Classification

Dan Lizotte

2018-10-16

Classification

• Space of outputs $\mathcal{Y}$ is finite. Often classes are given numbers starting from 0 or 1.

• Usually no notion of “similarity” between class labels in terms of loss. Remember our loss function $\ell(h(\mathbf{x}), y)$:
  - Regression: $\ell(9, 10)$ is better than $\ell(1, 10)$
  - Classification: $\ell(9, 10)$ and $\ell(1, 10)$ are equally bad.
    * Or, have explicit losses for every combination of predicted class and actual class.

“Linear models” in general (HTF Ch. 2.8.3)

• By linear models, we mean that the hypothesis function $h_\mathbf{w}(\mathbf{x})$ is a (transformed) linear function of the parameters $\mathbf{w}$.

• Predictions are a (transformed) linear combination of feature values

$$h_\mathbf{w}(\mathbf{x}) = g \left( \sum_{k=0}^{p} w_k \phi_k(\mathbf{x}) \right) = g(\phi(\mathbf{x})^T \mathbf{w})$$

• again, $\phi_k$ are called basis functions or feature functions As usual, we let $\phi_0(\mathbf{x}) = 1, \forall \mathbf{x}$, so that we don’t force $h_\mathbf{w}(0) = 0$

Linear Methods for Classification

• Classification tasks
  • Loss functions for classification
  • Logistic Regression
  • Support Vector Machines

Wisconsin Breast Cancer Prognostic Data

Cell samples were taken from tumors in breast cancer patients before surgery and imaged; tumors were excised; patients were followed to determine whether or not the cancer recurred, and how long until recurrence or disease free.
Wisconsin data (continued)

- 198 instances, 32 features for prediction
- Outcome (R=recurrence, N=non-recurrence)
- Time (until recurrence, for R, time healthy, for N).

Example: Given nucleus radius, predict cancer recurrence

```r
ggplot(bc, aes(Radius.Mean, fill=Outcome, color=Outcome)) + geom_density(alpha=1/2)
```
Example: Solution by linear regression

- Univariate real input: nucleus size
- Output coding: non-recurrence = 0, recurrence = 1
- Sum squared error minimized by the blue line
Linear regression for classification

- The predictor shows an increasing trend towards recurrence with larger nucleus size, as expected.
- Output cannot be directly interpreted as a class prediction.
- Thresholding output (e.g., at 0.5) could be used to predict 0 or 1.
  (In this case, prediction would be 0 except for extremely large nucleus size.)

Probabilistic view

- Suppose we have two possible classes: \( y \in \{0, 1\} \).
- The symbols “0” and “1” are unimportant. Could have been \( \{a, b\} \), \( \{up, down\} \), whatever.
- Rather than try to predict the class label directly, ask:
  What is the probability that a given input \( x \) has class \( y = 1 \)?
Aside: Relationships Between Random Variables

Conditional distributions
Conditional distributions

![Diagram showing conditional distributions with axes labeled as eruptions on the x-axis and waiting on the y-axis. The diagram includes two data clusters with varying distributions.]
Conditional distributions

\[ f(\text{eruptions}|\text{waiting}=60) \]
Conditional distributions

![Graph showing conditional distributions with x-axis for eruptions and y-axis for waiting times, with data points clustered in certain regions.]
Conditional distributions

\[ f(\text{eruptions}|\text{waiting}=80) \]
Conditional distributions

\[ f(\text{waiting} \mid 2 \leq \text{eruptions} \leq 2.1) \]
Conditional distributions
Conditional distributions

\[ f(\text{waiting} | 4.4 \leq \text{eruptions} \leq 4.5) \]
Predicting Waiting Time

## Mean: 70.90

### Conditional predictions

- If I know eruption time, can I do better?
## Mean: 55.60

**Conditional predictions**

- If I know eruption time, can I do better?
Conditional probability functions

Strategy: Assume that the probability $P(y = 1|X = x)$ is given by some function $h(x)$. Then find a function that “fits” the data. What kind of function do we use for $P(y = 1|X = x)$?

Idea: $h_w(x) = w^T x$ Why? Why not?

## Mean: 81.33
Sigmoid function

\[ \varsigma(x) = \frac{1}{1 + e^{-x}} \]
Logistic Regression HTF (Ch. 4.4)

- Represent the hypothesis as a logistic function of a linear combination of inputs, interpret $h(x)$ as $P(y = 1 | X = x)$:
  \[ h_w(x) = \varsigma(x^T w) \]

- \( \varsigma(a) = \frac{1}{1 + \exp(-a)} \) is the sigmoid or logistic function

- With a little algebra, we can write:
  \[ P(y = 1 | X = x) = \varsigma \left( \log \frac{P(y = 1 | X = x)}{P(y = 0 | X = x)} \right) \]
  - Interpret $x^T w$ as the log-odds

Logistic regression training HTF (Ch. 4.4)

- How do we choose $w$?
- In the probabilistic framework, observing $(x_i, 1)$ does not mean $h_w(x_i)$ should be as close to 1 as possible.
- Maximize probability the model assigns to the $y_i$ in the training set given the $x_i$ by adjusting $w$. 

18
Reminder: Independence

- Two random variables \( X \) and \( Y \) that are part of a random vector are independent iff:

\[
F_{X,Y}(x,y) = F_X(x)F_Y(y)
\]

If they have a joint density or joint PMF, then

\[
f_{X,Y}(x,y) = f_X(x)f_Y(y)
\]

Max Conditional Likelihood

- Maximize probability the model assigns to the \( y_i \) in the training set *given the* \( x_i \) by adjusting \( w \).
- Assumption 1: Examples are i.i.d. Probability of observing all \( y \)'s is product

\[
P(\text{all } y| \text{all } x) = P(Y_1 = y_1, Y_2 = y_2, ..., Y_n = y_n|X_1 = x_1, X_2 = x_2, ..., X_n = x_n)
\]

\[
= \prod_{i=1}^{n} P(Y_i = y_i|X_i = x_i = x_i)
\]

- Assumption 2: \( P(y = 1|X = x) = h_w(x) = \frac{1}{1 + \exp(-x^T w)} \)
  \( P(y = 0|X = x) = (1 - h_w(x)) \)

Max Conditional Likelihood

- Maximize probability the model assigns to the \( y_i \) in the training set *given the* \( x_i \) by adjusting \( w \).
- More numerically stable to maximize log probability. Note

\[
\log P(Y_i = y_i|X_i = x_i) = \begin{cases} 
\log h_w(x_i) & \text{if } y_i = 1 \\
\log(1 - h_w(x_i)) & \text{if } y_i = 0 
\end{cases}
\]

- Therefore,

\[
\log \prod_{i=1}^{n} P(Y_i = y_i|X_i = x_i) = \sum_{i=1}^{n} [y_i \log(h_w(x_i)) + (1 - y_i) \log(1 - h_w(x_i))]
\]

- Suggests an error

\[
J(h_w) = - \sum_{i=1}^{n} [y_i \log(h_w(x_i)) + (1 - y_i) \log(1 - h_w(x_i))]
\]

- This is the cross entropy. Number of bits to transmit \( y_i \) if both parties know \( h_w \) and \( x_i \).
Back to the breast cancer problem

Logistic Regression:

```
## (Intercept) Radius.Mean
## -3.4671348 0.1296493
```

Least Squares:

```
## (Intercept) Radius.Mean
## -0.17166939 0.02349159
```

Probability and Expectation

- Why are these so close?
- Recall the expected value of a discrete random variable $Y$ is denoted

$$E[Y] = \sum_{y \in Y} y \cdot p_Y(Y = y)$$

- Consider a random variable $Y \in \{0, 1\}$

$$E[Y] = \sum_{y \in \{0, 1\}} y \cdot p_Y(Y = y)$$

$$= 0 \cdot p_Y(Y = 0) + 1 \cdot p_Y(Y = 1)$$

$$= p_Y(Y = 1)$$
- Though we did not discuss in this way, linear regression tries to estimate the function $E[Y|X = x]$. So, makes sense that the OLS and logistic regression answers can be close.

### Supervised Learning Methods: “Objective-driven”

<table>
<thead>
<tr>
<th>Mthd.</th>
<th>Form</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLS</td>
<td>$h_w(x) = x^T w$</td>
<td>$\sum_{i=1}^n (h_w(x_i) - y_i)^2$</td>
</tr>
<tr>
<td></td>
<td>$\approx E[Y</td>
<td>X = x]...$</td>
</tr>
<tr>
<td>LR</td>
<td>$h_w(x) = \frac{1}{1+e^{-x^T w}}$</td>
<td>$-\sum_{i=1}^n y_i \log h_w(x_i) + (1 - y_i) \log(1 - h_w(x_i))$</td>
</tr>
<tr>
<td></td>
<td>$\approx P(Y = y</td>
<td>X = x)...$</td>
</tr>
</tbody>
</table>

- Both model the **conditional mean of $y$** using a (transformed) **linear function**
- Both use **maximum conditional likelihood** to estimate

### Decision boundary HTF Ch. 2.3.1,2.3.2

- How complicated is a classifier?
  - One way to think about it is in terms of its **decision boundary**, i.e. the line it defines for separating examples
  - **Linear classifiers** draw a hyperplane between examples of the different classes. **Non-linear classifiers** draw more complicated surfaces between the different classes.
  - For a probabilistic classifier with a cutoff of 0.5, the decision boundary is the curve on which:
    
    $\frac{P(y = 1|X = x)}{P(y = 0|X = x)} = 1$, i.e., where $\log \frac{P(y = 1|X = x)}{P(y = 0|X = x)} = 0$

- For logistic regression, this this is where $x^T w = 0$.

### Decision boundaries of linear classifiers

- Recall: predictions are a (transformed) **linear combination of feature values**
  
  $h_w(x) = g(x^T w)$

- Suppose our decision boundary is
  
  $h_w(x) = c$

- This is equivalent to
  
  $x^T w = c'$

where $c' = g^{-1}(c)$.
Decision boundary

Class = R if $\Pr(Y = 1|X = x) > 0.5$

Decision boundary

Class = R if $\Pr(Y = 1|X = x) > 0.25$
Decision boundary

Class = R if $\Pr(Y = 1|X = x) > 0.5$
Decision boundary

Class = R if $\Pr(Y = 1|X = x) > 0.25$
Supervised Learning Methods: “Objective-driven”

<table>
<thead>
<tr>
<th>Mthd.</th>
<th>Form</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLS</td>
<td>( h_w(x) = x^T w )</td>
<td>( \sum_{i=1}^n (h_w(x_i) - y_i)^2 ) using a linear function</td>
</tr>
<tr>
<td></td>
<td>( \approx E[Y</td>
<td>X = x] )</td>
</tr>
<tr>
<td>LR</td>
<td>( h_w(x) = \frac{1}{1 + e^{-x^T w}} )</td>
<td>( -\sum_{i=1}^n y_i \log h_w(x_i) + (1 - y_i) \log(1 - h_w(x_i)) )</td>
</tr>
<tr>
<td></td>
<td>( \approx P(Y = y</td>
<td>X = x) )</td>
</tr>
<tr>
<td>SVM</td>
<td>( h_w(x) = \text{sgn}(x^T w) )</td>
<td></td>
</tr>
</tbody>
</table>

Large Margin Classifiers:

Linear Support Vector Machines

- Linear classifiers that focus on learning the decision boundary rather than the conditional distribution \( P(Y = y|X = x) \)
  - Perceptrons
    * Definition
    * Perceptron learning rule
    * Convergence

25
– “Margin” idea and max margin classifiers
– (Linear) support vector machines
  * Formulation as optimization problem

Marvin Minsky, 1927-2016

Perceptrons HTF Ch. 4.5

- Consider a binary classification problem with data \( \{x_i, y_i\}_{i=1}^n \), \( y_i \in \{-1, +1\} \). **Note coding of** \( y_i \).
- A perceptron (Rosenblatt, 1957) is a classifier of the form:
  \[
  h_{w,w_0}(x) = \text{sign}(x^T w + w_0) = \begin{cases} 
  +1 & \text{if } x^T w + w_0 \geq 0 \\
  -1 & \text{otherwise}
  \end{cases}
  \]
  Here, \( w \) is a vector of weights, and \( w_0 \) is a constant offset. (**Note** \( x_0 = 1 \) **is omitted.**)
- The decision boundary is \( x^T w + w_0 = 0 \).
- Perceptrons output a class, not a probability
- An example \((x, y)\) is classified correctly if:
  \[
  y \cdot (x^T w + w_0) > 0
  \]

Linear separability

- The data set is **linearly separable** if and only if there exists \( w, w_0 \) such that:
  - For all \( i \), \( y_i(x_i^T w + w_0) > 0 \).
  - Or equivalently, the 0-1 loss \( \sum_i 1_{y_i(x_i^T w + w_0) < 0} \) is zero for some set of parameters \((w, w_0)\).
The Perceptron Learning Rule

- Consider the following procedure:
  1. Initialize $\mathbf{w}$ and $w_0$ randomly
  2. While any training examples remain incorrectly classified
     1. Loop through all misclassified examples
     2. For misclassified example $i$, perform the updates:
        \[
        \mathbf{w} \leftarrow \mathbf{w} + \delta y_i \mathbf{x}_i, \quad w_0 \leftarrow w_0 + \delta y_i
        \]
        where $\delta$ is a step-size parameter.
- The update equation, or sometimes the whole procedure, is called the perceptron learning rule.
- Intuition: For positive examples misclassified as negative, change $\mathbf{w}$ to increase $\mathbf{x}_i^T \mathbf{w} + w_0$, and vice versa
Error Minimization Interpretation

- PLR can be interpreted as a gradient descent on the following function:

\[
J(w, w_0) = \sum_{i=1}^{n} \begin{cases} 
0 & \text{if } y_i(x_i^T w + w_0) \geq 0 \\
-y_i(x_i^T w + w_0) & \text{if } y_i(x_i^T w + w_0) < 0
\end{cases}
\]

- For correctly classified examples, the error is zero.
- For incorrectly classified examples, the error is by how much \( x_i^T w + w_0 \) is on the wrong side of the decision boundary.
- \( J \) is piecewise linear, so it has a gradient almost everywhere; stochastic gradient descent gives the perceptron learning rule.
- \( J \) is zero if and only if all examples are classified correctly – just like the 0-1 loss function.

Perceptron convergence theorem

- **If** classes are linearly separable then the perceptron learning rule will find a separator after some finite number of updates.
- The number of updates depends on the data set, and also on the step size parameter.
- If the classes are not linearly separable, there will be oscillation (which can be detected automatically).
Perceptron Learning Example

Update: 0.5  \( w: [1.000, 1.000] \), \( w_0: 1.000 \)
Perceptron Learning Example

Update: 1.0  $w: [0.628, 0.788]$, $w_0: 0.000$
Perceptron Learning Example

Update: 1.5  \( w: [0.628, 0.788], w_0: 0.000 \)
Perceptron Learning Example

Update: 2.0  \( w: [0.426, 0.521] \), \( w_0: -1.000 \)
Perceptron Learning Example

Update: 2.5  \( w: [0.426, 0.521], w_0: -1.000 \)
Perceptron Learning Example

Update: 3.0  w: [1.325, 0.907], w0: 0.000
Perceptron Learning Example

Update: 3.5  w: [1.325, 0.907], w0: 0.000
Perceptron Learning Example

Update: 4.0  w: [0.380, 0.893], w0: -1.000
Perceptron Learning Example

Update: 4.5  $w: [0.380, 0.893]$, $w_0: -1.000$
Perceptron Learning Example

Update: 5.0  \( w: [1.041, 1.276] \), \( w_0: 0.000 \)
Perceptron Learning Example

Update: 5.5  w: [1.041, 1.276], w0: 0.000

yhat

y

x1

x2

yhat

y

1

−1

1

−1

Update: 5.5  w: [1.041, 1.276], w0: 0.000
Weight as a combination of input vectors

- Recall perceptron learning rule:
  \[ w \leftarrow w + \delta y_i x_i, \quad w_0 \leftarrow w_0 + \delta y_i \]

- If initial weights are zero, then at any step, the \textit{weights are a linear combination of feature vectors of the examples}:
  \[ w = \sum_{i=1}^{n} \alpha_i y_i x_i, \quad w_0 = \sum_{i=1}^{n} \alpha_i y_i \]

  where \( \alpha_i \) is the sum of step sizes used for all updates based on example \( i \).

- This is called the \textit{dual representation} of the classifier.

- Even by the end of training, some examples may have never participated in an update, just by chance. So their corresponding \( \alpha_i = 0 \).
Examples used (bold) and not used (faint) in updates

\[ w = [4.111 \ 3.8704] \quad w_0 = -4 \]
Comment: Solutions are nonunique

\[ w = [2.1395 \ 1.9372] \quad w_0 = -2 \]

Perceptron summary

- Perceptrons can be learned to fit linearly separable data, using a gradient descent rule.
- Blindingly fast
- Solutions are non-unique

Support Vector Machines

- Support vector machines (SVMs) for binary classification can be viewed as a way of training perceptrons
- Three main new ideas:
  - A optimization criterion (the “margin”) guarantees uniqueness and has theoretical advantages
  - Natural handling nonseparable data by allowing mistakes
  - An efficient way of operating in expanded feature spaces: “kernel trick”
• SVMs can also be used for multiclass classification and regression.

Returning to the non-uniqueness issue

• Consider a linearly separable binary classification data set
• There is an infinite number of hyperplanes that separate the classes:

![Diagram of hyperplanes and data points]

• Which plane is best?
• For a given plane, for which points should we be most confident in the classification?

The margin, and linear SVMs

• For a given separating hyperplane, the margin is two times the (Euclidean) distance from the hyperplane to the nearest training example.
• Width of the “strip” around the decision boundary containing no training examples.
• A linear SVM is a perceptron for which we choose $w, w_0$ so that margin is maximized

**Distance to the decision boundary**

• Suppose we have a decision boundary that separates the data.
Let $\gamma_i$ be the distance from instance $x_i$ to the decision boundary.

How can we write $\gamma_i$ in terms of $x_i, y_i, w, w_0$?

Distance to the decision boundary (II)

• $w$ is orthogonal to boundary, $\frac{w}{||w||}$ is the unit vector orthogonal to the boundary
• Vector from B to $x_i$ is $\gamma_i \frac{w}{||w||}$.
• B, the point on the boundary nearest $x_i$, is $x_i - \gamma_i \frac{w}{||w||}$.
• Since B is on the boundary,
$$\left( x_i - \gamma_i \frac{w}{||w||} \right)^T w + w_0 = 0$$
• Solving for $\gamma_i$ yields

$$\gamma_i = \frac{x_i^T w + w_0}{\|w\|}$$

The margin HTF Ch. 4.5, Ch 12

• The margin of the hyperplane is $2M$, where $M = \min_i y_i \gamma_i$

• The most direct statement of the problem of finding a maximum margin separating hyperplane is thus

$$\max \min_{w, w_0} y_i \gamma_i$$

$$\equiv \max \min_{w, w_0} \frac{x_i^T w + w_0}{\|w\|}$$

• This turns out to be inconvenient for optimization, however

Treating the $\gamma_i$ as constraints

• From the definition of margin, we have:

$$M \leq y_i \gamma_i = y_i \frac{x_i^T w + w_0}{\|w\|} \quad \forall i$$

• This suggests:

| maximize $M$ with respect to $M, w, w_0$ | subject to $M \leq y_i \frac{x_i^T w + w_0}{\|w\|}$ for all $i$ |

• Problems:
  
  – $w$ appears nonlinearly in the constraints.
  
  – This problem is underconstrained. If $(w, w_0, M)$ is an optimal solution, then so is $(\beta w, \beta w_0, M)$ for any $\beta > 0$.

Adding a constraint

Let’s add the constraint that $M = 1/\|w\|:$

• This allows us to rewrite the objective function:

| maximize $\frac{1}{\|w\|}$ with respect to $w, w_0$ | subject to $\frac{1}{\|w\|} \leq y_i \frac{x_i^T w + w_0}{\|w\|}$ for all $i$ |

which is the same as
maximize $\frac{1}{||w||}$ with respect to $w, w_0$
subject to $1 \leq y_i (x_i^T w + w_0)$ for all $i$

Final formulation

- Let’s minimize $\frac{1}{2} ||w||^2$ instead of maximizing $\frac{1}{||w||}$. (Taking the square is a monotone transformation, as $||w||$ is positive, so this doesn’t change the optimal solution.)
- This gets us to:

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} ||w||^2 \text{ w.r.t. } w, w_0 \\
\text{subject to} & \quad y_i (x_i^T w + w_0) \geq 1
\end{align*}$$

- This we can solve! How?
  - It is a convex quadratic programming (QP) problem—a standard type of optimization problem for which many efficient packages are available.

Perceptron vs. SVM

We have a solution, but no “support vectors” yet...

What are “Support Vectors”?

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} ||w||^2 \text{ w.r.t. } w, w_0 \\
\text{subject to} & \quad y_i (x_i^T w + w_0) \geq 1
\end{align*}$$

- Turns out (HTF Ch. 4.5.2) we can write:
  $$w = \sum_i \alpha_i y_i x_i, \quad \text{where } \alpha_i \geq 0$$
- As for the perceptron with zero initial weights, the optimal solution for $w$ and $w_0$ is a linear combination of the $x_i$.
- The output is therefore:
\[ h_{\mathbf{w}, w_0}(\mathbf{x}) = \text{sgn} \left( \sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + w_0 \right) \]

- Output depends on weighted dot product of input vector with training examples

**Solving “the dual”**

- We can actually solve directly for the \( \alpha_i \) (again see HTF Ch. 4.5.2):

  \[
  \max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\
  \text{with constraints: } \alpha_i \geq 0 \text{ and } \sum_{i} \alpha_i y_i = 0
  \]

- This is also a QP

**The support vectors**

- Suppose we find optimal \( \alpha \)s (e.g., using a standard QP package)
- The \( \alpha_i \) will be > 0 only for the points for which \( y_i (\mathbf{x}_i^T \mathbf{w} + w_0) = 1 \)
- These are the points lying on the edge of the margin, and they are called *support vectors*, because they define the decision boundary
- The output of the classifier for query point \( \mathbf{x} \) is computed as:

  \[
  \text{sgn} \left( \sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + w_0 \right)
  \]

Hence, the output is determined by computing the *dot product of the point with the support vectors*

**Example**

Support vectors are in bold

**But why all this work?**

- SVMs are a state-of-the-art for classification when you don’t need probability estimates
- Intuitively, the large-margin property makes sense. Theory backs this up.
- SVMs offer “off-the-shelf” *non*-linear classification without having to do explicit feature construction, as we will see.

**Soft margin classifiers**

- Recall that in the linearly separable case, we compute the solution to the following optimization problem:

  \[
  \begin{aligned}
  \min \quad & \frac{1}{2} \| \mathbf{w} \|^2 \\
  \text{s.t.} \quad & y_i (\mathbf{x}_i^T \mathbf{w} + w_0) \geq 1
  \end{aligned}
  \quad \text{w.r.t. } \mathbf{w}, w_0
  \]

48
• What if we can’t satisfy the constraints?

Soft margin classifiers

• To allow misclassifications, we relax the constraints to:

\[ y_i(x_i^T w + w_0) \geq 1 - \xi_i \]

• If \( \xi_i \in (0, 1) \), the data point is within the margin
• If \( \xi_i \geq 1 \), then the data point is misclassified
• We define the soft error as \( \sum_i \xi_i \); each \( \xi_i \) is a slack variable

Problem formulation with soft errors

• Instead of:

\[
\begin{align*}
\min & \quad \frac{1}{2}\|w\|^2 \text{ w.r.t. } w, w_0 \\
\text{s.t.} & \quad y_i(x_i^T w + w_0) \geq 1
\end{align*}
\]

we want to solve:

\[
\begin{align*}
\min & \quad \frac{1}{2}\|w\|^2 + C \sum_i \xi_i \text{ w.r.t. } w, w_0, \xi_i \\
\text{s.t.} & \quad y_i(x_i^T w + w_0) \geq 1 - \xi_i, \xi_i \geq 0
\end{align*}
\]

• Note that soft errors include points that are misclassified, as well as points within the margin
• There is a linear penalty for both categories
• The choice of the constant \( C \) controls boundary-fitting

A built-in boundary-fitting knob

\[
\begin{align*}
\min & \quad \frac{1}{2}\|w\|^2 + C \sum_i \xi_i \\
\text{w.r.t.} & \quad w, w_0, \xi_i \\
\text{s.t.} & \quad y_i(x_i^T w + w_0) \geq 1 - \xi_i, \xi_i \geq 0
\end{align*}
\]

• If \( C \) is very small, there is almost no penalty for soft errors, so the focus is on maximizing the margin, even if this means more mistakes
• If \( C \) is very large, the emphasis on the soft errors will decrease the margin, if this helps to classify more examples correctly.
• How could we choose \( C \)?
Example, $C = 100$

```
SVM classification plot

Example, $C = 10$

SVM classification plot
```
Example, $C = 1$

SVM classification plot

Example, $C = 0.1$
Example, $C = 0.01$

SVM classification plot

Example, $C = 0.001$

SVM classification plot
Dual form for the soft margin problem

- Like before, we can formulate a “dual” problem that identifies the support vectors:

**Primal form:**

\[
\begin{align*}
\text{min } & \|w\|^2 + C \sum_{i} \xi_i \quad \text{w.r.t. } w, w_0, \xi_i \\
\text{s.t. } & y_i (x_i^T w + w_0) \geq (1 - \xi_i), \xi_i \geq 0
\end{align*}
\]

**Dual form:**

\[
\begin{align*}
\text{max } & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) \quad \text{w.r.t. } \alpha_i \\
\text{s.t. } & 0 \leq \alpha_i \leq C, \sum_{i=1}^{n} \alpha_i y_i = 0
\end{align*}
\]

- All the previously described machinery can be used to solve this problem

**Supervised Learning Methods: “Objective-driven”**

<table>
<thead>
<tr>
<th>Mthd.</th>
<th>Form</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLS</td>
<td>(h_w(x) = x^T w) (\approx \frac{1}{2} \sum_{i=1}^{n} (h_w(x_i) - y_i)^2)</td>
<td>using a linear function</td>
</tr>
<tr>
<td>LR</td>
<td>(h_w(x) = \frac{1}{1 + e^{-x^T w}}) (\approx P(Y = y</td>
<td>X = x))</td>
</tr>
<tr>
<td>SVM</td>
<td>(h_w(x) = \text{sgn}(x^T w)) (\approx \text{decision boundary})</td>
<td>using a linear separator</td>
</tr>
</tbody>
</table>