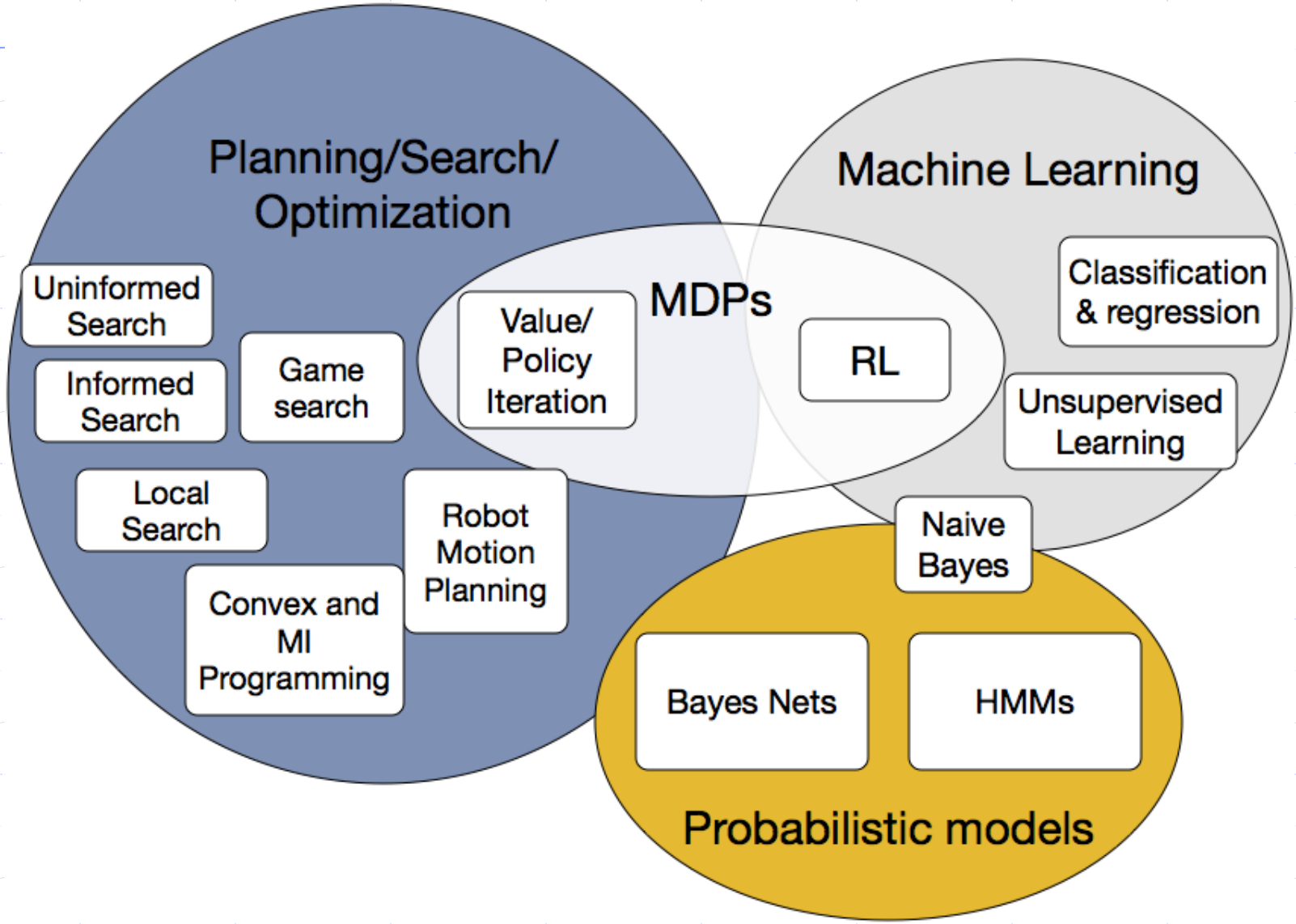
Classification and Regression

Big Picture

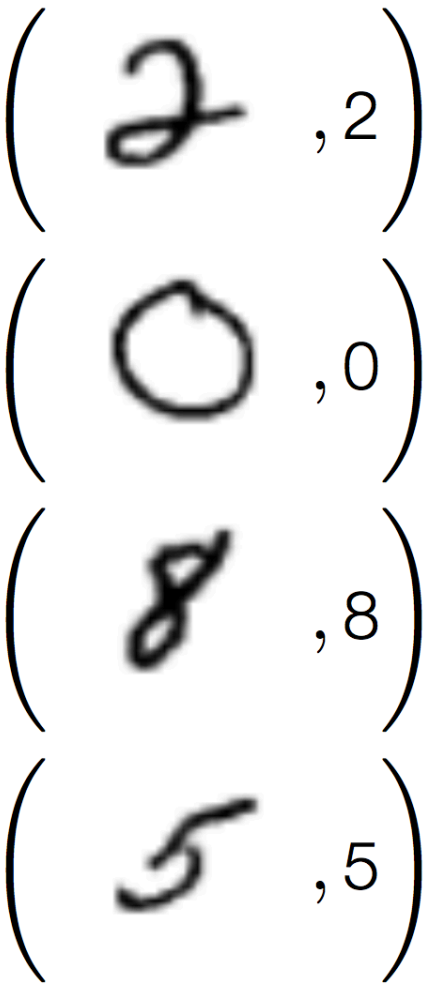

Planning/Search/
Optimization
Machine Learning
Uninformed
Search
Informed
Search
Game
search
Local
Search
Convex and
MI
Programming
Robot
Motion
Planning
MDPs
Value/
Policy
Iteration
RL
Classification
& regression
Unsupervised
Learning
Naive
Bayes
Bayes Nets
HMMs
Probabilistic models

Machine learning

- Three broad classes of problems:
  - ✓ Reinforcement learning (RL):
    - *Learn a policy that maximizes expected return given a time-delayed scalar reward signal*
  - Supervised learning:
    - *Learn a function $y=h(f)$ that maps input features, f, to a target value, y*
    - y continuous: regression, y discrete: classification
  - Unsupervised learning
    - *No labels or rewards, goal is to discover "structure" in data*
    - Clustering, dimensionality reduction, anomaly detection, …

Supervised Learning

Training data

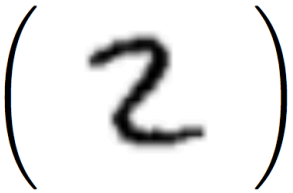


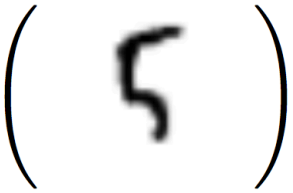
X

Learning Algorithm

h_θ

Testing

Prediction $= h_\theta$ 

Prediction $= h_\theta$ 

...

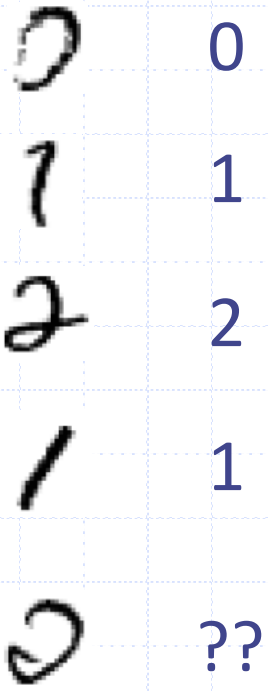
Classification Problems

- Classification Problem:
  - *Input*: A sample set of inputs (i.e. feature vectors), $f_1,\ldots,f_k$
  - *Output*: A list of labels, $y_1,\ldots,y_k$, corresponding to each input
  - *f* can be continuous or discrete
  - *y* is discrete (otherwise it's a regression problem)

- Goal: given a set of labeled data $(f_1,y_1),\ldots,(f_n,y_n)$, build a model that predicts the labels with high accuracy *on unseen data*

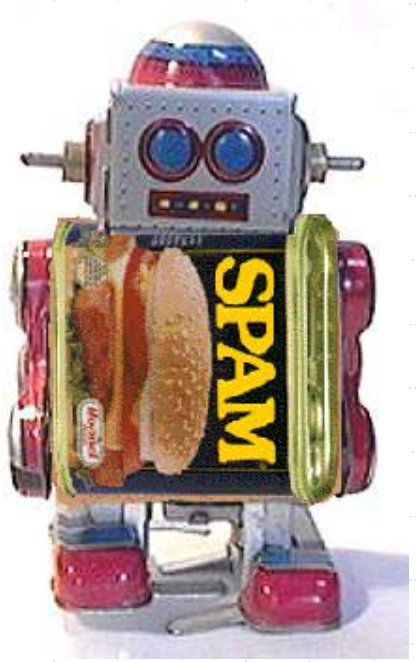
Example: Digit Recognition

- Input: images / pixel grids
- Output: a digit prediction 0-9
- Setup:
  - Get a large collection of example images, each labeled with a digit
    - Someone has to manually label all this data!
  - Want to learn to predict labels of ***new***, ***future digit images***
- Features:
  - Binary pixels: (13,17)=ON
  - Shape Patterns: NumComponents, AspectRatio, NumLoops
  - …



Example: Spam Filter

- Input: an email (array of strings)
- Output: spam or not
- Setup:
  - Get a large collection of example emails, each labeled "spam" or "ham / not spam"
  - Want to learn to predict labels of ***new, future emails***
- Features:
  - Words: "FREE!", "Nigeria"
  - Text Patterns: $dd, CAPS
  - Non-text metadata: SenderInContacts
  - …



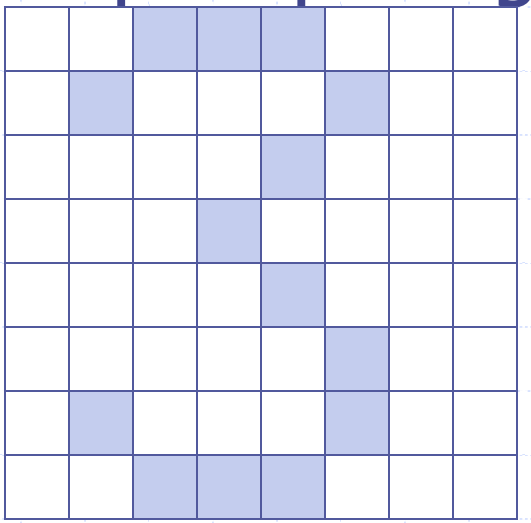
Model-Based Classification

- Model-based approach
  - Build a model (e.g., Bayes' net) where ***both the label and features are random variables***
  - Instantiate any observed features
  - Query for the distribution of the label conditioned on the features
- Challenges
  - What structure should the BN have?
  - How should we learn its parameters?
    - E.g. What is *P(spam | "ROCKSTAR")*? etc.

Naïve Bayes

A Digit Recognizer

- Input: pixel grids

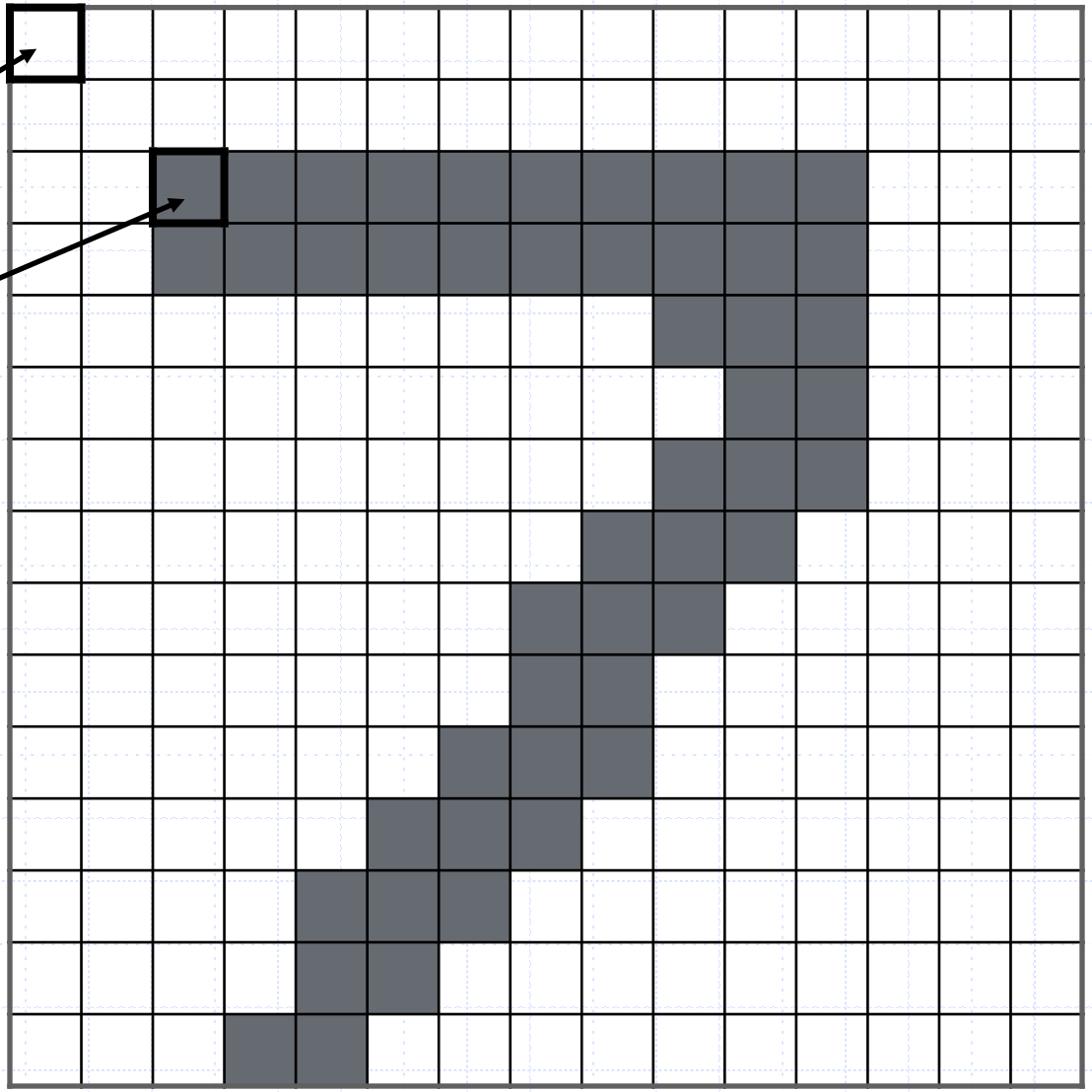


- Output: a digit 0-9

Digit Features

$F_{1,1} = 0$

$F_{3,3} = 1$



Naïve Bayes for Digits

- Simple version:
  - One feature F_{ij} for each grid position <i,j>
  - Possible feature values are on / off, based on whether intensity is more or less than 0.5 in underlying image
  - Each input maps to a feature vector, e.g.

$$\rightarrow \langle F_{0,0} = 0 \;\; F_{0,1} = 0 \;\; F_{0,2} = 1 \;\; F_{0,3} = 1 \;\; F_{0,4} = 0 \;\; \ldots F_{15,15} = 0 \rangle$$

  - Here: lots of features, each is binary valued
- Naïve Bayes model:

$$P(Y|F_{0,0} \ldots F_{15,15}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$$

- What do we need to learn?

General Naïve Bayes

- A general *naive Bayes* model:

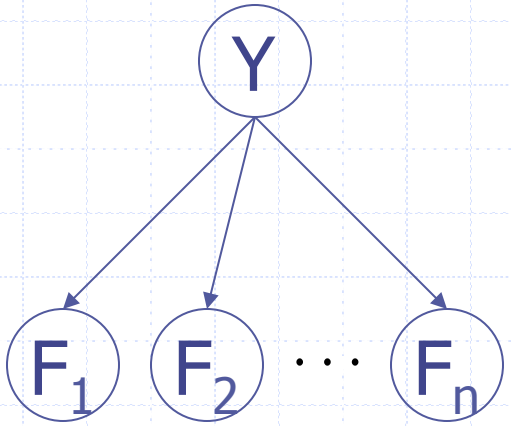
|Y| x |F|^n parameters

$$P(\mathsf{Y}, \mathsf{F}_1 \ldots \mathsf{F}_n) =$$

$$P(\mathsf{Y}) \prod_i P(\mathsf{F}_i|\mathsf{Y})$$

|Y| parameters

n x |F| x |Y| parameters



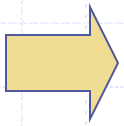
- We only specify how each feature depends on the class
- Total number of parameters is *linear* in n

Inference for Naïve Bayes

- Goal: compute posterior over Y,F
  - Step 1: get joint probability of Y (class) and F (evidence)

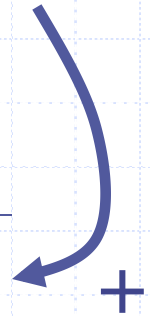
$$P(Y, f_1 \dots f_n) =$$

$$\begin{bmatrix} P(y_1, f_1 \dots f_n) \\ P(y_2, f_1 \dots f_n) \\ \vdots \\ P(y_k, f_1 \dots f_n) \end{bmatrix}$$

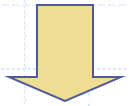


$$\begin{bmatrix} P(y_1) \prod_i P(f_i \mid y_1) \\ P(y_2) \prod_i P(f_i \mid y_2) \\ \vdots \\ P(y_k) \prod_i P(f_i \mid y_k) \end{bmatrix}$$

$$P(f_1 \dots f_n)$$



  - Step 2: get probability of evidence
  - Step 3: renormalize



$$P(Y \mid f_1 \dots f_n)$$

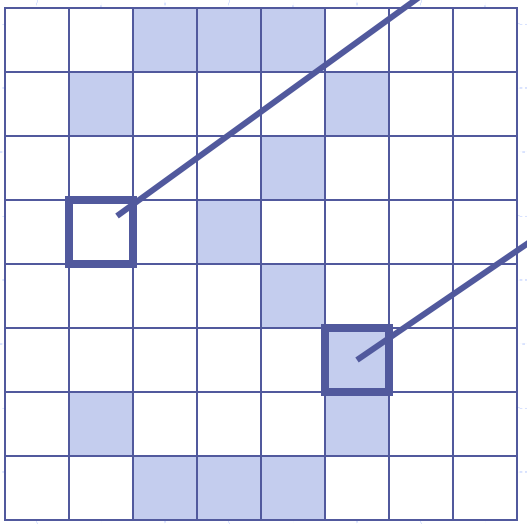
General Naïve Bayes

- What do we need in order to use naïve Bayes?
  - Inference (you know this part)
    - Start with a bunch of conditionals, P(Y) and the $P(F_i|Y)$ tables
    - Use standard inference to compute $P(Y|F_1 \ldots F_n)$
    - Nothing new here
  - Estimates of local conditional probability tables
    - P(Y), the prior over labels
    - $P(F_i|Y)$ for each feature (evidence variable)
    - These probabilities are collectively called the *parameters* of the model and denoted by θ
    - Up until now, we assumed these appeared by magic, but…
    - …they typically come from training data: we'll look at this now

Examples: CPTs

$P(Y)$

1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1



$P(F_{3,1} = on|Y)$

1	0.01
2	0.05
3	0.05
4	0.30
5	0.80
6	0.90
7	0.05
8	0.60
9	0.50
0	0.80

$P(F_{5,5} = on|Y)$

1	0.05
2	0.01
3	0.90
4	0.80
5	0.90
6	0.90
7	0.25
8	0.85
9	0.60
0	0.80

Naïve Bayes for Text

- Bag-of-Words Naïve Bayes:
  - Predict unknown class label (spam vs. ham)
  - Assume evidence features (e.g. the words) are independent
  - Warning: subtly different assumptions than before!
- Generative model

$$P(C, W_1 \dots W_n) = P(C) \prod_i P(W_i|C)$$

Word at position i, not i^{th} word in the dictionary!

- In a bag-of-words model
  - Each position is identically distributed
  - All positions share the same conditional probs P(W|C)
  - Why make this assumption?

Example: Spam Filtering

- Model: $P(C, W_1 \dots W_n) = P(C) \prod_i P(W_i|C)$
- What are the parameters?

$P(C)$

```
ham : 0.66
spam: 0.33
```

$P(W|\text{spam})$

```
the :  0.0156
to  :  0.0153
and :  0.0115
of  :  0.0095
you :  0.0093
a   :  0.0086
with:  0.0080
from:  0.0075
...
```

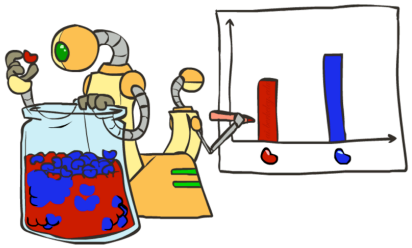
$P(W|\text{ham})$

```
the :  0.0210
to  :  0.0133
of  :  0.0119
2002:  0.0110
with:  0.0108
from:  0.0107
and :  0.0105
a   :  0.0100
...
```

- Where do these tables come from?

Parameter Estimation

- Estimating the distribution of a random variable
- *Elicitation*: ask a human expert
- *Empirically*: learn it use training data



  - E.g.: for each outcome *x*, look at the ***empirical rate*** of that value:

$$P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

$P_{\text{ML}}(\text{r}) = 2/3$

  - This is the estimate that maximizes the ***likelihood of the data*** $L(x, \theta) = \prod_i P_\theta(x_i)$

e.g., find $P(r)=\theta$ that maximizes $L=\theta^2(1-\theta)$.
Take derivative, set equal to 0 ==> $\theta=2/3$

Maximum Likelihood

- Relative frequencies are the maximum likelihood estimates

$$\theta_{ML} = \arg\max_{\theta} P(\mathbf{X}|\theta)$$
$$= \arg\max_{\theta} \prod_i P_\theta(X_i)$$

$$\Rightarrow \quad P_{\mathsf{ML}}(x) = \frac{\mathsf{count}(x)}{\mathsf{total\ samples}}$$

- Another option is to consider the most likely parameter value given the data

$$\theta_{MAP} = \arg\max_{\theta} P(\theta|\mathbf{X})$$
$$= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)/P(\mathbf{X})$$
$$= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)$$

Counting!!!

$$P(c_i) = \frac{\#c_i}{N}$$

$$P(f_i \mid c_i) = \frac{\#c_i, f_i}{\#c_i}$$

A Spam Filter

- Naïve Bayes spam filter
- Data:
  - Collection of emails, labeled spam or ham
  - Note: someone has to hand label all this data!
  - Split into training, held-out, test sets
- Classifiers
  - Learn on the training set
  - (Tune it on a held-out set)
  - Test it on new emails

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES FOR ONLY $99

Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Name:

Ex. Learning NB

W1	W2	.....	Class
The	Deal		Spam
Free	$$		Spam
Dear	Joe		Ham
$$	Free		Spam
Hello	Fred		Ham
Fred	Hello		Ham
Dear	Fred		Ham
Dear	Winner		Spam
Free	Firewood		Ham
Hello	Winner		Spam

P(Ham) =

P(Spam) =

P(Hello|Ham) =

P(Hello|Spam) =

P(Free|Ham) =

P(Free|Spam) =

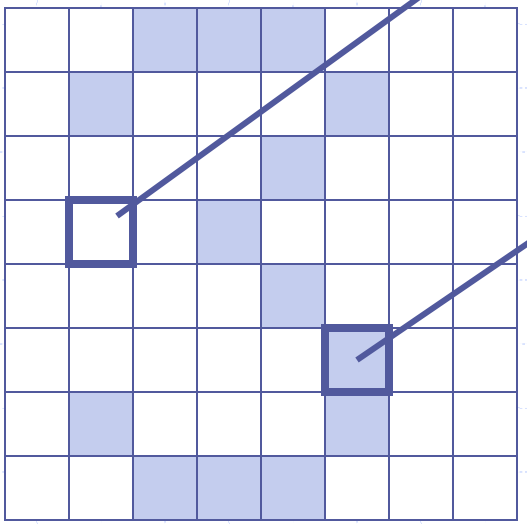
P(Dear|Ham) =

P(Dear|Spam) =

Similar for General NB: CPT are just counts

$P(Y)$

1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1



$P(F_{3,1} = on|Y)$

1	0.01
2	0.05
3	0.05
4	0.30
5	0.80
6	0.90
7	0.05
8	0.60
9	0.50
0	0.80

$P(F_{5,5} = on|Y)$

1	0.05
2	0.01
3	0.90
4	0.80
5	0.90
6	0.90
7	0.25
8	0.85
9	0.60
0	0.80

Example: Overfitting

$P(\text{features}, C = 2)$

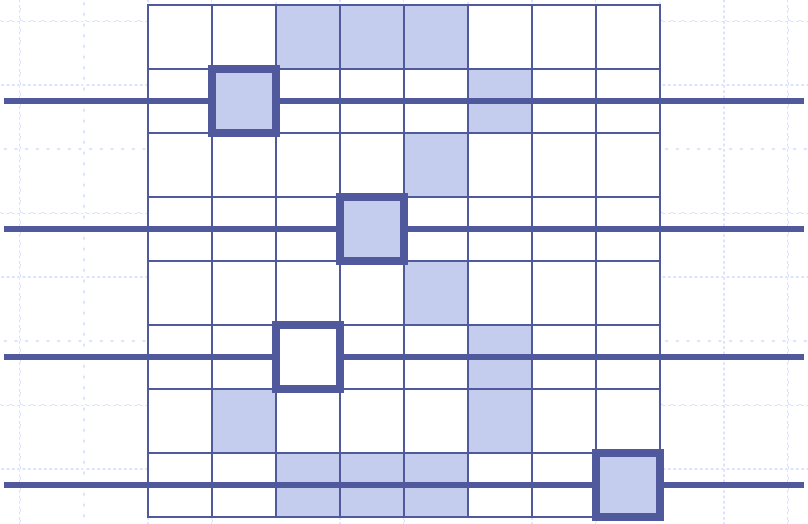
$P(C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.8$

$P(\text{on}|C = 2) = 0.1$

$P(\text{off}|C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.01$



$P(\text{features}, C = 3)$

$P(C = 3) = 0.1$

$P(\text{on}|C = 3) = 0.8$

$P(\text{on}|C = 3) = 0.9$

$P(\text{off}|C = 3) = 0.7$

$P(\text{on}|C = 3) = 0.0$

2 wins!!

Example: Overfitting

- Posteriors determined by *relative* probabilities (odds ratios):

$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

```
south-west : inf
nation     : inf
morally    : inf
nicely     : inf
extent     : inf
seriously  : inf
...
```

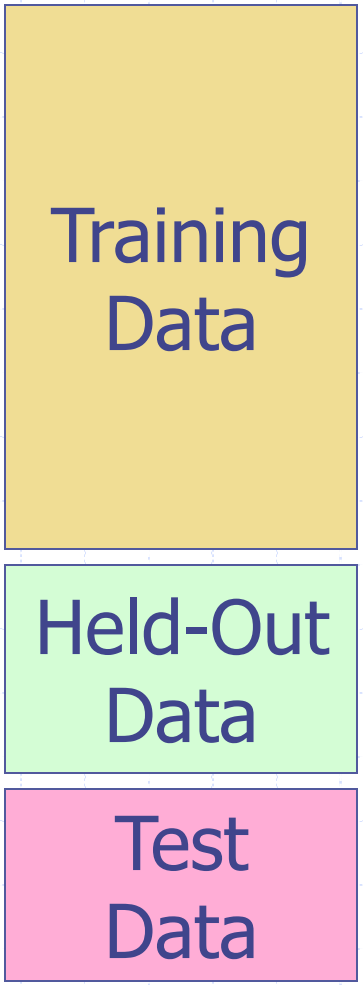
$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

```
screens    : inf
minute     : inf
guaranteed : inf
$205.00    : inf
delivery   : inf
signature  : inf
...
```

What went wrong here?

Important Concepts

- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out set
  - Test set
- Features: attribute-value pairs which characterize each x
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - (Tune hyperparameters on held-out set)
  - Compute accuracy of test set
  - Very important: never "peek" at the test set!
- Evaluation
  - Accuracy: fraction of instances predicted correctly
- Overfitting and generalization
  - Want a classifier which does well on *test* data
  - Overfitting: fitting the training data very closely, but not generalizing well



Generalization and Overfitting

- Relative frequency parameters will overfit the training data!
  - Just because we never saw a 3 with pixel (15,15) on during training doesn't mean we won't see it at test time
  - Unlikely that every occurrence of "minute" is 100% spam
  - Unlikely that every occurrence of "seriously" is 100% ham
  - What about all the words that don't occur in the training set at all?
  - In general, we can't go around giving unseen events zero probability
- As an extreme case, imagine using the entire email as the only feature
  - Would get the training data perfect (if deterministic labeling)
  - Wouldn't *generalize* at all
  - Just making the bag-of-words assumption gives us some generalization, but isn't enough
- To generalize better: we need to smooth or regularize the estimates

Estimation: Smoothing

- Maximum likelihood estimates:

$$P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

P_{ML} (T) = 1/3

- Problems with maximum likelihood estimates:
  - If I flip a coin once, and it's heads, what's the estimate for P(heads)?
  - What if I flip 10 times with 8 heads?
  - What if I flip 10M times with 8M heads?

- Basic idea:
  - We have some prior expectation about parameters (here, the probability of heads)
  - Given little evidence, we should skew towards our prior
  - Given a lot of evidence, we should listen to the data

Estimation: Laplace Smoothing

- Laplace's estimate (extended):
  - Pretend you saw every outcome k extra times

$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

  - What's Laplace with k = 0?
  - k is the strength of the prior

- Laplace for conditionals:
  - Smooth each condition independently:

$$P_{LAP,k}(x|y) = \frac{c(x,y) + k}{c(y) + k|X|}$$

$$P_{LAP,0}(X) =$$

$$P_{LAP,1}(X) =$$

$$P_{LAP,100}(X) =$$

Real NB: Smoothing

- For real classification problems, smoothing is critical
- New odds ratios:

$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

```
helvetica : 11.4
seems     : 10.8
group     : 10.2
ago       :  8.4
areas     :  8.3
...
```

$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

```
verdana : 28.8
Credit  : 28.4
ORDER   : 27.2
<FONT>  : 26.9
money   : 26.5
...
```

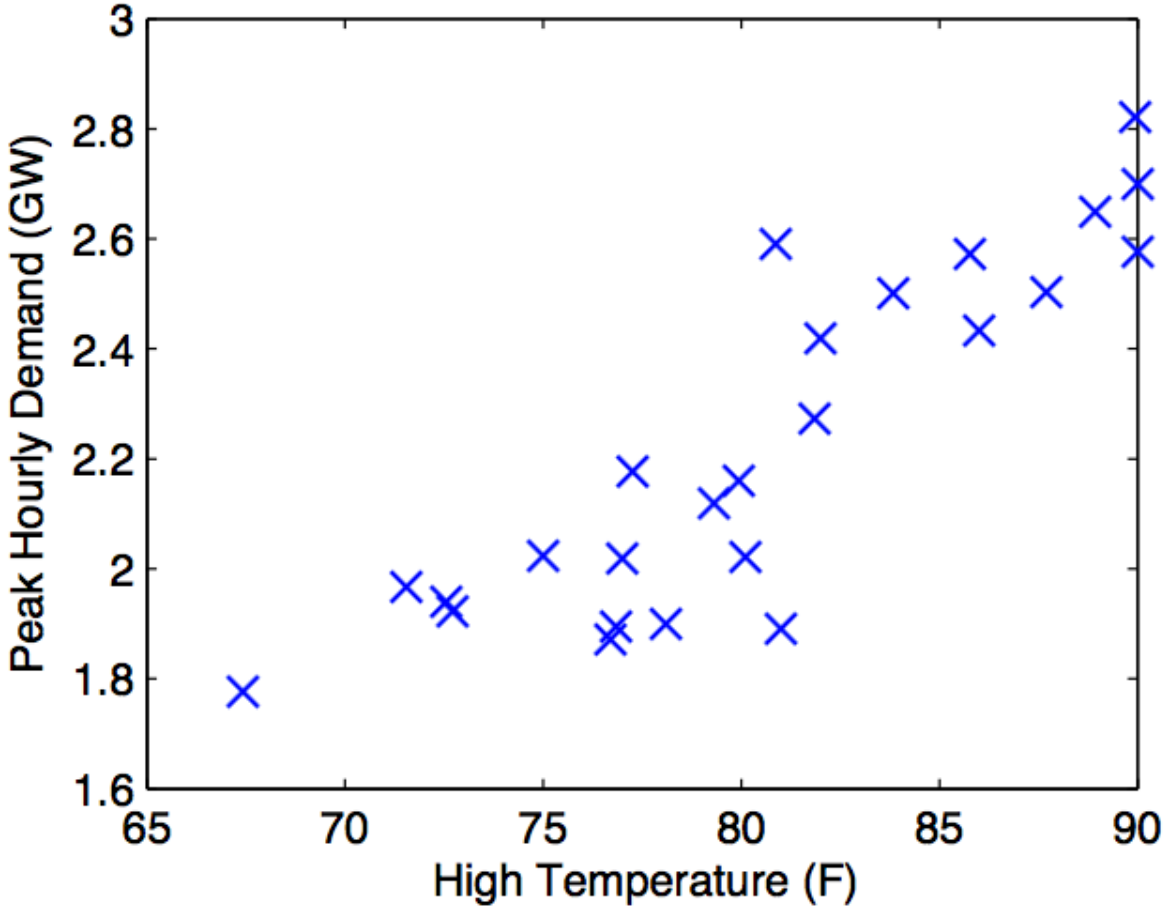
Do these make more sense?

Regression Problems

Regression Problems

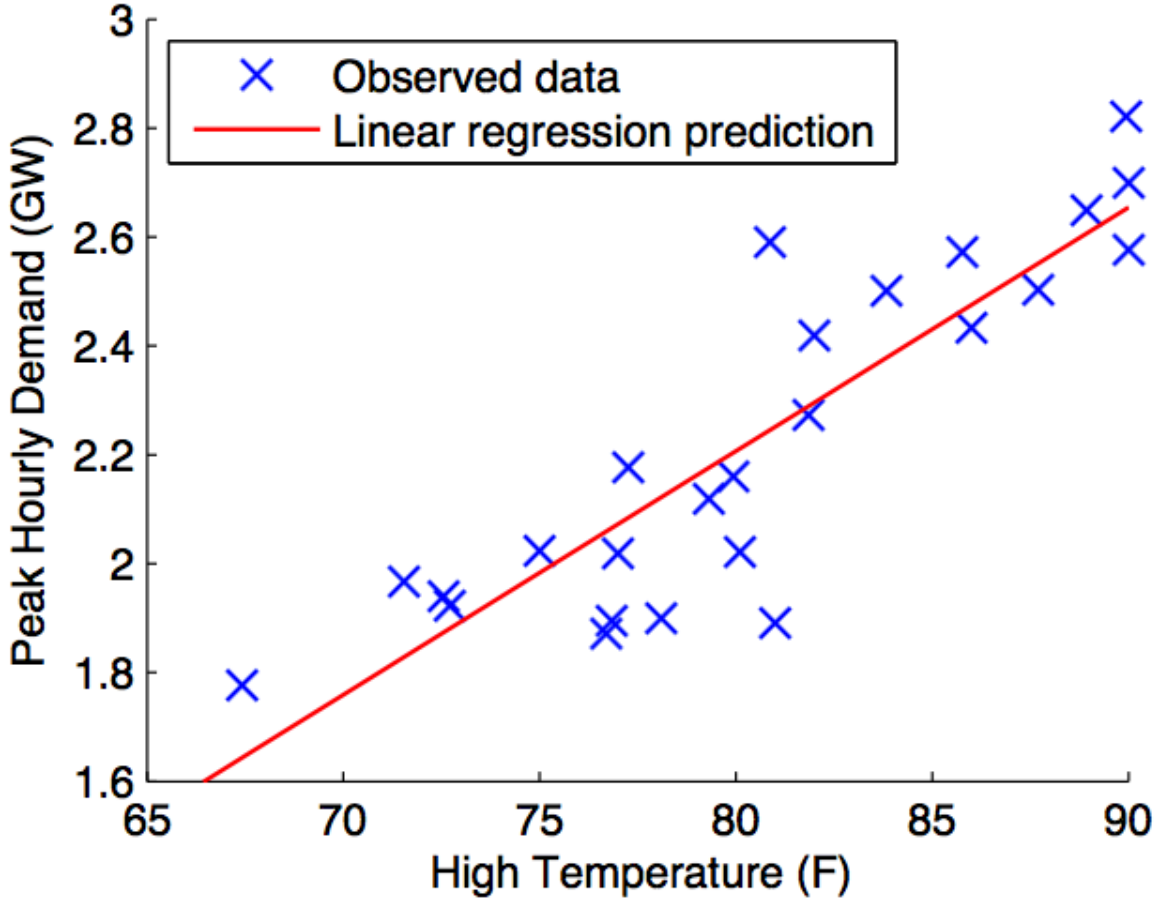
- Sometimes we care about predicting a real-valued label *y* given features $f_1,\ldots,f_k$
- There are ways to think about this problem using probabilistic models, but often we think about this as a parametric regression problem
- Example: what will the peak power consumption in Santa Cruz be on a given day?
  - ***Features***: high temperature on past days
  - ***Target values***: peak demand

Regression



$$\text{Peak demand} \approx \theta_1(\text{high temp}) + \theta_2$$

Regression



$$\text{Peak demand} \approx \theta_1(\text{high temp}) + \theta_2$$

Optimizing θ

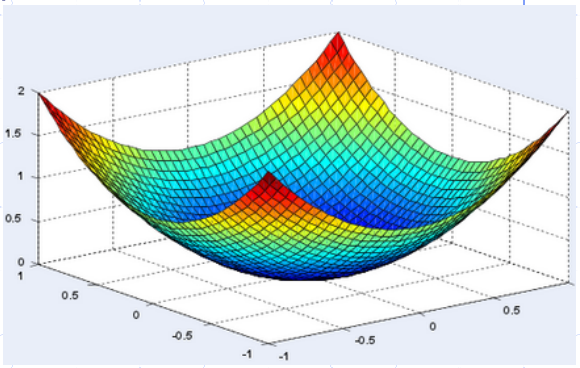
- We have described the ***features***, ***targets***, and a ***model*** we want to learn to predict target values
- We haven't specified a learning problem yet —> need an objective or *loss function*
- Loss functions are designed to measure how "good" a predictor is against the data
- For regression, we usually use a squared loss:

$$L(h_\theta(f^{(i)}), y^{(i)}) = (h_\theta(f^{(i)}) - y^{(i)})^2 = (\theta^T f^{(i)} - y^{(i)})^2$$

Optimizing θ

- Summing over all of the data, we have the cost:

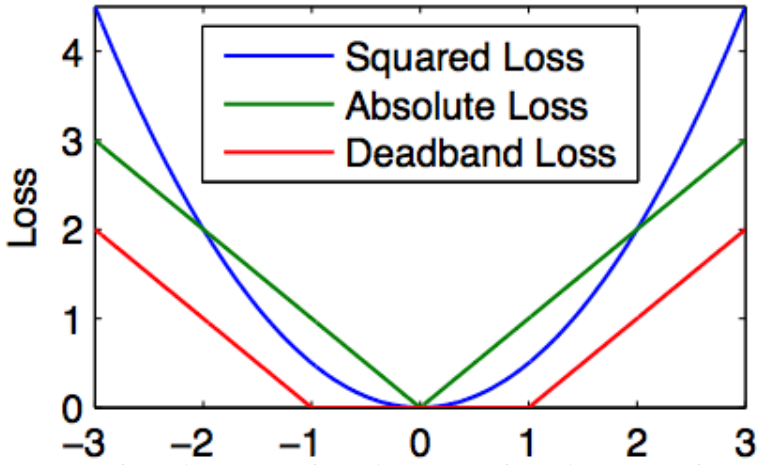
$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{m} (\theta^T f^{(i)} - y^{(i)})^2$$



- This is convex in θ
  - Set the derivative w.r.t. θ equal to zero and solve for θ
    - Results in familiar closed-form solution you've seen in your statistics course
    - This is a rare benefit, usually have to do (stochastic) gradient descent!

Other loss functions

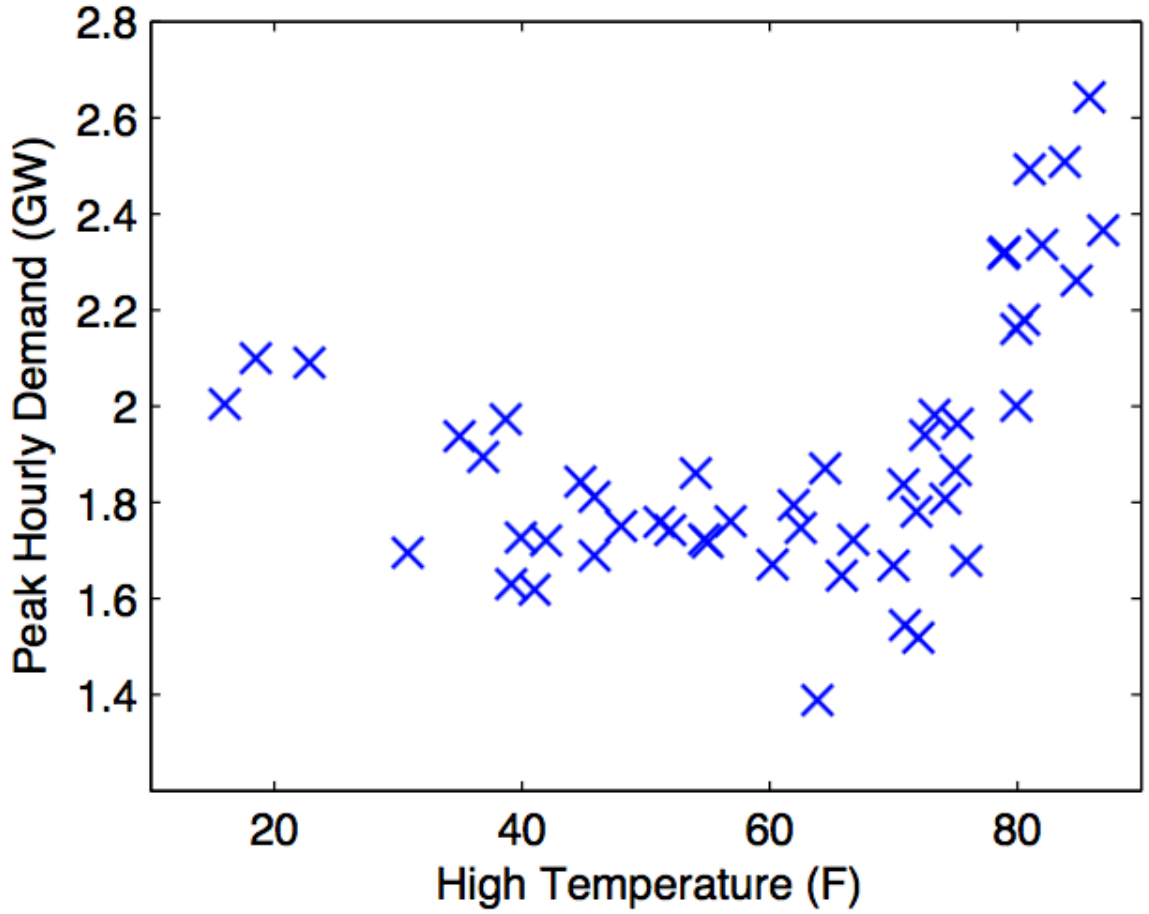
- Loss functions are a design choice
- Other choices of loss functions in linear regression do not admit closed form solutions, e.g.,



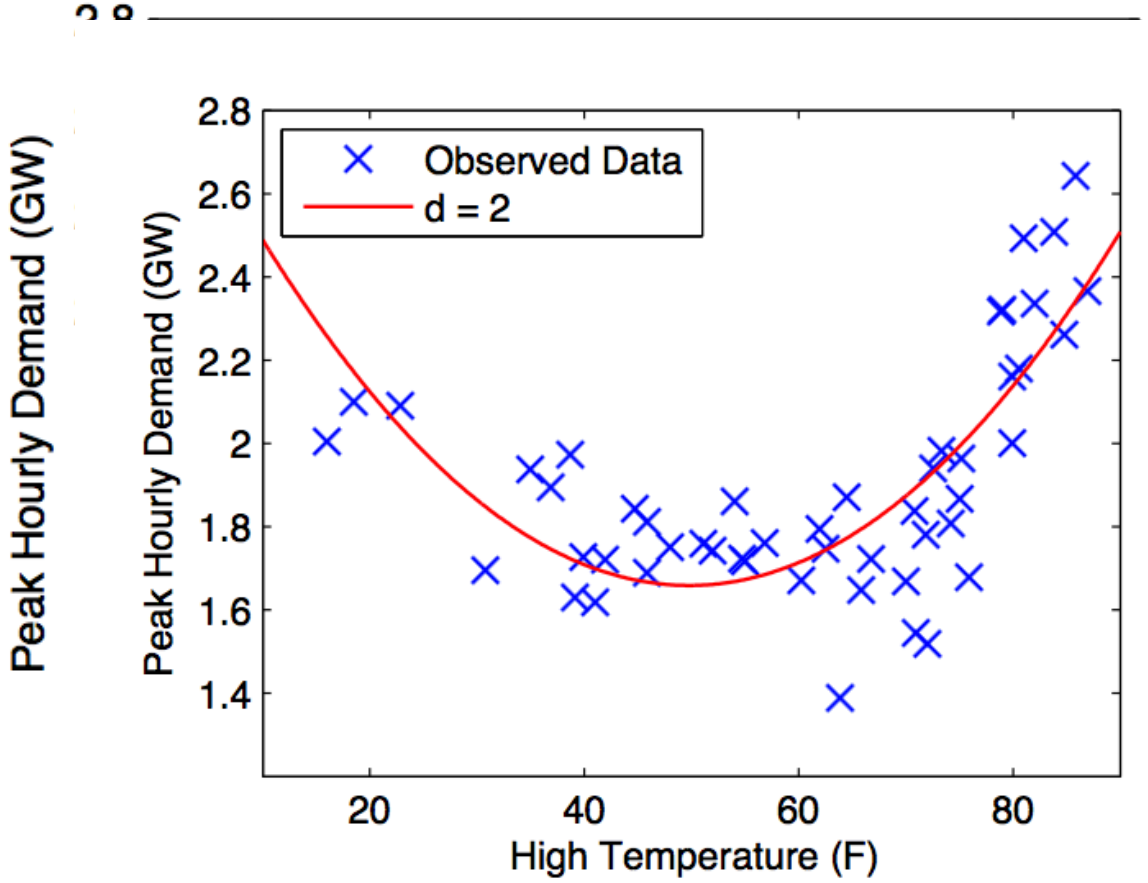
- But we can still perform gradient descent:

$$\theta \leftarrow \theta - \alpha \sum_{i=1}^{m} \nabla_\theta L(h_\theta(f^{(i)}), y^{(i)})$$

Nonlinear Regression



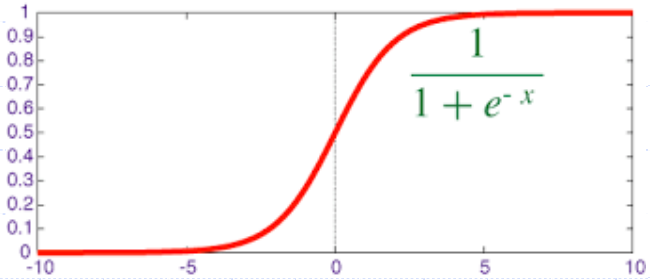
Nonlinear Regression



$$h_\theta(f) = \theta^T \phi(f) = \theta_2 \cdot f^2 + \theta_1 \cdot f + \theta_0$$

Can we use this type of model for classification?

- For example, suppose I wanted h_θ to output the probability of spam/ham… $h_\theta(f) = P(spam \mid f)$
  - Not immediately useful since $h_\theta(f) = f^T\theta$ is not bounded [0,1]
  - Idea: "squash" the function using a sigmoid: $h_\theta(f) = \sigma(f^T\theta)$



  - This is called ***logistic regression***
- We can train logistic regression classifiers using gradient descent
- Q: What loss function should we use?

Logistic Regression Loss

- Can we still use the squared error?

$$L(h_\theta(f), y) = (h_\theta(f) - y)^2 \equiv (\hat{y} - y)^2 \qquad \hat{y} = h_\theta(f)$$

- Sure, but the resulting optimization is no longer convex in θ! (convince yourself!)
- Could define a different loss that ***is*** convex

$$L(h_\theta(f), y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

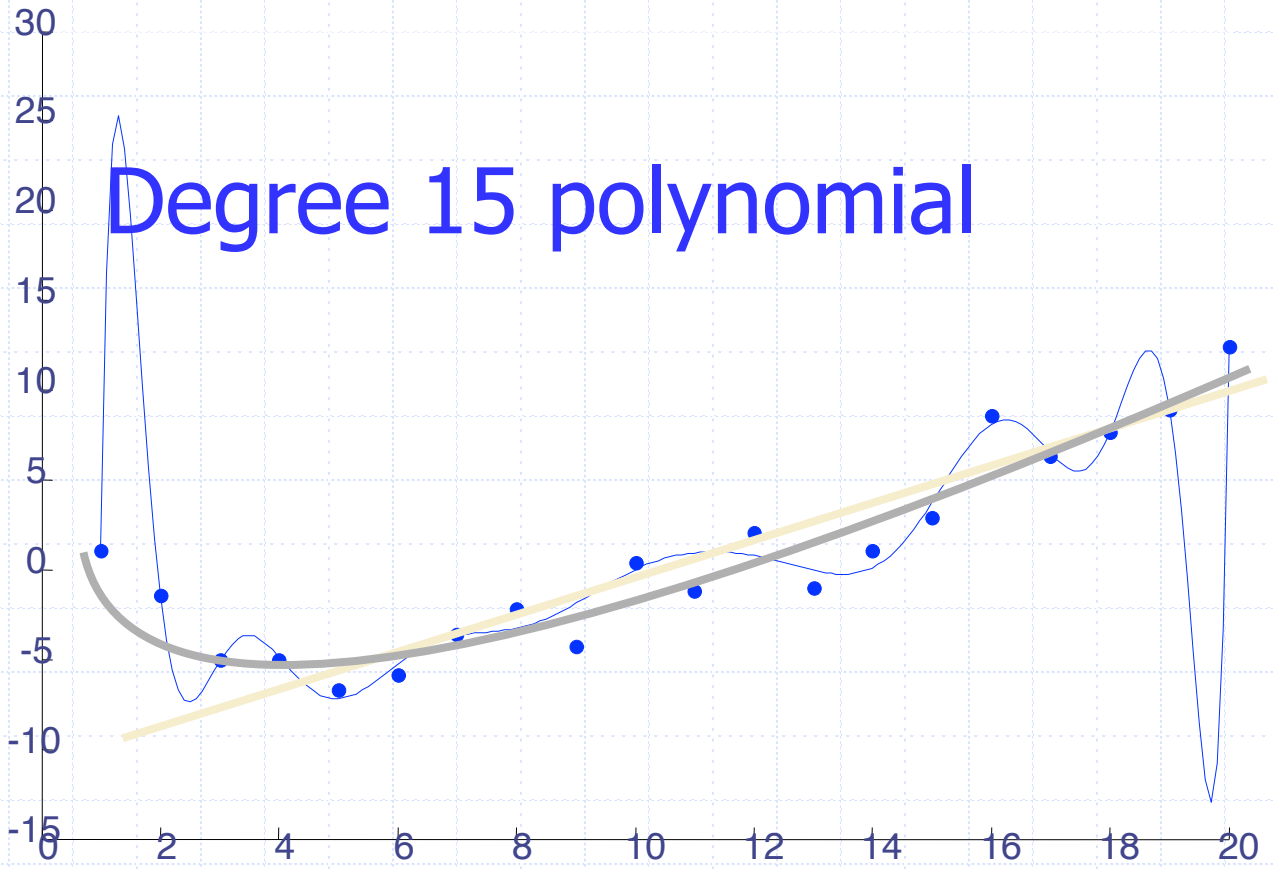
- Intuition:

$$y = 1 \Rightarrow L = -\log \hat{y} \quad \text{“want } \hat{y} \text{ large”}$$
$$y = 0 \Rightarrow L = -\log(1 - \hat{y}) \quad \text{“want } \hat{y} \text{ small”}$$

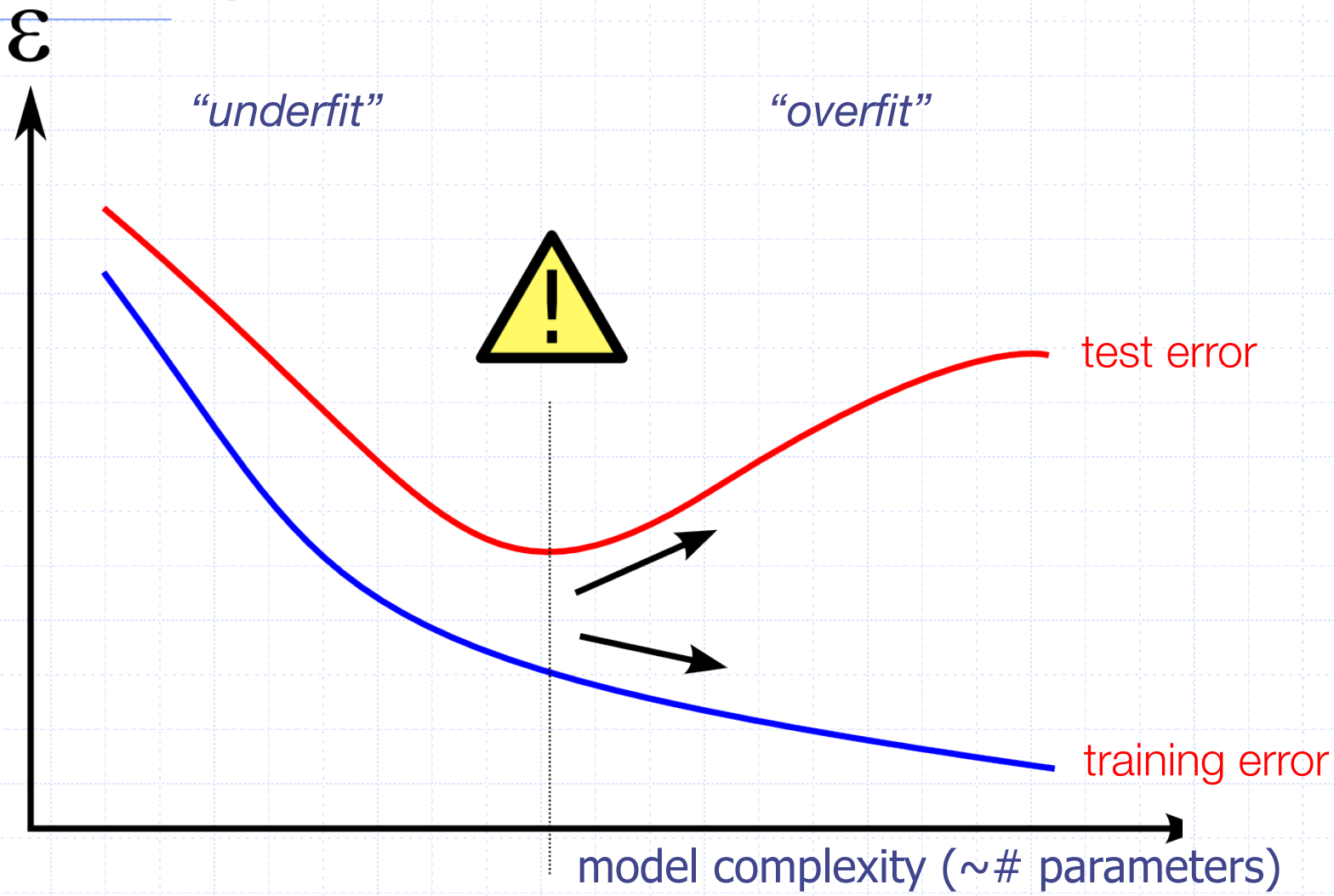
Other interpretations of loss*

- We've given rather informal convexity-based justifications for both the squared loss in (non)linear regression and log loss in logistic regression
- It turns out there's also a clean probabilistic motivation for both
- Regression:
  - Assume independent, identically distributed (i.i.d) Gaussian noise
  - Maximize log-likelihood of the data given parameters —> least squares
- Logistic regression:
  - Assume i.i.d. Bernoulli distribution
  - Maximize log-likelihood of the data given parameters —> logistic loss

Overfitting in Regression



Training vs. Test Error

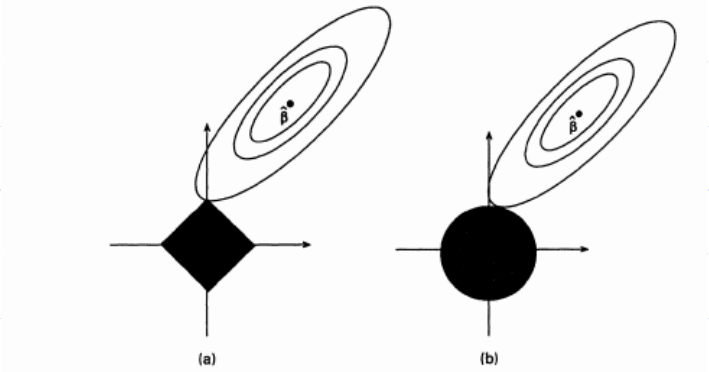


Smoothing in Regression

- How do we apply smoothing to regression problems?
- One simple way is to add a *regularizer* to the loss function

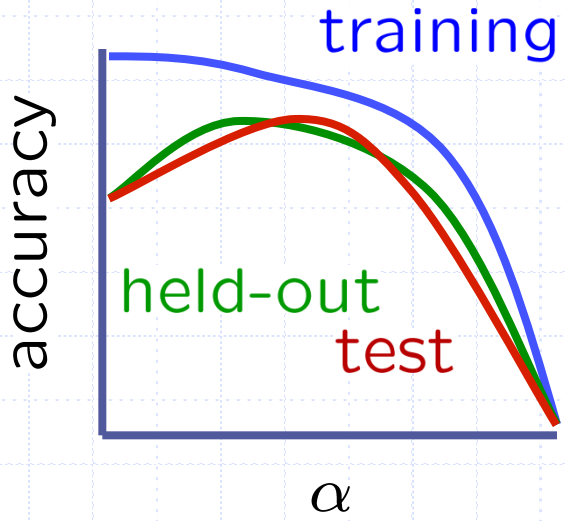
"ridge regression" $$L(h_\theta(f^{(i)}), y^{(i)}) = (\theta^T f^{(i)} - y^{(i)})^2 + \lambda \|\theta\|_2^2$$

"lasso" regression $$L(h_\theta(f^{(i)}), y^{(i)}) = (\theta^T f^{(i)} - y^{(i)})^2 + \lambda \|\theta\|_1$$



Tuning on Held-Out Data

- Now we've got two kinds of unknowns
  - Parameters: the probabilities P(Y|X), P(Y)
  - Hyperparameters, like the amount of smoothing to do: k, etc.
- Where to learn?
  - Learn parameters from training data
  - Must tune hyperparameters on different data
    - Why?
  - For each value of the hyperparameters, train and test on the held-out data
  - Choose the best value and do a final test on the test data



Errors, and What to Do

- Examples of errors

> Dear GlobalSCAPE Customer,
>
> GlobalSCAPE has partnered with ScanSoft to offer you the latest version of OmniPage Pro, for just $99.99* - the regular list price is $499! The most common question we've received about this offer is - Is this genuine? We would like to assure you that this offer is authorized by ScanSoft, is genuine and valid. You can get the . . .

> . . . To receive your $30 Amazon.com promotional certificate, click through to
>
> http://www.amazon.com/apparel
>
> and see the prominent link for the $30 offer. All details are there. We hope you enjoyed receiving this message. However, if you'd rather not receive future e-mails announcing new store launches, please click . . .

What to Do About Errors?

- Need more features– words aren't enough!
  - Have you emailed the sender before?
  - Have 1K other people just gotten the same email?
  - Is the sending information consistent?
  - Is the email in ALL CAPS?
  - Do inline URLs point where they say they point?
  - Does the email address you by (your) name?
- Can add these information sources as new variables in the NB model

Summary

- NB is a simple probabilistic classifier
- (Variations) widely used in practice
- Can also learn more complex Bayesian networks in a similar manner