CPSC 340: Machine Learning and Data Mining

Feature Engineering

Last Time: Multi-Class Linear Classifiers

- We discussed multi-class classification: y_i in {1,2,…,k}.
- One vs. all with +1/-1 binary classifier:
  - Train weights w_c separately to predict +1 for class 'c', -1 otherwise.

$$W = \underbrace{\begin{bmatrix} — & w_1^T & — \\ — & w_2^T & — \\ & \vdots & \\ — & w_\ell^T & — \end{bmatrix}}_{d} \Big\} \ell$$

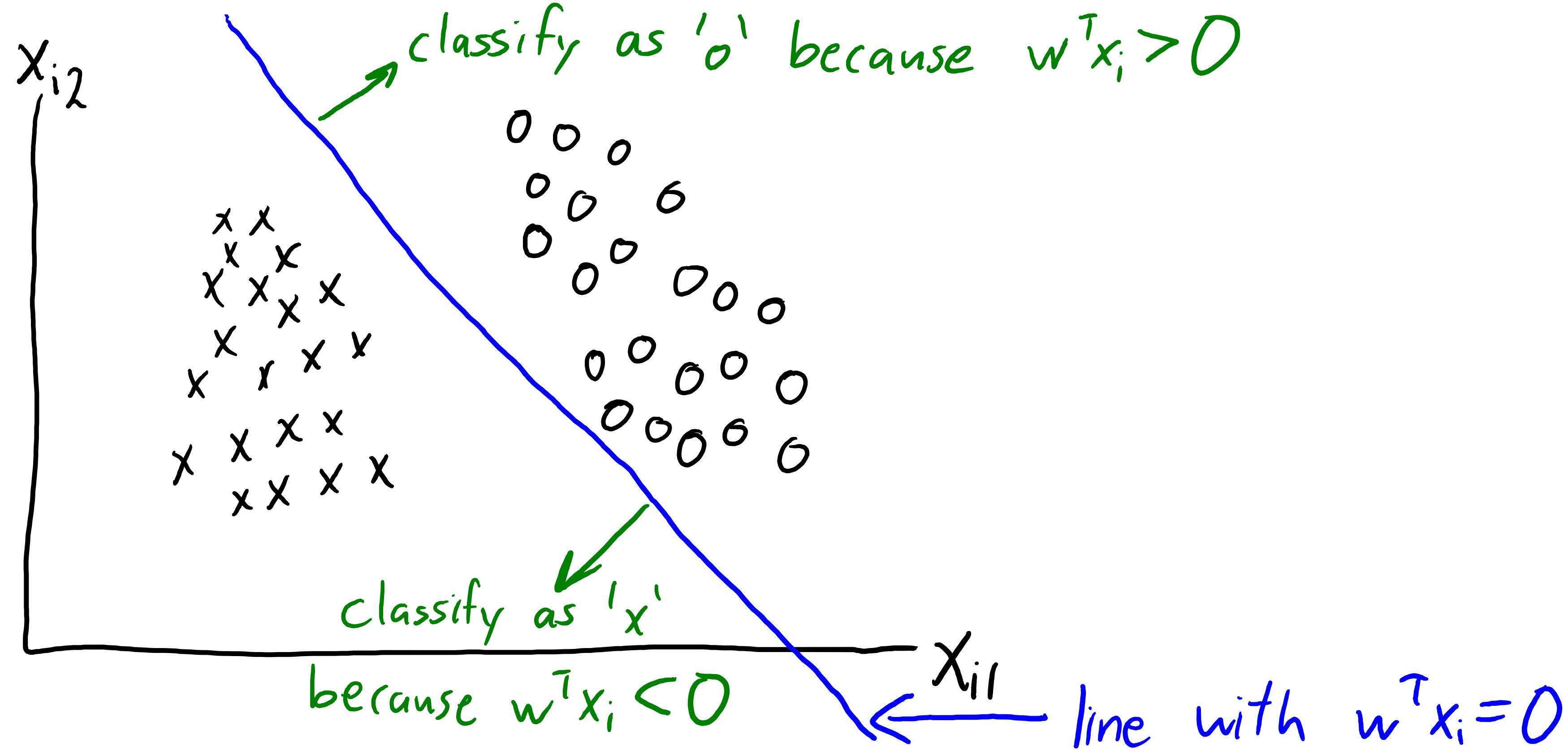
  - Predict by taking 'c' maximizing class output $o_{ic} = w_c^T x_i$.
- Multi-class SVMs and multi-class logistic regression:
  - Train the w_c jointly to encourage maximum o_{ic} to be o_{iy_i}.

$$f(W) = \sum_{i=1}^{n}\Big[-\underbrace{w_{y_i}^T x_i}_{o_{iy_i}} + \log\Big(\sum_{c=1}^{k} \exp(\underbrace{w_c^T x_i}_{o_{ic}})\Big)\Big] + \frac{\lambda}{2}\|W\|_F^2$$

"Frobenius" norm

Shape of Decision Boundaries

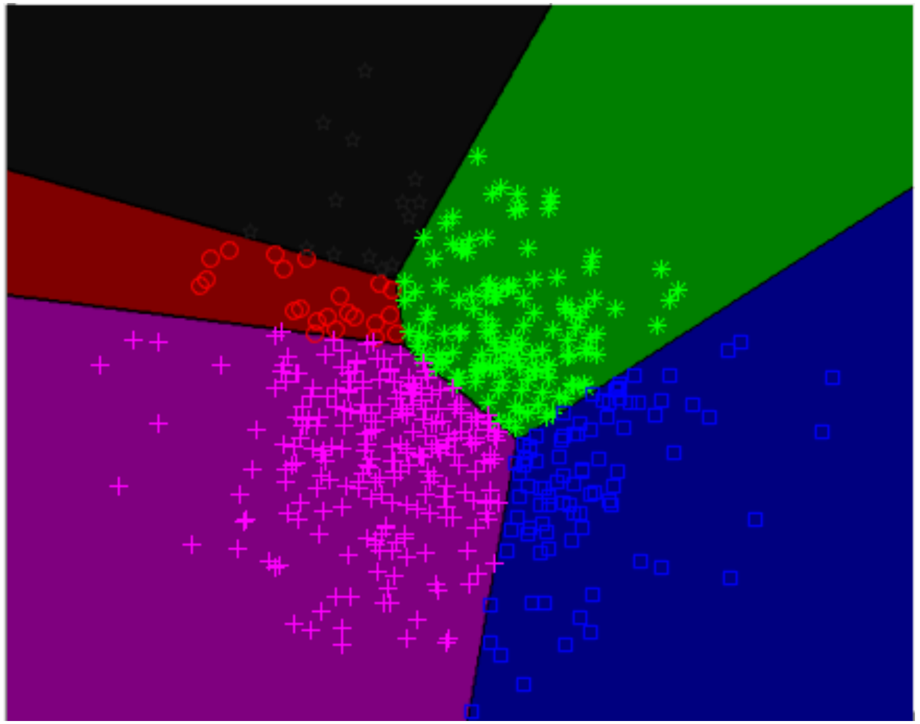
- Recall that a binary linear classifier splits space using a hyper-plane:



- Divides x_i space into 2 "half-spaces".

Shape of Decision Boundaries

- Multi-class linear classifier is intersection of these "half-spaces":
  - This divides the space into convex regions (like k-means):



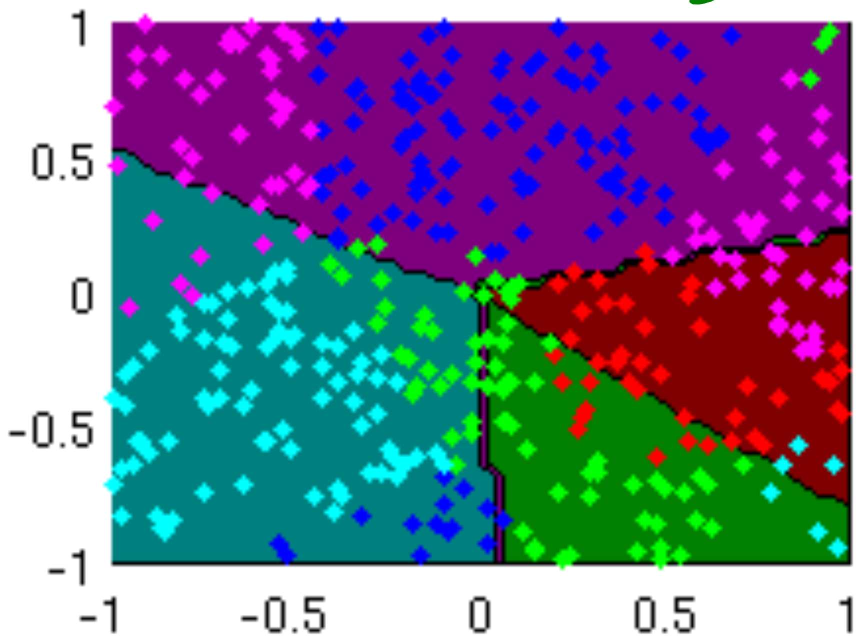
"Blue" region is region where we have:

$$w_{blue}^T x_i \geq w_{green}^T x_i$$
$$w_{blue}^T x_i \geq w_{magenta}^T x_i$$
$$w_{blue}^T x_i \geq w_{red}^T x_i$$
$$w_{blue}^T x_i \geq w_{black}^T x_i$$

Shape of Decision Boundaries

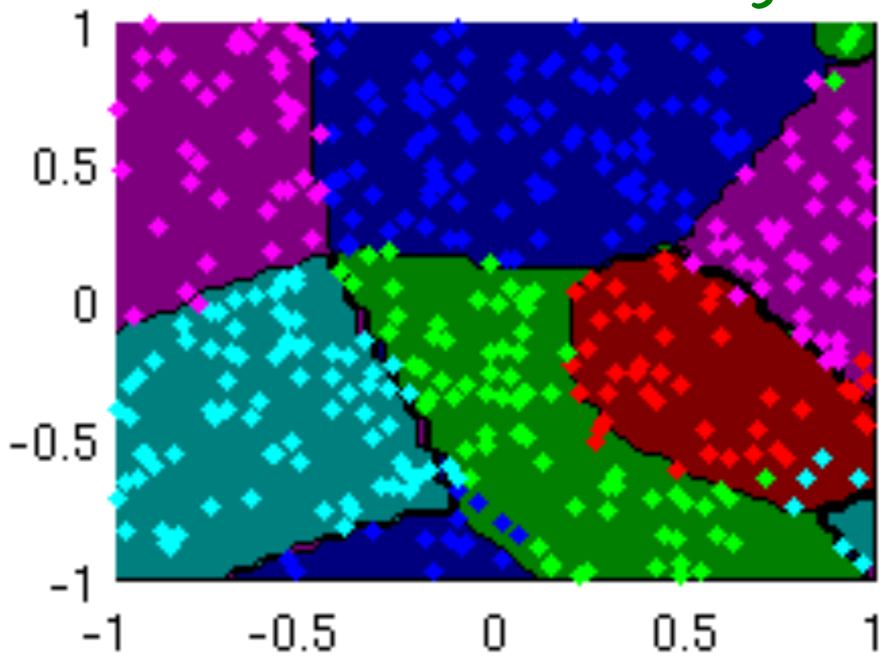
- Multi-class linear classifier is intersection of these "half-spaces":
  - Though regions could be non-convex with non-linear feature transforms:

Original Features



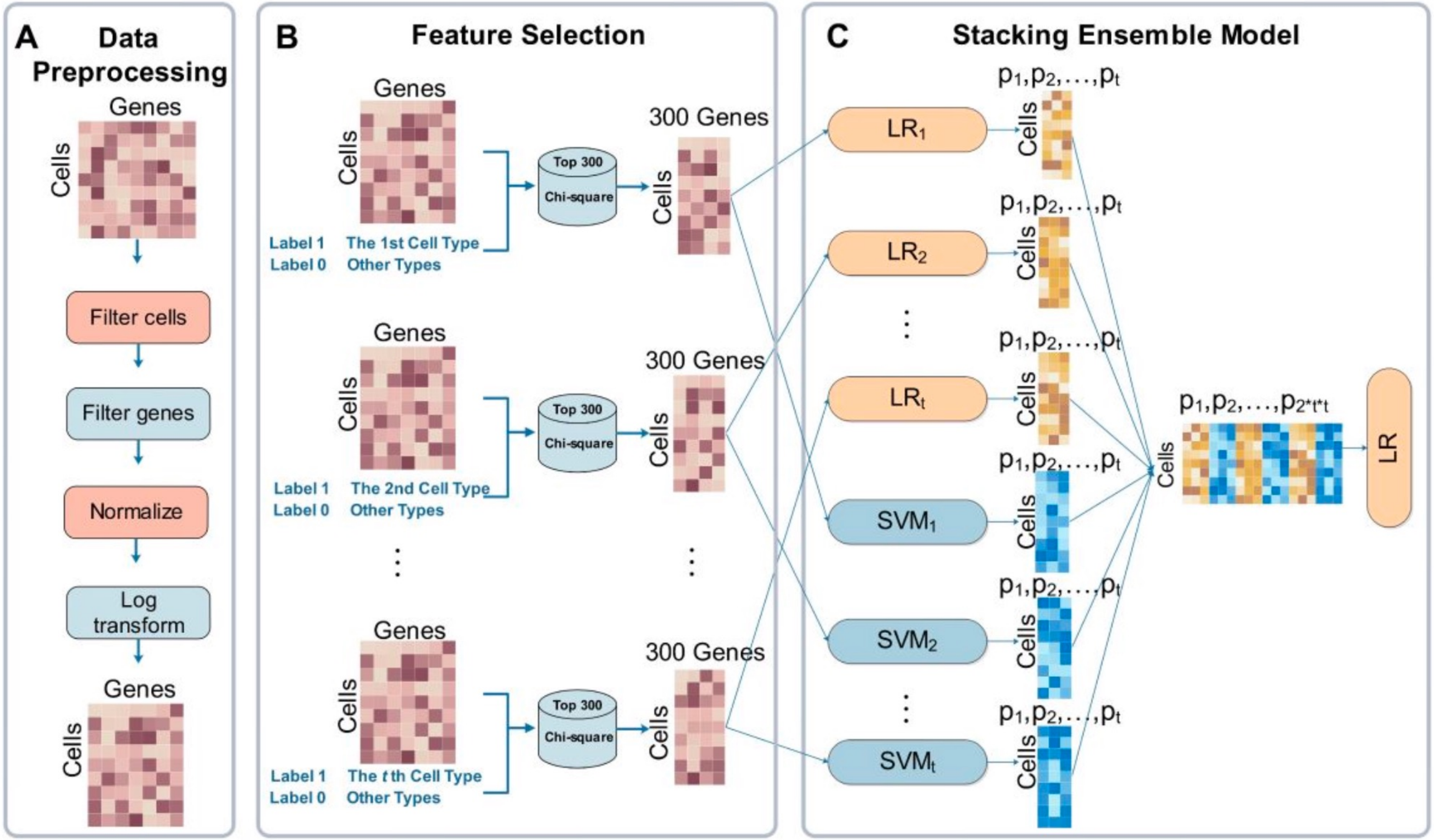
(onvex regions (not a good classifier)

Gaussian RBFs as Features



Non-convex regions (much better classifier on this data)

Example Applications



https://academic.oup.com/bioinformatics/article/40/2/btae063/7601321?login=false

Next Topic: Probabilistic Outputs

Previously: Identifying Important E-mails

- Recall problem of identifying ‘important’ e-mails:

- We can do binary classification by taking sign of linear model:

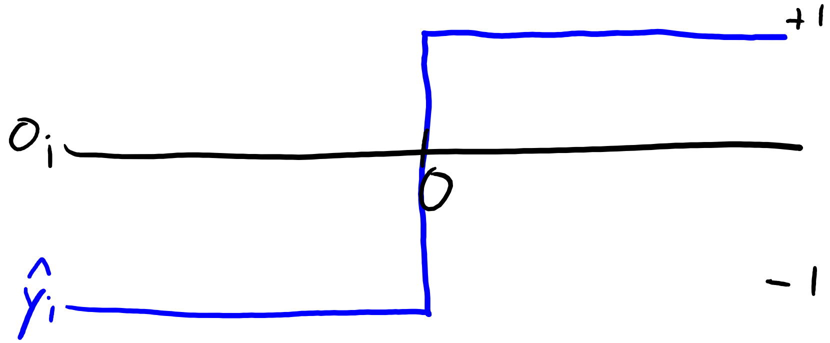
$$\hat{y}_i = \text{sign}(w^T x_i)$$

  – Convex loss functions (hinge/logistic loss) let us find an appropriate ‘w’.

- But what if we want a probabilistic classifier?
  – Want a model of $P(y_i = \text{“important”} \mid x_i)$ for use in decision theory.

Predictions vs. Probabilities

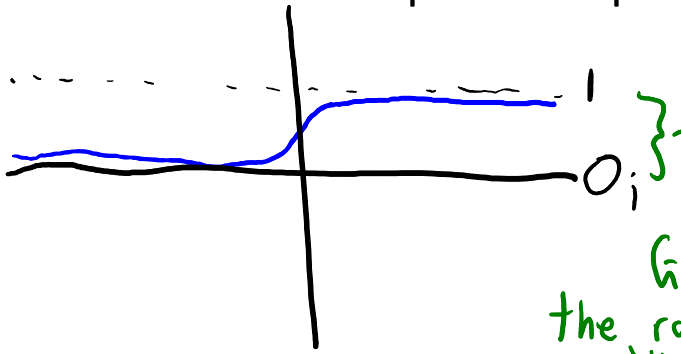
- With $o_i = w^T x_i$, linear classifiers make prediction using $\text{sign}(o_i)$:



- For predictions, "sign" maps from o_i to the elements {-1,+1}.
  - If o_i is positive we predict +1, if o_i negative we predict -1.
- For probabilities, we want to map from o_i to the range [0,1].
  - If o_i is very positive, we output a value close to +1 (confident y_i=1).
  - If o_i is very negative, we output a value close to 0 (confident y_i=-1).
  - If o_i is close to 0, we output a value close to 0.5 (classes equally likely).

Sigmoid Function

- So we want a transformation of $o_i = w^T x_i$ that looks like this:



Gives values in the range (0,1) that we'll interpret as probabilities

- The most common choice is the sigmoid function:

$$h(o_i) = \frac{1}{1 + \exp(-o_i)}$$

- Values of $h(o_i)$ match what we want:

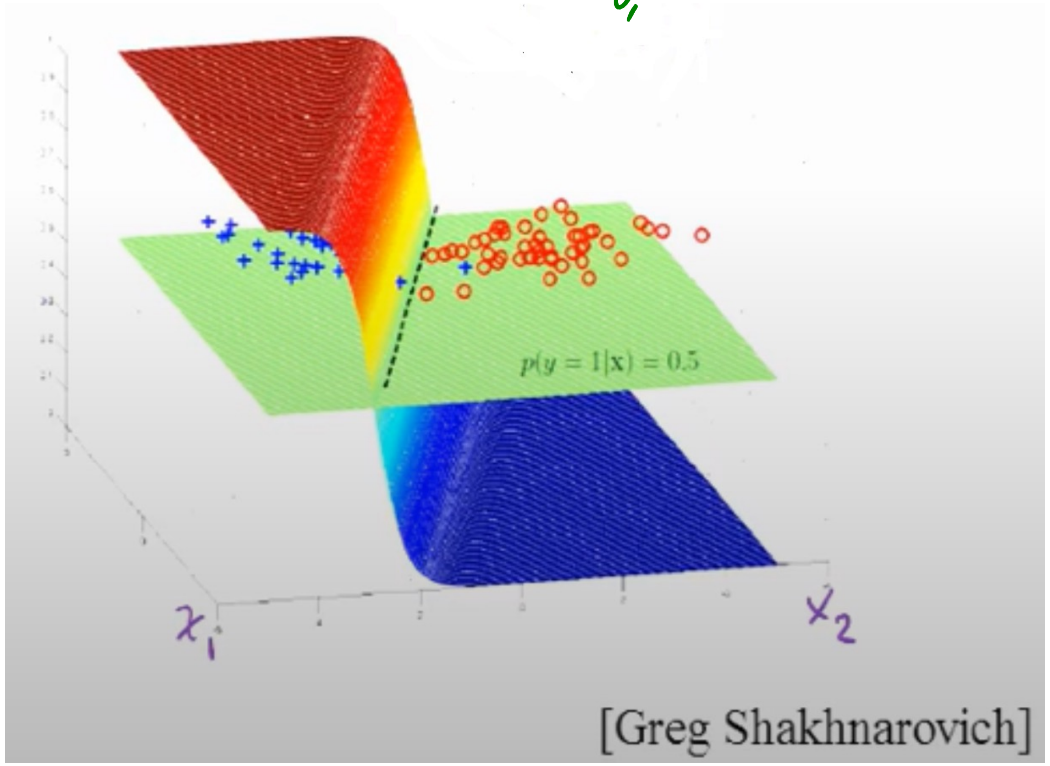
$h(-\infty) = 0$ $\quad h(-1) \simeq 0.27$ $\quad h(0) = 0.5$ $\quad h(0.5) \simeq 0.62$ $\quad h(+1) \simeq 0.73$ $\quad h(+\infty) = 1$

Probabilities for Linear Classifiers using Sigmoid

- Using sigmoid function, we output probabilities for linear models using:

$$p(y_i = 1 \mid w, x_i) = \frac{1}{1 + \exp(-\underbrace{w^T x_i}_{o_i})}$$

- Visualization for 2 features:



Probabilities for Linear Classifiers using Sigmoid

- Using sigmoid function, we output probabilities for linear models using:

$$p(y_i = 1 \mid w, x_i) = \frac{1}{1 + \exp(-\underbrace{w^T x_i}_{o_i})}$$

- By rules of probability:

$$p(y_i = -1 \mid w, x_i) = 1 - p(y_i = 1 \mid w, x_i)$$

$$= \frac{1}{1 + \exp(\underbrace{w^T x_i}_{-o_i})} \quad \text{(with some effort)}$$

- We then use these for "probability that e-mail is important".
- This may seem heuristic, but later we'll see that:
  - Minimizing logistic loss does "maximum likelihood estimation" in this model.

Softmax Function: Multi-Class Probabilities

- We have thus far considered the binary case:.
  - We start with an $o_i = w^T x_i$ in $(-\infty,\infty)$.
  - And we converted to probabilities in [0,1] using sigmoid(o_i).

$$o_i = -1.1 \Rightarrow p(y_i = 1 \mid o_i) = 0.25$$

- Now consider outputting probabilities in the multi-class case:
  - We have ℓ real numbers $o_{ic} = w_c^T x_i$ in $(-\infty,\infty)$.
  - We want to convert to ℓ numbers in [0,1] that sum to 1.

$$\begin{aligned} o_{i1} &= -0.1 \\ o_{i2} &= -0.8 \\ o_{i3} &= 0.9 \end{aligned} \Rightarrow \begin{aligned} p(y_i = 1 \mid o_{i1}, o_{i2}, o_{i3}) &= 0.24 \\ p(y_i = 2 \mid o_{i1}, o_{i2}, o_{i3}) &= 0.12 \\ p(y_i = 3 \mid o_{i1}, o_{i2}, o_{i3}) &= 0.64 \end{aligned}$$

Softmax Function: Multi-Class Probabilities

- Most common way to convert to probabilities is with softmax:

$$p(y_i = c \mid o_{i1}, o_{i2}, \dots, o_{i\ell}) \propto \exp(o_{ic})$$

  - Taking $\exp(o_{ic})$ makes it non-negative.
- To sum to one over value of s 'c', denominator sums over classes:

$$p(y_i = c \mid o_{i1}, o_{i2}, \dots, o_{i\ell}) = \frac{\exp(o_{ic})}{\sum_{c'=1}^{\ell} \exp(o_{ic'})}$$

  - So this gives a probability for each of the 'ℓ' possible values of 'c'.
    - Minimizing softmax loss does "maximum likelihood estimation" in this model.

Next Topic: Feature Engineering

Feature Engineering

- “…some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.”
  - Pedro Domingos

- “Coming up with features is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering.”
  - Andrew Ng

Feature Engineering

- Better features usually help more than a better model.

- Good features would ideally:
  - Allow learning with few examples, be hard to overfit with many examples.
  - Capture most important aspects of problem.
  - Reflects invariances (generalize to new scenarios).

- There is a trade-off between simple and expressive features:
  - With simple features overfitting risk is low, but accuracy might be low.
  - With complicated features accuracy can be high, but so is overfitting risk.

Feature Engineering

- The best features may be dependent on the model you use.

- For counting-based methods like naïve Bayes and decision trees:
  – Need to address coupon collecting, but separate relevant “groups”.

- For distance-based methods like KNN:
  – Want different class labels to be “far”.

- For regression-based methods like linear regression:
  – Want labels to have a linear dependency on features.

Discretization for Counting-Based Methods

- For counting-based methods:
  - Discretization: turn continuous into discrete.

Age
23
23
22
25
19
22

< 20	>= 20, < 25	>= 25
0	1	0
0	1	0
0	1	0
0	0	1
1	0	0
0	1	0

  - Counting age "groups" could let us learn more quickly than exact ages.
    - But we wouldn't do this for a distance-based method.

Standardization for Distance-Based Methods

- Consider features with different scales:

Egg (#)	Milk (mL)	Fish (g)	Pasta (cups)
0	250	0	1
1	250	200	1
0	0	0	0.5
2	250	150	0

- Should we convert to some standard 'unit'?
  - It doesn't matter for counting-based methods.
- It matters for distance-based methods:
  - KNN will focus on large values more than small values.
  - Often we "standardize" scales of different variables (e.g., convert everything to grams).
  - Also need to worry about correlated features

Non-Linear Transformations for Regression-Based

- Non-linear feature/label transforms can make things more linear:
  - Polynomial, exponential/logarithm, sines/cosines, RBFs.

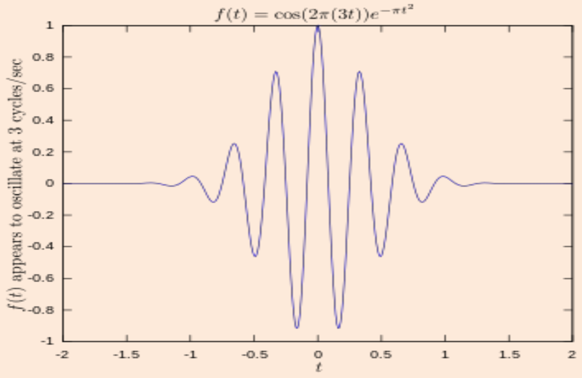
www.google.com/finance

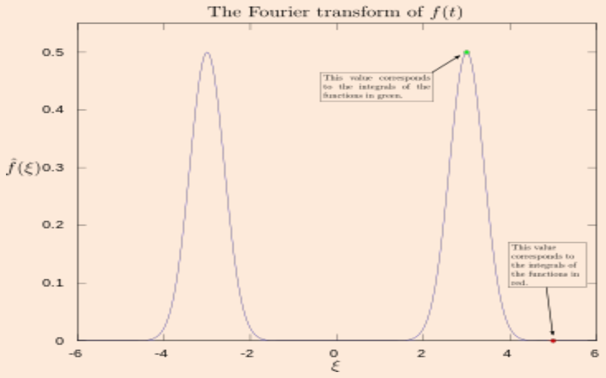
Discussion of Feature Engineering

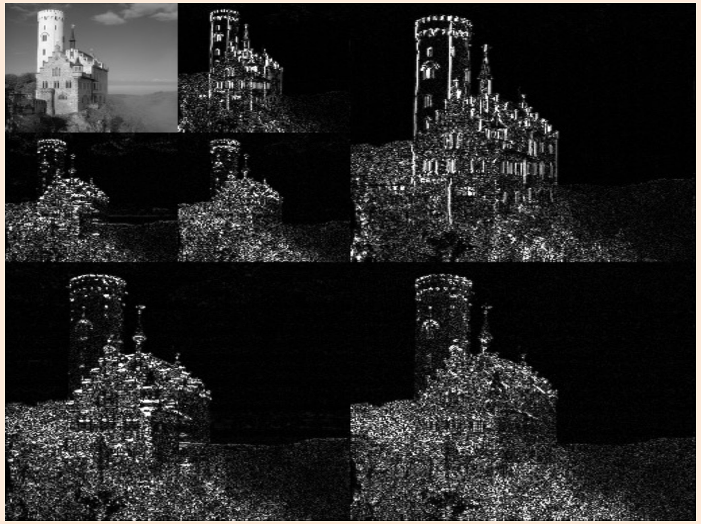
- The best feature transformations are application-dependent.
  - It’s hard to give general advice.
- Advice: ask the domain experts.
  - Often have idea of right discretization/standardization/transformation.
- If no domain expert, cross-validation will help.
  - Or if you have lots of data, use deep learning methods from Part 5.
- Next: we’ll discuss features used for text/image applications.

Domain-Specific Transformations

- In some domains there are natural transformations to do:
  - Fourier coefficients and spectrograms (sound data).
  - Wavelets (image data).
  - Convolutions (we'll talk about these soon).







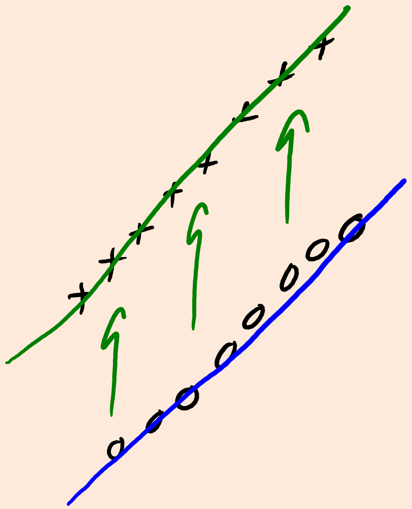
https://en.wikipedia.org/wiki/Fourier_transform
https://en.wikipedia.org/wiki/Spectrogram
https://en.wikipedia.org/wiki/Discrete_wavelet_transform

Digression: Linear Models with Binary Features

- What is the effect of a binary features on linear regression?

- Suppose we use a bag of words:
  - With 3 words {"hello", "Vicodin", "340"} our model would be:

$$\hat{y}_i = \underbrace{w_1 x_{i1}}_{\text{whether "hello" appears}} + w_2 x_{i2} + \underbrace{w_3 x_{i3}}_{\text{whether "340" appears}}$$



  - If e-mail only has "hello" and "340" our prediction is:

$$\hat{y}_i = \underbrace{w_1}_{\text{"hello" weight}} + \underbrace{w_3}_{\text{"340" weight}}$$

- So having the binary feature 'j' increases $\hat{y}_i$ by the fixed amount w_j.
  - Predictions are a bit like naïve Bayes where we combine features independently.
  - But now we're learning all w_j together so this tends to work better.

Next Topic: Features for Text Data

Text Example 1: Language Identification

- Consider data that doesn't look like this:

$$X = \begin{bmatrix} 0.5377 & 0.3188 & 3.5784 \\ 1.8339 & -1.3077 & 2.7694 \\ -2.2588 & -0.4336 & -1.3499 \\ 0.8622 & 0.3426 & 3.0349 \end{bmatrix}, \quad y = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix},$$

- But instead looks like this:

$$X = \begin{bmatrix} \text{Do you want to go for a drink sometime?} \\ \text{J'achète du pain tous les jours.} \\ \text{Fais ce que tu veux.} \\ \text{There are inner products between sentences?} \end{bmatrix}, y = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix}.$$

- How should we represent sentences using features?

A (Bad) Universal Representation

- Treat character in position ‘j’ of the sentence as a categorical feature.
  - “fais ce que tu veux” => x_i = [f a i s ‘’ c e ‘’ q u e ‘’ t u ‘’ v e u x .]
- “Pad” end of the sentence up to maximum #characters:
  - “fais ce que tu veux” => x_i = [f a i s ‘’ c e ‘’ q u e ‘’ t u ‘’ v e u x . γ γ γ γ γ γ γ γ …]
- Advantage:
  – No information is lost, KNN can eventually solve the problem.
- Disadvantage: throws out everything we know about language.
  – Needs to learn that “veux” starting from any position indicates “French”.
    - Doesn’t even use that sentences are made of words (this must be learned).
  – High overfitting risk, you will need a lot of examples for this easy task.

Bag of Words Representation

- Bag of words represents sentences/documents by word counts:

> The **International Conference on Machine Learning** (ICML) is the leading international academic conference in machine learning

ICML	International	Conference	Machine	Learning	Leading	Academic
1	2	2	2	2	1	1

- Bag of words loses a ton of information/meaning:
  - But it easily solves language identification problem

Universal Representation vs. Bag of Words

- Why is bag of words better than "string of characters" here?
  - It needs less data because it captures invariances for the task:
    - Most features give strong indication of one language or the other.
    - It doesn't matter *where* the French words appear.
  - It overfits less because it throws away irrelevant information.
    - Exact sequence of words isn't particularly relevant here.

Text Example 2: Word Sense Disambiguation

- Consider the following two sentences:
  – “The cat ran after the mouse.”
  – “Move the mouse cursor to the File menu.”

- Word sense disambiguation (WSD): classify “meaning” of a word:
  – A surprisingly difficult task.

- You can do ok with bag of words, but it will have problems:
  – “Her mouse clicked on one cat video after another.”
  – “We saw the mouse run out from behind the computer.”
  – “The mouse was gray.” (ambiguous without more context)

Bigrams and Trigrams

- A bigram is an ordered set of two words:
  - Like “computer mouse” or “mouse ran”.
- A trigram is an ordered set of three words:
  - Like “cat and mouse” or “clicked mouse on”.

- These give more context/meaning than bag of words:
  - Includes neighbouring words as well as order of words.
  - Trigrams are widely-used for various language tasks.

- General case is called n-gram.
  - Unfortunately, coupon collecting becomes a problem with larger ‘n’.

Text Example 3: Part of Speech (POS) Tagging

- Consider problem of finding the verb in a sentence:
  - "The 340 students jumped at the chance to hear about POS features."

- Part of speech (POS) tagging is the problem of labeling all words.
  - >40 common syntactic POS tags.
  - Current systems have ~97% accuracy on standard ("clean") test sets.
  - You can achieve this by applying a "word-level" classifier to each word.
    - That independently classifies each word with one of the 40 tags.

- What features of a word should we use for POS tagging?

POS Features

- Regularized multi-class logistic regression with these features gives ~97% accuracy:
  - Categorical features whose domain is all words ("lexical" features):
    - The word (e.g., "jumped" is usually a verb).
    - The previous word (e.g., "he" hit vs. "a" hit).
    - The previous previous word.
    - The next word.
    - The next next word.
  - Categorical features whose domain is combinations of letters ("stem" features):
    - Prefix of length 1 ("what letter does the word start with?")
    - Prefix of length 2.
    - Prefix of length 3.
    - Prefix of length 4 ("does it start with JUMP?")
    - Suffix of length 1.
    - Suffix of length 2.
    - Suffix of length 3 ("does it end in ING?")
    - Suffix of length 4.
  - Binary features ("shape" features):
    - Does word contain a number?
    - Does word contain a capital?
    - Does word contain a hyphen?

well-dressed

$\text{prefix}(w_i) = \texttt{w}$

$\text{prefix}(w_i) = \texttt{we}$

$\text{prefix}(w_i) = \texttt{wel}$

$\text{prefix}(w_i) = \texttt{well}$

$\text{suffix}(w_i) = \texttt{ssed}$

$\text{suffix}(w_i) = \texttt{sed}$

$\text{suffix}(w_i) = \texttt{ed}$

$\text{suffix}(w_i) = \texttt{d}$

$\text{has-hyphen}(w_i)$

$\text{word-shape}(w_i) = \texttt{xxxx-xxxxxxx}$

$\text{short-word-shape}(w_i) = \texttt{x-x}$

Ordinal Features

- Categorical features with an ordering are called ordinal features.

Rating
Bad
Very Good
Good
Good
Very Bad
Good
Medium

Rating
2
5
4
4
1
4
3

- If using decision trees, makes sense to replace with numbers.
  - Captures ordering between the ratings.
  - A rule like (rating ≥ 3) means (rating ≥ Good), which make sense.

Ordinal Features

- With linear models, "convert to number" assumes ratings are equally spaced.
  - "Bad" and "Medium" distance is similar to "Good" and "Very Good" distance.
- One alternative that preserves ordering with binary features:

Rating
Bad
Very Good
Good
Good
Very Bad
Good
Medium

≥ Bad	≥ Medium	≥ Good	Very Good
1	0	0	0
1	1	1	1
1	1	1	0
1	1	1	0
0	0	0	0
1	1	1	0
1	1	0	0

- Regression weight w_{medium} represents:
  - "How much medium changes prediction over bad".
- Bonus slides discuss "cyclic" features like "time of day".

Next Topic: Personalized Features

Motivation: “Personalized” Important E-mails

- Features: bag of words, trigrams, regular expressions, and so on.

- There might be some “globally” important messages:
  - “This is your mother, something terrible happened, give me a call ASAP.”

- But your “important” message may be unimportant to others.
  - Similar for spam: “spam” for one user could be “not spam” for another.

“Global” and “Local” Features

- Consider the following weird feature transformation:

“340”
1
1
1
0
1

“340” (any user)	“340” (user?)
1	User 1
1	User 1
1	User 2
0	<no “340”>
1	User 3

- First feature: did “340” appear in this e-mail?
- Second feature: if “340” appeared in this e-mail, who was it addressed to?

- First feature will increase/decrease importance of “340” for every user (including new users).
- Second (categorical feature) increases/decreases importance of “340” for a specific user.
  - Lets us learn more about specific users where we have a lot of data

"Global" and "Local" Features

- Recall we usually represent categorical features using "1 of k" binaries:

"340"
1
1
1
0
1

$\Rightarrow$

"340" (any user)	"340" (user = 1)	"340" (user = 2)
1	1	0
1	1	0
1	0	1
0	0	0
1	0	0

- First feature "moves the line up" for all users.
- Second feature "moves the line up" when the e-mail is to user 1.
- Third feature "moves the line up" when the e-mail is to user 2.

The Big Global/Local Feature Table for E-mails

- Each row is one e-mail (there are lots of rows):

$$X = \begin{bmatrix} \quad \end{bmatrix} \qquad y = \begin{bmatrix} \text{"important"} \\ \text{"not important"} \\ \vdots \end{bmatrix}$$

"global" features: shared by all users

"local" features for user "1": set to 0 for all other users.

"local" features for user "2"

We only need to store the user ID and list of non-zero features.

Predicting Importance of E-mail For New User

- Consider a new user:
  - We start out with no information about them.
    - We initialize local weights w_u to zero (so they have not effect new users).
  - So we use global features to predict what is important to a generic user.

$$\hat{y}_i = \text{sign}(\underbrace{w_g^T x_{ig}}_{\text{features/weights shared across users.}})$$

- With more data, update global features and user's local features:
  - Local features make prediction *personalized*.

$$\hat{y}_i = \text{sign}(w_g^T x_{ig} + \underbrace{w_u^T x_{iu}}_{\text{features/weights specific to user.}})$$

  - What is important to *this* user?
    - Global weight for "Bitcoin" might be negative, but local weight is positive for some users.
- G-mail system: classification with logistic regression.
  - Trained with a variant of stochastic gradient descent (later).

Summary

- Sigmoid function turns binary linear predictions into probabilities.
  - Softmax functions turns multi-class linear predictions into probabilities.
- Feature engineering can be a key factor affecting performance.
  - Good features depend on the task and the model.
- Bag of words: not a good representation in general.
  - But good features if word order isn’t needed to solve problem.
- Universal text representation: also not a good general representation.
  - But can solve any problem if you have enough data.
- Text features (beyond bag of words): trigrams, lexical, stem, shape.
  - Try to capture important invariances in text data.
- Global vs. local features allow “personalized” predictions.

- Next time:
  - A trick that lets you find gold and use the polynomial basis with $d > 1$.

“All-Pairs” and ECOC Classification

- Alternative to “one vs. all” to convert binary classifier to multi-class is “all pairs”.
  - For each pair of labels ‘c’ and ‘d’, fit a classifier that predicts +1 for examples of class ‘c’ and -1 for examples of class ‘d’ (so each classifier only trains on examples from two classes).
  - To make prediction, take a vote of how many of the (k-1) classifiers for class ‘c’ predict +1.
  - Often works better than “one vs. all”, but not so fun for large ‘k’.
    - Need $O(k^2)$ classifiers.
- A variation on this is using “error correcting output codes” from information theory (see Math 342).
  - Each classifier trains to predict +1 for some of the classes and -1 for others.
  - You setup the +1/-1 code so that it has an “error correcting” property.
    - It will make the right decision even if some of the classifiers are wrong.

Motivation: Dog Image Classification

- Suppose we're classifying images of dogs into breeds:

- What if we have images where class label isn't obvious?
  – Syberian husky vs. Inuit dog?

https://www.slideshare.net/angjoo/dog-breed-classification-using-part-localization
https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements

Learning with Preferences

- Do we need to throw out images where label is ambiguous?
  - We don't have the y_i.

  - We want classifier to prefer Syberian husky over bulldog, Chihuahua, etc.
    - Even though we don't know if these are Syberian huskies or Inuit dogs.
  - Can we design a loss that enforces preferences rather than "true" labels?

https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements

Learning with Pairwise Preferences (Ranking)

- Instead of y_i, we're given list of (c_1,c_2) preferences for each 'i':

We want $w_{c_1}^T x_i > w_{c_2}^T x_i$ for these particular (c_1, c_2) values

- Multi-class classification is special case of choosing (y_i,c) for all 'c'.

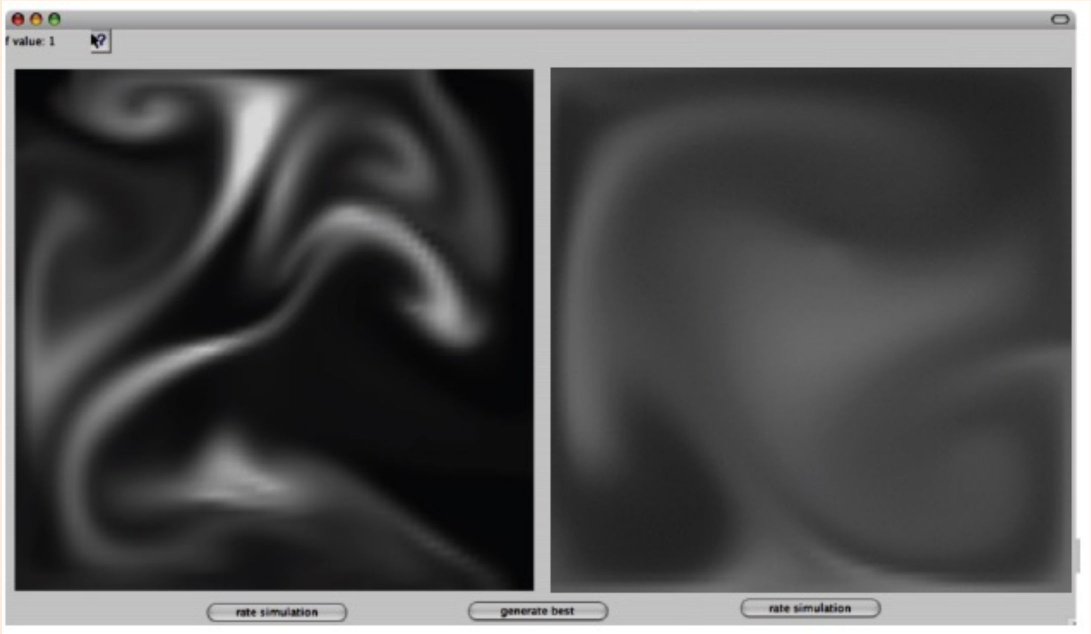
- By following the earlier steps, we can get objectives for this setting:

$$\sum_{i=1}^{n} \underbrace{\sum_{(c_1,c_2)} \max\{0, 1 - w_{c_1}^T x_i + w_{c_2}^T x_i\}}_{\text{"sum" version of multi-class SVM}} + \frac{\lambda}{2}\|W\|_F^2$$

https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements

Learning with Pairwise Preferences (Ranking)

- Pairwise preferences for computer graphics:
  - We have a smoke simulator, with several parameters:

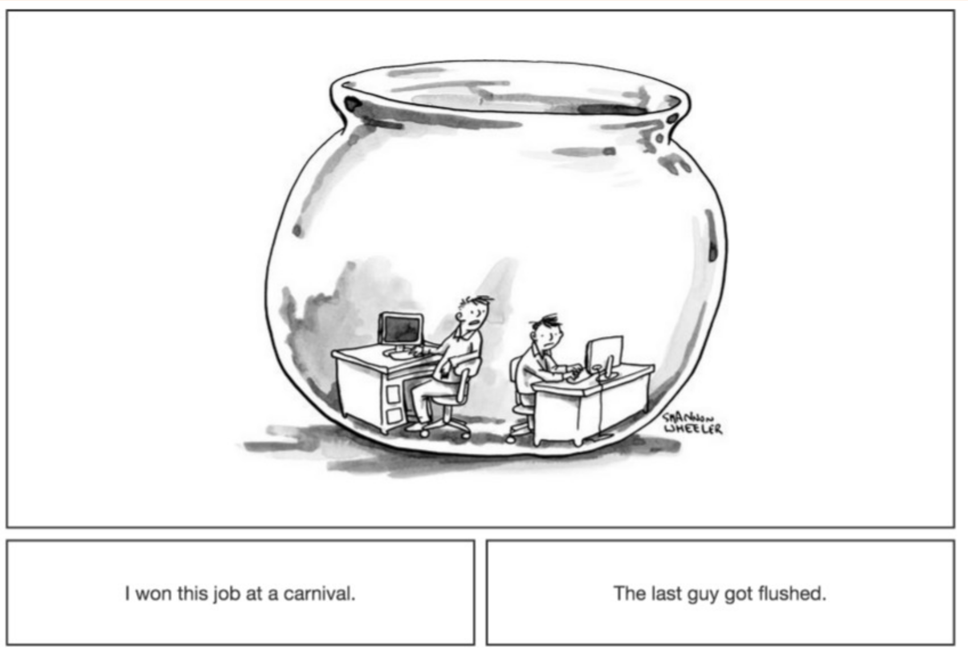


  - Don't know what the optimal parameters are, but we can ask the artist:
    - "Which one looks more like smoke"?

https://circle.ubc.ca/bitstream/handle/2429/30519/ubc_2011_spring_brochu_eric.pdf?sequence=3

Learning with Pairwise Preferences (Ranking)

- Pairwise preferences for humour:
  – New Yorker caption contest:



  – "Which one is funnier"?

https://homes.cs.washington.edu/~jamieson/resources/next.pdf

Risk Scores

- In medicine/law/finance, risk scores are sometimes used to give probabilities:

1.	**C**ongestive Heart Failure	1 point	…
2.	**H**ypertension	1 point	+ …
3.	**A**ge ≥ 75	1 point	+ …
4.	**D**iabetes Mellitus	1 point	+ …
5.	Prior **S**troke or Transient Ischemic Attack	**2** points	+
		SCORE	=

SCORE	0	1	2	3	4	5	6
RISK	1.9%	2.8%	4.0%	5.9%	8.5%	12.5%	18.2%

Figure 1: CHADS$_2$ risk score of Gage et al. (2001) to assess stroke risk (see www.mdcalc.com for other medical scoring systems). The variables and points of this model were determined by a panel of experts, and the risk estimates were computed empirically from data.

  – Get integer-valued "points" for each "risk factor", and probability is computed from data based on people with same number of points.
  – Less accurate than fancy models, but interpretable and can be done by hand.
    - Some work on trying to "learn" the whole thing (like doing feature selection then rounding).

https://arxiv.org/pdf/1610.00168.pdf