Natural Language Processing

Classification I
Dan Klein – UC Berkeley
Classification
Classification

- **Automatically make a decision about inputs**
  - Example: document → category
  - Example: image of digit → digit
  - Example: image of object → object type
  - Example: query + webpages → best match
  - Example: symptoms → diagnosis
  - ...

- **Three main ideas**
  - Representation as feature vectors / kernel functions
  - Scoring by linear functions
  - Learning by optimization
Some Definitions

INPUTS

$X_i$

close the ____

CANDIDATE SET

$\mathcal{Y}(x)$

{door, table, ...}

CANDIDATES

$y$

table

TRUE OUTPUTS

$y^*$

door

FEATURE VECTORS

$f(x, y) \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

$x_1 = \text{"the"} \land y = \text{"door"}$

"close" in $x \land y = \text{"door"}$

$x_1 = \text{"the"} \land y = \text{"table"}$

y occurs in $x$
Features
Feature Vectors

- Example: web page ranking (not actually classification)

\[ x_i = \text{“Apple Computers”} \]

\[
\mathbf{f}_i(x_i) = [0.3 \ 5 \ 0 \ 0 \ \ldots]
\]

\[
\mathbf{f}_i(x_i) = [0.8 \ 4 \ 2 \ 1 \ \ldots]
\]
Sometimes, we think of the input as having features, which are multiplied by outputs to form the candidates

\[
x \quad \text{... win the election ...}
\]

\[
\text{“f(x)”} \quad \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}
\]

\[
\text{“win”} \quad \text{“election”}
\]

\[
f(\text{SPORTS}) = [1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]
\]

\[
f(\text{POLITICS}) = [0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]
\]

\[
f(\text{OTHER}) = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0]
\]
Non-Block Feature Vectors

- Sometimes the features of candidates cannot be decomposed in this regular way
- Example: a parse tree’s features may be the productions present in the tree

\[
f(\text{NP VP}) = [1 \ 0 \ 1 \ 0 \ 1]
\]

\[
f(\text{NP VP}) = [1 \ 1 \ 0 \ 1 \ 0]
\]

- Different candidates will thus often share features
- We’ll return to the non-block case later
Linear Models
Linear Models: Scoring

- In a linear model, each feature gets a weight $w$

$$f(POLITICS) = [0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$

$$f(SPORTS) = [1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$

$$w = [1\ 1\ -1\ -2\ 1\ -1\ 1\ -2\ -2\ -1\ -1\ 1]$$

- We score hypotheses by multiplying features and weights:

$$score(y, w) = w^T f(y)$$

$$f(POLITICS) = [0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$

$$w = [1\ 1\ -1\ -2\ 1\ -1\ 1\ -2\ -2\ -1\ -1\ 1]$$

$$score(POLITICS, w) = 1 \times 1 + 1 \times 1 = 2$$
Linear Models: Decision Rule

- The linear decision rule:

\[
prediction(... \text{win the election} ..., \mathbf{w}) = \arg \max_{y \in \mathcal{Y}(x)} \mathbf{w}^\top f(y)
\]

\[
score(\text{SPORTS}, \mathbf{w}) = 1 \times 1 + (-1) \times 1 = 0
\]

\[
score(\text{POLITICS}, \mathbf{w}) = 1 \times 1 + 1 \times 1 = 2
\]

\[
score(\text{OTHER}, \mathbf{w}) = (-2) \times 1 + (-1) \times 1 = -3
\]

\[
prediction(... \text{win the election} ..., \mathbf{w}) = \text{POLITICS}
\]

- We’ve said nothing about where weights come from
Binary Classification

- Important special case: binary classification
  - Classes are $y=+1/-1$
    
    $$f(x, -1) = -f(x, +1)$$
    $$f(x) = 2f(x, +1)$$
  - Decision boundary is a hyperplane
    $$\mathbf{w}^\top f(x) = 0$$

**Diagram:**
- $\mathbf{w}$
  - BIAS : -3
  - free : 4
  - money : 2

+1 = SPAM
-1 = HAM

$$\mathbf{w}^\top f = 0$$
Multiclass Decision Rule

- If more than two classes:
  - Highest score wins
  - Boundaries are more complex
  - Harder to visualize

\[ \text{prediction}(x_i, w) = \arg \max_{y \in \mathcal{Y}} w^T f_i(y) \]

- There are other ways: e.g. reconcile pairwise decisions
Learning
Learning Classifier Weights

- Two broad approaches to learning weights

- Generative: work with a probabilistic model of the data, weights are (log) local conditional probabilities
  - Advantages: learning weights is easy, smoothing is well-understood, backed by understanding of modeling

- Discriminative: set weights based on some error-related criterion
  - Advantages: error-driven, often weights which are good for classification aren’t the ones which best describe the data

- We’ll mainly talk about the latter for now
How to pick weights?

- **Goal:** choose “best” vector \( w \) given training data
  - For now, we mean “best for classification”

- **The ideal:** the weights which have greatest test set accuracy / F1 / whatever
  - But, don’t have the test set
  - Must compute weights from training set

- **Maybe we want weights which give best training set accuracy?**
  - Hard discontinuous optimization problem
  - May not (does not) generalize to test set
  - Easy to overfit

*Though, min-error training for MT does exactly this.*
Minimize Training Error?

- A loss function declares how costly each mistake is

\[ \ell_i(y) = \ell(y, y_i^*) \]

- E.g. 0 loss for correct label, 1 loss for wrong label
- Can weight mistakes differently (e.g. false positives worse than false negatives or Hamming distance over structured labels)

- We could, in principle, minimize training loss:

\[ \min_w \sum_i \ell_i \left( \arg \max_y w^\top f_i(y) \right) \]

- This is a hard, discontinuous optimization problem
Linear Models: Perceptron

- **The perceptron algorithm**
  - Iteratively processes the training set, reacting to training errors
  - Can be thought of as trying to drive down training error

- **The (online) perceptron algorithm:**
  - Start with zero weights \( w \)
  - Visit training instances one by one
    - Try to classify
      \[
      \hat{y} = \arg \max_{y \in Y(x)} w^\top f(y)
      \]
    - If correct, no change!
    - If wrong: adjust weights
      \[
      w \leftarrow w + f(y_i^*) \quad \text{and} \quad w \leftarrow w - f(\hat{y})
      \]
Example: “Best” Web Page

\[
w = [1 \quad 2 \quad 0 \quad 0 \quad \ldots]
\]

\[
x_i = “Apple Computers”
\]

\[
f_i( ) = [0.3 \quad 5 \quad 0 \quad 0 \quad \ldots] \quad w^\top f = 10.3 \quad \hat{y}
\]

\[
f_i( ) = [0.8 \quad 4 \quad 2 \quad 1 \quad \ldots] \quad w^\top f = 8.8 \quad y_i^*
\]

\[
w \leftarrow w + f(y_i^*) - f(\hat{y})
\]

\[
w = [1.5 \quad 1 \quad 2 \quad 1 \quad \ldots]
\]
Examples: Perceptron

- Separable Case
A data set is separable if some parameters classify it perfectly.

Convergence: if training data separable, perceptron will separate (binary case).

Mistake Bound: the maximum number of mistakes (binary case) related to the margin or degree of separability.
Examples: Perceptron

- Non-Separable Case
Issues with Perceptrons

- **Overtraining**: test / held-out accuracy usually rises, then falls
  - Overtraining isn’t the typically discussed source of overfitting, but it can be important

- **Regularization**: if the data isn’t separable, weights often thrash around
  - Averaging weight vectors over time can help (averaged perceptron)
  - [Freund & Schapire 99, Collins 02]

- **Mediocre generalization**: finds a “barely” separating solution
Problems with Perceptrons

- Perceptron “goal”: separate the training data

\[ \forall i, \forall y \neq y^i \quad w^T f_i(y^i) \geq w^T f_i(y) \]

1. This may be an entire feasible space
2. Or it may be impossible
Margin
Objective Functions

- What do we want from our weights?
  - Depends!
  - So far: minimize (training) errors:

\[
\sum_i \text{step} \left( w^T f_i(y_i^*) - \max_{y \neq y_i^*} w^T f_i(y) \right)
\]

- This is the “zero-one loss”
  - Discontinuous, minimizing is NP-complete
  - Not really what we want anyway

- Maximum entropy and SVMs have other objectives related to zero-one loss
Linear Separators

Which of these linear separators is optimal?
Classification Margin (Binary)

- Distance of $x_i$ to separator is its margin, $m_i$
- Examples closest to the hyperplane are support vectors
- Margin $\gamma$ of the separator is the minimum $m$
Classification Margin

- For each example \( x_i \) and possible mistaken candidate \( y \), we avoid that mistake by a margin \( m_i(y) \) (with zero-one loss)

\[
m_i(y) = w^T f_i(y^*_i) - w^T f_i(y)
\]

- Margin \( \gamma \) of the entire separator is the minimum \( m \)

\[
\gamma = \min_i \left( w^T f_i(y^*_i) - \max_{y \neq y^*_i} w^T f_i(y) \right)
\]

- It is also the largest \( \gamma \) for which the following constraints hold

\[
\forall i, \forall y \quad w^T f_i(y^*_i) \geq w^T f_i(y) + \gamma \ell_i(y)
\]
Separable SVMs: find the max-margin $w$

$$\max_{||w||=1} \gamma$$

$$\ell_i(y) = \begin{cases} 0 & \text{if } y = y_i^* \\ 1 & \text{if } y \neq y_i^* \end{cases}$$

$$\forall i, \forall y \quad w^\top f_i(y_i^*) \geq w^\top f_i(y) + \gamma \ell_i(y)$$

- Can stick this into Matlab and (slowly) get an SVM
- Won’t work (well) if non-separable
Why Max Margin?

- **Why do this? Various arguments:**
  - Solution depends only on the boundary cases, or *support vectors* (but remember how this diagram is broken!)
  - Solution robust to movement of support vectors
  - Sparse solutions (features not in support vectors get zero weight)
  - Generalization bound arguments
  - Works well in practice for many problems

Support vectors
Max Margin / Small Norm

- **Reformulation:** find the smallest $w$ which separates data

Remember this condition?

\[
\max_{\|w\|=1} \gamma \\
\forall i, y \quad w^T f_i(y_i^*) \geq w^T f_i(y) + \gamma \ell_i(y)
\]

- $\gamma$ scales linearly in $w$, so if $\|w\|$ isn’t constrained, we can take any separating $w$ and scale up our margin

\[
\gamma = \min_{i, y \neq y_i^*} \frac{[w^T f_i(y_i^*) - w^T f_i(y)]}{\ell_i(y)}
\]

- Instead of fixing the scale of $w$, we can fix $\gamma = 1$

\[
\min_w \frac{1}{2} \|w\|^2 \\
\forall i, y \quad w^T f_i(y_i^*) \geq w^T f_i(y) + 1 \ell_i(y)
\]
Soft Margin Classification

- What if the training set is not linearly separable?
- *Slack variables* $\xi_i$ can be added to allow misclassification of difficult or noisy examples, resulting in a *soft margin* classifier.
Maximum Margin

- **Non-separable SVMs**
  - Add slack to the constraints
  - Make objective pay (linearly) for slack:

\[
\min_{\mathbf{w}, \xi} \frac{1}{2} ||\mathbf{w}||^2 + C \sum_i \xi_i
\]

\[
\forall i, y, \quad \mathbf{w}^T \mathbf{f}_i(y^*_i) + \xi_i \geq \mathbf{w}^T \mathbf{f}_i(y) + \ell_i(y)
\]

- C is called the *capacity* of the SVM – the smoothing knob

- **Learning:**
  - Can still stick this into Matlab if you want
  - Constrained optimization is hard; better methods!
  - We’ll come back to this later

*Note: exist other choices of how to penalize slacks!*
Maximum Margin
Likelihood
Linear Models: Maximum Entropy

- **Maximum entropy (logistic regression)**
  - Use the scores as probabilities:
    \[
    P(y|x, w) = \frac{\exp(w^T f(y))}{\sum_{y'} \exp(w^T f(y'))}
    \]
  - Maximize the (log) conditional likelihood of training data
    \[
    L(w) = \log \prod_i P(y^*_i|x_i, w) = \sum_i \log \left( \frac{\exp(w^T f_i(y^*_i))}{\sum_y \exp(w^T f_i(y))} \right)
    \]
    \[
    = \sum_i \left( w^T f_i(y^*_i) - \log \sum_y \exp(w^T f_i(y)) \right)
    \]
Maximum Entropy II

- **Motivation for maximum entropy:**
  - Connection to maximum entropy principle (sort of)
  - Might want to do a good job of being uncertain on noisy cases...
  - ... in practice, though, posteriors are pretty peaked

- **Regularization (smoothing)**

\[
\max_w \sum_i \left( w^T f_i(y_i^*) - \log \sum_y \exp(w^T f_i(y)) \right) - k ||w||^2
\]

\[
\min_w k ||w||^2 - \sum_i \left( w^T f_i(y_i^*) - \log \sum_y \exp(w^T f_i(y)) \right)
\]
Maximum Entropy
Loss Comparison
Log-Loss

- If we view maxent as a minimization problem:

\[
\min_w k||w||^2 + \sum_i - \left( w^T f_i(y_i^*) - \log \sum_y \exp(w^T f_i(y)) \right)
\]

- This minimizes the “log loss” on each example

\[
- \left( w^T f_i(y_i^*) - \log \sum_y \exp(w^T f_i(y)) \right) = -\log P(y_i^*|x_i, w)
\]

- One view: log loss is an upper bound on zero-one loss
Remember SVMs...

- We had a constrained minimization
  \[
  \min_{\mathbf{w}, \xi} \frac{1}{2}||\mathbf{w}||^2 + C \sum_i \xi_i
  \]
  \[
  \forall i, y, \quad \mathbf{w}^T \mathbf{f}_i(y^*_i) + \xi_i \geq \mathbf{w}^T \mathbf{f}_i(y) + \ell_i(y)
  \]

- ...but we can solve for \( \xi_i \)
  \[
  \forall i, y, \quad \xi_i \geq \mathbf{w}^T \mathbf{f}_i(y) + \ell_i(y) - \mathbf{w}^T \mathbf{f}_i(y^*_i)
  \]
  \[
  \forall i, \quad \xi_i = \max_y \left( \mathbf{w}^T \mathbf{f}_i(y) + \ell_i(y) \right) - \mathbf{w}^T \mathbf{f}_i(y^*_i)
  \]

- Giving
  \[
  \min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C \sum_i \left( \max_y \left( \mathbf{w}^T \mathbf{f}_i(y) + \ell_i(y) \right) - \mathbf{w}^T \mathbf{f}_i(y^*_i) \right)
  \]
Hinge Loss

- Consider the per-instance objective:

$$\min \ k ||w||^2 + \sum_i \left( \max_y \left( w^T f_i(y) + \ell_i(y) \right) - w^T f_i(y_i^*) \right)$$

- This is called the “hinge loss”
  - Unlike maxent / log loss, you stop gaining objective once the true label wins by enough
  - You can start from here and derive the SVM objective
  - Can solve directly with sub-gradient decent (e.g. Pegasos: Shalev-Shwartz et al 07)
Max vs “Soft-Max” Margin

- **SVMs:**
  \[
  \min_w k\|w\|^2 - \sum_i \left( w^T f_i(y_i^*) - \max_y \left( w^T f_i(y) + \ell_i(y) \right) \right)
  \]
  You can make this zero

- **Maxent:**
  \[
  \min_w k\|w\|^2 - \sum_i \left( w^T f_i(y_i^*) - \log \sum_y \exp \left( w^T f_i(y) \right) \right)
  \]
  ... but not this one

- **Very similar! Both try to make the true score better than a function of the other scores**
  - The SVM tries to beat the augmented runner-up
  - The Maxent classifier tries to beat the “soft-max"
Loss Functions: Comparison

- **Zero-One Loss**
  \[ \sum_i \text{step} \left( \mathbf{w}^T \mathbf{f}_i(y_i^*) - \max_{y \neq y_i^*} \mathbf{w}^T \mathbf{f}_i(y) \right) \]

- **Hinge**
  \[ \sum_i \left( \mathbf{w}^T \mathbf{f}_i(y_i^*) - \max \left( \mathbf{w}^T \mathbf{f}_i(y) + \ell_i(y) \right) \right) \]

- **Log**
  \[ \sum_i \left( \mathbf{w}^T \mathbf{f}_i(y_i^*) - \log \sum_y \exp \left( \mathbf{w}^T \mathbf{f}_i(y) \right) \right) \]
Separators: Comparison
Conditional vs Joint Likelihood
Example: Sensors

**Reality**

<table>
<thead>
<tr>
<th>Raining</th>
<th>Sunny</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Cylinder" /></td>
<td><img src="image2.png" alt="Cylinder" /></td>
</tr>
<tr>
<td><img src="image3.png" alt="Cylinder" /></td>
<td><img src="image4.png" alt="Cylinder" /></td>
</tr>
</tbody>
</table>

- $P(+,+,r) = 3/8$
- $P(-,-,r) = 1/8$
- $P(+,+,s) = 1/8$
- $P(-,-,s) = 3/8$

**NB Model**

- **Raining?**
  - M1
  - M2

**NB FACTORS:**
- $P(s) = 1/2$
- $P(+) | s) = 1/4$
- $P(+) | r) = 3/4$

**PREDICTIONS:**
- $P(r,+,+) = (1/2)(3/4)(3/4)$
- $P(s,+,+) = (1/2)(1/4)(1/4)$
- $P(r|+,+) = 9/10$
- $P(s|+,+) = 1/10$
Example: Stoplights

Reality

<table>
<thead>
<tr>
<th>Lights Working</th>
<th>Lights Broken</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Stoplight Diagram" /></td>
<td><img src="image" alt="Stoplight Diagram" /></td>
</tr>
<tr>
<td>( P(g, r, w) = \frac{3}{7} )</td>
<td>( P(r, r, b) = \frac{1}{7} )</td>
</tr>
<tr>
<td>( P(r, g, w) = \frac{3}{7} )</td>
<td></td>
</tr>
</tbody>
</table>

**NB Model**

**Working?**

- **NS**
- **EW**

**NB FACTORS:**

- \( P(w) = \frac{6}{7} \)
- \( P(r \mid w) = \frac{1}{2} \)
- \( P(g \mid w) = \frac{1}{2} \)
- \( P(b) = \frac{1}{7} \)
- \( P(r \mid b) = 1 \)
- \( P(g \mid b) = 0 \)
Example: Stoplights

- What does the model say when both lights are red?
  - \( P(b, r, r) = (1/7)(1)(1) = 1/7 = 4/28 \)
  - \( P(w, r, r) = (6/7)(1/2)(1/2) = 6/28 = 6/28 \)
  - \( P(w | r, r) = 6/10! \)

- We’ll guess that \((r, r)\) indicates lights are working!

- Imagine if \(P(b)\) were boosted higher, to 1/2:
  - \( P(b, r, r) = (1/2)(1)(1) = 1/2 = 4/8 \)
  - \( P(w, r, r) = (1/2)(1/2)(1/2) = 1/8 = 1/8 \)
  - \( P(w | r, r) = 1/5! \)

- Changing the parameters bought accuracy at the expense of data likelihood