Statistical NLP
Spring 2008

Lecture 6: Classification

Dan Klein – UC Berkeley

A Discriminative Approach

- View WSD as a discrimination task (regression, really)

  P(sense | context:jail, context:county,
  context:feeding, …
  local-context:jail, local-context:meals
  subcat:NP, direct-object-head:meals, ….)

- Have to estimate multinomial (over senses) where there are a huge number of things to condition on
  - History is too complex to think about this as a smoothing / back-off problem
- Many feature-based classification techniques out there
- We tend to need ones that output distributions over classes (why?)

Feature Representations

d $\quad\quad$ $\{f_i(d)\}$

Washington County jail served 11,166 meals last month - a figure that translates to feeding some 120 people three times daily for 31 days.

context:jail = 1
context:county = 1
context:feeding = 1
context:game = 0
…
local-context:jail = 1
local-context:meals = 1
…
subcat:NP = 1
subcat:PP = 0
…
object-head:meals = 1
object-head:ball = 0

- Features are indicator functions f_i which count the occurrences of certain patterns in the input
- We map each input to a vector of feature predicate counts

Example: Text Classification

- We want to classify documents into categories

DOCUMENT	CATEGORY
… win the election …	POLITICS
… win the game …	SPORTS
… see a movie …	OTHER

- Classically, do this on the basis of words in the document, but other information sources are potentially relevant:
  - Document length
  - Average word length
  - Document's source
  - Document layout

Some Definitions

Sometimes, we want Y to depend on x

INPUTS	$\mathbf{x}^i$	… win the election …
OUTPUT SPACE	$\mathcal{Y}$	SPORTS, POLITICS, OTHER
OUTPUTS	$\mathbf{y}$	SPORTS
TRUE OUTPUTS	$\mathbf{y}^i$	POLITICS
FEATURE VECTORS	$\mathbf{f}_i(\mathbf{y})$	[0 0 0 0 1 0 1 0 0 0 0 0]

Either x is implicit, or y contains x

SPORTS ∧ "win" $\quad$ POLITICS ∧ "election" $\quad$ POLITICS ∧ "win"

Block Feature Vectors

- Sometimes, we think of the input as having features, which are multiplied by outputs to form the candidates

$\mathbf{x}$ $\quad$ … win the election …

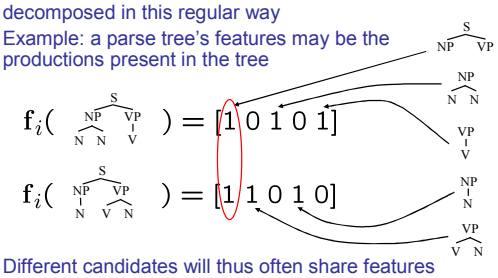
"$\mathbf{f}_i(\mathbf{x})$" $\quad$ [1 0 1 0]

"win" $\quad$ "election"

$$\mathbf{f}_i(SPORTS) = [1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$
$$\mathbf{f}_i(POLITICS) = [0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0]$$
$$\mathbf{f}_i(OTHER) = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0]$$

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



- Different candidates will thus often share features
- We'll return to the non-block case later

Linear Models: Scoring

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

$$\mathbf{f}_i(POLITICS) = [\ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$$
$$\mathbf{f}_i(SPORTS) = [\ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$$
$$\mathbf{w} = [\ 1\ \ 1\ -1\ -2\ \ 1\ -1\ \ 1\ -2\ -2\ -1\ -1\ \ 1]$$

- We compare hypotheses on the basis of their linear scores:

$$score(\mathbf{x}^i, \mathbf{y}, \mathbf{w}) = \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$$

$$\mathbf{f}_i(POLITICS) = [\ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$$
$$\mathbf{w} = [\ 1\ \ 1\ -1\ -2\ \ 1\ -1\ \ 1\ -2\ -2\ -1\ -1\ \ 1]$$

$$score(\mathbf{x}^i, POLITICS, \mathbf{w}) = 1 \times 1 + 1 \times 1 = 2$$

Linear Models: Prediction Rule

- The linear prediction rule:

$$prediction(\mathbf{x}^i, \mathbf{w}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$$

$$score(\mathbf{x}^i, SPORTS, \mathbf{w}) = 1 \times 1 + (-1) \times 1 = 0$$
$$score(\mathbf{x}^i, POLITICS, \mathbf{w}) = 1 \times 1 + 1 \times 1 = 2$$
$$score(\mathbf{x}^i, OTHER, \mathbf{w}) = (-2) \times 1 + (-1) \times 1 = -3$$

$$prediction(\mathbf{x}^i, \mathbf{w}) = POLITICS$$

- We've said nothing about where weights come from!

Binary Decision Rule

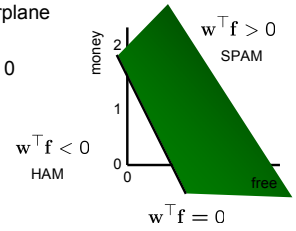
- Heavily studied case: binary classification
  - Simplifed: only class "1" has features

$$prediction(\mathbf{x}^i, \mathbf{w}) = (\mathbf{w}^\top \mathbf{f}_i > 0)$$

  - Decision rule is a hyperplane
  - One side will be class 1
  - Other side will be class 0

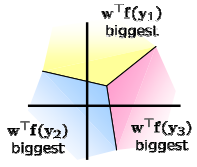
w

BIAS	-3
free	4
money	2
the	0
...	



Multiclass Decision Rule

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



$$prediction(\mathbf{x}^i, \mathbf{w}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$$

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

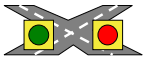
Learning Classifier Weights

- 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

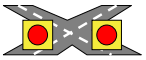
Example: Stoplights

Reality

Lights Working

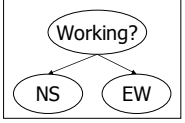


Lights Broken



P(g,r,w) = 3/7 P(r,g,w) = 3/7 P(r,r,b) = 1/7

NB Model



NB FACTORS:

- P(w) = 6/7
- P(r|w) = 1/2
- P(g|w) = 1/2
- P(b) = 1/7
- P(r|b) = 1
- P(g|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!
- Non-generative values can give better classification

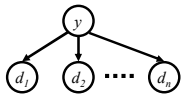
Linear Models: Naïve-Bayes

- (Multinomial) Naïve-Bayes is a linear model, where:

$$\mathbf{x}^i = d_1, d_2, \cdots d_n$$

$$\begin{array}{rcl} \mathrm{f}_i(\mathrm{y}) & = & [\ \cdots 0 \cdots \quad 1, \quad \#v_1, \quad \#v_2, \quad \cdots \quad \#v_{|V|} \quad \cdots 0 \cdots\ ] \\ \mathrm{w} & = & [\ \cdots \quad \log \mathrm{P}(y), \ \log \mathrm{P}(v_1|y), \ \log \mathrm{P}(v_2|y), \ \cdots \ \log \mathrm{P}(v_n|y) \quad \cdots\ ] \end{array}$$

$$\begin{array}{rcl} score(\mathbf{x}^i, \mathrm{y}, \mathrm{w}) & = & \log \mathrm{P}(\mathbf{x}^i, \mathrm{y}) \\ & = & \log \left(\mathrm{P}(\mathrm{y}) \prod_{d \in \mathbf{x}^i} \mathrm{P}(d|\mathrm{y}) \right) \\ & = & \log \left(\mathrm{P}(\mathrm{y}) \prod_k \mathrm{P}(v_k|\mathrm{y})^{\#v_k} \right) \\ & = & \log \mathrm{P}(\mathrm{y}) + \sum_k \#v_k \log \mathrm{P}(v_k|\mathrm{y}) \\ & = & \mathrm{w}^\top \mathrm{f}_i(\mathrm{y}) \end{array}$$



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

$$\ell_i(\mathrm{y}) = \ell(\mathrm{y}, \mathrm{y}^i)$$

  - 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:

$$\min_{\mathrm{w}} \sum_i \ell_i \left(\arg\max_{\mathrm{y}} \mathrm{w}^\top \mathrm{f}_i(\mathrm{y}) \right)$$

- This is a hard, discontinuous optimization problem

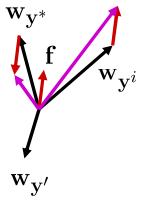
Linear Models: Perceptron

- 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
  - Visit training instances one by one
    - Try to classify

$$\mathrm{y}^* = \arg\max_{\mathrm{y} \in \mathcal{Y}} \mathrm{w}^\top \mathrm{f}_i(\mathrm{y})$$

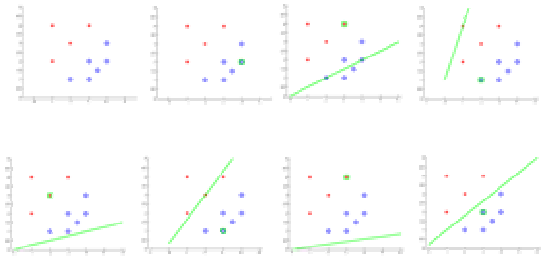
    - If correct, no change!
    - If wrong: adjust weights

$$\mathrm{w} \leftarrow \mathrm{w} + \mathrm{f}_i(\mathrm{y}^i)$$
$$\mathrm{w} \leftarrow \mathrm{w} - \mathrm{f}_i(\mathrm{y}^*)$$



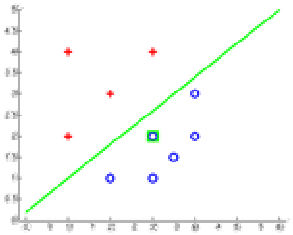
Examples: Perceptron

- Separable Case



Examples: Perceptron

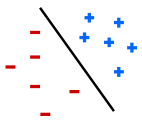
- Separable Case



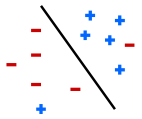
Perceptrons and Separability

- 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

Separable

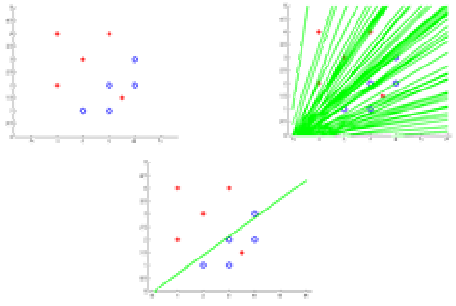


Non-Separable



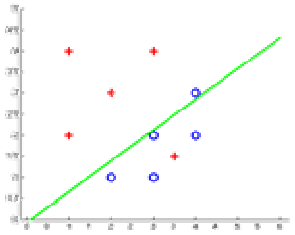
Examples: Perceptron

- Non-Separable Case



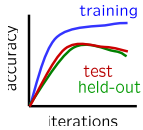
Examples: Perceptron

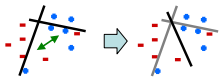
- Non-Separable Case



Issues with Perceptrons

- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining isn't quite as bad as overfitting, but is similar
- 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



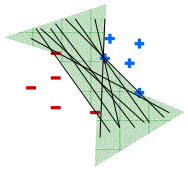


Problems with Perceptrons

- Perceptron "goal": separate the training data

$$\forall i, \forall \mathbf{y} \neq \mathbf{y}^i \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$$

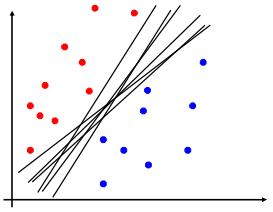
1. This may be an entire feasible space



2. Or it may be impossible

Linear Separators

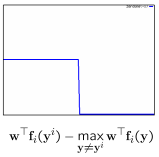
- Which of these linear separators is optimal?



Objective Functions

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

$$\sum_i step\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y} \neq \mathbf{y}^i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})\right)$$



  - 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 Models: Maximum Entropy

- Maximum entropy (logistic regression)
  - Use the scores as probabilities:

$$P(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}))}{\sum_{\mathbf{y}'} \exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}'))}$$

Make positive

Normalize

  - Maximize the (log) conditional likelihood of training data

$$L(\mathbf{w}) = \log \prod_i P(\mathbf{y}^i|\mathbf{x}^i, \mathbf{w}) = \sum_i \log\left(\frac{\exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i))}{\sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}))}\right)$$

$$= \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}))\right)$$

Derivative for Maximum Entropy

$$L(\mathbf{w}) = \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}))\right)$$

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}_n} = \sum_i \left(\mathbf{f}_i(\mathbf{y}^i)_n - \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_i)\mathbf{f}_i(\mathbf{y})_n\right)$$

Total count of feature n in correct candidates

Expected count of feature n in predicted candidates

Expected Counts

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}_n} = \sum_i \left(\mathbf{f}_i(\mathbf{y}^i)_n - \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_i)\mathbf{f}_i(\mathbf{y})_n\right)$$

		$\mathbf{y}^i$	$P(\mathbf{y} \mid \mathbf{x}^i, \mathbf{w})$
$\mathbf{x}^i$'s	meal, jail, …	food	.4
	jail, term, …	prison	.8
The weight for the "context-word:jail and cat:prison" feature:		actual = 1	empirical = 1.2

- The optimum parameters are the ones for which each feature's predicted expectation equals its empirical expectation. The optimum distribution is:
  - Always unique (but parameters may not be unique)
  - Always exists (if features counts are from actual data).

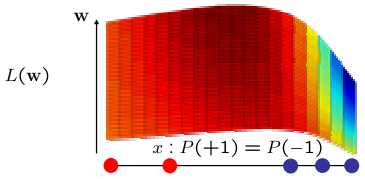
Maximum Entropy II

- 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)

$$\max_{\mathbf{w}} \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right) - k||\mathbf{w}||^2$$

$$\min_{\mathbf{w}} \; k||\mathbf{w}||^2 - \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

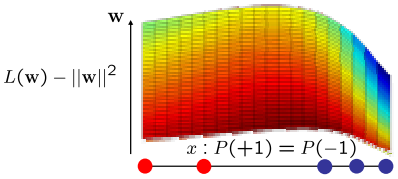
Maximum Entropy Separators



$$\mathbf{f}(+1) = [1 \quad x \quad 0 \quad 0]$$
$$\mathbf{f}(-1) = [0 \quad 0 \quad 1 \quad x]$$
$$\mathbf{w} = [a \quad b \quad -a \quad -b]$$

$$\mathrm{P}(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}))}{\sum_{\mathbf{y}'} \exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}'))}$$

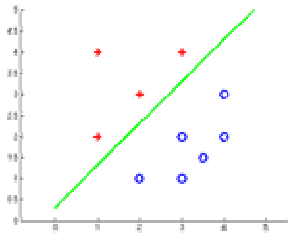
Maximum Entropy Separators



$$\mathbf{f}(+1) = [1 \quad x \quad 0 \quad 0]$$
$$\mathbf{f}(-1) = [0 \quad 0 \quad 1 \quad x]$$
$$\mathbf{w} = [a \quad b \quad -a \quad -b]$$

$$\mathrm{P}(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}))}{\sum_{\mathbf{y}'} \exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}'))}$$

Maximum Entropy



Example: NER Smoothing

Because of smoothing, the more common prefixes have larger weights even though entire-word features are more specific.

Local Context

	Prev	Cur	Next
State	Other	???	???
Word	at	Grace	Road
Tag	IN	NNP	NNP
Sig	x	Xx	Xx

Feature Weights

Feature Type	Feature	PERS	LOC
Previous word	*at*	-0.73	0.94
Current word	*Grace*	0.03	0.00
Beginning bigram	*<G*	0.45	-0.04
Current POS tag	NNP	0.47	0.45
Prev and cur tags	IN NNP	-0.10	0.14
Previous state	Other	-0.70	-0.92
Current signature	Xx	0.80	0.46
Prev state, cur sig	O-Xx	0.68	0.37
Prev-cur-next sig	x-Xx-Xx	-0.69	0.37
P. state - p-cur sig	O-x-Xx	-0.20	0.82
…			
Total:		**-0.58**	**2.68**

Log-Loss

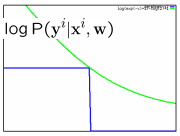
- If we view maxent as a minimization problem:

$$\min_{\mathbf{w}} \; k||\mathbf{w}||^2 - \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

- This minimizes the "log loss" on each example

$$-\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right) = -\log \mathrm{P}(\mathbf{y}^i|\mathbf{x}^i, \mathbf{w})$$

$$step\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y} \neq \mathbf{y}^i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) \right)$$



- One view: log loss is an *upper bound* on zero-one loss

Derivative for Maximum Entropy

$$L(\mathbf{w}) = -k||\mathbf{w}||^2 + \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}_n} = -2k\mathbf{w}_n + \sum_i \left(\mathbf{f}_i(\mathbf{y}^i)_n - \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_i)\mathbf{f}_i(\mathbf{y})_n \right)$$

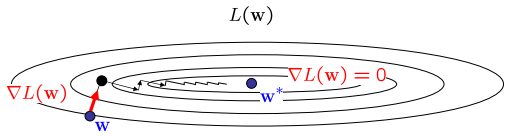
Big weights are bad

Total count of feature n in correct candidates

Expected count of feature n in predicted candidates

Unconstrained Optimization

- The maxent objective is an unconstrained optimization problem

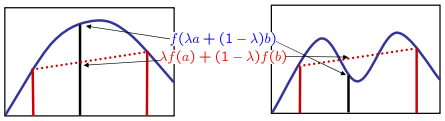


- Basic idea: move uphill from current guess
- Gradient ascent / descent follows the gradient incrementally
- At local optimum, derivative vector is zero
- Will converge if step sizes are small enough, but not efficient
- All we need is to be able to evaluate the function and its derivative

Convexity

- The maxent objective is nicely behaved:
  - *Differentiable* (so many ways to optimize)
  - *Convex* (so no local optima)

$$f(\lambda a + (1-\lambda)b) \geq \lambda f(a) + (1-\lambda) f(b)$$

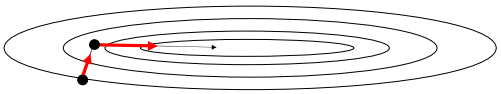


Convex Non-Convex

Convexity guarantees a single, global maximum value because any higher points are greedily reachable

Unconstrained Optimization

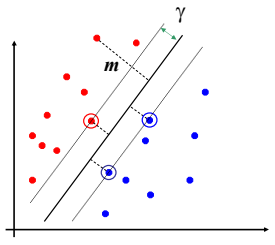
- Once we have a function f, we can find a local optimum by iteratively following the gradient



- For convex functions, a local optimum will be global
- Basic gradient ascent isn't very efficient, but there are simple enhancements which take into account previous gradients: conjugate gradient, L-BFGs
- There are special-purpose optimization techniques for maxent, like iterative scaling, but they aren't better

Classification Margin (Binary)

- Distance of $\mathbf{x}_i$ to separator is its margin, $\boldsymbol{m}_i$
- Examples closest to the hyperplane are support vectors
- Margin γ of the separator is the minimum $\boldsymbol{m}$



Classification Margin

- For each example $\mathbf{x}_i$ and possible mistaken candidate $\mathbf{y}$, we avoid that mistake by a margin $\boldsymbol{m}_i(\mathbf{y})$ (with zero-one loss)

$$m_i(\mathbf{y}) = \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$$

- Margin γ of the entire separator is the minimum $\boldsymbol{m}$

$$\gamma = \min_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \min_{\mathbf{y} \neq \mathbf{y}^i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) \right)$$

- It is also the largest γ for which the following constraints hold

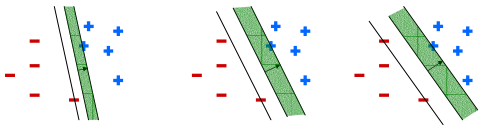
$$\forall i, \forall \mathbf{y} \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \gamma \ell_i(\mathbf{y})$$

Maximum Margin

- Separable SVMs: find the max-margin w

$$\max_{||\mathbf{w}||=1} \gamma \qquad \ell_i(\mathbf{y}) = \begin{cases} 0 & \text{if } \mathbf{y} = \mathbf{y}^i \\ 1 & \text{if } \mathbf{y} \neq \mathbf{y}^i \end{cases}$$

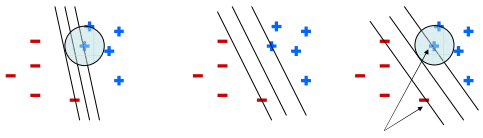
$$\forall i, \forall \mathbf{y} \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \gamma \ell_i(\mathbf{y})$$



- 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



Max Margin / Small Norm

- Reformulation: find the smallest w which separates data

Remember this condition?

$$\max_{||\mathbf{w}||=1} \gamma \quad \forall i, \mathbf{y} \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \gamma \ell_i(\mathbf{y})$$

- γ scales linearly in w, so if ||w|| isn't constrained, we can take any separating w and scale up our margin

$$\gamma = \min_{i, \mathbf{y} \neq \mathbf{y}^i} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})] / \ell_i(\mathbf{y})$$

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

$$\min_{\mathbf{w}} \frac{1}{2} ||\mathbf{w}||^2$$
$$\forall i, \mathbf{y} \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^*) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + 1 \ell_i(\mathbf{y})$$

Gamma to w

$$\max_{||\mathbf{w}||=1} \gamma \quad \forall i, \mathbf{y} \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \gamma \ell_i(\mathbf{y})$$

$$\mathbf{w} = \gamma u$$

$$\gamma = 1/||u||$$

$$\max_{||\gamma u||=1} 1/||u||^2 \quad \forall i, \mathbf{y} \quad \gamma u^\top \mathbf{f}_i(\mathbf{y}^i) \geq \gamma u^\top \mathbf{f}_i(\mathbf{y}) + \gamma \ell_i(\mathbf{y})$$

$$\max_{||\gamma u||=1} 1/||u||^2 \quad \forall i, \mathbf{y} \quad u^\top \mathbf{f}_i(\mathbf{y}^i) \geq u^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$$

$$\min_{||\gamma u||=1} ||u||^2 \quad \forall i, \mathbf{y} \quad u^\top \mathbf{f}_i(\mathbf{y}^i) \geq u^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$$

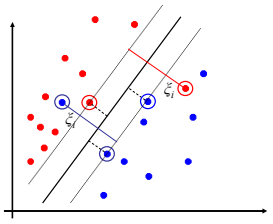
$$\min_{u} ||u||^2 \quad \forall i, \mathbf{y} \quad u^\top \mathbf{f}_i(\mathbf{y}^i) \geq u^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$$

$$\min_{u} \frac{1}{2} ||u||^2 \quad \forall i, \mathbf{y} \quad u^\top \mathbf{f}_i(\mathbf{y}^i) \geq u^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$$

$$\min_{\mathbf{w}} \frac{1}{2} ||\mathbf{w}||^2 \quad \forall i, \mathbf{y} \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$$

Soft Margin Classification

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

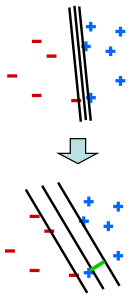


Maximum Margin

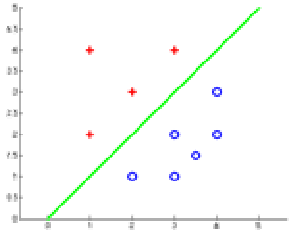
- Non-separable SVMs
  - Add slack to the constraints
  - Make objective pay (linearly) for slack:

$$\min_{\mathbf{w}, \xi} \frac{1}{2} ||\mathbf{w}||^2 + C \sum_i \xi_i$$
$$\forall i, \mathbf{y}, \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) + \xi_i \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$$

  - 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



Maximum Margin



Remember SVMs…

- We had a constrained minimization

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i$$
$$\forall i, \mathbf{y}, \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) + \xi_i \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$$

- …but we can solve for ξ_i

$$\forall i, \mathbf{y}, \quad \xi_i \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y}) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i)$$
$$\forall i, \quad \xi_i = \max_{\mathbf{y}} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})\right) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i)$$

- Giving

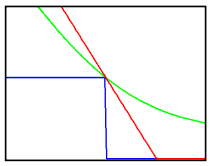
$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 - C\sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y}} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})\right)\right)$$

Hinge Loss

- Consider the per-instance objective:

$$\min_{\mathbf{w}} \; k||\mathbf{w}||^2 - \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y}} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(y)\right)\right)$$

- 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



Plot really only right in binary case

Max vs "Soft-Max" Margin

- SVMs:

$$\min_{\mathbf{w}} \; k||\mathbf{w}||^2 - \sum_i \underbrace{\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y}} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(y)\right)\right)}_{\text{You can make this zero}}$$

- Maxent:

$$\min_{\mathbf{w}} \; k||\mathbf{w}||^2 - \sum_i \underbrace{\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})\right)\right)}_{\text{… but not this one}}$$

- 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

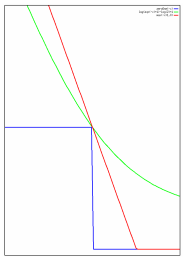
$$\sum_i step\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y}\neq\mathbf{y}^i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})\right)$$

- Hinge

$$\sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y}} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(y)\right)\right)$$

- Log

$$\sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})\right)\right)$$



Separators: Comparison