updated after
Going through whole dataset)
Converging to global minima is
Guaranteed (given small alpha)
Stochastic Gradient Descent
(Where weights are updates after
Each training example is seen)
Convergence to global minima
Not guaranteed (choose a decay rate
To come close to global minima)

The L2 function is spherical, which makes it rotationally invariant: Imagine a set of points in a plane, measured by their x and y coordinates. Now imagine rotating the axes by 45 o . You'd get a different set of (x , y) values representing the same points. If you apply L2 regularization before and after rotating, you get exactly the same point as the answer (although the point would be described with the new (x, y) coordinates). That is appropriate when the choice of axes really is arbitrary—when it doesn't matter whether your two dimensions are distances north and east; or distances north-east and south-east. With L1 regularization you'd get a different answer. That is appropriate when the axes are not interchangeable; it doesn't make sense to rotate "number of bathrooms" 45o Towards "lot size."



Conditional Gaussian is defined when Parent is discrete.

GP > 0

/

Age

[u=19.25]

[s=3.91]

\

Age

[u=19.13]

[s=3.12]

$$\frac{\partial}{\partial w_i} Loss(\mathbf{w}) = \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(\mathbf{x}))^2$$
$$= 2(y - h_{\mathbf{w}}(\mathbf{x})) \times \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(\mathbf{x}))$$
$$= -2(y - h_{\mathbf{w}}(\mathbf{x})) \times g'(\mathbf{w} \cdot \mathbf{x}) \times \frac{\partial}{\partial w_i}\mathbf{w} \cdot \mathbf{x}$$
$$= -2(y - h_{\mathbf{w}}(\mathbf{x})) \times g'(\mathbf{w} \cdot \mathbf{x}) \times x_i \ .$$

$$w_i \leftarrow w_i + \alpha\,(y - h_{\mathbf{w}}(\mathbf{x})) \times h_{\mathbf{w}}(\mathbf{x})(1 - h_{\mathbf{w}}(\mathbf{x})) \times x_i \ .$$

In (a), the linearly separable case, logistic regression is somewhat slower to converge, but behaves much more predictably. In (b) and (c), where the data are noisy and nonseparable, logistic Regression converges far more quickly and reliably.

Solution to Overfitting

- Recall for discrete data: smooth towards the uniform distribution (Laplace correction).
- Bayesian perspective: maximize the posterior P(θ|D), which includes a (uniform) prior, not just the likelihood.
- For linear regression: smooth or **regularize** towards w= 0 weight.
- Bayesian perspective: maximize the posterior P(θ|D), which includes a *shrinkage prior* that assigns highest probability to 0.
- Interpretation: assume irrelevance until the data prove otherwise.

Log-Odds Classifier

- For a probabilistic classifier, a natural discriminative function are the log-odds h(**x**)=ln{P(class=1|**x**)/P(class=0|**x**)}
- max probability classifier ⇔ 0 log-odds threshold:
  - class =1 if P(class=1|**x**)> P(class=0|**x**)
  - P(class=1|**x**)/P(class=0|**x**)>1
  - ln{P(class=1|**x**)/P(class=0|**x**)}>0
- Log-odds focus on features that discriminate the classes
  - e.g. P(**x**) in Bayes formula cancels out
  - in Naive Bayes, if P(**x**$_i$|class=1)= P(**x**$_i$|class=0), then ln{P(**x**$_i$|class=1)/P(**x**$_i$|class=0)}=0
  - non-discriminative features are ignored

$$p(x_i|\underline{\theta}) = p(x_i|\mu,\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} exp^{-\frac{1}{2}(\frac{x_i-\mu}{\sigma})^2}$$

$$\log L(\underline{\theta}) = l(\underline{\theta}) = -\frac{n}{2}\log(2\pi\theta_2) - \frac{1}{2\theta_2}\sum_{i=1}\Big(x_i - \theta_1\Big) \ .$$

Smoothing Frequency Estimates

- *h* heads, *t* tails, *n* = *h*+*t*.
- Prior probability estimate *p*.
- Equivalent Sample Size *m*.
- m-estimate = $\frac{h + mp}{n + m}$
- Interpretation: we started with a "virtual" sample of *m* tosses with *mp* heads.
- *P* = ½,*m*=2 ➔ **Laplace correction** = $\frac{h+1}{n+2}$

Parent Node/ Child Node	Discrete	Continuous
Discrete	**Maximum Likelihood** Decision Trees	logit distribution (logistic regression)
Continuous	conditional Gaussian (not discussed)	linear Gaussian (linear regression)