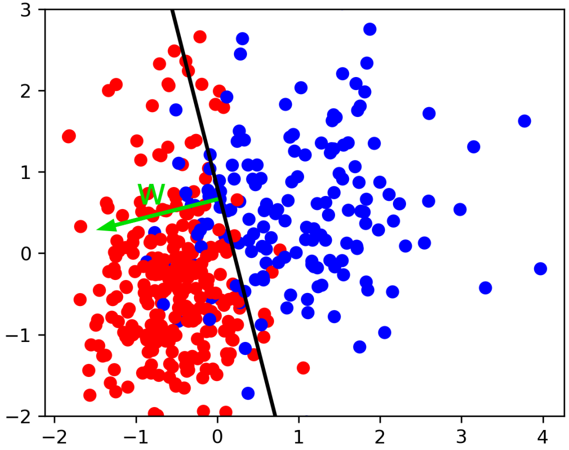
Applied Machine Learning

Linear Models for Classification

Linear models for **binary** classification

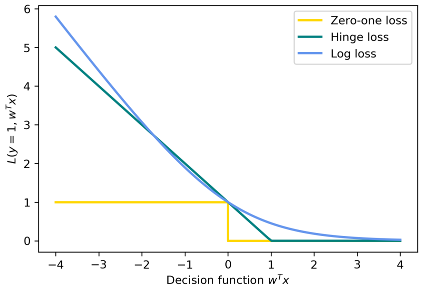


$$\hat{y} = \text{sign}(w^T \mathbf{x} + b) = \text{sign}\left(\sum_i w_i x_i + b\right)$$

Picking a loss?

$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} \sum_{i=1}^{n} 1_{y_i \neq \text{sign}(w^T \mathbf{x} + b)}$$



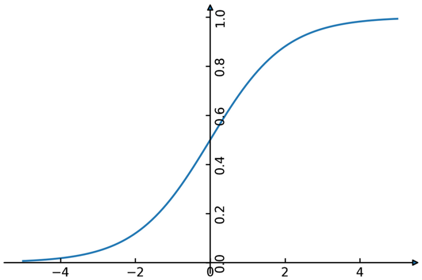
Logistic Regression

$$\log\left(\frac{p(y=1|x)}{p(y=0|x)}\right) = w^T\mathbf{x} + b$$

$$p(y|\mathbf{x}) = \frac{1}{1+e^{-w^T\mathbf{x}-b}}$$

$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}} -\sum_{i=1}^{n} \log(\exp(-y_i(w^T\mathbf{x}_i + b)) + 1)$$

$$\hat{y} = \text{sign}(w^T\mathbf{x} + b)$$



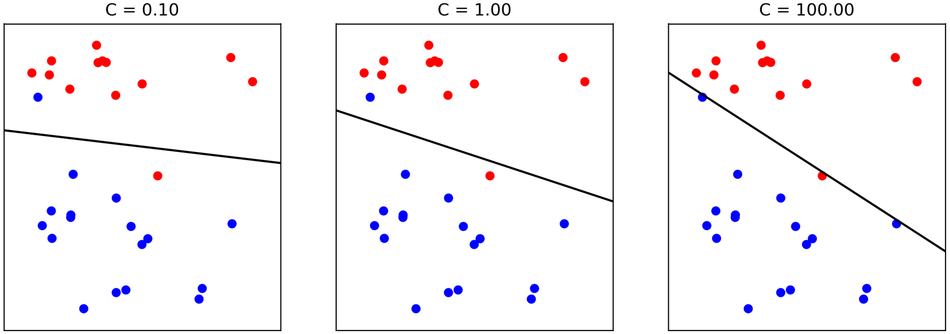
Penalized Logistic Regression

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} C \sum_{i=1}^{n} \log(\exp(-y_i(w^T \mathbf{x}_i + b) + 1) + ||w||_2^2$$

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} C \sum_{i=1}^{n} \log(\exp(-y_i(w^T \mathbf{x}_i + b) + 1) + ||w||_1$$

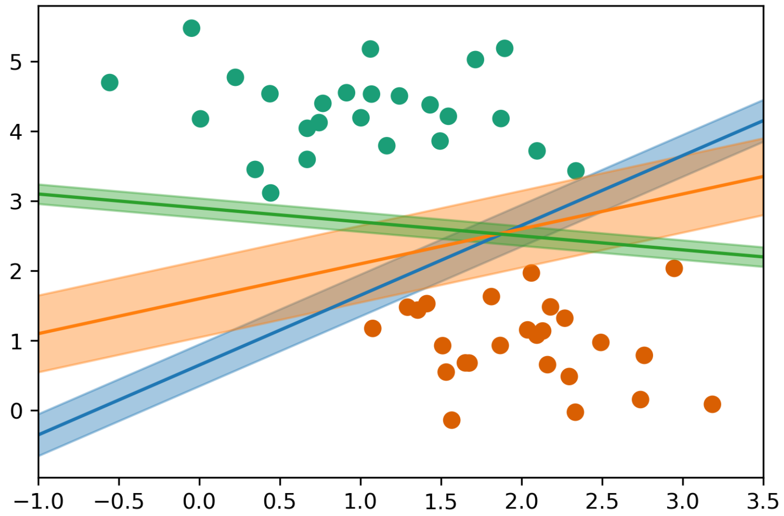
- C is inverse to alpha (or alpha / n_samples)

Effect of regularization



- Small C (a lot of regularization) limits the influence of individual points!

Max-Margin and Support Vectors



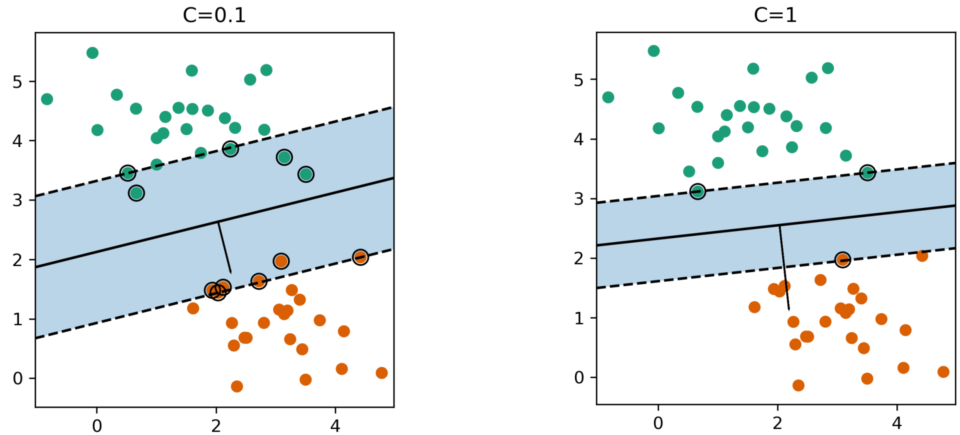
Max-Margin and Support Vectors

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} C \sum_{i=1}^{n} \max(0, 1 - y_i(w^T \mathbf{x} + b)) + ||w||_2^2$$

$$\text{Within margin} \Leftrightarrow y_i(w^T x + b) < 1$$

Smaller $w \Rightarrow$ larger margin

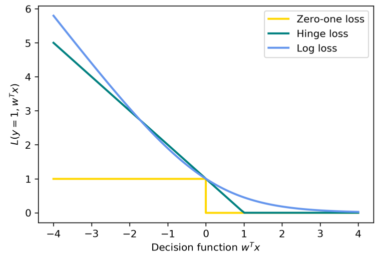
Max-Margin and Support Vectors



Logistic Regression vs SVM

$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}} C \sum_{i=1}^{n} \log(\exp(-y_i(w^T\mathbf{x}_i + b)) + 1) + ||w||_2^2$$

$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}} C \sum_{i=1}^{n} \max(0, 1 - y_i(w^T\mathbf{x}_i + b)) + ||w||_2^2$$

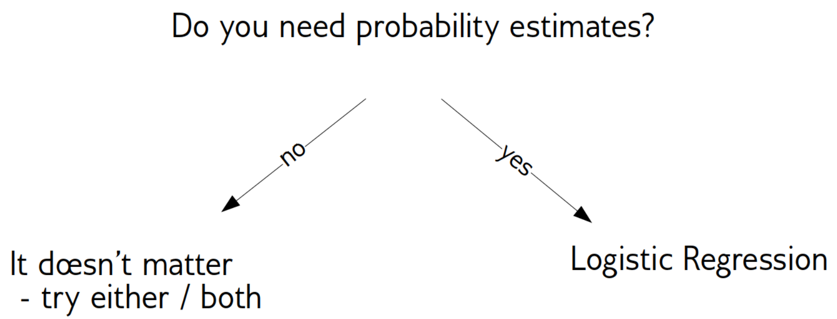


(soft margin) linear SVM

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} C \sum_{i=1}^{n} \max(0, 1 - y_i(w^T \mathbf{x}_i + b)) + ||w||_2^2$$

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} C \sum_{i=1}^{n} \max(0, 1 - y_i(w^T \mathbf{x}_i + b)) + ||w||_1$$

SVM or LogReg?



- Need compact model or believe solution is sparse? Use L1

Multiclass classification

Reduction to Binary Classification

One vs Rest

One vs One

One Vs Rest

For 4 classes:

1v{2,3,4}, 2v{1,3,4}, 3v{1,2,4}, 4v{1,2,3}

In general:

n binary classifiers - each on all data

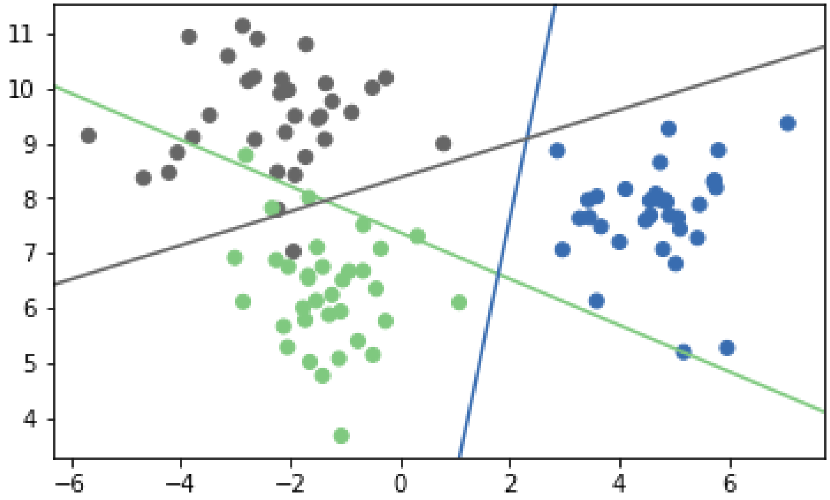
Prediction with One Vs Rest

"Class with highest score"

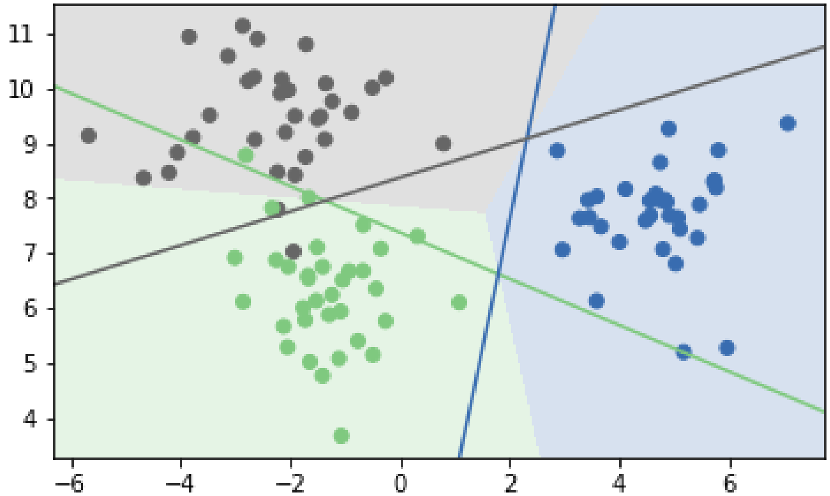
$$\hat{y} = \arg\max_{i \in Y} \mathbf{w}_i^T \mathbf{x}$$

Unclear why it works, but work well.

One vs Rest Prediction



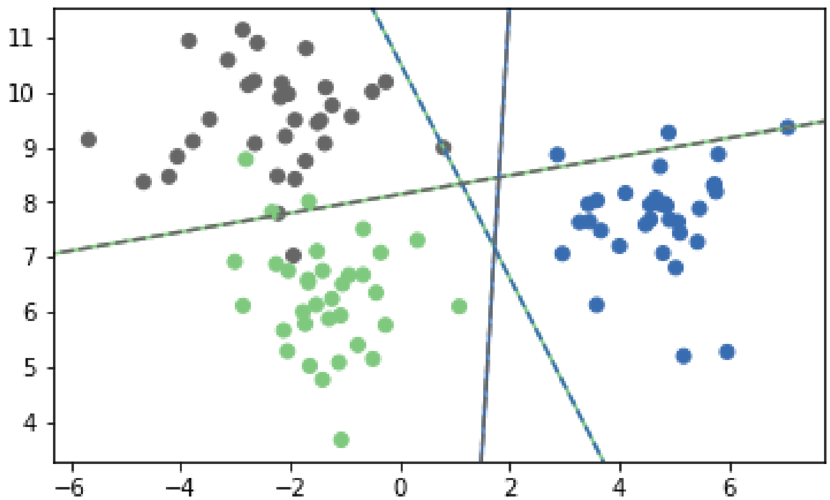
One vs Rest Prediction



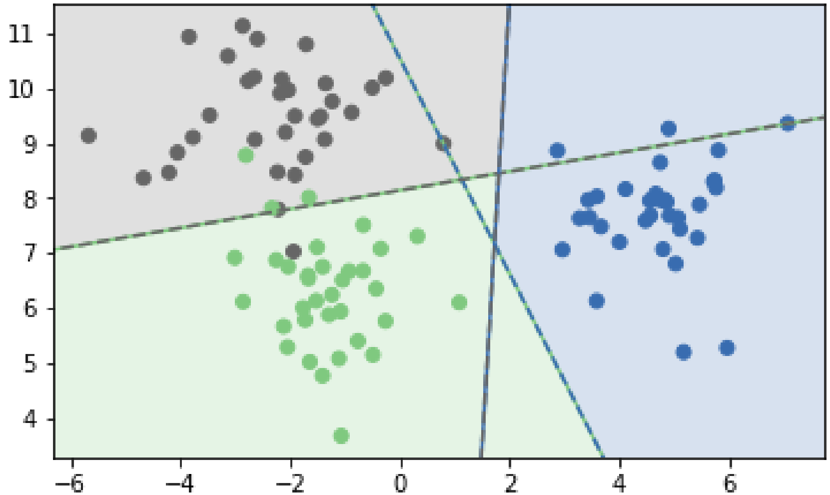
One Vs One

- 1v2, 1v3, 1v4, 2v3, 2v4, 3v4
- n * (n-1) / 2 binary classifiers - each on a fraction of the data
- "Vote for highest positives"
- Classify by all classifiers.
- Count how often each class was predicted.
- Return most commonly predicted class.
- Again - just a heuristic.

One vs One Prediction

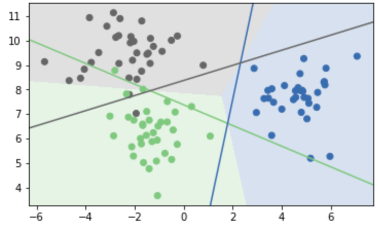


One vs One Prediction



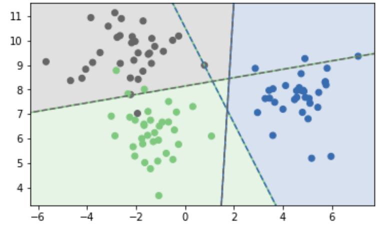
One vs Rest

- n_classes classifiers
- trained on imbalanced datasets of original size



One vs One

- n_classes * (n_classes - 1)/2 classifiers
- trained on balanced subsets



Multinomial Logistic Regression

Probabilistic multi-class model:

$$p(y = i|x) = \frac{e^{\mathbf{w}_i^T \mathbf{x}}}{\sum_j e^{\mathbf{w}_j^T \mathbf{x}}}$$

$$\min_{w \in \mathbb{R}^p} -x \sum_{i=1}^{n} \log(p(y = y_i | x_i))$$

$$\hat{y} = \arg\max_{i \in Y} \mathbf{w}_i \mathbf{x}$$

- Same prediction rule as OvR !

Multi-Class in Practice

OvR and multinomial LogReg produce one coef per class:

```
from sklearn.datasets import load_iris
iris = load_iris()
X, y = iris.data, iris.target
print(X.shape)
print(np.bincount(y))
```

```
(150, 4)
[50 50 50]
```

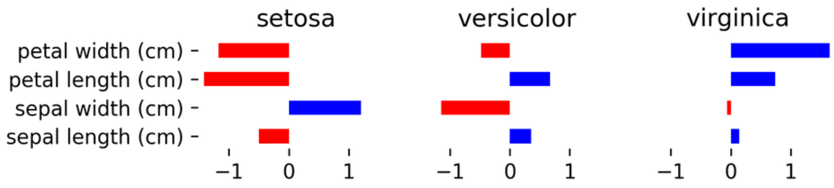
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC

logreg = LogisticRegression(multi_class="multinomial", solver="lbfgs").fit(X, y)
linearsvm = LinearSVC().fit(X, y)
print(logreg.coef_.shape)
print(linearsvm.coef_.shape)
```

```
(3, 4)
(3, 4)
```

```python
logreg.coef_
```

```
array([[-0.42339232,  0.96169329, -2.51946669, -1.0860205 ],
       [ 0.53411332, -0.31794321, -0.20537377, -0.93961515],
       [-0.11072101, -0.64375008,  2.72484045,  2.02563566]])
```



(after centering data, without intercept)

Questions ?