Machine Learning

Regression Methods 2

Topics of previous lectures

- ✓ Ingredients of Machine Learning
- ✓ Classification Basics, Basic Linear Classifier
- ✓ K-Nearest Neighbours Classifier
- ✓ Naive Bayes Classifier
- ✓ Linear and Quadratic Discriminant Analysis
- ✓ Support Vector Machines (SVM)
- ✓ Decision Trees
- ✓ Ensemble Methods (Bagging, Weighted Voting, Stacking)
- ✓ Linear Regression

Topics of today's lecture

- Regularization
- Logistic Regression
- Softmax Classifier
- Support Vector Regression (SVR)
- Regression Trees
- Evaluation and Scoring of Classifiers

Regularization

- The main idea of regularization in machine learning is to penalize complex models

Regularization

- The main idea of regularization in machine learning is to penalize complex models
- If two models have a similar loss on the training data then the simpler tends to be better on test data

Regularization

- The main idea of regularization in machine learning is to penalize complex models
- If two models have a similar loss on the training data then the simpler tends to be better on test data
- Simpler models have less capacity to overfit the noise

Ridge regression

- Ridge regression is a regularisation (also known as L2-norm regularization) method for linear regression

Ridge regression

- Ridge regression is a regularisation (also known as L2-norm regularization) method for linear regression
- It regularises the task by penalising vector length:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \|\mathbf{w}\|^2 \right)$$

Ridge regression

- Ridge regression is a regularisation (also known as L2-norm regularization) method for linear regression
- It regularises the task by penalising vector length:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \|\mathbf{w}\|^2 \right)$$

- Here λ is a regularisation parameter, which we can choose by cross-validation:
  – Higher λ means stronger regularization
  – If $\lambda = 0$ then we are back at OLS

Calculating the partial derivatives

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \Big((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \|\mathbf{w}\|^2 \Big)$$

- Based on properties of matrix derivatives:

$$\frac{\partial \Big((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w} \Big)}{\partial \mathbf{w}} =$$

Calculating the partial derivatives

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \|\mathbf{w}\|^2 \right)$$

- Based on properties of matrix derivatives:

$$\frac{\partial \left((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w} \right)}{\partial \mathbf{w}} = -2\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + 2\lambda \mathbf{w} = 0$$

Calculating the partial derivatives

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \|\mathbf{w}\|^2 \right)$$

- Based on properties of matrix derivatives:

$$\frac{\partial \left((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w} \right)}{\partial \mathbf{w}} = -2\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + 2\lambda \mathbf{w} = 0$$

$$\mathbf{X}^T \mathbf{X}\mathbf{w} + \lambda \mathbf{w} = \mathbf{X}^T \mathbf{y}$$

Calculating the partial derivatives

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \|\mathbf{w}\|^2 \right)$$

- Based on properties of matrix derivatives:

$$\frac{\partial \left((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w} \right)}{\partial \mathbf{w}} = -2\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + 2\lambda \mathbf{w} = 0$$

$$\mathbf{X}^T \mathbf{X}\mathbf{w} + \lambda \mathbf{w} = \mathbf{X}^T \mathbf{y}$$

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})\mathbf{w} = \mathbf{X}^T \mathbf{y}$$

Calculating the partial derivatives

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \|\mathbf{w}\|^2 \right)$$

- Based on properties of matrix derivatives:

$$\frac{\partial \left((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w} \right)}{\partial \mathbf{w}} = -2\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + 2\lambda \mathbf{w} = 0$$

$$\mathbf{X}^T \mathbf{X}\mathbf{w} + \lambda \mathbf{w} = \mathbf{X}^T \mathbf{y}$$

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})\mathbf{w} = \mathbf{X}^T \mathbf{y}$$

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} (\mathbf{X}^T \mathbf{y})$$

Calculating the partial derivatives

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \|\mathbf{w}\|^2 \right)$$

- Based on properties of matrix derivatives:

$$\frac{\partial \left((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w} \right)}{\partial \mathbf{w}} = -2\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + 2\lambda \mathbf{w} = 0$$

$$\mathbf{X}^T \mathbf{X}\mathbf{w} + \lambda \mathbf{w} = \mathbf{X}^T \mathbf{y}$$

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})\mathbf{w} = \mathbf{X}^T \mathbf{y}$$

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} (\mathbf{X}^T \mathbf{y})$$

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} (\mathbf{X}^T \mathbf{y})$$

Calculating the partial derivatives

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \Big((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \|\mathbf{w}\|^2 \Big)$$

- Based on properties of matrix derivatives:

$$\frac{\partial \Big((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w} \Big)}{\partial \mathbf{w}} = -2\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + 2\lambda \mathbf{w} = 0$$

$$\begin{aligned}
\mathbf{X}^T \mathbf{X}\mathbf{w} + \lambda \mathbf{w} &= \mathbf{X}^T \mathbf{y} \\
(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})\mathbf{w} &= \mathbf{X}^T \mathbf{y} \\
(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})\mathbf{w} &= (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} (\mathbf{X}^T \mathbf{y}) \\
\mathbf{w} &= (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} (\mathbf{X}^T \mathbf{y})
\end{aligned}$$

- For ridge regression there exists a **closed-form solution** (i.e. can be explicitly calculated without numerical optimization):

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} (\mathbf{X}^T \mathbf{y})$$

Ridge regression

Pro: effective in reducing overfitting compared to OLS

Ridge regression

Pro: effective in reducing overfitting compared to OLS

Con: all coefficients are still non-zero, even if the true model is very sparse (very few non-zero coefficients)

Lasso regression

- Lasso regression is another regularisation (also known as L1-norm regularization) method for linear regression

Lasso regression

- Lasso regression is another regularisation (also known as L1-norm regularization) method for linear regression
- It regularises the task by penalising the sum of absolute values of weights:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \sum_{i=1}^{d} |w_i| \right)$$

Lasso regression

- Lasso regression is another regularisation (also known as L1-norm regularization) method for linear regression
- It regularises the task by penalising the sum of absolute values of weights:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left((\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \sum_{i=1}^{d} |w_i| \right)$$

- No closed-form solution exists, must be optimised numerically

Lasso vs Ridge regression

- Lasso can produce sparse solutions:
  - Many coefficients are exactly zero
  - How many? Depends on λ

Lasso vs Ridge regression

- Lasso can produce sparse solutions:
  – Many coefficients are exactly zero
  – How many? Depends on λ
- This is beneficial for two reasons:
  – The learned model becomes easier to interpret
  – If the true model is sparse, then lasso tends to give lower test MSE than ridge regression

Lasso vs Ridge regression

- Lasso can produce sparse solutions:
  – Many coefficients are exactly zero
  – How many? Depends on λ
- This is beneficial for two reasons:
  – The learned model becomes easier to interpret
  – If the true model is sparse, then lasso tends to give lower test MSE than ridge regression
- The parameter can be learned by using a hold-out validation dataset or K-fold cross-validation

Lasso regression

Pro: if the true model is sparse then outperforms ridge regression and OLS

Lasso regression

Pro: if the true model is sparse then outperforms ridge regression and OLS

Con: if several features are highly correlated then tends to put high weight on an arbitrary one of those and zero to others

Logistic Regression

- Despite the name, **logistic regression is a classification method**, or it can be view as a regression method restriced into $[0, 1]$ interval

Logistic Regression

- Despite the name, **logistic regression is a classification method**, or it can be view as a regression method restriced into $[0, 1]$ interval
- We have used **Linear Regression** when y was continuous, but will it be easily used when y is binary?

Logistic Regression

- Despite the name, **logistic regression is a classification method**, or it can be view as a regression method restriced into $[0, 1]$ interval
- We have used **Linear Regression** when y was continuous, but will it be easily used when y is binary?
- We need a way to transform the predictions, so that we can get binary labels, instead of continous quantites

Logistic Regression

- Despite the name, **logistic regression is a classification method**, or it can be view as a regression method restriced into $[0, 1]$ interval
- We have used **Linear Regression** when y was continuous, but will it be easily used when y is binary?
- We need a way to transform the predictions, so that we can get binary labels, instead of continous quantites
- We can use the logistic sigmoid function on the predictions!

$$\sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

Logistic Regression

- Despite the name, **logistic regression is a classification method**, or it can be view as a regression method restriced into $[0, 1]$ interval
- We have used **Linear Regression** when y was continuous, but will it be easily used when y is binary?
- We need a way to transform the predictions, so that we can get binary labels, instead of continous quantites
- We can use the logistic sigmoid function on the predictions!

$$\sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

- In Logistic Regression the parameters are usually fit by maximum likelihood method (see Lecture 3)

MLE for Logistic Regression

- The likelihood function will be

$$L = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1-y_i},$$

where $p_i = P(y_i = \oplus | \mathbf{x}_i) = \sigma(\mathbf{w}^T \mathbf{x}_i)$

MLE for Logistic Regression

- The likelihood function will be

$$L = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1-y_i},$$

where $p_i = P(y_i = \oplus | \mathbf{x}_i) = \sigma(\mathbf{w}^T \mathbf{x}_i)$

- It is more convenient to work with the log likelihood function

$$\begin{aligned}
\log(L) &= \sum_i \log(p_i^{y_i} (1 - p_i)^{1-y_i}) \\
&= \sum_i \Big(\log(p_i^{y_i}) + \log((1 - p_i)^{1-y_i}) \Big) = \\
&= \sum_i \Big(y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \Big)
\end{aligned}$$

Does this look familiar?

MLE for Logistic Regression

Now that we have the log likelihood function, we can compute the partial derivative w.r.t. $\mathbf{w}$

$$\frac{\partial \log(L(\mathbf{w}))}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \sum_i (y_i \log(\sigma(\mathbf{w}^T \mathbf{x}_i)) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) =$$

MLE for Logistic Regression

Now that we have the log likelihood function, we can compute the partial derivative w.r.t. $\mathbf{w}$

$$\frac{\partial \log(L(\mathbf{w}))}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \sum_i (y_i \log(\sigma(\mathbf{w}^T \mathbf{x}_i)) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) =$$

$$= \sum_i (\frac{y_i}{\sigma(\mathbf{w}^T \mathbf{x}_i)} \frac{\partial}{\partial \mathbf{w}} (\sigma(\mathbf{w}^T \mathbf{x}_i))] +$$

$$+ \sum_i (\frac{1 - y_i}{1 - \sigma(\mathbf{w}^T \mathbf{x}_i)} \frac{\partial}{\partial \mathbf{w}} (1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) =$$

MLE for Logistic Regression

Now that we have the log likelihood function, we can compute the partial derivative w.r.t. $\mathbf{w}$

$$\frac{\partial \log(L(\mathbf{w}))}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \sum_i (y_i \log(\sigma(\mathbf{w}^T \mathbf{x}_i)) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) =$$

$$= \sum_i (\frac{y_i}{\sigma(\mathbf{w}^T \mathbf{x}_i)} \frac{\partial}{\partial \mathbf{w}} (\sigma(\mathbf{w}^T \mathbf{x}_i))] +$$

$$+ \sum_i (\frac{1 - y_i}{1 - \sigma(\mathbf{w}^T \mathbf{x}_i)} \frac{\partial}{\partial \mathbf{w}} (1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) =$$

MLE for Logistic Regression

Now that we have the log likelihood function, we can compute the partial derivative w.r.t. $\mathbf{w}$

$$\frac{\partial \log(L(\mathbf{w}))}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \sum_i (y_i \log(\sigma(\mathbf{w}^T \mathbf{x}_i)) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) =$$

$$= \sum_i (\frac{y_i}{\sigma(\mathbf{w}^T \mathbf{x}_i)} \frac{\partial}{\partial \mathbf{w}} (\sigma(\mathbf{w}^T \mathbf{x}_i))] +$$

$$+ \sum_i (\frac{1 - y_i}{1 - \sigma(\mathbf{w}^T \mathbf{x}_i)} \frac{\partial}{\partial \mathbf{w}} (1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) =$$

the derivative of sigmoid is $\frac{d}{dz}\sigma(z) = \sigma(z)(1 - \sigma(z))$

MLE for Logistic Regression

Now that we have the log likelihood function, we can compute the partial derivative w.r.t. $\mathbf{w}$

$$\frac{\partial \log(L(\mathbf{w}))}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \sum_i (y_i \log(\sigma(\mathbf{w}^T \mathbf{x}_i)) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) =$$

$$= \sum_i (\frac{y_i}{\sigma(\mathbf{w}^T \mathbf{x}_i)} \frac{\partial}{\partial \mathbf{w}} (\sigma(\mathbf{w}^T \mathbf{x}_i))] +$$

$$+ \sum_i (\frac{1 - y_i}{1 - \sigma(\mathbf{w}^T \mathbf{x}_i)} \frac{\partial}{\partial \mathbf{w}} (1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) =$$

the derivative of sigmoid is $\frac{d}{dz}\sigma(z) = \sigma(z)(1 - \sigma(z))$

$$= \sum_i (\frac{y_i}{\sigma(\mathbf{w}^T \mathbf{x}_i)} \sigma(\mathbf{w}^T \mathbf{x}_i)(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))\mathbf{x}_i) +$$

$$+ \sum_i (-\frac{1 - y_i}{1 - \sigma(\mathbf{w}^T \mathbf{x}_i)} \sigma(\mathbf{w}^T \mathbf{x}_i)(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))\mathbf{x}_i)$$

MLE for Logistic Regression

Now that we have the log likelihood function, we can compute the partial derivative w.r.t. $\mathbf{w}$

$$\frac{\partial \log(L(\mathbf{w}))}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \sum_i (y_i \log(\sigma(\mathbf{w}^T \mathbf{x}_i)) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) =$$

$$= \sum_i (y_i (1 - \sigma(\mathbf{w}^T \mathbf{x}_i)) \mathbf{x}_i) + \sum_i (-(1 - y_i) \sigma(\mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i) =$$

MLE for Logistic Regression

Now that we have the log likelihood function, we can compute the partial derivative w.r.t. $\mathbf{w}$

$$\frac{\partial \log(L(\mathbf{w}))}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \sum_i (y_i \log(\sigma(\mathbf{w}^T \mathbf{x}_i)) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) =$$

$$= \sum_i (y_i (1 - \sigma(\mathbf{w}^T \mathbf{x}_i)) \mathbf{x}_i) + \sum_i (-(1 - y_i) \sigma(\mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i) =$$

$$= \sum_i (\mathbf{x}_i (y_i - y_i \sigma(\mathbf{w}^T \mathbf{x}_i) - \sigma(\mathbf{w}^T \mathbf{x}_i) + y_i \sigma(\mathbf{w}^T \mathbf{x}_i))) =$$

MLE for Logistic Regression

Now that we have the log likelihood function, we can compute the partial derivative w.r.t. $\mathbf{w}$

$$\begin{aligned}
\frac{\partial \log(L(\mathbf{w}))}{\partial \mathbf{w}} &= \frac{\partial}{\partial \mathbf{w}} \sum_i (y_i \log(\sigma(\mathbf{w}^T \mathbf{x}_i)) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) = \\
&= \sum_i (y_i (1 - \sigma(\mathbf{w}^T \mathbf{x}_i)) \mathbf{x}_i) + \sum_i (-(1 - y_i) \sigma(\mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i) = \\
&= \sum_i (\mathbf{x}_i (y_i - y_i \sigma(\mathbf{w}^T \mathbf{x}_i) - \sigma(\mathbf{w}^T \mathbf{x}_i) + y_i \sigma(\mathbf{w}^T \mathbf{x}_i))) = \\
&= \sum_i \mathbf{x}_i (y_i - \sigma(\mathbf{w}^T \mathbf{x}_i)) = 0
\end{aligned}$$

MLE for Logistic Regression

Now that we have the log likelihood function, we can compute the partial derivative w.r.t. $\mathbf{w}$

$$\begin{aligned}
\frac{\partial \log(L(\mathbf{w}))}{\partial \mathbf{w}} &= \frac{\partial}{\partial \mathbf{w}} \sum_i (y_i \log(\sigma(\mathbf{w}^T \mathbf{x}_i)) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) = \\
&= \sum_i (y_i (1 - \sigma(\mathbf{w}^T \mathbf{x}_i)) \mathbf{x}_i) + \sum_i (-(1 - y_i) \sigma(\mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i) = \\
&= \sum_i (\mathbf{x}_i (y_i - y_i \sigma(\mathbf{w}^T \mathbf{x}_i) - \sigma(\mathbf{w}^T \mathbf{x}_i) + y_i \sigma(\mathbf{w}^T \mathbf{x}_i))) = \\
&= \sum_i \mathbf{x}_i (y_i - \sigma(\mathbf{w}^T \mathbf{x}_i)) = 0
\end{aligned}$$

$$\sum_i \mathbf{x}_i y_i = \sum_i \mathbf{x}_i \sigma(\mathbf{w}^T \mathbf{x}_i)$$

MLE for Logistic Regression

Now that we have the log likelihood function, we can compute the partial derivative w.r.t. $\mathbf{w}$

$$\begin{aligned}
\frac{\partial \log(L(\mathbf{w}))}{\partial \mathbf{w}} &= \frac{\partial}{\partial \mathbf{w}} \sum_i (y_i \log(\sigma(\mathbf{w}^T \mathbf{x}_i)) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) = \\
&= \sum_i (y_i (1 - \sigma(\mathbf{w}^T \mathbf{x}_i)) \mathbf{x}_i) + \sum_i (-(1 - y_i) \sigma(\mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i) = \\
&= \sum_i (\mathbf{x}_i (y_i - y_i \sigma(\mathbf{w}^T \mathbf{x}_i) - \sigma(\mathbf{w}^T \mathbf{x}_i) + y_i \sigma(\mathbf{w}^T \mathbf{x}_i))) = \\
&= \sum_i \mathbf{x}_i (y_i - \sigma(\mathbf{w}^T \mathbf{x}_i)) = 0
\end{aligned}$$

$$\sum_i \mathbf{x}_i y_i = \sum_i \mathbf{x}_i \sigma(\mathbf{w}^T \mathbf{x}_i)$$

This system of equalities is hard to solve directly!

Logistic Regression Solution

- The above system of equalities is hard to solve directly

Logistic Regression Solution

- The above system of equalities is hard to solve directly
- Instead, it is easier to optimize

$$\underset{\mathbf{w}}{\operatorname{argmin}} - \sum_i \Big(y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \Big)$$

Logistic Regression Solution

- The above system of equalities is hard to solve directly
- Instead, it is easier to optimize

$$\underset{\mathbf{w}}{\operatorname{argmin}} - \sum_i \Big(y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \Big)$$

- This can be done with a numerical method such as gradient descent

Logistic Regression Solution

- The above system of equalities is hard to solve directly
- Instead, it is easier to optimize

$$\underset{\mathbf{w}}{\operatorname{argmin}} - \sum_i \Big(y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \Big)$$

- This can be done with a numerical method such as gradient descent
- The loss function that logistic regression minimizes is known as **cross entropy** (also known as log-loss or binomial deviance)

Softmax Classifier

- Softmax classifier is the multi-class generalization of Logistic Regression

Softmax Classifier

- Softmax classifier is the multi-class generalization of Logistic Regression
- In case of a C class classification task, we will have a weight matrix $\mathbf{W}$ of size $C \times (d+1)$, instead of a $d+1$ dimensional vector

Softmax Classifier

- Softmax classifier is the multi-class generalization of Logistic Regression
- In case of a C class classification task, we will have a weight matrix $\mathbf{W}$ of size $C \times (d+1)$, instead of a $d+1$ dimensional vector
- The softmax function is used on each of the prediction scores

$$f_j(\mathbf{z}) = \frac{e^{z_j}}{\sum_k e^{z_k}}, \quad \text{where} \quad z_j = \mathbf{W}_j \mathbf{x}^T, \mathbf{z} = \mathbf{W}\mathbf{x}^T$$

Softmax Classifier

- Softmax classifier is the multi-class generalization of Logistic Regression
- In case of a C class classification task, we will have a weight matrix $\mathbf{W}$ of size $C \times (d+1)$, instead of a $d+1$ dimensional vector
- The softmax function is used on each of the prediction scores

$$f_j(\mathbf{z}) = \frac{e^{z_j}}{\sum_k e^{z_k}}, \quad \text{where} \quad z_j = \mathbf{W}_j \mathbf{x}^T, \mathbf{z} = \mathbf{W}\mathbf{x}^T$$

- The loss function will be cross-entropy again

$$L(\mathbf{y}, f(\mathbf{x})) = -\sum_{j=1}^{C} y_j \log f_j(\mathbf{z}),$$

where $\mathbf{y}$ is a one-hot encoded vector for the ground truth class.

Softmax Classifier

- Softmax classifier is the multi-class generalization of Logistic Regression
- In case of a C class classification task, we will have a weight matrix $\mathbf{W}$ of size $C \times (d+1)$, instead of a $d+1$ dimensional vector
- The softmax function is used on each of the prediction scores

$$f_j(\mathbf{z}) = \frac{e^{z_j}}{\sum_k e^{z_k}}, \quad \text{where} \quad z_j = \mathbf{W}_j \mathbf{x}^T, \mathbf{z} = \mathbf{W}\mathbf{x}^T$$

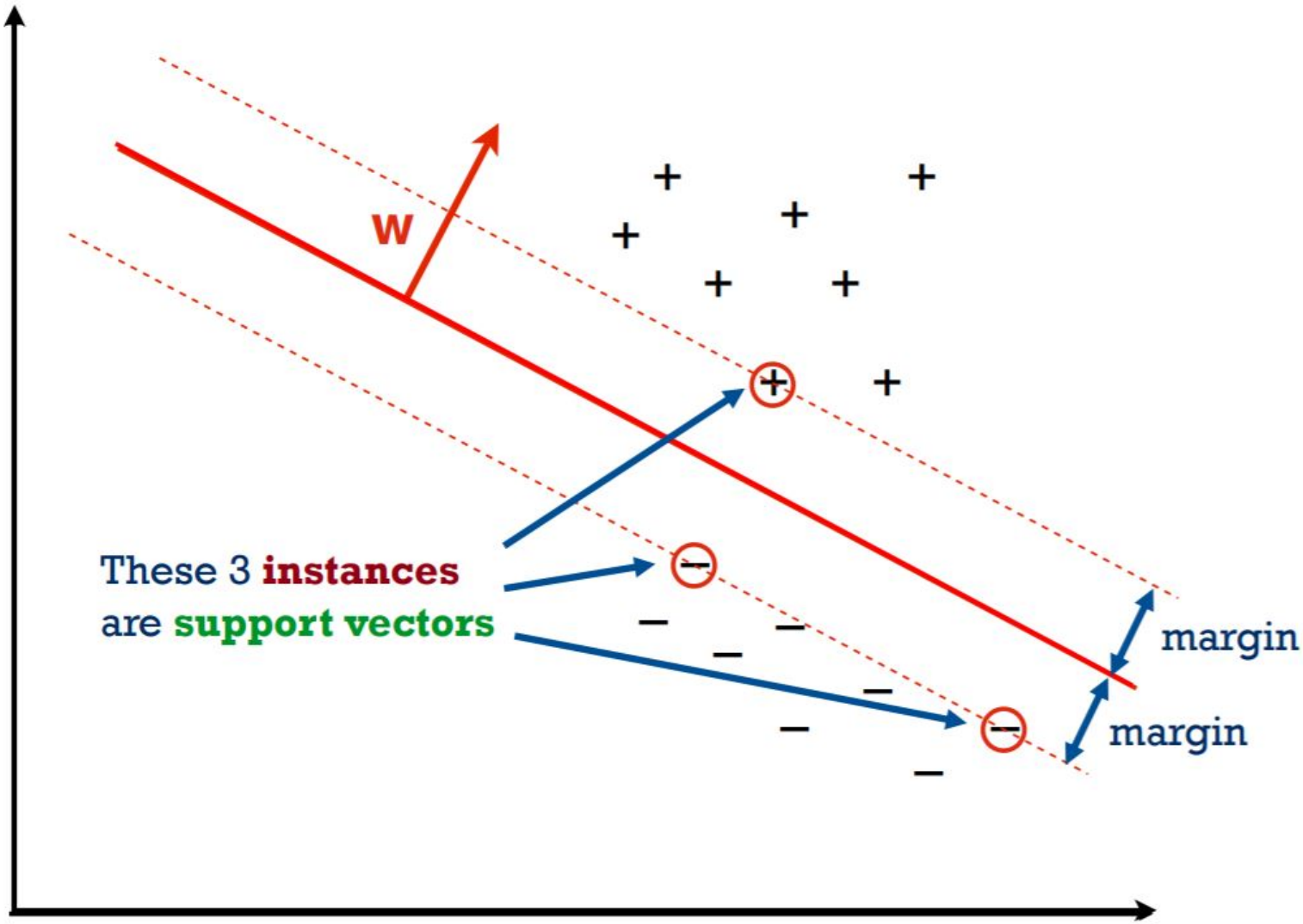
- The loss function will be cross-entropy again

$$L(\mathbf{y}, f(\mathbf{x})) = -\sum_{j=1}^{C} y_j \log f_j(\mathbf{z}),$$

where $\mathbf{y}$ is a one-hot encoded vector for the ground truth class.

- The problem can be solved with gradient descent.

Support Vector Machine (SVM) for Classification



Hard-margin SVM for classification

$$\mathbf{w}^*, t^* = \underset{\mathbf{w},t}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{w}\|^2$$

subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i - t) \geq 1, \quad y_i \in \{-1, 1\}, \quad i = 1, \ldots, n$

Hard-margin SVM for classification

$$\mathbf{w}^*, t^* = \underset{\mathbf{w},t}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i - t) \geq 1, \quad y_i \in \{-1, 1\}, \quad i = 1, \ldots, n$$

We can solve the equivalent dual problem

$$\alpha_1^*, \ldots, \alpha_n^* = \underset{\alpha_1,\ldots,\alpha_n}{\operatorname{argmax}} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j + \sum_{i=1}^{n}\alpha_i$$

$$\text{subject to } \alpha_i \geq 0, \quad i = 1, \ldots, n \text{ and } \sum_{i=1}^{n}\alpha_i y_i = 0$$

Hard-margin SVM for classification

$$\mathbf{w}^*, t^* = \underset{\mathbf{w},t}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i - t) \geq 1, \quad y_i \in \{-1, 1\}, \quad i = 1, \ldots, n$$

We can solve the equivalent dual problem

$$\alpha_1^*, \ldots, \alpha_n^* = \underset{\alpha_1,\ldots,\alpha_n}{\operatorname{argmax}} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j + \sum_{i=1}^{n}\alpha_i$$

$$\text{subject to } \alpha_i \geq 0, \quad i = 1, \ldots, n \text{ and } \sum_{i=1}^{n}\alpha_i y_i = 0$$

From the result we can calculate:

$$\mathbf{w}^* = \sum_{i=1}^{n}\alpha_i^* y_i \mathbf{x}_i \qquad t^* = \mathbf{w}^* \cdot \mathbf{x}_i - y_i,$$

where $\mathbf{x}_i$ is a support vector and α_i is its weight.

Soft-margin SVM for classification

$$\mathbf{w}^*, t^*, \xi_i^* = \underset{\mathbf{w},t,\xi_i}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to } y_i(\mathbf{w}\cdot\mathbf{x}_i - t) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n$$

Soft-margin SVM for classification

$$\mathbf{w}^*, t^*, \xi_i^* = \underset{\mathbf{w},t,\xi_i}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to } y_i(\mathbf{w}\cdot\mathbf{x}_i - t) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n$$

We can solve the equivalent dual problem

$$\alpha_1^*, \ldots, \alpha_n^* = \underset{\alpha_1,\ldots,\alpha_n}{\operatorname{argmax}} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i$$

$$\text{subject to } 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \text{ and } \sum_{i=1}^{n}\alpha_i y_i = 0$$

Soft-margin SVM for classification

$$\mathbf{w}^*, t^*, \xi_i^* = \underset{\mathbf{w},t,\xi_i}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to } y_i(\mathbf{w}\cdot\mathbf{x}_i - t) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n$$

We can solve the equivalent dual problem

$$\alpha_1^*, \ldots, \alpha_n^* = \underset{\alpha_1,\ldots,\alpha_n}{\operatorname{argmax}} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i$$

$$\text{subject to } 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \text{ and } \sum_{i=1}^{n}\alpha_i y_i = 0$$

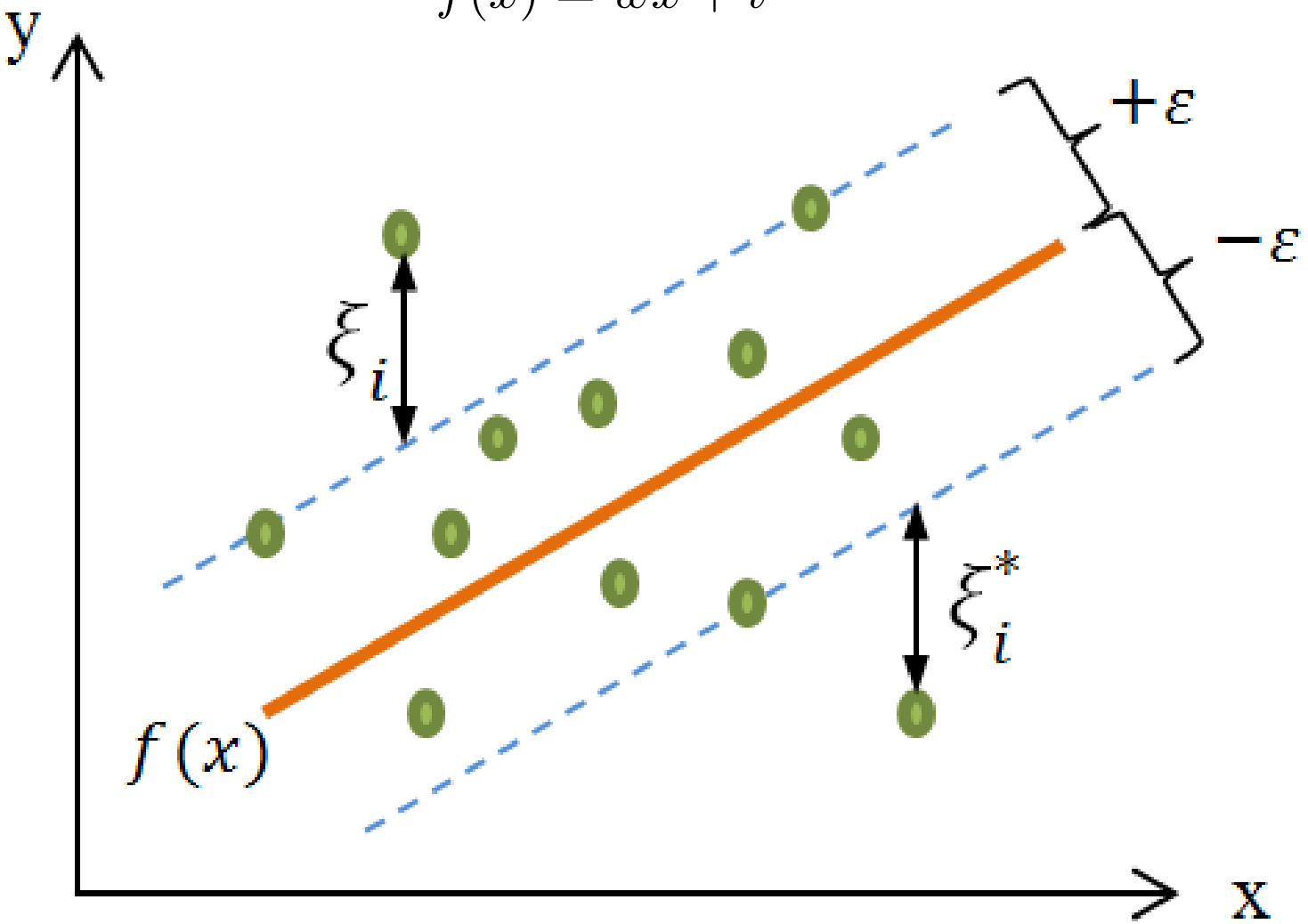
From the result we can calculate:

$$\mathbf{w}^* = \sum_{i=1}^{n}\alpha_i^* y_i \mathbf{x}_i \qquad t^* = \mathbf{w}^*\cdot\mathbf{x}_i - y_i,$$

where $\mathbf{x}_i$ is a support vector and α_i is its weight.

SVM for Regression

$$f(x) = wx + t$$



Support Vector Regression

$$\min_{\mathbf{w},t,\xi_i,\xi_i'} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i')$$

$$\text{subject to } y_i - \mathbf{w}\cdot\mathbf{x}_i - t \leq \epsilon + \xi_i, \quad \xi_i \geq 0$$

$$\mathbf{w}\cdot\mathbf{x}_i + t - y_i \leq \epsilon + \xi_i', \quad \xi_i' \geq 0, \quad i = 1,\ldots,n$$

Support Vector Regression

$$\min_{\mathbf{w},t,\xi_i,\xi_i'} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i')$$

$$\text{subject to } y_i - \mathbf{w}\cdot\mathbf{x}_i - t \leq \epsilon + \xi_i, \quad \xi_i \geq 0$$

$$\mathbf{w}\cdot\mathbf{x}_i + t - y_i \leq \epsilon + \xi_i', \quad \xi_i' \geq 0, \quad i = 1,\ldots,n$$

We can solve the equivalent dual problem

$$\max_{\boldsymbol{\alpha};\boldsymbol{\alpha}'} -\frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}')^T Q(\boldsymbol{\alpha}-\boldsymbol{\alpha}') - \epsilon e^T(\boldsymbol{\alpha}+\boldsymbol{\alpha}') + y^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}')$$

$$\text{subject to } 0 \leq \alpha_i, \alpha_i' \leq C, \quad i = 1,\ldots,n \text{ and } e^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}') = 0$$

where $Q = \mathbf{x}\mathbf{x}^T$ and $e = (1,\ldots,1)^T$

Support Vector Regression

$$\min_{\mathbf{w},t,\xi_i,\xi_i'} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i')$$

$$\text{subject to } y_i - \mathbf{w}\cdot\mathbf{x}_i - t \le \epsilon + \xi_i, \quad \xi_i \ge 0$$

$$\mathbf{w}\cdot\mathbf{x}_i + t - y_i \le \epsilon + \xi_i', \quad \xi_i' \ge 0, \quad i = 1,\ldots,n$$

We can solve the equivalent dual problem

$$\max_{\boldsymbol{\alpha};\boldsymbol{\alpha}'} -\frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}')^T Q(\boldsymbol{\alpha}-\boldsymbol{\alpha}') - \epsilon e^T(\boldsymbol{\alpha}+\boldsymbol{\alpha}') + y^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}')$$

$$\text{subject to } 0 \le \alpha_i, \alpha_i' \le C, \quad i = 1,\ldots,n \text{ and } e^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}') = 0$$

where $Q = \mathbf{x}\mathbf{x}^T$ and $e = (1,\ldots,1)^T$

From this we can calculate the prediction: $\hat{f}(\mathbf{x}) = \mathbf{w}^*\cdot\mathbf{x} + t$, with

$$\mathbf{w}^* = \sum_{i=1}^{n}(\alpha_i - \alpha_i')\mathbf{x}_i, \quad t^* = y_i - \epsilon - \mathbf{w}^*\cdot\mathbf{x}_i$$

Support Vector Regression

Can we apply kernels in this case?

$$\min_{\mathbf{w},t,\xi_i,\xi_i'} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i')$$

$$\text{subject to } y_i - \mathbf{w}\cdot\mathbf{x}_i - t \leq \epsilon + \xi_i, \quad \xi_i \geq 0$$

$$\mathbf{w}\cdot\mathbf{x}_i + t - y_i \leq \epsilon + \xi_i', \quad \xi_i' \geq 0, \quad i = 1,\ldots,n$$

We can solve the equivalent dual problem

$$\max_{\boldsymbol{\alpha};\boldsymbol{\alpha}'} -\frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}')^T Q(\boldsymbol{\alpha}-\boldsymbol{\alpha}') - \epsilon e^T(\boldsymbol{\alpha}+\boldsymbol{\alpha}') + y^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}')$$

$$\text{subject to } 0 \leq \alpha_i, \alpha_i' \leq C, \quad i = 1,\ldots,n \text{ and } e^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}') = 0$$

where $Q = \mathbf{x}\mathbf{x}^T$ and $e = (1,\ldots,1)^T$

From this we can calculate the prediction: $\hat{f}(\mathbf{x}) = \mathbf{w}^*\cdot\mathbf{x} + t$, with

$$\mathbf{w}^* = \sum_{i=1}^{n}(\alpha_i - \alpha_i')\mathbf{x}_i, \quad t^* = y_i - \epsilon - \mathbf{w}^*\cdot\mathbf{x}_i$$

Kernel SVR

- Fitting (soft-margin kernel SVR):

$$\max_{\boldsymbol{\alpha};\boldsymbol{\alpha}'} -\frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}')^T Q(\boldsymbol{\alpha}-\boldsymbol{\alpha}') + \epsilon e^T(\boldsymbol{\alpha}+\boldsymbol{\alpha}') - y^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}')$$

subject to $0 \leq \alpha_i, \alpha_i^{'} \leq C, \quad i = 1, \ldots, n$ and $e^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}') = 0$

where $Q = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ and $e = (1, \ldots, 1)^T$

$$t^* = y_j - \epsilon - \sum_{i=1}^{n}(\alpha_i - \alpha_i^{'})\kappa(\mathbf{x}_i, \mathbf{x}_j),$$

where $\mathbf{x}_j$ is a support vector of target value y_j.

Kernel SVR

- Fitting (soft-margin kernel SVR):

$$\max_{\boldsymbol{\alpha};\boldsymbol{\alpha}'} -\frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}')^T Q(\boldsymbol{\alpha}-\boldsymbol{\alpha}') + \epsilon e^T(\boldsymbol{\alpha}+\boldsymbol{\alpha}') - y^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}')$$

subject to $0 \leq \alpha_i, \alpha_i^{'} \leq C, \quad i = 1, \ldots, n$ and $e^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}') = 0$

where $Q = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ and $e = (1, \ldots, 1)^T$

$$t^* = y_j - \epsilon - \sum_{i=1}^{n}(\alpha_i - \alpha_i^{'})\kappa(\mathbf{x}_i, \mathbf{x}_j),$$

where $\mathbf{x}_j$ is a support vector of target value y_j.

- Prediction:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^{'})\kappa(\mathbf{x}_i, \mathbf{x}) + t^*$$

Regression Trees

- In order to adapt decision trees to regression problem, we need to think about a suitable impurity function

Regression Trees

- In order to adapt decision trees to regression problem, we need to think about a suitable impurity function
- We can use the **weighted average variance** of the target values as the measure of impurity

Regression Trees

- In order to adapt decision trees to regression problem, we need to think about a suitable impurity function
- We can use the **weighted average variance** of the target values as the measure of impurity
- If a split partitions the set of target values Y into mutually exclusive sets $\{Y_1, ..., Y_l\}$, the weighted average variance is then

$$\mathbb{V}ar(\{Y_1, ..., Y_l\}) = \sum_{j=1}^{l} \frac{|Y_j|}{|Y|} \mathbb{V}ar(Y_j) =$$

Regression Trees

- In order to adapt decision trees to regression problem, we need to think about a suitable impurity function
- We can use the **weighted average variance** of the target values as the measure of impurity
- If a split partitions the set of target values Y into mutually exclusive sets $\{Y_1, ..., Y_l\}$, the weighted average variance is then

$$\mathbb{V}ar(\{Y_1, ..., Y_l\}) = \sum_{j=1}^{l} \frac{|Y_j|}{|Y|} \mathbb{V}ar(Y_j) = \sum_{j=1}^{l} \frac{|Y_j|}{|Y|} \frac{1}{|Y_j|} \sum_{y \in |Y_j|} (y - \bar{y})^2 =$$

Regression Trees

- In order to adapt decision trees to regression problem, we need to think about a suitable impurity function
- We can use the **weighted average variance** of the target values as the measure of impurity
- If a split partitions the set of target values Y into mutually exclusive sets $\{Y_1, ..., Y_l\}$, the weighted average variance is then

$$\mathbb{V}ar(\{Y_1, ..., Y_l\}) = \sum_{j=1}^{l} \frac{|Y_j|}{|Y|} \mathbb{V}ar(Y_j) = \sum_{j=1}^{l} \frac{|Y_j|}{|Y|} \frac{1}{|Y_j|} \sum_{y \in |Y_j|} (y - \bar{y})^2 =$$

$$= \sum_{j=1}^{l} \Big(\frac{|Y_j|}{|Y|} \frac{1}{|Y_j|} \sum_{y \in |Y_j|} y^2 - \bar{y}^2 \Big) =$$

Regression Trees

- In order to adapt decision trees to regression problem, we need to think about a suitable impurity function
- We can use the **weighted average variance** of the target values as the measure of impurity
- If a split partitions the set of target values Y into mutually exclusive sets $\{Y_1, ..., Y_l\}$, the weighted average variance is then

$$\mathbb{V}ar(\{Y_1, ..., Y_l\}) = \sum_{j=1}^{l} \frac{|Y_j|}{|Y|} \mathbb{V}ar(Y_j) = \sum_{j=1}^{l} \frac{|Y_j|}{|Y|} \frac{1}{|Y_j|} \sum_{y \in |Y_j|} (y - \bar{y})^2 =$$

$$= \sum_{j=1}^{l} \Big(\frac{|Y_j|}{|Y|} \frac{1}{|Y_j|} \sum_{y \in |Y_j|} y^2 - \bar{y}^2 \Big) =$$

$$= \frac{1}{|Y|} \sum_{y \in |Y_j|} y^2 - \sum_{j=1}^{l} \frac{|Y_j|}{|Y|} \bar{y}^2$$

Regression Trees

- A split is considered good if it maximizes the impurity gain:

$$Imp(D) - Imp(\{D_1, \ldots, D_l\})$$

the overall impurity of child nodes should be maximally smaller than the impurity of the parent D

Regression Trees

- A split is considered good if it maximizes the impurity gain:

$$Imp(D) - Imp(\{D_1, \ldots, D_l\})$$

  the overall impurity of child nodes should be maximally smaller than the impurity of the parent D
- In case of regression we take $Imp = \mathbb{V}ar$ and we base our decision on **variance reduction**

Regression Trees

- A split is considered good if it maximizes the impurity gain:

$$Imp(D) - Imp(\{D_1, \ldots, D_l\})$$

  the overall impurity of child nodes should be maximally smaller than the impurity of the parent D
- In case of regression we take $Imp = \mathbb{V}ar$ and we base our decision on **variance reduction**
- Each leaf returns the average target value of the segment

Evaluating Classification Performance

- Accuracy and the error rate are the primary evaluation measures of classifiers

Evaluating Classification Performance

- Accuracy and the error rate are the primary evaluation measures of classifiers
- Accuracy is the proportion of correctly classified instances, its value on the test set is

$$acc = \frac{1}{|Te|} \sum_{\mathbf{x} \in Te} \mathbb{1}_{\{\hat{f}(\mathbf{x}) = f(\mathbf{x})\}}$$

Evaluating Classification Performance

- Accuracy and the error rate are the primary evaluation measures of classifiers
- Accuracy is the proportion of correctly classified instances, its value on the test set is

$$acc = \frac{1}{|Te|} \sum_{\mathbf{x} \in Te} \mathbb{1}_{\{\hat{f}(\mathbf{x}) = f(\mathbf{x})\}}$$

- Error rate is the complement of accuracy, that is the proportion of incorrectly classified instances

$$err = \frac{1}{|Te|} \sum_{\mathbf{x} \in Te} \mathbb{1}_{\{\hat{f}(\mathbf{x}) \neq f(\mathbf{x})\}} = 1 - acc$$

Evaluating Classification Performance

- Accuracy and the error rate are the primary evaluation measures of classifiers
- Accuracy is the proportion of correctly classified instances, its value on the test set is

$$acc = \frac{1}{|Te|} \sum_{\mathbf{x} \in Te} \mathbb{1}_{\{\hat{f}(\mathbf{x}) = f(\mathbf{x})\}}$$

- Error rate is the complement of accuracy, that is the proportion of incorrectly classified instances

$$err = \frac{1}{|Te|} \sum_{\mathbf{x} \in Te} \mathbb{1}_{\{\hat{f}(\mathbf{x}) \neq f(\mathbf{x})\}} = 1 - acc$$

- Test set error rate can be seen as an estimate of the probability of the model error on a random instance $\mathbf{x} \in \mathcal{X}$ from the instance space

$$\mathbb{P}(\hat{f}(\mathbf{x}) \neq f(\mathbf{x}))$$

Confusion Matrix

- Often it is useful to see the kind of errors that the classifier makes

	Predicted ⊕	Predicted ⊖	
Actual ⊕	**30**	**20**	50
Actual ⊖	**10**	**40**	50
	40	60	100

Confusion Matrix

- Often it is useful to see the kind of errors that the classifier makes

	Predicted ⊕	Predicted ⊖	
Actual ⊕	**30**	**20**	50
Actual ⊖	**10**	**40**	50
	40	60	100

- This example of a confusion matrix (also known as contingency table) shows prediction performance on 100 instances

Famous Meme


TRUE NEGATIVE
You're not pregnant
FALSE POSITIVE
You're pregnant
TYPE 1 ERROR
FALSE NEGATIVE
You're not pregnant
TYPE 2 ERROR
TRUE POSITIVE
You're pregnant

Terminology in binary classification

- The elements in the confusion matrix are called as follows:

	Predicted ⊕	Predicted ⊖	
Actual ⊕	**True positives**	**False negatives**	Actual positives
Actual ⊖	**False positives**	**True negatives**	Actual negatives
	Predicted positives	Predicted negatives	Total

Terminology in binary classification

- The elements in the confusion matrix are called as follows:

	Predicted ⊕	Predicted ⊖	
Actual ⊕	**True positives**	**False negatives**	Actual positives
Actual ⊖	**False positives**	**True negatives**	Actual negatives
	Predicted positives	Predicted negatives	Total

- We will denote

	Predicted ⊕	Predicted ⊖	
Actual ⊕	**TP**	**FN**	*Pos*
Actual ⊖	**FP**	**TN**	*Neg*
	PPos	*PNeg*	$\|Te\|$

Terminology in binary classification

- The elements in the confusion matrix are called as follows:

	Predicted ⊕	Predicted ⊖	
Actual ⊕	**True positives**	**False negatives**	Actual positives
Actual ⊖	**False positives**	**True negatives**	Actual negatives
	Predicted positives	Predicted negatives	Total

- We will denote

	Predicted ⊕	Predicted ⊖	
Actual ⊕	**TP**	**FN**	*Pos*
Actual ⊖	**FP**	**TN**	*Neg*
	PPos	*PNeg*	$\|Te\|$

- $acc = \frac{TP+TN}{|Te|}$ and $err = \frac{FP+FN}{|Te|}$

Terminology in binary classification

- The elements in the confusion matrix are called as follows:

	Predicted ⊕	Predicted ⊖	
Actual ⊕	**True positives**	**False negatives**	Actual positives
Actual ⊖	**False positives**	**True negatives**	Actual negatives
	Predicted positives	Predicted negatives	Total

- We will denote

	Predicted ⊕	Predicted ⊖	
Actual ⊕	**TP**	**FN**	*Pos*
Actual ⊖	**FP**	**TN**	*Neg*
	PPos	*PNeg*	$\|Te\|$

- $acc = \frac{TP+TN}{|Te|}$ and $err = \frac{FP+FN}{|Te|}$
- There are many other evaluation measures

Other measures

- Often it is usual to assess the classifier separately on positives and negatives

Other measures

- Often it is usual to assess the classifier separately on positives and negatives
- **True positive rate**(also known as **sensitivity**, **recall**) is the proportion of positives correctly classified $TPR = \frac{TP}{Pos} = rec$

Other measures

- Often it is usual to assess the classifier separately on positives and negatives
- **True positive rate**(also known as **sensitivity**, **recall**) is the proportion of positives correctly classified $TPR = \frac{TP}{Pos} = rec$
- **True negative rate**(also known as **specificity**) is the proportion of negatives correctly classified (accuracy on the negatives) $TNR = \frac{TN}{Neg}$

Other measures

- Often it is usual to assess the classifier separately on positives and negatives
- **True positive rate**(also known as **sensitivity**, **recall**) is the proportion of positives correctly classified $TPR = \frac{TP}{Pos} = rec$
- **True negative rate**(also known as **specificity**) is the proportion of negatives correctly classified (accuracy on the negatives) $TNR = \frac{TN}{Neg}$
- **False negative rate**(also known as **miss rate**) $FNR = \frac{FN}{Pos}$

Other measures

- Often it is usual to assess the classifier separately on positives and negatives
- **True positive rate**(also known as **sensitivity**, **recall**) is the proportion of positives correctly classified $TPR = \frac{TP}{Pos} = rec$
- **True negative rate**(also known as **specificity**) is the proportion of negatives correctly classified (accuracy on the negatives) $TNR = \frac{TN}{Neg}$
- **False negative rate**(also known as **miss rate**) $FNR = \frac{FN}{Pos}$
- **False positive rate**(also known as **false alarm rate**) $FPR = \frac{FP}{Neg}$

Other measures

- Often it is usual to assess the classifier separately on positives and negatives
- **True positive rate**(also known as **sensitivity**, **recall**) is the proportion of positives correctly classified $TPR = \frac{TP}{Pos} = rec$
- **True negative rate**(also known as **specificity**) is the proportion of negatives correctly classified (accuracy on the negatives) $TNR = \frac{TN}{Neg}$
- **False negative rate**(also known as **miss rate**) $FNR = \frac{FN}{Pos}$
- **False positive rate**(also known as **false alarm rate**) $FPR = \frac{FP}{Neg}$
- **Precision** (also known as positive predictive value) is the proportion of actual positives among predicted positives $prec = \frac{TP}{PPos}$

Other measures

- Often it is usual to assess the classifier separately on positives and negatives
- **True positive rate**(also known as **sensitivity**, **recall**) is the proportion of positives correctly classified $TPR = \frac{TP}{Pos} = rec$
- **True negative rate**(also known as **specificity**) is the proportion of negatives correctly classified (accuracy on the negatives) $TNR = \frac{TN}{Neg}$
- **False negative rate**(also known as **miss rate**) $FNR = \frac{FN}{Pos}$
- **False positive rate**(also known as **false alarm rate**) $FPR = \frac{FP}{Neg}$
- **Precision** (also known as positive predictive value) is the proportion of actual positives among predicted positives $prec = \frac{TP}{PPos}$
- **F-measure** is the harmonic mean of precision and recall $F = \frac{2 \cdot prec \cdot rec}{prec + rec}$

Imbalanced costs

- In imbalanced tasks the costs are also often imbalanced
  – False positives and false negatives can have very different costs

Imbalanced costs

- In imbalanced tasks the costs are also often imbalanced
  – False positives and false negatives can have very different costs
- We want to have many true positives, without having many false positives

Imbalanced costs

- In imbalanced tasks the costs are also often imbalanced
  – False positives and false negatives can have very different costs
- We want to have many true positives, without having many false positives
- For example in medical diagnostic testing
  – Few disease cases (positive class) – Many healthy cases (negative class)

Trade-off between TPR and FPR

- A classifier that outputs binary label hits a particular balance between TPR and FPR

Trade-off between TPR and FPR

- A classifier that outputs binary label hits a particular balance between TPR and FPR
- This cannot be changed without learning a new classifier

Trade-off between TPR and FPR

- A classifier that outputs binary label hits a particular balance between TPR and FPR
- This cannot be changed without learning a new classifier
- A better solution: ask classifiers to output scores:
  - Higher score means more likely positive
  - Lower score means more likely negative

Trade-off between TPR and FPR

- A classifier that outputs binary label hits a particular balance between TPR and FPR
- This cannot be changed without learning a new classifier
- A better solution: ask classifiers to output scores:
  – Higher score means more likely positive
  – Lower score means more likely negative
- By choosing the decision threshold we can change trade-off between TPR and FPR

Scoring classifiers for TP/FP trade-off

- Most classification models can output scores in addition to labels

Scoring classifiers for TP/FP trade-off

- Most classification models can output scores in addition to labels
- KNN (K nearest neighbours):
  - Score = proportion of positive neighbours

Scoring classifiers for TP/FP trade-off

- Most classification models can output scores in addition to labels
- KNN (K nearest neighbours):
  – Score = proportion of positive neighbours
- SVM (support vector machine):
  – Score = signed distance to the decision boundary

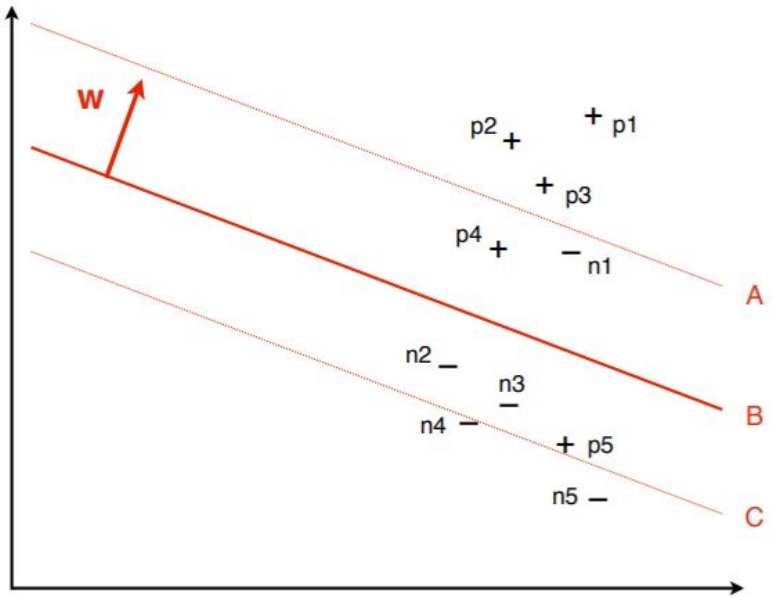
Scoring classifiers for TP/FP trade-off

- Most classification models can output scores in addition to labels
- KNN (K nearest neighbours):
  – Score = proportion of positive neighbours
- SVM (support vector machine):
  – Score = signed distance to the decision boundary
- DT (decision tree):
  – Score = proportion of positive instances in the decision leaf

Scoring classifiers for TP/FP trade-off

- Most classification models can output scores in addition to labels
- KNN (K nearest neighbours):
  - Score = proportion of positive neighbours
- SVM (support vector machine):
  - Score = signed distance to the decision boundary
- DT (decision tree):
  - Score = proportion of positive instances in the decision leaf
- RF (random forest):
  - Score = proportion of trees predicting positive

Scoring for SVM

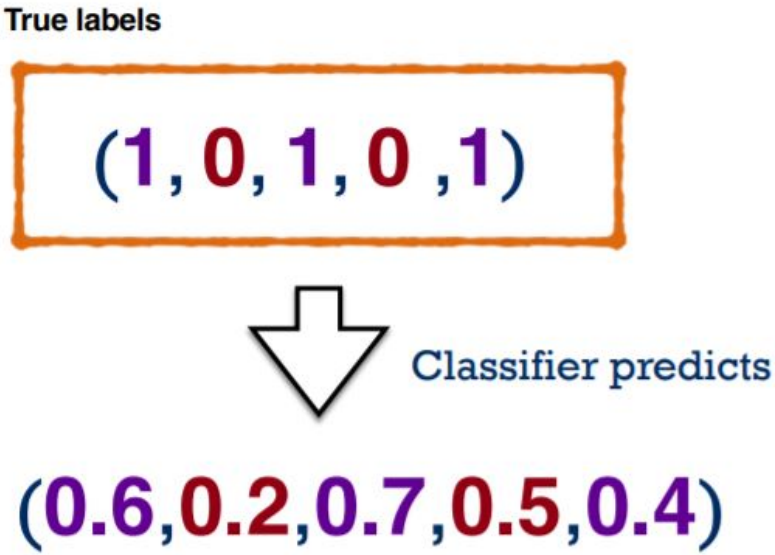


Threshold A: p1-p2-p3-n1-p4-n2-n3-p5-n4-n5

Threshold B: p1-p2-p3-n1-p4-n2-n3-p5-n4-n5

Threshold C: p1-p2-p3-n1-p4-n2-n3-p5-n4-n5

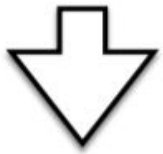
Example



Example

True labels

(1, 1, 0, 1, 0)

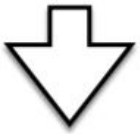


(0.7,0.6,0.5,0.4,0.2)

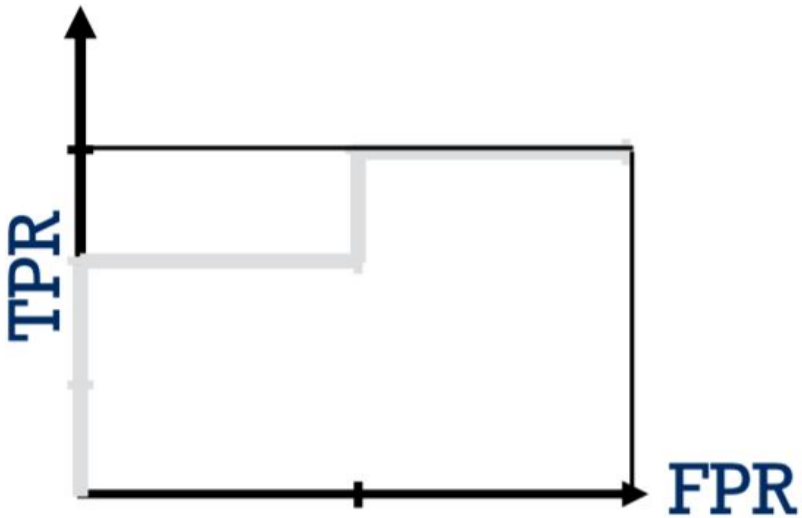
Example

True labels

(1, 1, 0, 1, 0)



(0.7,0.6,0.5,0.4,0.2)



$$TPR = TP/P$$

$$FPR = FP/(FP + TN)$$

Example

True labels

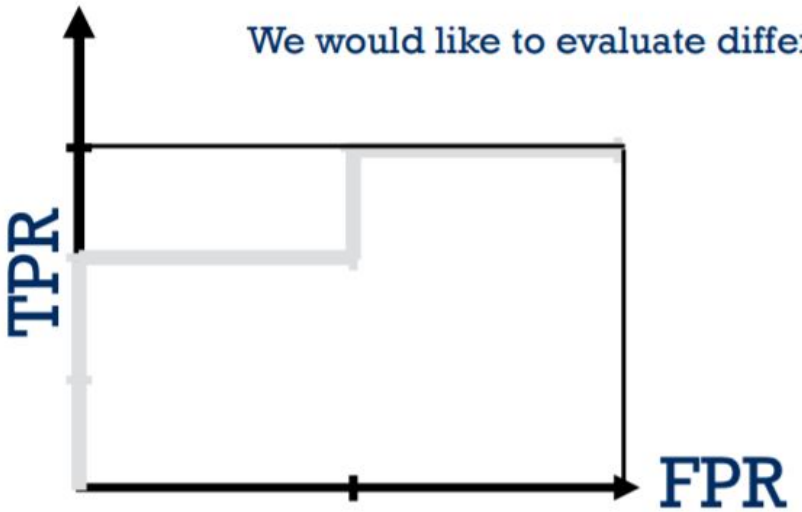
(1, 1, 0, 1, 0)

$$TPR = TP/P$$

$$FPR = FP/(FP + TN)$$

(0.7,0.6,0.5,0.4,0.2)

We would like to evaluate different **strictness levels** of our classifier



Example

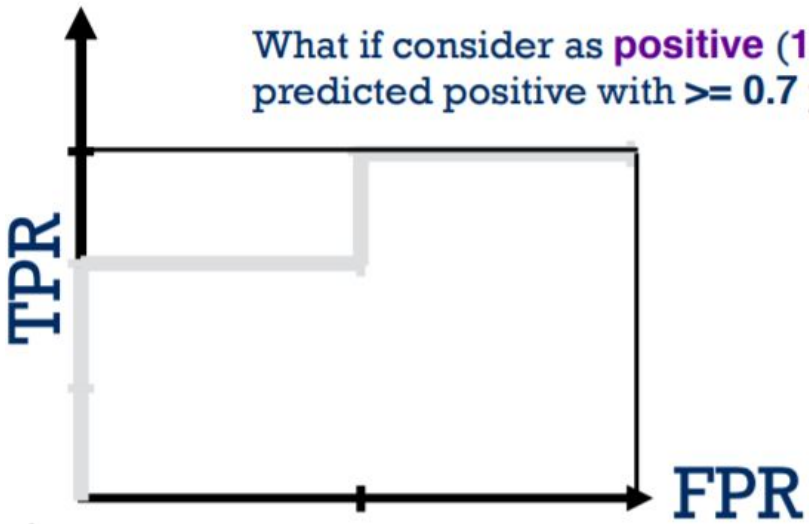
True labels

(1, 1, 0, 1, 0)

$$TPR = TP/P$$

$$FPR = FP/(FP + TN)$$

(0.7,0.6,0.5,0.4,0.2)



Example

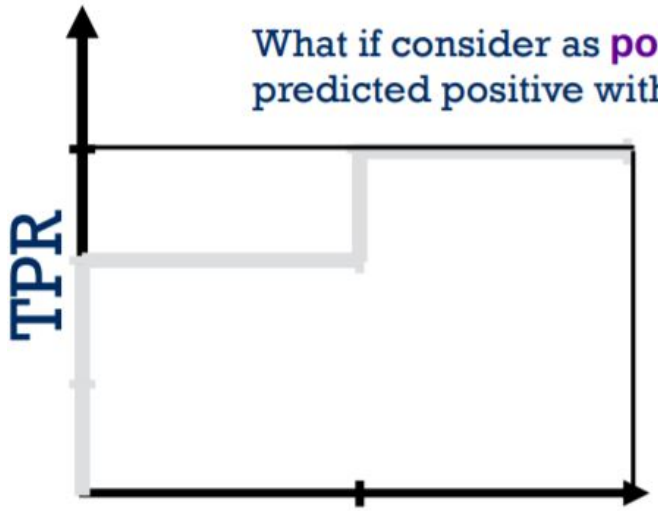
True labels

(1, 1, 0, 1, 0)

$$TPR = TP/P$$

$$FPR = FP/(FP + TN)$$

(0.7,0.6,0.5,0.4,0.2)



What if consider as **positive** (1) only instances that were predicted positive with **>= 0.7** probability?

What would **TPR** and **FPR** be in this case?

>= 0.7 $TPR = 1/3$

$FPR = 0/(0 + 2)$

Example

True labels

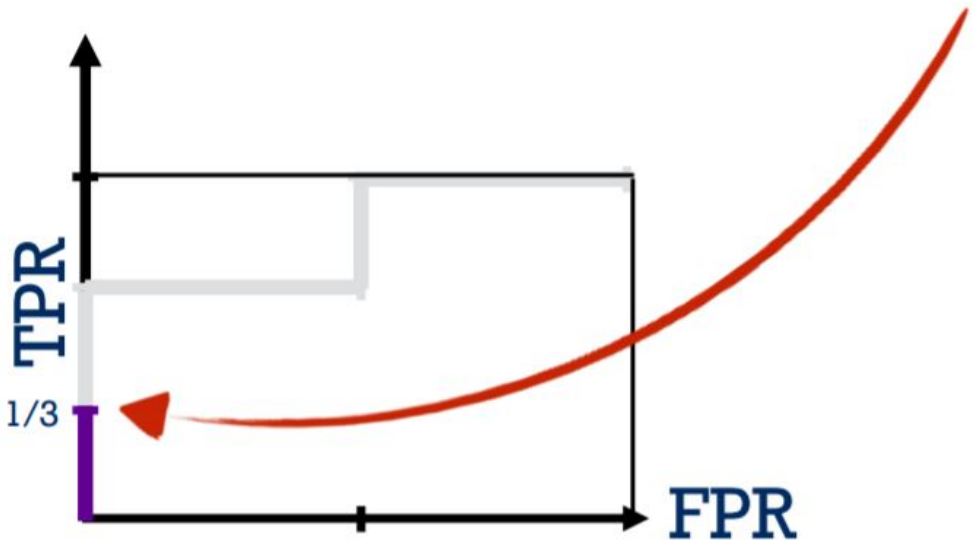
(1, 1, 0, 1, 0)

$$TPR = TP/P$$

$$FPR = FP/(FP + TN)$$

>= 0.7 TPR = 1/3 FPR = 0

(0.7,0.6,0.5,0.4,0.2) Let's plot this point on a graph



Example

True labels

(1, 1, 0, 1, 0)

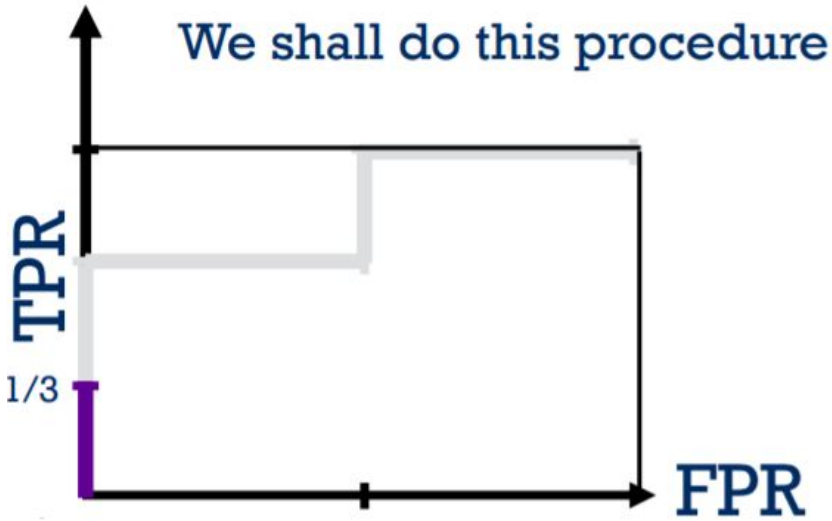
$$TPR = TP/P$$

$$FPR = FP/(FP + TN)$$

>= 0.7 TPR = 1/3 FPR = 0

(0.7,0.6,0.5,0.4,0.2)

We shall do this procedure for all **possible thresholds**



Example

True labels

(1, 1, 0, 1, 0)

$$TPR = TP/P$$

$$FPR = FP/(FP + TN)$$

(0.7,0.6,0.5,0.4,0.2)

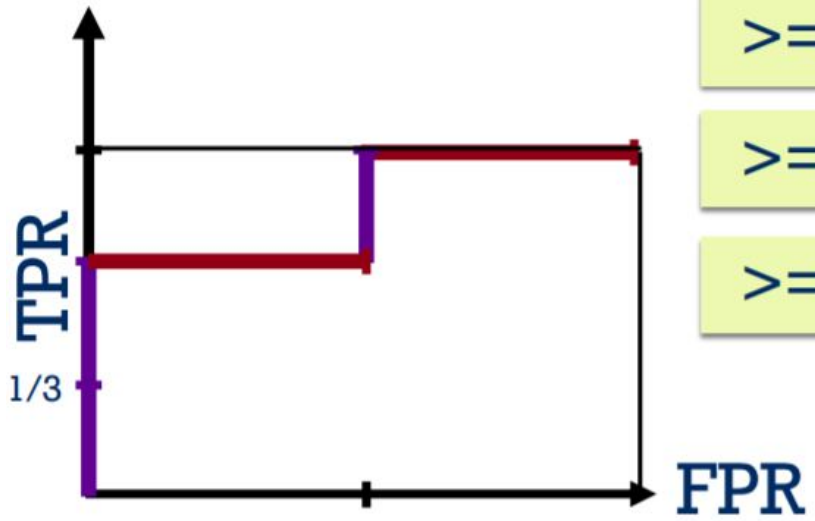
>= 0.7 TPR = 1/3 FPR = 0

>= 0.6 TPR = 2/3 FPR = 0

>= 0.5 TPR = 2/3 FPR =1/2

>= 0.4 TPR = 3/3 FPR =1/2

>= 0.2 TPR = 3/3 FPR =2/2



Example

True labels

(1, 1, 0, 1, 0)

$$TPR = TP/P$$

$$FPR = FP/(FP + TN)$$

(0.7,0.6,0.5,0.4,0.2)

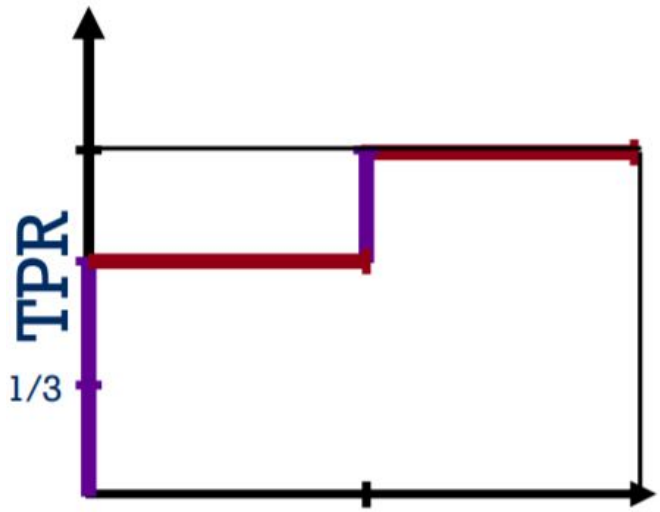
>= 0.7 TPR = 1/3 FPR = 0

>= 0.6 TPR = 2/3 FPR = 0

>= 0.5 TPR = 2/3 FPR =1/2

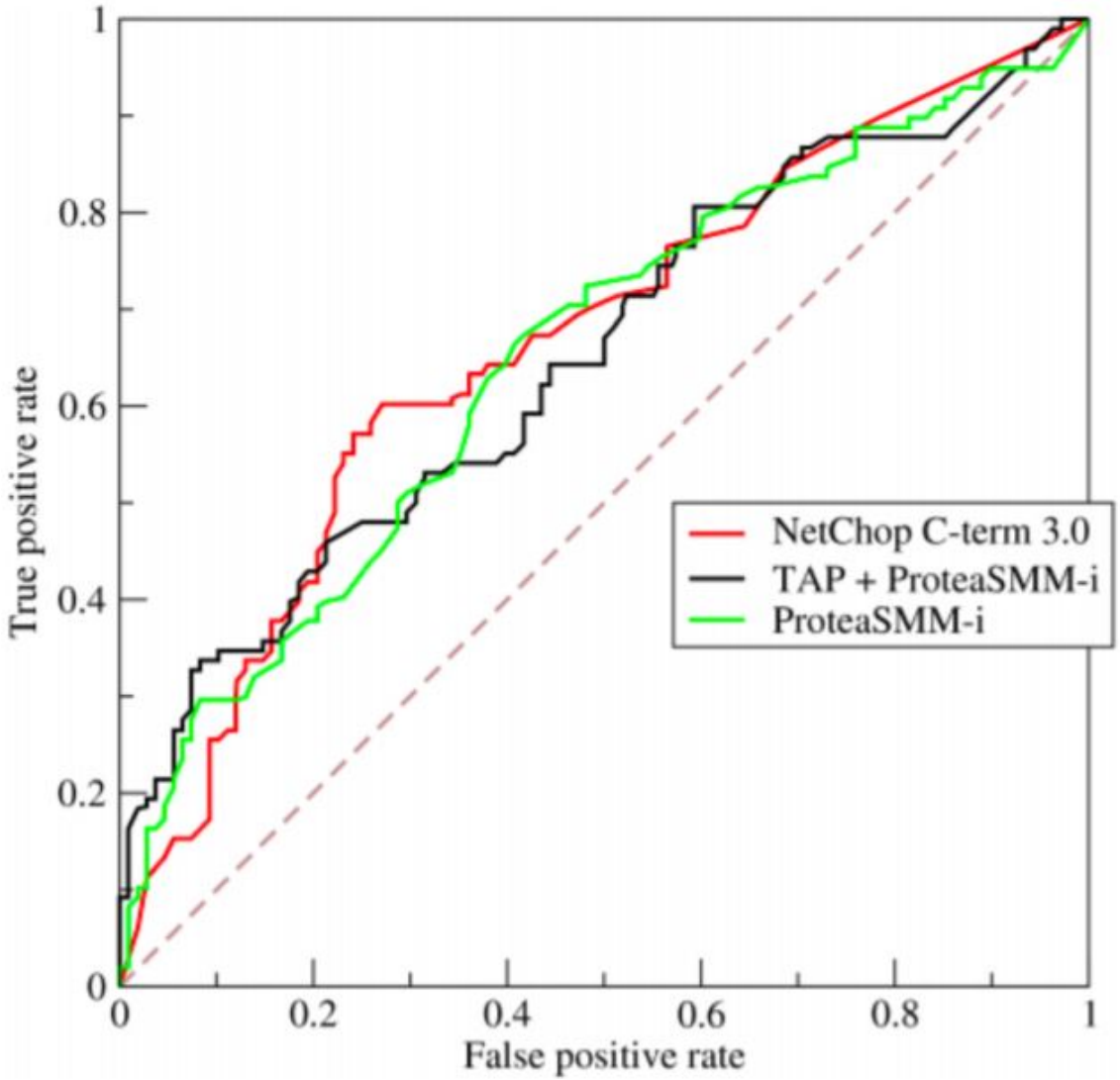
>= 0.4 TPR = 3/3 FPR =1/2

>



This curve is called the Receiver Operating Characteristic (ROC) and the area under it is denoted as AUC and for an ideal classifier it should be close to 1

Typical comparison of classifiers



What have we learned today?

- ✓ Regularization
- ✓ Logistic Regression
- ✓ Softmax Classifier
- ✓ Support Vector Regression (SVR)
- ✓ Regression Trees
- ✓ Evaluation and Scoring of Classifiers