Lecture 3: Classification: From Binary to Structure II

Kai-Wei Chang
CS @ UCLA
kw@kwchang.net
Announcements

- Waiting list: If you’re not enrolled, please sign up.

- We will use Piazza as an online discussion platform. Please sign up here: http://piazza.com/ucla/spring2020/cs263
Supervised learning

Input

$x \in \mathcal{X}$

An item $x$ drawn from an instance space $\mathcal{X}$

Output

$y \in \mathcal{Y}$

An item $y$ drawn from a label space $\mathcal{Y}$

Target function

$y = f^*(x)$

Learned Model

$y = f(x)$
Supervised learning

Input

\[ x \in \mathcal{X} \]

An item \( x \) drawn from an instance space \( \mathcal{X} \)

\( x \) is represented in a feature space
- Typically \( x \in \{0,1\}^n \) or \( R^N \)
- Usually represented as a vector
- We call it input vector
Supervised learning

An item $y$ drawn from a label space $Y$

$y$ is represented in output space (label space)

Different kinds of output:

- Binary classification: $y \in \{-1, 1\}$
- Multiclass classification: $y \in \{1, 2, 3, \ldots K\}$
- Regression: $y \in R$
- Structured output $y \in \{1, 2, 3, \ldots K\}^N$
Learning the mapping

\[ y \in Y \]

An item \( y \) drawn from a label space \( Y \)

Input

\[ x \in X \]

An item \( x \) drawn from an instance space \( X \)

Target function

\[ y = f^*(x) \]

Learned Model

\[ y = f(x) \]

Output

\[ y \in Y \]

An item \( y \) drawn from a label space \( Y \)
Linear classification

Teacher

Today, we are going to learn about matrix

<table>
<thead>
<tr>
<th>Expectation</th>
<th>Reality</th>
</tr>
</thead>
</table>
| b = \[
  1 & 2 & 3 \\
  3 & 1 & 2 
\] | 0 0 1  |
| e = \[
  1 & 2 & 3 \\
  3 & 2 & 1 
\] | 0 0 1  |

[Image of a person from The Matrix with bullet holes]
Linear classifiers

- For now, we consider binary classification.
- Given a training set $\mathcal{D} = \{(x, y)\}$, find a linear threshold units classify an example $x$ using the classification rule:

$$\text{sgn}(b + w^T x) = \text{sgn}(b + \sum_i w_i x_i)$$

- $b + w^T x \geq 0 \Rightarrow$ Predict $y = 1$
- $b + w^T x < 0 \Rightarrow$ Predict $y = -1$
The geometry interpretation

\[ \text{sgn}(b + w_1 x_1 + w_2 x_2) \]

\[ b + w_1 x_1 + w_2 x_2 = 0 \]

In \( n \) dimensions, a linear classifier represents a hyperplane that separates the space into two half-spaces.
Some data are not linearly separable
But they can be **made** liner

Using a different representation e.g., feature conjunctions, non-linear mapping
Linear classifiers

- Let’s take a look at a few linear classifiers
- We will show later, they can be written in the same framework!

- Perceptron
- (Linear) Support Vector Machines
- Logistic Regression
The Perceptron Algorithm [Rosenblatt 1958]

- Goal: find a separating hyperplane
- Can be used in an online setting: considers one example at a time
- Converges if data is separable
  -- mistake bound
The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. **Initialize** $w \leftarrow 0 \in \mathbb{R}^n$
2. **For** epoch $1 \ldots T$:
3. **For** $(x, y)$ in $\mathcal{D}$:
4. \[ \hat{y} = sgn(w^T x) \] \hspace{1cm} (predict)
5. **if** $\hat{y} \neq y$, $w \leftarrow w + \eta y x$ \hspace{1cm} (update)
6. **Return** $w$

Prediction: $y^{test} \leftarrow sgn(w^T x^{test})$
The Perceptron Algorithm [Rosenblatt 1958]

Given a training set \( D = \{(x, y)\} \)

1. Initialize \( w \leftarrow 0 \in \mathbb{R}^n \)
2. For epoch 1...\( T \):
3. For \( (x, y) \) in \( D \):
4. if \( y(w^T x) \leq 0 \)
5. \( w \leftarrow w + \eta y x \)
6. Return \( w \)

Prediction: \( y^{\text{test}} \leftarrow \text{sgn}(w^T x^{\text{test}}) \)
How about batch setting?

- Learning as loss minimization
  1. Collect training data $\tilde{D} = \{(x, y)\}$
  2. Pick a hypothesis class
     - E.g., linear classifiers, deep neural networks
  3. Choose a loss function
     - Hinge loss, negative log-likelihood
     - We can impose a preference (i.e., prior) over hypotheses, e.g., simpler is better
  4. Minimize the expected loss
     - SGD, coordinate descent, Newton methods, LBFGS
Batch learning setup

- \( \hat{D} = \{(x, y)\} \) drawn from a fixed, unknown distribution \( \mathcal{D} \)
- A hidden oracle classifier \( f^* \), \( y = f^*(x) \)
- We wish to find a hypothesis \( f \in H \) that mimics \( f^* \)
- We define a loss function \( L(f(x), f^*(x)) \) that penalizes mistakes
- What is the ideal \( f \)?

\[
\arg\min_{f \in H} \mathbb{E}_{x \sim \mathcal{D}} \left[ L(f(x), f^*(x)) \right]
\]

expected loss
Batch learning setup

- \( \hat{D} = \{(x, y)\} \) drawn from a fixed, unknown distribution \( \mathcal{D} \)
- A hidden oracle classifier \( f^* \), \( y = f^*(x) \)
- We wish to find a hypothesis \( f \in H \) that mimics \( f^* \)
- We define a loss function \( L(f(x), f^*(x)) \) that penalizes mistakes

What is the ideal \( f \)?

Let’s define

\[
L_{0-1}(y, y') = \begin{cases} 
1 & \text{if } y \neq y' \\
0 & \text{if } y = y'
\end{cases}
\]

\[
\min_{f \in H} E_{x \sim D} \left[ L_{0-1}(f(x), f^*(x)) \right] = \min_{f \in H} E_{x \sim D}[\#\text{mistakes}]
\]
How can we learn $f$ from $\hat{D}$

- We don’t know $D$, we only see samples in $\hat{D}$
- Instead, we minimize empirical loss

$$\min_{f \in H} \frac{1}{|\hat{D}|} \sum_{(x,y) \in \hat{D}} [L(f(x), y)]$$
How can we prevent over-fitting?

- With sufficient data, \( \hat{D} \approx D \)
- However, if data is insufficient \( \Rightarrow \) overfitting
- We can impose a preference over models

\[
\min_{f \in H} R(f) + \frac{1}{|\hat{D}|} \sum_{(x,y) \in \hat{D}} [L(f(x), y)]
\]

- We will discuss the choices of \( R(f) \) later
Many choices

- We are minimizing with $R, L, H$ with your choice

$$\min_{f \in H} R(f) + \frac{1}{|D|} \sum_{(x,y) \in \tilde{D}} [L(f(x), y)]$$

- Let consider $H$ is a set of $d$-dimensional linear function

- $H$ can be parameterized as

$$\{f(x): w^T x \geq 0\}, w \in R^d$$
Back to Linear model

- We are minimizing with R, L, H with your choice

\[
\min_{f \in H} R(f) + \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} [L(f(x), y)]
\]

- Let decide H to be a set of d-dimensional linear function

- H can be parameterized as

\[\{ f(x): w^T x \geq 0 \}, w \in \mathbb{R}^d \]

- We are going to fine the best one based on \( \mathcal{D} \)

  - i.e., find the best setting of w and b
Rewrite our optimization problem

- Minimizing the empirical loss:
  \[
  \min_{f \in H} R(f) + \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} [L(f(x), y)]
  \]

- Minimizing the empirical loss with linear function
  \[
  \min_{w \in \mathbb{R}^d} R(w) + \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} [L(x, w, y)]
  \]

- What choices of R and L we have?
Many choices of loss function (L)

Many loss functions exist

- Perceptron loss
  \[ L_{\text{Perceptron}}(y, x, w) = \max(0, -y w^T x) \]

- Hinge loss (SVM)
  \[ L_{\text{Hinge}}(y, x, w) = \max(0, 1 - y w^T x) \]

- Exponential loss (AdaBoost)
  \[ L_{\text{Exponential}}(y, x, w) = e^{-y w^T x} \]

- Logistic loss (logistic regression)
  \[ L_{\text{Logistic}}(y, x, w) = \log(1 + e^{-y w^T x}) \]
Many choices of $R(w)$

- Minimizing the empirical loss with linear function

$$\min_{w \in \mathbb{R}^d} R(w) + \frac{1}{|D|} \sum_{(x,y) \in \hat{D}} [L(x, w, y)]$$

- Prefer simpler model: (how?)
  - Sparse:
    $$R(w) = \# \text{non-zero elements in } w \quad (\text{L0 regularizer})$$
    $$R(w) = \sum_i |w_i| \quad (\text{L1 regularizer})$$
  - Gaussian prior (large margin w/ hinge loss):
    $$R(w) = \sum_i w_i^2 = w^T w \quad (\text{L2 regularizer})$$
Support Vector Machines

CMU ML protest
Support Vector Machines (SVMs)

- R(w): l2-loss, \( L(w, x, y) \): hinge loss
  \[
  \min_w \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i (w^T x_i))
  \]

- Maximizing margin (why?!!)

![Diagram showing maximizing margin](image)
Let’s view it from another direction

- SVM learns a model $w$ on $\mathcal{D} = \{(x_i, y_i)\}$ by solving:

$$
\begin{align*}
\min_{w, b} & \quad \frac{1}{2} w^T w \\
\text{s.t.} & \quad y_i (w^T x_i + b) \geq 1, \forall (x_i, y_i) \in \mathcal{D}
\end{align*}
$$

(Hard SVM)

Why the margin is $\frac{2}{||w||}$?

$$
\begin{align*}
\arg \max & \quad \frac{2}{||w||} \\
\equiv & \quad \arg \min \quad ||w|| \\
\equiv & \quad \arg \min \quad ||w||^2 \\
\equiv & \quad \arg \min \quad w^T w
\end{align*}
$$
Balance between regularization and empirical loss

(a) Training data and an over-fitting classifier

(b) Testing data and an over-fitting classifier
Balance between regularization and empirical loss

(c) Training data and a better classifier

(d) Testing data and a better classifier
Soft SVMs

- Data is not separable ⇒ hard SVM fails
  Why?
- Introduce a set of slack variable \( \{\xi_i\} \) ⇒ relax the constraints
- Given \( \mathcal{D} = \{(x_i, y_i)\} \), soft SVM solves:

\[
\min_{w, b, \xi} \frac{1}{2} w^T w + C \sum_i \xi_i \\
\text{s.t. } y_i (w^T x_i + b) \geq 1 - \xi_i; \quad \xi_i \geq 0 \quad \forall i
\] (Soft SVM)

penalty parameter
An alternative formulation

\[
\min_{\mathbf{w}, \mathbf{b}, \xi} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \xi_i \\
\text{s.t.} \quad y_i (\mathbf{w}^T \mathbf{x}_i + \mathbf{b}) \geq 1 - \xi_i; \quad \xi_i \geq 0 \quad \forall i
\]

- Rewrite the constraints:
  \[
  \xi_i \geq 1 - y_i (\mathbf{w}^T \mathbf{x}_i + \mathbf{b}); \quad \xi_i \geq 0 \quad \forall i
  \]

- In the optimum, \( \xi_i = \max(0, 1 - y_i (\mathbf{w}^T \mathbf{x}_i + \mathbf{b})) \)

- Soft SVM can be rewritten as:
  \[
  \min_{\mathbf{w}, \mathbf{b}} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \max(0, 1 - y_i (\mathbf{w}^T \mathbf{x}_i + \mathbf{b}))
  \]

Empirical loss

Regularization term

CS263- Lecture 2 ML in NLP
The hinge loss
**Hinge**: Incorrect predictions get a linearly increasing penalty with $w^T x$.

**Hinge**: Penalize predictions even if they are correct, but too close to the margin.

**Hinge**: No penalty if $w^T x$ is far away from 1 (-1 for negative examples).

$y w^T x$
Regularized loss minimization

- **L1-Loss SVM**
  \[
  \min_w \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i (w^T x_i))
  \]

- **L2-Loss SVM**
  \[
  \min_w \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i (w^T x_i))^2
  \]

- **Logistic Regression (regularized)**
  \[
  \min_w \frac{1}{2} w^T w + C \sum_i \log(1 + e^{-y_i (w^T x_i)})
  \]

- **Loss over training data + regularizer**
Logistic Regression

Regression

Logistic regression

Logistic function
logistic function / sigmoid function

- When $z \to \infty$ what is $\sigma(z)$?
- When $z \to -\infty$ what is $\sigma(z)$?
- When $z = 0$ what is $\sigma(z)$?

\[
\sigma(z) = \frac{1}{1 + e^{-z}}
\]
Why sigmoid?

Least squares fit

\[ \sigma(wx + b) \text{ fit to } y \]

\[ wx + b \text{ fit to } y \]
Probabilistic Interpretation

$$\min_w \frac{1}{2} w^T w + C \sum_i \log(1 + e^{-y_i(w^T x_i)})$$

Assume labels are generated using the following probability distribution:

$$P(y = 1|x, w) = \frac{e^{w^T x}}{1 + e^{w^T x}} = \frac{1}{1 + e^{-w^T x}}$$

$$P(y = -1|x, w) = \frac{1}{1 + e^{w^T x}}$$
How to make prediction?

Predict $y=1$ if $P(y=1|x,w) > p(y=-1|x,w)$
Structured Output Prediction
A General Formula

\[ \hat{y} = \arg\max_{y \in \mathcal{Y}} f(y; \mathbf{w}, \mathbf{x}) \]

- **Inference/Test:** Given \( \mathbf{w}, \mathbf{x} \), solve \( \arg\max \)
- **Learning/Training:** Find a good \( \mathbf{w} \)
- **Today:** \( \mathbf{x} \in \mathbb{R}^n, \mathcal{Y} = \{1, 2, \ldots, K\} \) (multiclass)
Outline – Multiclass classification

- Reducing multiclass to binary
  - One-against-all & One-vs-one
  - Error correcting codes

- Training a single classifier
  - Multiclass Perceptron: Kesler’s construction
  - Multiclass SVMs: Crammer & Singer formulation
  - Multinomial logistic regression
One against all strategy
One against All learning

- Multiclass classifier
  - Function $f : \mathbb{R}^n \rightarrow \{1,2,3,\ldots,k\}$
- Decompose into binary problems
One-again-All learning algorithm

- Learning: Given a dataset $D = \{(x_i, y_i)\}$
  $x_i \in \mathbb{R}^n$, $y_i \in \{1, 2, 3, \ldots K\}$

- Decompose into K binary classification tasks
  - Learn K models: $w_1, w_2, w_3, \ldots w_K$
  - For class k, construct a binary classification task as:
    - Positive examples: Elements of D with label k
    - Negative examples: All other elements of D
  - The binary classification can be solved by any algorithm we have seen
One against All learning

- Multiclass classifier
  - Function \( f : \mathbb{R}^n \rightarrow \{1,2,3,\ldots,k\} \)
  - Decompose into binary problems

Ideal case: only the correct label will have a positive score

\[
\begin{align*}
  w_{black}^T x &> 0 \\
  w_{blue}^T x &> 0 \\
  w_{green}^T x &> 0
\end{align*}
\]
One-again-All Inference

- **Learning**: Given a dataset \( D = \{(x_i, y_i)\} \)
  \( x_i \in \mathbb{R}^n, y_i \in \{1,2,3, ... K\} \)
- **Decompose into K binary classification tasks**
  - Learn K models: \( w_1, w_2, w_3, ... w_K \)
- **Inference**: “Winner takes all”
  - \( \hat{y} = \arg\max_{y \in \{1,2, ... K\}} w_y^T x \)

  **For example**: \( y = \arg\max (w_{black}^T x, w_{blue}^T x, w_{green}^T x) \)

- **An instance of the general form**
  \[ \hat{y} = \arg\max_{y \in \mathbb{Y}} f(y; w, x) \]

  \( w = \{w_1, w_2, ... w_K\}, f(y; w, x) = w_y^T x \)
One v.s. One (All against All) strategy
One v.s. One learning

- Multiclass classifier
  - Function \( f : \mathbb{R}^n \rightarrow \{1, 2, 3, \ldots, k\} \)
- Decompose into binary problems
One-v.s-One learning algorithm

- Learning: Given a dataset $D = \{(x_i, y_i)\}$
  \[x_i \in \mathbb{R}^n, y_i \in \{1, 2, 3, \ldots, K\}\]
- Decompose into $C(K,2)$ binary classification tasks
  - Learn $C(K,2)$ models: $w_1, w_2, w_3, \ldots, w_{K*(K-1)/2}$
  - For each class pair $(i,j)$, construct a binary classification task as:
    - Positive examples: Elements of $D$ with label $i$
    - Negative examples: Elements of $D$ with label $j$
    - The binary classification can be solved by any algorithm we have seen
One-v.s-One Inference algorithm

- Decision Options:
  - More complex; each label gets k-1 votes
  - Output of binary classifier may not cohere.
  - **Majority**: classify example $x$ to take label $i$ if $i$ wins on $x$ more often than $j$ ($j=1,...,k$)
  - **A tournament**: start with $n/2$ pairs; continue with winners
Problems with Decompositions

- Learning optimizes over *local* metrics
  - Does not guarantee good *global* performance
  - We don’t care about the performance of the *local* classifiers
- Poor decomposition $\Rightarrow$ poor performance
  - Difficult local problems
  - Irrelevant local problems
- Efficiency: e.g., All vs. All vs. One vs. All
- Not clear how to generalize multi-class to problems with a very large # of output
This Lecture

- Multiclass classification overview
- Reducing multiclass to binary
  - One-against-all & One-vs-one
  - Error correcting codes
- Training a single classifier
  - Multiclass Perceptron: Kesler’s construction
  - Multiclass SVMs: Crammer&Singer formulation
  - Multinomial logistic regression
Revisit One-again-All learning algorithm

- Learning: Given a dataset $D = \{(x_i, y_i)\}$
  $x_i \in \mathbb{R}^n, y_i \in \{1, 2, 3, \ldots K\}$

- Decompose into K binary classification tasks
  - Learn K models: $w_1, w_2, w_3, \ldots w_K$
  - $w_k$: separate class $k$ from others

- Prediction
  $$\hat{y} = \arg\max_{y \in \{1, 2, \ldots K\}} w^T_y x$$
Observation

- At training time, we require $w_i^T x$ to be positive for examples of class $i$.
- Really, all we need is for $w_i^T x$ to be more than all others $\Rightarrow$ this is a weaker requirement

For examples with label $i$, we need

$$w_i^T x > w_j^T x \quad \text{for all } j$$
### Perceptron-style algorithm

For each training example \((x, y)\)

- If for some \(y'\), \(w_{y'}^T x \leq w_y^T x\) mistake!
  - \(w_y \leftarrow w_y + \eta x\) update to promote label \(y\)
  - \(w_{y'} \leftarrow w_{y'} - \eta x\) update to demote label \(y'\)

---

**Why add \(\eta x\) to \(w_y\) to promote label \(y\):**

Before update \(s(y) = <w_y^{\text{old}}, x>\)

After update \(s(y) = <w_y^{\text{new}}, x> = <w_y^{\text{old}} + \eta x, x>\)

\[= <w_y^{\text{old}}, x> + \eta <x, x>\]

**Note!** \(<x, x> = x^T x > 0\)
A Perceptron-style Algorithm

Given a training set \( D = \{(x, y)\} \)

Initialize \( w \leftarrow 0 \in \mathbb{R}^n \)

For epoch 1...\( T \): 
For \((x, y)\) in \( D \):
For \( y' \neq y \)
if \( w_y^T x < w_y'^T x \)
   make a mistake 
   \( w_y \leftarrow w_y + \eta x \) promote \( y \)
   \( w_{y'} \leftarrow w_{y'} - \eta x \) demote \( y' \)

Return \( w \)

Prediction: \( \arg\max_y \ w_y^T x \)
Linear Separability with multiple classes

- Let’s rewrite the equation
  \[ w_i^T x > w_j^T x \quad \text{for all } j \]
- Instead of having \( w_1, w_2, w_3, \ldots w_K \), we want to represent the model using a single vector \( w \)
  \[ w^T \text{?} > w^T \text{?} \quad \text{for all } j \]

- How?
  Change the input representation
  Let’s define \( \phi(x, y) \), such that
  \[ w^T \phi(x, i) > w^T \phi(x, j) \quad \forall j \]

multiple models v.s. multiple data points
Kesler construction

Assume we have a multi-class problem with K class and n features.

\[ w_i^T x > w_j^T x \quad \forall \ j \]

- models:
  \[ w_1, w_2, \ldots w_K, \quad w_k \in R^n \]

- Input:
  \[ x \in R^n \]

\[ w^T \phi(x, i) > w^T \phi(x, j) \quad \forall \ j \]
Kesler construction

Assume we have a multi-class problem with $K$ class and $n$ features.

$$w_i^T x > w_j^T x \quad \forall j$$

- models:
  $$w_1, w_2, \ldots, w_K, \quad w_k \in \mathbb{R}^n$$

- Input:
  $$x \in \mathbb{R}^n$$

$$w^T \phi(x, i) > w^T \phi(x, j) \quad \forall j$$

- Only one model:
  $$w \in \mathbb{R}^{n \times K}$$

\[
\begin{bmatrix}
w_1 \\
w_2 \\
\vdots \\
w_K
\end{bmatrix}_{nK \times 1}
\]
Kesler construction

Assume we have a multi-class problem with K class and n features.

\[ w_i^T x > w_j^T x \quad \forall j \]

- **models:**
  \[ w_1, w_2, \ldots, w_K, \quad w_k \in \mathbb{R}^n \]

- **Input:**
  \[ x \in \mathbb{R}^n \]

\[ w^T \phi(x, i) > w^T \phi(x, j) \quad \forall j \]

- **Only one model:**
  \[ w \in \mathbb{R}^{nK} \]

- Define \( \phi(x, y) \) for label \( y \) being associated to input \( x \)

\[
\phi(x, y) = \begin{bmatrix}
0^n \\
\vdots \\
x \\
\vdots \\
0^n
\end{bmatrix}_{nK \times 1}
\]

\( x \) in \( y^{th} \) block; Zeros elsewhere
Kesler construction

Assume we have a multi-class problem with K class and n features.

\[ w_i^T x > w_i^T x \quad \forall \ i \]

- **models:**
  \[ w_1, w_2, \ldots, w_K, \]
  \[ w_k \in \mathbb{R}^n \]

- **Input:**
  \[ x \in \mathbb{R}^n \]

\[ w^T \phi(x, i) > w^T \phi(x, j) \quad \forall \ j \]

\[ w = \begin{bmatrix} w_1 \\ \vdots \\ w_y \\ \vdots \\ w_n \end{bmatrix}_{nK \times 1} \]

\[ \phi(x, y) = \begin{bmatrix} x \\ \vdots \\ 0_n \end{bmatrix}_{nK \times 1} \]

\[ w^T \phi(x, y) = w_y^T x \]

x in y\textsuperscript{th} block; Zeros elsewhere
Kesler construction

Assume we have a multi-class problem with K class and n features.

\[ w^T \phi(x, i) > w^T \phi(x, j) \quad \forall j \]
\[ \Rightarrow w^T[\phi(x, i) - \phi(x, j)] > 0 \quad \forall j \]

\[ w = \begin{bmatrix} w_1 \\ \vdots \\ w_K \end{bmatrix}_{nK \times 1} \]
\[ [\phi(x, i) - \phi(x, j)] = \begin{bmatrix} 0_n \\ \vdots \\ x \\ \vdots \\ -x \\ \vdots \\ 0_n \end{bmatrix}_{nK \times 1} \]

x in \( i^{th} \) block;

\(-x\) in \( j^{th} \) block;
Linear Separability with multiple classes

What we want

$$\mathbf{w}^T \phi(x, i) > \mathbf{w}^T \phi(x, j) \quad \forall j$$

$$\Rightarrow \mathbf{w}^T [\phi(x, i) - \phi(x, j)] > 0 \quad \forall j$$

For all example $$(x, y)$$ with all other labels $$y'$$ in dataset, $$\mathbf{w}$$ in $$nK$$ dimension should linearly separate $$\phi(x, i) - \phi(x, j)$$ and $$- [\phi(x, i) - \phi(x, j)]$$
How can we predict?

\[ \text{argmax}_y \ w^T \phi(x, y) \]

\[ w = \begin{bmatrix} w_1 \\ \vdots \\ w_y \\ \vdots \\ w_n \end{bmatrix}_{nK \times 1} \]

\[ \phi(x, y) = \begin{bmatrix} 0_n \\ \vdots \\ x \\ \vdots \\ 0_n \end{bmatrix}_{nK \times 1} \]

For input an input \( x \), the model predict label is 3.
How can we predict?

\[
\text{argmax}_y \ w^T \phi(x, y)
\]

\[
w = \begin{bmatrix} w_1 \\ \vdots \\ w_y \\ \vdots \\ w_n \end{bmatrix}_{nK \times 1} \quad \phi(x, y) = \begin{bmatrix} 0_n \\ \vdots \\ x \\ \vdots \\ 0_n \end{bmatrix}_{nK \times 1}
\]

This is equivalent to

\[
\text{argmax}_{y \in \{1, 2, \ldots, K\}} \ w_y^T \ x
\]

For input an input x, the model predict label is 3.
Goal: \( w[\phi(x, i) - \phi(x, j)] \geq 0 \ \forall j \)

Training:

- For each example \((x, i)\)
- Update model if \( w^T[\phi(x, i) - \phi(x, j)] < 0 , \forall j \)
A Perceptron-style Algorithm

Given a training set $\mathcal{D} = \{(x, y)\}$

Initialize $w \leftarrow 0 \in \mathbb{R}^n$

For epoch $1 \ldots T$:
  For $(x, y)$ in $\mathcal{D}$:
    For $y' \neq y$
      if $w^T [\phi(x, y) - \phi(x, y')] < 0$
        $w \leftarrow w + \eta [\phi(x, y) - \phi(x, y')]$

Return $w$

Prediction: $\arg\max_y w^T \phi(x, y)$
An alternative training algorithm

- **Goal:** \( w[\phi(x, i) - \phi(x, j)] \geq 0 \quad \forall j \)
  \( \Rightarrow w^T \phi(x, i) - \max_{j \neq i} [w^T \phi(x, i)] \geq 0 \)

- **Training:**
  - For each example \((x, i)\)
    - Find the prediction of the current model:
      \[ \hat{y} = \arg\max_j w^T \phi(x, j) \]
    - Update model if \( w^T [\phi(x, i) - \phi(x, \hat{y})] < 0 , \forall y' \)
A Perceptron-style Algorithm

Given a training set $\mathcal{D} = \{(x, y)\}$

Initialize $w \leftarrow 0 \in \mathbb{R}^n$

For epoch 1...$T$:

For $(x, y)$ in $\mathcal{D}$:

$\hat{y} = \text{argmax}_y \, w^T \phi(x, y')$

if $w^T [\phi(x, y) - \phi(x, \hat{y})] < 0$

$w \leftarrow w + \eta [\phi(x, y) - \phi(x, \hat{y})]$

Return $w$

How to interpret this update rule?

Prediction: $\text{argmax}_y \, w^T \phi(x, y)$
A Perceptron-style Algorithm

Given a training set $\mathcal{D} = \{(x, y)\}$

Initialize $w \leftarrow 0 \in \mathbb{R}^n$

For epoch $1 \ldots T$:

For $(x, y)$ in $\mathcal{D}$:

$$\hat{y} = \text{argmax}_y w^T \phi(x, y)$$

if $w^T [\phi(x, y) - \phi(x, \hat{y})] < 0$

$$w \leftarrow w + \eta [\phi(x, y) - \phi(x, \hat{y})]$$

Return $w$

How to interpret this update rule?

Prediction: $\text{argmax}_y w^T \phi(x, y)$
A Perceptron-style Algorithm

Given a training set $\mathcal{D} = \{(x, y)\}$

Initialize $w \leftarrow 0 \in \mathbb{R}^n$

For epoch 1...$T$:

For $(x, y)$ in $\mathcal{D}$:

$\hat{y} = \text{argmax}_y, \; w^T \phi(x, y')$

$w \leftarrow w + \eta [\phi(x, y) - \phi(x, \hat{y})]$

Return $w$

How to interpret this update rule?

Prediction: $\text{argmax}_y \; w^T \phi(x, y)$
Consider multiclass margin

In terms of Kesler construction

$$\min_{y' \neq y} \mathbf{w}^T [\phi(x, y) - \phi(x, y')]$$

Here $y$ is the label that has the highest score
Marginal constraint classifier

❖ Goal: for every \((x,y)\) in the training data set

\[
\min_{y' \neq y} w^T [\phi(x, y) - \phi(x, y')] \geq \delta \\
\Rightarrow w^T \phi(x, y) - \max_{y \neq y'} w^T \phi(x, y') \geq \delta \\
\Rightarrow w^T \phi(x, i) - \left[ \max_{y \neq y'} w^T \phi(x, j) + \delta \right] \geq 0
\]

Constraints violated \(\Rightarrow\) need an update

Let’s define:

\[
\Delta(y, y') = \begin{cases} 
\delta & \text{if } y \neq y' \\
0 & \text{if } y = y' 
\end{cases}
\]

Check if

\[
y = \arg \max_{y'} w^T \phi(x, y') + \Delta(y, y')
\]
A Perceptron-style Algorithm

Given a training set \( \mathcal{D} = \{(x, y)\} \)

Initialize \( \mathbf{w} \leftarrow 0 \in \mathbb{R}^n \)

For epoch 1...\( T \):

For \( (x, y) \) in \( \mathcal{D} \):

\[
\hat{y} = \text{argmax}_y, \ w^T \phi(x, y') + \Delta(y, y')
\]

\[
\mathbf{w} \leftarrow \mathbf{w} + \eta [\phi(x, y) - \phi(x, \hat{y})]
\]

Return \( \mathbf{w} \)

How to interpret this update rule?

Prediction: \( \text{argmax}_y \ w^T \phi(x, y) \)
Remarks

- This approach can be generalized to train a ranker; in fact, any output structure
  - We have preference over label assignments
  - E.g., rank search results, rank movies / products
This Lecture

- Multiclass classification overview
- Reducing multiclass to binary
  - One-against-all & One-vs-one
  - Error correcting codes
- Training a single classifier
  - Multiclass Perceptron: Kesler’s construction
  - Multiclass SVMs: Crammer&Singer formulation
  - Multinomial logistic regression
Recall: Margin for binary classifiers

- The **margin** of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.
Multi-class SVM

- In a risk minimization framework
- Goal: \( D = \{ (x_i, y_i) \}_{i=1}^N \)
  1. \( w_{y_i}^T x_i > w_{y'}^T x_i \) for all \( i, y' \)
  2. Maximizing the margin

Defined as the score difference between the highest scoring label and the second one.
Multiclass Margin

Defined as the score difference between the highest scoring label and the second one

Score for a label = $w_{\text{label}}^T x$

- Blue
- Red
- Green
- Black

Labels
Multiclass SVM (Intuition)

- Binary SVM
  - Maximize margin. Equivalently,
    Minimize norm of weights such that the closest points to the hyperplane have a score 1

- Multiclass SVM
  - Each label has a different weight vector (like one-vs-all)
  - Maximize multiclass margin. Equivalently,
    Minimize total norm of the weights such that the true label is scored at least 1 more than the second best one
Multiclass SVM in the separable case

Recall hard binary SVM

$$\begin{align*}
\min_{\mathbf{w}} & \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} \\
\text{s.t.} & \quad \forall i, \quad y_i \mathbf{w}^T \mathbf{x}_i \geq 1
\end{align*}$$

Size of the weights. Effectively, regularizer

$$\begin{align*}
\min_{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K} & \quad \frac{1}{2} \sum_k \mathbf{w}_k^T \mathbf{w}_k \\
\text{s.t.} & \quad \mathbf{w}_{y_i}^T \mathbf{x} - \mathbf{w}_k^T \mathbf{x} \geq 1, \quad \forall (\mathbf{x}_i, y_i) \in D, \\
& \quad k \in \{1, 2, \ldots, K\}, k \neq y_i, \\
& \quad \forall i.
\end{align*}$$

The score for the true label is higher than the score for any other label by 1

The score for the true label is higher than the score for any other label by 1
Multiclass SVM: General case

\[ \min_{w_1, w_2, \ldots, w_K} \frac{1}{2} \sum_k w_k^T w_k \]

s.t. \[ w_{y_i}^T x - w_k^T x \geq 1 \]

\[ \forall (x_i, y_i) \in D, \quad k \in \{1, 2, \ldots, K\}, k \neq y_i, \]

The score for the true label is higher than the score for any other label by 1

Slack variables. Not all examples need to satisfy the margin constraint.

Slack variables can only be positive

Size of the weights. Effectively, regularizer

Total slack. Effectively, don’t allow too many examples to violate the margin constraint.
Multiclass SVM: General case

The score for the true label is higher than the score for any other label by \( 1 - \|w\|_i \)

Size of the weights. Effectively, regularizer

\[
\begin{align*}
\min_{w_1, w_2, \ldots, w_K, \xi} & \quad \frac{1}{2} \sum_k w_k^T w_k + C \sum_{(x_i, y_i) \in D} \xi_i \\
\text{s.t.} & \quad w_{y_i}^T x - w_k^T x \geq 1 - \xi_i, \quad \forall (x_i, y_i) \in D, \\
& \quad \xi_i \geq 0, \quad k \in \{1, 2, \ldots, K\}, k \neq y_i, \\
& \quad \forall i.
\end{align*}
\]

Total slack. Effectively, don’t allow too many examples to violate the margin constraint

Slack variables. Not all examples need to satisfy the margin constraint.

Slack variables can only be positive
Recap: An alternative SVM formulation

\[
\begin{align*}
\min_{w,b,\xi} & \quad \frac{1}{2} w^T w + C \sum_i \xi_i \\
\text{s.t} & \quad y_i(w^T x_i + b) \geq 1 - \xi_i; \quad \xi_i \geq 0 \quad \forall i
\end{align*}
\]

- Rewrite the constraints:

  \[
  \xi_i \geq 1 - y_i(w^T x_i + b); \quad \xi_i \geq 0 \quad \forall i
  \]

- In the optimum, \( \xi_i = \max(0, 1 - y_i(w^T x_i + b)) \)

- Soft SVM can be rewritten as:

  \[
  \min_{w,b} \quad \frac{1}{2} w^T w + C \sum \max(0, 1 - y_i(w^T x_i + b))
  \]

\(\text{Regularization term}\)

\(\text{Empirical loss}\)
Rewrite it as unconstraint problem

\[
\min_{w_1,w_2,\ldots,w_K,\xi} \quad \frac{1}{2} \sum_{k} w_k^T w_k + C \sum_{(x_i,y_i) \in D} \xi_i \\
\text{s.t.} \quad w_{y_i}^T x - w_k^T x \geq 1 - \xi_i, \quad \forall (x_i,y_i) \in D, \\
\quad k \in \{1,2,\ldots,K\}, k \neq y_i, \\
\quad \xi_i \geq 0, \quad \forall i.
\]

Let's define:

\[
\Delta(y,y') = \begin{cases} 
\delta & \text{if } y \neq y' \\
0 & \text{if } y = y'
\end{cases}
\]

\[
\min_w \frac{1}{2} \sum_k w_k^T w_k + C \sum_i (\max_k (\Delta(y_i,k) + w_k^T x) - w_{y_i}^T x )
\]
Multiclass SVM

- Generalizes binary SVM algorithm
  - If we have only two classes, this reduces to the binary (up to scale)

- Comes with similar generalization guarantees as the binary SVM

- Can be trained using different optimization methods
  - Stochastic sub-gradient descent can be generalized
Exercise!

- Write down SGD for multiclass SVM

- Write down multiclass SVM with Kesler construction
This Lecture

- Multiclass classification overview
- Reducing multiclass to binary
  - One-against-all & One-vs-one
  - Error correcting codes
- Training a single classifier
  - Multiclass Perceptron: Kesler’s construction
  - Multiclass SVMs: Crammer&Singer formulation
  - Multinomial logistic regression
Recall: (binary) logistic regression

\[ \min_w \frac{1}{2} w^T w + C \sum_i \log(1 + e^{-y_i(w^T x_i)}) \]

Assume labels are generated using the following probability distribution:

\[ P(y = 1|x, w) = \frac{e^{w^T x}}{1 + e^{w^T x}} = \frac{1}{1 + e^{-w^T x}} \]
\[ P(y = -1|x, w) = \frac{1}{1 + e^{w^T x}} \]
(multi-class) log-linear model

Assumption:

\[
P(y|x, w) = \frac{\exp(w_y^T x)}{\sum_{y' \in \{1, 2, \ldots, K\}} \exp(w_{y'}^T x)}
\]

This is a valid probability assumption. Why?

Another way to write this (with Kesler construction) is

\[
P(y|x, w) = \frac{\exp(w^T \phi(x, y))}{\sum_{y' \in \{1, 2, \ldots, K\}} \exp(w^T \phi(x, y'))}
\]
Softmax

- Softmax: let $s(y)$ be the score for output $y$ here $s(y)=w^T \phi(x, y)$ (or $w_y^T x$) but it can be computed by other metric.

$$P(y) = \frac{\exp(s(y))}{\sum_{y' \in \{1, 2, \ldots, K\}} \exp(s(y'))}$$

- We can control the peakedness of the distribution

$$P(y | \sigma) = \frac{\exp(s(y)/\sigma)}{\sum_{y' \in \{1, 2, \ldots, K\}} \exp(s(y'/\sigma))}$$
Example

\[ S(\text{dog}) = 0.5; \quad s(\text{cat}) = 1; \quad s(\text{mouse}) = 0.3; \quad s(\text{duck}) = -0.2 \]
Log linear model

\[ P(y|x, w) = \frac{\exp(w_y^T x)}{\sum_{y' \in \{1, 2, \ldots, K\}} \exp(w_{y'}^T x)} \]

\[ \log P(y|x, w) = \log(\exp(w_y^T x)) - \log(\sum_{y' \in \{1, 2, \ldots, K\}} \exp(w_{y'}^T x)) \]
\[ = w_y^T x - \log(\sum_{y' \in \{1, 2, \ldots, K\}} \exp(w_{y'}^T x)) \]

Note:

\[ p(y) \propto \exp(\theta^T f(y)) \]
Maximum log-likelihood estimation

- Training can be done by maximum log-likelihood estimation i.e. $\max_w \log P(D|w)$

\[ D = \{(x_i, y_i)\} \]

\[ P(D|w) = \prod_i \frac{\exp(w_{y_i}^T x_i)}{\sum_{y' \in \{1, 2, \ldots, K\}} \exp(w_{y'}^T x_i)} \]

\[ \log P(D|w) = \sum_i [w_{y_i}^T x_i - \log \sum_{y' \in \{1, 2, \ldots, K\}} \exp(w_{y'}^T x_i)] \]
Maximum a posteriori

\[ D = \{(x_i, y_i)\} \]

\[ P(w|D) \propto P(w)P(D|w) \]

\[ \max_w - \frac{1}{2} \sum_y w_y^T w_y + C \sum_i [w_{y_i}^T x_i - \log \sum_{y \in \{1, 2, \ldots, K\}} \exp (w_{y_i}^T x_i)] \]

or

\[ \min_w \frac{1}{2} \sum_y w_y^T w_y + C \sum_i [\log \sum_{y \in \{1, 2, \ldots, K\}} \exp (w_{y_i}^T x_i) - w_{y_i}^T x_i] \]
Comparisons

- Multi-class SVM:
  \[
  \min_\mathbf{w} \frac{1}{2} \sum_k \mathbf{w}_k^T \mathbf{w}_k + C \sum_i \left( \max \left( \Delta(y_i, k) + \mathbf{w}_k^T \mathbf{x} - \mathbf{w}_{y_i}^T \mathbf{x} \right) \right)
  \]

- Log-linear model w/ MAP (multi-class)
  \[
  \min_\mathbf{w} \frac{1}{2} \sum_k \mathbf{w}_k^T \mathbf{w}_k + C \sum_i \left[ \log \sum_{k \in \{1, 2, \ldots, K\}} \exp \left( \mathbf{w}_k^T \mathbf{x}_i \right) - \mathbf{w}_{y_i}^T \mathbf{x}_i \right]
  \]

- Binary SVM:
  \[
  \min_\mathbf{w} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \max(0, 1 - y_i (\mathbf{w}^T \mathbf{x}_i))
  \]

- Log-linear mode (logistic regression)
  \[
  \min_\mathbf{w} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \log \left( 1 + e^{-y_i (\mathbf{w}^T \mathbf{x}_i)} \right)
  \]